HYBRID PARTICLE SWARM OPTIMIZATION DAN *K-MEANS* UNTUK *CLUSTERING* DATA PENENTUAN UKT PROPORSIONAL

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Maulian Eka Kusuma
NIM. 115060807111122

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

HYBRID PARTICLE SWARM OPTIMIZATION DAN *K-MEANS* UNTUK *CLUSTERING* DATA PENENTUAN UKT PROPORSIONAL

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Maulian Eka Kusuma
NIM. 115060807111122

Skripsi ini telah diuji dan dinyatakan lulus pada
15 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D
NIK. 850719 16 1 1 0268	NIP. 19720919 199702 1001

Mengetahui,
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T
NIP. 19670801 1 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2016

Maulian Eka Kusuma
NIM. 115060807111122

KATA PENGANTAR

Dengan segala kerendahan hati dan diri memanjatkan syukur kehadirat Allah SWT atas limpahan rahmat dan hidayahNya sehingga penulis dapat menyelesaikan penulisan skripsi dengan baik. Shalawat dan salam semoga senantiasa terlimpah curahkan kepada Rasulullah Muhammad shallallahu 'alaihi wasallam beserta keluarga dan sahabat-sahabat beliau.

Penulisan skripsi diajukan untuk memenuhi salah satu persayaratan dalam memperoleh gelar Sarjana Komputer pada Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Topik skripsi yang diajukan adalah **"*Hybrid Particle Swarm Optimization* Dan *K-Means* Untuk *Clustering* Data Penentuan UKT Proporsional".**

Kelancaran penulisan skripsi ini tidak lepas dari bantuan berbagai pihak, oleh karena itu penulis mengucapkan terimakasih dan rasa hormat kepada:

1. Bapak Imam Cholissodin, S.Si, M.Kom selaku dosen pembimbing I yang telah memberikan arahan dan menyediakan waktu untuk berdiskusi dengan penulis selama proses penulisan skripsi.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku dosen pembimbing II yang telah memberikan banyak masukan dan motivasi kepada penulis selama proses penulisan skripsi.
3. Bapak Heri Prawoto selaku narasumber dalam wawancara penentuan UKT Proporsional beserta staff dan karyawan sub bagian keuangan Universitas Brawijaya.
4. Bapak Drs. Marji, M.T selaku ketua Program studi Informatika/Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
5. Bapak Issa Arwani, S.Kom, M.Sc selaku sekretaris Program studi Informatika/Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
6. Bapak Denny Sagita Rusdianto, S.Kom, M.Kom selaku dosen penasehat akademik.
7. Laboratorium Komputasi Cerdas dan Visualisasi, utamanya Ibu Lailil Muflikhah, S.Kom, M.Sc selaku kepala laboratorium dan Bapak Fransiscus Priharsono, S.Kom selaku laboran.
8. Ibu Indriati, S.T, M.Kom dan Bapak Agus Wahyu Widodo, S.T, M.Cs selaku dosen penguji I dan dosen penguji II yang telah banyak memberikan koreksi dan saran terkait dengan perbaikan skripsi.
9. Kedua orang tua penulis, Bapak Suyadi dan Ibu Toyibatun. Terimakasih untuk doa, nasihat dan dukungan baik moril maupun materiil.
10. Kakak dan adik penulis, Dian Nur, Rima Savitri, Maulidya, Mei Risa, Qurrota Ayun. Terimakasih untuk doa dan dukungan.
11. Rini Roshida, Rizky Hetari, Kentia Dea Hapsari. Terimakasih untuk bantuan dan semangat selama pengerjaan skripsi ini.
12. Seluruh *civitas academica* Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa dalam penulisan skripsi ini masih jauh dari kata sempurna, oleh karenanya segala kritik dan saran yang bersifat membangun sangat penulis harapkan demi perbaikan skripsi ini. Akhirnya, semoga karya sederhana ini dapat memberikan manfaat bagi semua pihak sekaligus sebagai sarana pendukung pada pengembangan karya-karya terkait dimasa mendatang.

Malang, Januari 2016

Maulian Eka Kusuma
maulian.eka@gmail.com

ABSTRAK

Maulian Eka Kusuma. 2016: *Hybrid Particle Swarm Optimization* dan *K-Means* Untuk *Clustering* Data Penentuan UKT Proporsional. Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Pembimbing: Imam Cholissodin, S.Si., M.Kom dan Wayan Firdaus Mahmudy, S.Si., MT., Ph.D

Evaluasi terkait jumlah kategori penentuan dilakukan sebagai langkah penting menuju keseimbangan administrasi di lingkungan pendidikan. *Clustering* merupakan salah satu teknik pada *data mining* yang dapat digunakan untuk menentukan jumlah kategori dengan mengelompokkan data ke dalam beberapa kelompok sehingga data dalam satu kelompok memiliki kemiripan karakteristik yang tinggi dibandingkan data pada kelompok lain. Namun, pengelompokan secara manual membutuhkan banyak waktu terlebih jika data berjumlah besar atau kompleks. Sehingga, diperlukan sebuah sistem yang tidak hanya dapat mengelompokkan data secara otomatis namun juga mampu menghasilkan pengelompokkan lebih baik dengan waktu lebih efisien. *K-Means* merupakan algoritma pengelompokan yang sering digunakan karena mudah untuk diimplementasikan, efisien dan *powerful* untuk penanganan data dalam jumlah besar. Namun, kinerja *K-Means* sangat bergantung pada pemilihan pusat cluster awal sehingga solusi yang dihasilkan rentan terjebak pada daerah optimum lokal. Selain itu, *K-Means* tidak menjamin hasil pengelompokkan yang unik karena algoritma ini akan selalu menghasilkan hasil berbeda setiap kali program dijalankan. Salah satu pendekatan yang dapat digunakan untuk mengatasi permasalahan ini adalah dengan menerapkan algoritma optimasi yaitu *Particle Swarm Optimization* (PSO). PSO mengeksplorasi ruang pencarian untuk menemukan pusat *cluster* optimum berdasarkan nilai *cost* partikel. Nilai *cost* dirancang untuk meminimalkan jarak antara data dengan pusat *cluster* sehingga semakin kecil nilai *cost* maka semakin besar peluang sebuah partikel terpilih sebagai solusi.

Penelitian ini menggunakan *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM) untuk *clustering* data UKT Proporsional. Kualitas *clustering* dievaluasi menggunakan metode *Silhouette Coefficient* dengan memeriksa seberapa baik *separation* dan *compactness* sebuah *cluster*. Hasil percobaan menunjukkan bahwa algoritma HPSOKM menghasilkan pengelompokan lebih baik dibandingkan dengan algoritma *K-Means*.

Kata kunci: Pengelompokan, Optimasi, *Particle Swarm Optimization, K-Means, Silhouette Coefficient*

ABSTRACT

Maulian Eka Kusuma. 2016: A Hybrid Particle Swarm Optimization and K-Means For Clustering Of Data UKT Proporsional. Programme of Information Technology and Computer Sience, Brawijaya University. Advisors: Imam Cholissodin, S.Si., M.Kom and Wayan Firdaus Mahmudy, S.Si., MT., Ph.D

Evaluation of the number of categories determination is always done as an important step toward the administration balance in educational environment. Clustering, one of techniques in data mining, can be used to determine the number of categories by grouping the objects into clusters so that objects within a cluster have high similarity than objects in other clusters. However, clustering manually is time-consuming when the datasets are large or complex. We need a system that cannot only automatically clustering, but it produces better results and more efficient in terms of execution time. K-Means is the most commonly used partitioned clustering algorithms because it can be easily implemented, efficient, and powerful for the large datasets. As the weakness of algorithms, the performance of K-Means depends to the selection of initial cluster centers and may be trapped in local optimum solution. In addition, K-Means does not guaranteed unique clustering because it will always generates different results with randomly chosen initial cluster centers for each program runs. In order to solve this problem, one approach is to use the optimization algorithms such as Particle Swarm Optimization (PSO). PSO will explore the search space to find the optimum cluster centers based on the cost of the particles. The cost values were designed to minimize the distance between data and cluster centers so that the smaller the values of cost, the greater chance of the particles selected to be a solution.

This research uses Hybrid Particle Swarm Optimization and K-Means (HPSOKM) for clustering of UKT Proportional. The quality of clustering were evaluated using Silhouette Coefficient by examining how well the clusters are separated and how compact the clusters are. The experiments show that the HPSOKM algorithms produces better results of clustering compared to K-Means.

Keyword: Clustering, Optimization, Particle Swarm Optimization, K-Means, Silhouette Coefficient

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bab ini menjelaskan tentang latar belakang pemilihan masalah, alternatif penyelesaian masalah, rumusan masalah, manfaat dan tujuan penelitian, batasan masalah, serta sistematika penulisan yang digunakan dalam penelitian.

1.1 Latar Belakang

Uang Kuliah Tunggal (UKT) Proporsional merupakan aturan penentuan biaya kuliah dengan sistem pembayaran sekaligus mencakup SPP Proporsional dan Sumbangan Pengembangan Fasilitas Pendidikan (SPFP) untuk masa studi delapan semester pada jenjang Strata 1 (S1). Berdasarkan kebijakan Universitas Brawijaya, UKT Proporsional dikelompokkan ke dalam beberapa kategori. Kebijakan tersebut bertujuan agar besarnya biaya UKT Proporsional yang dibayarkan tepat sasaran yakni sesuai dengan kemampuan mahasiswa sekaligus memberikan kesempatan bagi mahasiswa yang kurang mampu untuk dapat melanjutkan pendidikan ke jenjang perguruan tinggi (Prawoto, 2015).

Pengelompokan data ke dalam beberapa kategori menimbulkan aliran informasi dalam jumlah yang besar. Selain itu, evaluasi terkait jumlah kategori penentuan UKT Proporsional juga dilakukan secara berkala. Evaluasi tersebut bertujuan untuk menjaga agar sebaran kategori konsisten normal sehingga jalannya administrasi di lingkungan perguruan tinggi berlangsung lancar (Prawoto, 2015). Oleh karena itu, dibutuhkan pendekatan khusus untuk mengelompokan data secara otomatis. Pengelompokan juga berfungsi untuk memudahkan pencarian informasi mengenai suatu kejadian dalam kurun waktu tertentu. Sehingga, kajian terhadap pengelompokkan data semakin dibutuhkan agar data dalam jumlah besar dapat diproses dan dimanfaatkan dengan tetap memperhatikan estimasi waktu dan kualitas hasil (Afivi, 2005).

Penyelesaian pengelompokan data salah satunya dapat menggunakan teknik *data mining* yaitu *clustering*. *Clustering* secara praktik membagi sejumlah data menjadi beberapa kelompok dimana variabel target (*group* atau kategori) untuk pengelompokan data tidak diketahui dan didefinisikan sebelumnya. Prinsip pengelompokan data menggunakan teknik *clustering* adalah memaksimalkan nilai kemiripan data dalam satu kelompok dan meminimalkan nilai kemiripan data pada kelompok yang lain (Kusrini, et al., 2009).

K-Means merupakan algoritma *clustering* yang paling sering digunakan karena mudah diimplementasikan, efisien, dan *powerful* untuk penanganan data dalam jumlah besar. Sebaliknya, algoritma ini sangat sensitif terhadap inisialisasi pusat *cluster* awal sehingga solusi yang dihasilkan mudah terjebak pada daerah optimum lokal. Salah satu pendekatan untuk menginisialisasi pusat *cluster* adalah dengan menerapkan sebuah algoritma pencarian global seperti *Particle Swarm Optimization* (PSO) (Rana, et al., 2010). Sama halnya dengan *K-Means,* algoritma PSO juga mudah diimplementasikan karena algoritma ini memiliki parameter yang lebih sedikit jika dibandingkan dengan teknik optimasi lain. Setiap partikel pada

PSO juga akan mengingat posisi terbaik yang pernah dicapai sehingga kemampuan memorinya menjadi lebih efektif (Valle, et al., 2008). Meskipun sederhana, PSO terbukti sukses diterapkan pada beberapa permasalahan yang kompleks. Penelitian oleh Mahmudy (2014) menerapkan PSO untuk menyelesaikan optimasi *part type selection* dan *machine loading problem* pada *flexible manufacturing system* (FMS). Permasalahan tersebut sulit dicari penyelesaiannya namun menjadi penting karena memiliki pengaruh besar terhadap produktivitas FMS (Mahmudy, 2014a). Pada pengembangan penelitian selanjutnya, metode *random injection* ditambahkan sebagai penanganan konvergensi dini dan untuk meningkatkan kemampuan eksplorasi algoritma PSO (Mahmudy, 2015). Kesimpulan dari kedua penelitian menunjukkan bahwa untuk memproses data dalam jumlah besar, PSO mampu menghasilkan solusi baik optimum maupun mendekati optimum dengan waktu komputasi yang relatif cepat.

Penggunaan algoritma PSO untuk menyelesaikan optimasi pusat *cluster* pada algoritma *K-Means* telah dilakukan oleh beberapa peneliti. Penelitian pertama oleh Merwe, et al (2003) yang menerapkan algoritma *K-Means* terlebih dahulu sebanyak satu kali kemudian hasil yang telah diperoleh dijadikan sebagai salah satu inisialisasi partikel pada algoritma PSO. Penelitian kedua oleh Rana, et al (2010) yang menerapkan algoritma PSO dan *K-Means* secara sekuensial dan mengamati pengaruh penambahan jumlah atribut data terhadap hasil *clustering.* Penelitian ketiga oleh Cui, et al (2005) yang menerapkan algoritma PSO+*K-Means* pada objek yang berbeda untuk penyelesaian permasalahan *clustering* dokumen.

Kajian dari beberapa penelitian diatas menunjukkan bahwa algoritma *Particle Swarm Optimization* dapat digunakan untuk mengoptimasi pusat *cluster* awal pada algoritma *K-Means.* Selain itu, penggabungan dua metode dapat meningkatkan kinerja algoritma dalam menemukan solusi optimum (Rana, et al., 2010). Sehingga, kombinasi antara kemampuan menemukan solusi optimum pada ruang pencarian global PSO dengan kecepatan konvergensi *K-Means* diharapkan dapat menghasilkan hasil *clustering* yang lebih baik (Cui, et al., 2005).

Berdasarkan latar belakang yang telah dijabarkan sebelumnya, penulis mengajukan penelitian dengan judul **"*Hybrid Particle Swarm Optimization* Dan *K-Means* Untuk *Clustering* Data Penentuan UKT Proporsional*".***

1.2 Rumusan Masalah

Rumusan masalah pada skripsi ini antara lain:

1. Bagaimana mengimplementasikan *Hybrid Particle Swarm Optimization* dan *K-Means* untuk *clustering* data UKT Proporsional?
2. Bagaimana nilai evaluasi yang dihasilkan sistem menggunakan *Hybrid Particle Swarm Optimization* dan *K-means* untuk *clustering* data UKT Proporsional?
3. Bagaimana peningkatan kualitas solusi menggunakan *Hybrid Particle Swarm Optimization* dan *K-means* dibandingkan dengan *K-Means* untuk *clustering* data UKT Proporsional?

1.3 Tujuan

Tujuan pada skripsi ini antara lain:

1. Mengimplementasikan sistem *clustering* data penentuan UKT Proporsional
2. Mengetahui nilai evaluasi yang dihasilkan sistem dengan menggunakan algoritma *Hybrid Particle Swarm Optimization* dan *K-Means*

1.4 Manfaat

Hasil daripada skripsi ini diharapkan dapat memberikan manfaat antara lain:

1. Mengelompokkan data UKT Proporsional ke dalam *k* kategori
2. Memberikan rekomendasi jumlah kategori penentuan UKT Proporsional
3. Meringankan beban biaya kuliah karena penentuan UKT proporsional disesuaikan dengan kemampuan mahasiswa
4. Sebagai bahan evaluasi untuk menentukan jumlah kategori UKT Proporsional dimasa mendatang

1.5 Batasan Masalah

Batasan masalah pada skripsi ini antara lain:

1. Data yang digunakan merupakan data UKT Proporsional tahun 2014 dari jalur SNMPTN dan SBMPTN Jurusan Teknik Informatika PTIIK UB
2. Data terdiri dari 6 atribut antara lain panghasilan orang tua, rekening listrik, rekening telepon, rekening PAM/PDAM, rekening pajak bumi dan bangunan (PBB), serta rekening pajak kendaraan bermotor (PKB)
3. Sistem hanya melakukan pengelompokan data UKT Proporsional ke dalam *k* kategori dan bukan menentukan besar nominal untuk setiap kategori yang dihasilkan
4. Tidak memperhatikan kriteria pengajuan penurunan biaya UKT Proporsional

1.6 Sistematika Penulisan

Sistematika penulisan pada skripsi ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Menjelaskan latar belakang pemilihan masalah, perumusan masalah, batasan masalah, manfaat penelitian, tujuan penelitian, dan sistematika penulisan skripsi.

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Menjelaskan kajian pustaka dan dasar teori yang mendasari proses implementasi algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* untuk menyelesaikan *clustering* data penentapan UKT Proporsional.

BAB 3 METODOLOGI PENELITIAN

Menjelaskan tahapan penelitian, teknik pengumpulan data, implemetasi algoritma, spesifikasi kebutuhan sistem meliputi kebutuhan perangkat keras dan kebutuhan perangkat lunak, serta rancangan pengujian sistem.

BAB 4 PERANCANGAN

Menjelaskan deskripsi sistem, formulasi permasalahan, siklus algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM), siklus penyelesaian *clustering* data penentuan UKT Proporsional menggunakan algoritma HPSOKM, serta perancangan antar muka.

BAB 5 IMPLEMENTASI

Menjelaskan proses implementasi sistem *clustering* data penentuan UKT Proporsional menggunakan algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* sesuai dengan perancangan sistem yang telah dibuat.

BAB 6 PENGUJIAN DAN PEMBAHASAN

Menjelaskan tahapan pengujian dan pembahasan dari sistem *clustering* data penentuan UKT Proporsioanl menggunakan algoritma *Hybrid Particle Swarm Optimization* dan *K-Means.*

BAB 7 PENUTUP

Menjelaskan kesimpulan yang diperoleh berdasarkan hasil implementasi dan pengujian serta saran untuk pengembangan sistem lebih lanjut.

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini menjelaskan kajian pustaka dan dasar teori untuk mendukung penelitian yang diajukan, diantaranya: kajian pustaka, gambaran umum seleksi masuk UB, teori: *data mining*, *clustering,* algoritma *Particle Swarm Optimization* (PSO)*,* algoritma *K-Means,* algoritma *Hybrid Particle Swarm Optimization* dan *K-Means,* metode *Min-Max Normalization,* dan metode *Silhouette Coefficient*.

2.1 Kajian Pustaka

Penelitian tentang penerapan algoritma pencarian global untuk mengoptimasi pusat *cluster* awal pada algoritma *K-Means* telah dilakukan oleh beberapa peneliti. Kajian pustaka bertujuan untuk mendukung usulan penelitian sekaligus membandingkannya dengan penelitian sebelumnya.

Penelitian pertama dilakukan oleh Rana, et al., (2010) berjudul *"A Hybrid Squential Approach for Data Clustering Using K-Means and Particle Swarm Optimization Algorithm"*. Pada penelitian tersebut menunjukkan bahwa pembentukan *cluster* hanya dengan menggunakan PSO tidaklah cukup, sebaliknya perbaikan lebih lanjut oleh *K-Means* menyebabkan pembentukan *cluster* menjadi lebih komposit dan kental. Selain itu pada data yang sama, penambahan jumlah atribut menyebabkan jarak *intra-cluster* semakin kecil dan jarak *inter-cluster* semakin besar dibandingkan sebelum dilakukan penambahan atribut data. Kesimpulan dari penelitian ini adalah kelemahan *K-Means* terhadap inisialisasi pusat *cluster* dapat diminimalkan dengan menerapkan algoritma *Particle Swarm Optimization* (Rana, et al., 2010).

Penelitian kedua dilakukan oleh Cui et al., (2005) berjudul *"A Hybrid Squential Approach for Data Clustering Using K-Means and Particle Swarm Optimization Algorithm".* Penelitian tersebut menunjukkan bahwa selain digunakan untuk mengukur jarak rata-rata antara pusat *cluster* dengan dokumen*,* nilai *fitness* juga digunakan untuk mengevaluasi kualitas *cluster.* Metode PSO menghasilkan nilai *Euclidean* lebih kecil dibandingkan *K-Means* sebaliknya *K-Means* menghasilkan nilai *Cosine* lebih kecil dibandingkan PSO. Namun jika metode ini digabungan (PSO+*K-Means)* maka baik nilai *Euclidean* dan *Cosine* nya berturut-turut menjadi paling kecil. Kesimpulan dari penelitian ini adalah algoritma *Hybrid* PSO + *K-Means* menghasilkan pengelompokkan yang lebih baik dibandingkan jika kedua algoritma dijalankan secara terpisah (Cui, et al., 2005).

Penelitian ketiga dilakukan oleh Merwe, et al., (2003) berjudul *"Data Clustering Using Particle Swarm Optimization".* Pada penelitian tersebut algoritma *K-Means* dijalankan terlebih dahulu sebanyak satu kali kemudian dilanjutkan dengan algoritma PSO. Inisialisasi partikel pada algoritma PSO salah satunya diperoleh dari pusat *cluster* hasil *clustering K-Means.* Prosedur PSO yang digunakan adalah standar *global best* PSO dan nilai *fitness* dihitung berdasarkan penjumlahan jarak minimum data dalam satu *cluster* dibagi dengan banyaknya anggota dalam satu *cluster* tersebut. Kesimpulan dari penelitian ini adalah

algoritma PSO untuk *clustering* memiliki tingkat konvergensi lebih baik yaitu nilai *quantization error* lebih rendah, dan secara umum jarak *inter-cluster* lebih besar serta jarak *intra-cluster* lebih kecil (Merwe, et al., 2003).

Penelitian keempat dilakukan oleh Karegowda, et al., (2012) berjudul *"Genetic Algorithm Based Dimensionality Reduction for Improving Performance of K-Means Clustering: A Case Study for Categorization of Medical Dataset".* Pada penelitian tersebut terdiri dari dua tahap yaitu optimasi pusat *cluster* dan reduksi dimensi. Algoritma Genetika *(Genetic Algorithm)* digunakan untuk mengoptimasi pusat *cluster* sekaligus mereduksi dimensi sementara *Entropy Fuzzy Clustering* (EFC) hanya digunakan untuk mereduksi dimensi. Hasil yang diperoleh menunjukkan bahwa akurasi tertinggi diperoleh dengan menerapkan Algoritma Genetika untuk mengoptimasi pusat *cluster* dan mereduksi dimensi. Kesimpulan dari penelitian ini adalah performa dari algoritma *K-Means clustering* tidak hanya bergantung pada inisialisasi pusat *cluster* namun juga pada pemilihan dimensi data yang signifikan (Karegowda, et al., 2012).

Pada penelitian yang diajukan, algoritma *Particle Swarm Optimization* (PSO) dikombinasikan dengan algoritma *K-Means* untuk *clustering* data penentapan UKT Proporsional. Kedua algoritama dijalankan secara sekuensial dengan algoritma PSO untuk mengoptimasi pusat *cluster* terlebih dahulu kemudian hasil pusat *cluster* yang paling optimum digunakan sebagai inisialisasi pusat *cluster* awal pada algoritma *K-Means.*

Perbandingan objek, metode, hasil, masukan, proses, dan keluaran antara penelitian sebelum dan usulan penelitian ditunjukkan pada Tabel 2.1

Tabel 2.1 Perbandingan Penelitian Sebelum Dan Usulan

Judul Penelitian	Objek Penelitian	Metode Penelitian	Hasil Penenlitian
	Masukan	Proses	Keluaran
Document Clustering Analysis Based on Hybrid PSO + K-Means Algorithm. (Cui et al, 2005)	*Dataset* dokumen dari TREC, *Text Retrieval Conference* tahun 1999	*Hybrid Particle Swarm Optimization* (PSO) + *K-Means*	▪ Persamaan *fitness* juga digunakan untuk mengevaluasi kualitas *cluster* ▪ Hasil *clustering* mengggunakan *hybrid* PSO + *K-Means* lebih tinggi dibandingkan jika PSO atau *K-Means* dijalankan secara terpisah
	▪ bobot inersia ▪ koefisien akselerasi 1&2	Proses PSO: ▪ Memilih *k* dokumen sebagai	▪ Nilai *Euclidean* ▪ Nilai *Cosine Correlation*

	▪ ukuran *swarm* ▪ jumlah *cluster* ▪ jumlah iterasi	pusat *cluster (centroid)* ▪ Mengelompokkan dokumen ke *cluster* terdekat ▪ Menghitung nilai *fitness* ▪ Meng-*update* kecepatan dan posisi partikel Proses *K-Means* ▪ Inisialisasi *centroid* dari hasil PSO ▪ Mengelompokkan dokumen ke *cluster* terdekat ▪ Menghitung pusat *cluster* baru	
A Hybrid Squential Approach For Data Clustering Using K-Means and Particle Swarm Optimization Algorithm. (Rana et al, 2010)	*Dataset: Artificial Problem 1, Artificial Problem 2, Wine,* dan *Iris*	*Hybrid Particle Swarm Optimization* (PSO) dan *K-Means*	▪ Nilai *quantization error* ▪ Nilai jarak *intra-cluster* ▪ Nilai jarak *inter-cluster*
	▪ bobot inersia, ▪ koefisien akselerasi 1&2 ▪ ukuran *swarm* ▪ jumlah *cluster* ▪ jumlah iterasi	Proses PSO ▪ Inisialisasi partikel ▪ Inisialisasi posisi dan kecepatan awal partikel ▪ Evaluasi *fitness* ▪ Meng-*update personal best global best* ▪ Meng-*update* posisi dan kecepatan partikel Proses *K-Means* ▪ Mengelompokan data ke *cluster* terdekat dengan *centroid* awal	▪ Parameter PSO mempengaruhi hasi *clustering* ▪ Penambahan jumlah atribut mempengaruhi nilai *intra-cluster* dan *inter-cluster* ▪ Kelemahan dalam pemilihan pusat *cluster* pada *K-Means* dapat diminimalkan dengan PSO

		diperoleh dari PSO ▪ Menghitung pusat *cluster* baru	
Data Clustering Using Particle Swarm optimization (Merwe et al, 2003)	*Dataset: Artificial 1, Artificial 2, Iris, Wine, Breast-cancer, Automotive*	*Hybrid PSO* dan *K-Means*	▪ Nilai *quantization error* ▪ Nilai jarak *intra-cluster* ▪ Nilai jarak *inter-cluster*
	Paremeter *K-Means:* Jumlah *cluster* Parameter PSO: ▪ bobot inersia, ▪ koefisien akselerasi 1&2 ▪ ukuran *swarm* ▪ jumlah *cluster* ▪ jumlah iterasi	Proses *K-Means:* ▪ Memilih *k* pusat *cluster* ▪ Mengelompokkan data ke pusat *cluster* terdekat ▪ Menghitung pusat *cluster* baru Proses PSO: ▪ Inisialisasi partikel, dengan salah satu partikel merupakan pusat *cluster* yang diperoleh dari *K-Means* ▪ Menghitung nilai *fitness* ▪ *Update global best* dan *local best* ▪ *Update* kecepatan dan posisi partikel	Pencarian berbasis populasi pada PSO mengurangi dampak sensitifitas *K-Means* terhadap inisialisasi pusat *cluster* awal
Genetic Algorithm Based Dimensionality Reduction for Improving Performance of K-Means Clustering: A	*Medical dataset "Pima Indian Diabetes* (PIDD) *and Heart Statlog Dataset"*	Reduksi dimensi menggunakan *Genetic Algorithm* (GA) dan inisialisasi pusat *cluster* awal *K-Means* menggunakan GA dan *Entropy based*	Kinerja *K-Means* tidak hanya bergantung pada inisialisasi pusat *cluster* awal namun juga pada jumlah dimensi data

Case Study for Categorization of Medical Dataset (Karegowda et al, 2012)		*Fuzzy Clustering* (EFC)	
	Parameter GA: ▪ *mutation rate* ▪ *crossover rate* ▪ jumlah populasi ▪ jumlah generasi Parameter EFC: ▪ *dataset* dengan *n* sampel ▪ nilai threshold *(β)* ▪ konstanta *α*	▪ Reduksi dimensi (menggunakan *binary* dan *integer encoded* GA) ▪ Inisialisasi pusat *cluster* (GA): - *Input* parameter GA - *Crossover* - *Mutation* ▪ Inisialisasi pusat *cluster* (GA): - Menghitung *entropy* - Identifikasi x_i - Menghapus x_i dan data	*K-Means* dengan pusat *cluster* secara acak: ▪ Jumlah fitur, TP, FP, TN, FN, jumlah iterasi, *Sum of Square Error* (SSE) dan *error* *K-Means* dengan pusat *cluster* diperoleh dari GA dan EFC: ▪ Jumlah fitur, TP, FP, TN, FN, Jumlah iterasi, *recall, F-measure, precision*, dan *error*
Hybrid Particle Swarm Optimization Dan *K-Means* Untuk *Clustering* Data Penentuan UKT Proporsional (Usulan Penelitian)	Data UKT Proporsional Tahun 2014 dari prodi Teknik Informatika PTIIK UB	*Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM)	Algoritma HPSOKM lebih baik dan lebih stabil dibandingkan algoritma *K-Means* standar dalam menyelesaikan *clustering* data
	▪ bobot inersia max (w_{max}) ▪ bobot inersia min (w_{min}) ▪ koefisien akselerasi 1 (c_1) ▪ koefisien akselerasi 2 (c_2) ▪ jumlah *cluster* ▪ jumlah iterasi ▪ ukuran *swarm*	Proses PSO ▪ Normalisasi data ▪ Memilih *k* jumlah *cluster* ▪ Menghitung nilai *fitness* ▪ Meng-*update pBest* dan *gBest* ▪ Meng-*update* kecepatan dan posisi partikel Proses *K-Means* ▪ Inisialisasi *centroid* dari hasil PSO	Hasil pengelompokkan data setiap *cluster* dan nilai evaluasi hasil *clustering* data UKT Proporsional

		▪ Mengelompokkan data ke *cluster* terdekat ▪ Menghitung pusat *cluster* baru	

Sumber: (Rana, et al., 2010), (Cui, et al., 2005), (Merwe, et al., 2003), (Karegowda, et al., 2012)

2.2 Gambaran Umum Seleksi Masuk Universitas Brawijaya

Seleksi Masuk Universitas Brawijaya (SELMA UB) merupakan seleksi penerimaan mahasiswa baru yang dilaksanakan oleh Universitas Brawijaya. SELMA UB memiliki beberapa jalur masuk diantaranya, jalur Seleksi Nasional Masuk perguruan Tinggi Negeri (SNMPTN), jalur Seleksi Bersama Nasional Masuk Perguruan Tinggi Negeri (SBMPTN), jalur Seleksi Program Minat dan Kemampuan (SPMK), jalur Seleksi Program Khusus Penyandang Disabilitas (SPKPD), jalur *International Student Admissions,* dan jalur Seleksi Alih Program (SAP) (Selma, 2015a). Dari beberapa jalur masuk tersebut, fokus pembahasan penelitian ini adalah penentuan kategori UKT Proporsional melalui jalur SNMPTN dan SBMPTN.

SNMPTN 2014 dan SBMPTN 2014 merupakan seleksi penerimaan mahasiswa baru yang dilaksanakan secara nasional. Pada SNMPTN 2014 kriteria penerimaan dinilai berdasarkan nilai rapor dan daftar prestasi. Calon mahasiswa baru yang masuk melalui jalur SNMPTN seluruh biaya administrasi ditanggung oleh pemerintah. Sementara pada SBMPTN 2015 kriteria penerimaan dinilai berdasarkan hasil test tulis atau ujian ketrampilan dengan biaya pendaftaran yang dibebankan pada peserta (Selma, 2015b).

Berdasarkan Surat Keputusan Rektor Universitas Brawijaya Nomor 276/SK/2013, penentuan UKT Proporsional melalui jalur SNMPTN dan SBMPTN mulai tahun 2013 dibayarkan sekaligus untuk satu tahun pertama dan dibayarkan per semester pada tahun-tahun berikutnya. Sedangkan mulai tahun 2014 terdapat sedikit perubahan yaitu berdasarkan Surat Keputusan Rektor Universitas Brawijaya Nomor 340/SK/2014, penentuan UKT Proporsional dibayarkan per semester. Biaya UKT Proporsional mencakup SPP Proporsional dan Sumbangan Pembangunan Fasilitas Pendidikan (SPFP) yang dalam pengelolaannya, dana SPFP tersebut dapat dialokasikan untuk pembiayaan uang pangkal atau uang gedung, biaya praktikum, dan beberapa biaya administrasi pendukung lain. Sistem UKT Proprosional dirancang untuk meng-*cover* biaya kuliah mahasiswa selama masa studi delapan semeter pada jenjang Strata 1 (S1) dan tidak ada pungutan biaya lain sehingga jika mahasiswa yang bersangkutan menempuh pendidikan lebih dari delapan semester maka pembayaran biaya kuliah untuk semester sembilan dan seterusnya menggunakan sistem SPP progresif yang besarnya ditentukan oleh kebijakan universitas (Prawoto, 2015).

Penentuan UKT Proporsional berdasarkan variabel penghasilan orang tua, rekening listrik, rekening telepon, rekening PAM/PDAM, rekening pajak bumi dan

bangunan (PBB) dan rekening pajak kendaraan bermotor (PKB). Variabel tersebut kemudian diakumulasi untuk menentukan mahasiswa masuk ke salah satu kategori. Jumlah kategori UKT Proporsional yang ditetapkan Universitas Brawijaya pada tahun 2014 untuk jalur SNMPTN dan SBMPTN adalah enam kategori. Selanjutnya, besar nominal untuk setiap kategori bervariasi antara rentang Rp. 500.000 sampai dengan Rp. 23.450.000 (Selma, 2015b). Variasi tersebut salah satu nya dipengaruhi oleh perhitungan unit *cost* yang berbeda pada setiap fakultas. Tidak hanya itu, biaya yang harus dikeluarkan oleh fakultas juga disesuaikan dengan rencana strategi dan target yang ingin dicapai (Puspitarini. 2013).

Program Teknologi Informasi dan Ilmu Komputer menentukan nilai UKT Proporsional antara rentang Rp. 500.000,00 sampai dengan Rp. 9.500.000 yang dibagi ke dalam enam kategori. Besar UKT Proporsional tersebut berlaku untuk program studi informatika/ilmu komputer, sistem komputer, dan sistem informasi (Selma, 2015b).

2.3 *Data mining*

Pengertian *data mining* dikutip dari beberapa sumber pustaka adalah sebagai berikut (Kusrini, et al., 2009):

- Istilah untuk menguraikan pengetahuan di dalam *database.* Proses *data mining* menggunakan teknik statistika, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi dan pengetahuan terkait dari *database* besar (Turban, 2005).
- Proses menemukan hubungan, pola, dan kecerdasan dengan memeriksa sekumpulan besar data yang tersimpan dengan memanfaatkan teknik pengenalan pola seperti teknik statistik dan matematika (Larose, 2005).
- Analisis otomatis dari data yang berjumlah besar dengan tujuan untuk menemukan pola atau kecenderungan yang keberadaannya sering tidak disadari (Pramudiono, 2006).

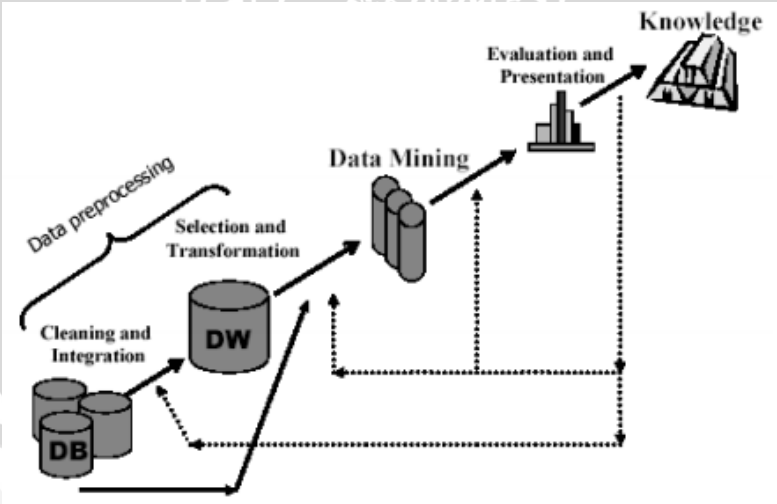
Dari definisi tesebut diatas, beberapa *point* penting terkait dengan *data mining* adalah sebagai berikut (Kusrini, et al., 2009):

- *Data mining* merupakan proses otomatis terhadap data yang sudah ada
- *Data mining* memproses data dalam jumlah besar
- Tujuan *data mining* adalah memperoleh hubungan atau pola yang cenderung memberikan indikasi bermanfaat

2.3.1 Tahapan *Data mining*

Istilah *data mining* dan *knowledge discovery in database* (KDD) sering kali digunakan bergantian yang merujuk pada proses pencarian informasi tersembunyi dalam basis data yang besar. Kedua istilah tersebut pada dasarnya memiliki konsep yang berbeda namun berkaitan satu sama lain. Pada keseluruhan proses KDD terdapat satu tahapan proses *data mining* sebagai berikut (Kusrini, et al., 2009):

1. *Data Selection*
   Pemilihan/seleksi data dari sekumpulan data operasional dilakukan sebelum melangkah pada tahap pencarian informasi. Selanjutnya, data hasil seleksi tersebut disimpan pada suatu berkas dan terpisah dari basis data operasional.
2. *Pre-processing/Cleaning*
   Proses *cleaning* data dilakukan terlebih dahulu sebelum proses *data mining.* Proses ini terdiri dari: membuang duplikasi data, memeriksa data yang konsisten, memperbaiki kesalahan pada data. Pada tahap ini juga dilakukan proses *enrichment* untuk "memperkaya" data yang sudah ada dengan data atau informasi lain yang relevan dan diperlukan oleh KDD.
3. *Transformation*
   Proses transformasi data atau disebut *coding* bertujuan agar data yang telah dipilih sesuai untuk proses *data mining.* Proses *coding* merupakan proses kreatif dan sangat bergantung pada jenis atau pola informasi yang akan dicari dalam basis data.
4. *Data mining*
   Data mining merupakan proses mencari pola atau informasi dalam data terpilih dengan menggunakan algoritma, metode atau teknik tertentu. Pemilihan algoritma, metode, atau teknik yang tepat bergantung pada tujuan dan proses KDD secara keseluruhan.
5. *Interpretation/Evaluation*
   Interpretation merupakan bagian dari proses KDD yang bertujuan agar pola informasi yang dihasilkan dari proses *data mining* dapat ditampilkan dalam bentuk yang mudah dimengerti oleh pihak berkepentingan. Proses ini meliputi pemerikasaan terkait pola atau informasi yang bertentangan dengan fakta atau hipotesis yang ada sebelumnya.



Gambar 2.1 Proses *Data Mining*
Sumber: (Junaedi, et al., 2011)

2.3.2 Pengelompokkan *Data mining*

Menurut Larose (dalam Kusrini, et al., 2009) berdasarkan tugas yang dapat dilakukan, *data mining* dapat dibagi menjadi beberapa kelompok antara lain:

1. Deskripsi
   Deskripsi tentang pola dan kecenderungan dapat memberikan kemungkinan kejelasan bagi pola dan kecenderungan itu sendiri. Deskripsi tersebut dapat berupa gambaran pola dan kecenderungan yang terdapat dalam sebuah data.
2. Estimasi
   Meskipun hampir sama dengan klasifikasi namun variabel target pada estimasi lebih mengarah ke numerik daripada kategori. Model dibangun dengan melibatkan *record* lengkap yang menyediakan nilai dari variabel target sebagai nilai prediksi. Pada peninjauan selanjutnya, estimasi nilai dari variabel target dibuat berdasarkan nilai variabel prediksi.
3. Prediksi
   Prediksi hampir sama dengan klasifikasi dan estimasi namun pada prediksi nilai dari hasil terdapat pada masa mendatang. Selain itu, beberapa metode dan teknik yang digunakan pada klasifikasi dan estimasi dapat pula digunakan pada prediksi untuk kondisi keadaan yang tepat.
4. Klasifikasi *(classification)*
   Klasifikasi memiliki target variabel kategori yang jelas, artinya kelompok atau kategori sudah didefinisikan sebelumnya. Hal ini pula yang menjadi perbedaan mendasar antara klasifikasi dan klasterisasi.
5. Klasterisasi *(clustering)*
   Klaterisasi merupakan pengelompokan *record,* pengamatan, atau memperhatikan dan membentuk kelas objek yang memiliki kemiripan. *Cluster* sendiri merupakan kumpulan *record* yang memiliki kemiripan satu sama lain dan ketidakmiripan dengan *record* pada *cluster* lain.
6. Asosiasi
   Asosiasi bertugas menemukan atribut yang muncul dalam satu waktu.

2.4 *Clustering*

Clustering atau klaterisasi merupakan salah satu teknik dalam *data mining*. Menurut Tan (dalam Irwansyah, et al., 2015) *clustering* merupakan proses mengelompokkan data ke dalam beberapa *cluster* atau kelompok dengan tujuan memaksimalkan kemiripan data dalam satu *cluster* dan sebaliknya meminimalkan kemiripan data pada *cluster* berbeda.

Dalam proses *clustering,* satu set objek data akan dipartisi ke dalam himpunan bagian yang disebut dengan *cluster.* Setiap objek dalam satu *cluster* memiliki kemiripan karakteritik antara satu sama lain sekaligus ketidakmiripan karakteristik dengan objek pada *cluster* berbeda. Proses ini melibatkan algoritma *clustering* dimana *group* atau kelompok tidak didefinisikan sebelumnya. Dalam kehidupan sehari-hari, *clustering* banyak digunakan pada berbagai aplikasi diantaranya: *business intelligence,* pengenalan pola citra, *web search,* bidang ilmu biologi dan bidang keamanan (Irwansyah, at al., 2015).

2.4.1 Syarat *Clustering*

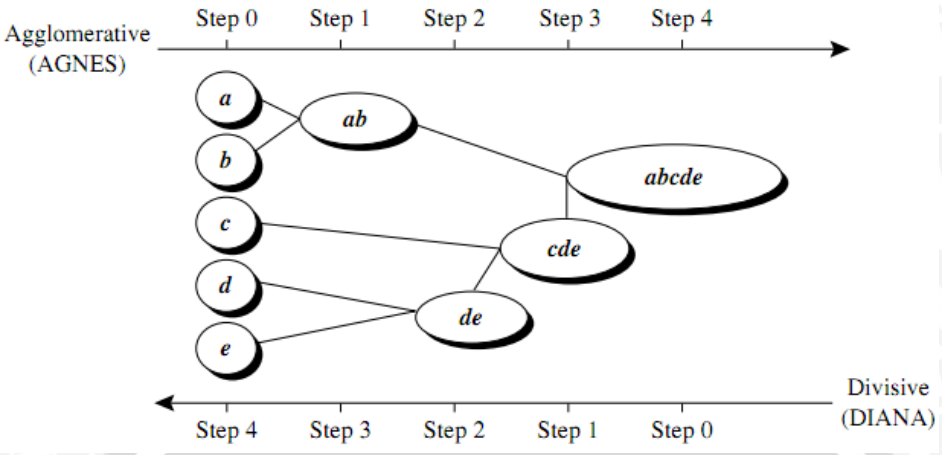
Menurut Han dan Kamber (dalam Irwansyah, at al., 2015) sebuah algoritma *clustering* harus dapat memenuhi syarat dan tantangan sebagai berikut:

1. Skalabilitas
   Algoritma *clustering* dapat menangani data dalam jumlah besar. Dewasa ini, data dalam jumlah besar sangat umum digunakan pada berbagai bidang seperti *database.* Pada struktur *database* berukuran besar*,* penanganan tidak terbatas pada ratusan objek data melainkan lebih dari jutaan objek data.
2. Kemampuan menganalisa beragam bentuk data
   Algoritma *clustering* dapat diimplementasikan pada berbagai bentuk data seperti data nominal, ordinal maupun gabungannya.
3. Kemampuan menemukan *cluster* dalam bentuk tidak terduga
   Algoritma *clustering* dapat menganalisa *cluster* dalam berbagai bentuk.
4. Kemampuan menangani *noise*
   Algoritma *clustering* dapat menangani data pada kondisi kurang baik seperti data yang rusak, tidak dimengerti maupun hilang.
5. Sensitifitas terhadap perubahan *input*
   Perubahan atau penambahan data dapat menyebabkan perubahan pada *cluster* yang telah dibentuk. Hal tersebut dapat terjadi jika dalam proses klasterisasi menggunakan algoritma *clustering* yang memiliki tingkat sensitifitas rendah.
6. Kemampuan melakukan *clustering* untuk data berdimensi tinggi
   Algoritma *clustering* dapat menangani objek data yang memiliki dimensi atau atribut dalam jumlah besar.
7. Interpretasi dan kegunaan
   Hasil daripada *clustering* dapat diinterpretasikan dan bermanfaat.

2.4.2 Metode C*lustering*

Menurut Tan (dalam Irwansyah, at al., 2015) metode *clustering* secara umum dibagi menjadi dua yaitu *hierarchical clustering* dan *partitional clustering.* Penjelasan dari masing-masing metode adalah sebagi berikut:

1. *Hierarchical Clustering*
   Pada *hierarchical clustering* objek data dikelompokkan melalui bagan berupa hirarki dimana pada setiap iterasi terdapat penggabungan dua *group* atau kelompok atau pembagian seluruh objek data ke dalam *cluster.*



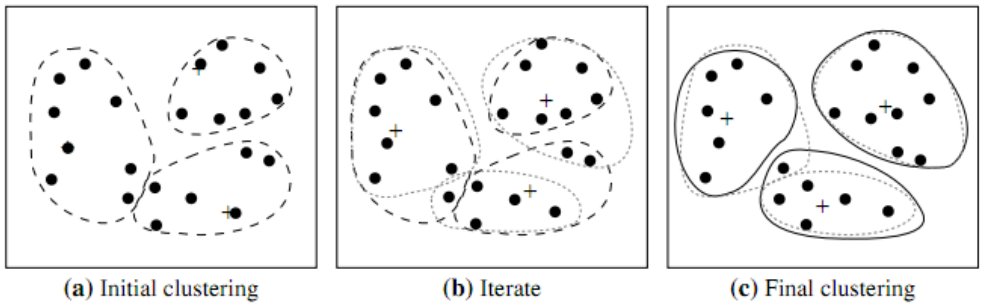
Gambar 2.2 *Hierarchical Clustering*
Sumber: (Han, et al., 2012)

Langkah-langkah *hierarchical clustering* adalah sebagai berikut:

- Identifikasi data dengan jarak terdekat
- Gambungakan data dengan jarak terdekat ke dalam satu *cluster*
- Hitung jarak antar *cluster*
- Ulangi dari awal sampai semua terhubung

Contoh metode *Hierarchical Clustering* adalah *Single Linkage*, *Complete Linkage, Average Linkage*, *Average Group Linkage.*

2. *Partitional Clustering*
   Pada *partitional clustering* data dikelompokkan ke dalam beberapa *cluster* tanpa adanya struktur hirarki antara satu dengan yang lain. Metode ini memiliki titik pusat *cluster* untuk setiap iterasi dan secara umum memiliki fungsi yang tujuan untuk meminimalkan jarak *(dissimilarity)* antara data dengan pusat *cluster* masing-masing.



Gambar 2.3 Proses *Clustering* Menggunakan Algoritma *K-Means*
Sumber: (Han, et al., 2012)

Contoh *Partitional Clustering* adalah *K-Means, Fuzzy K-Means*, dan *Mixture Modelling.*

2.5 *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) merupakan salah satu cabang *"swarm intelligence"* berdasarkan algoritma metaheuristik yang pertama kali diperkenalkan oleh Kennedy dan Eberhart pada tahun 1995. Algoritma PSO terinspirasi dari perilaku sekawanan burung dalam hal kerjasama dan komunikasi. Kecerdasan yang muncul dari perilaku tersebut menyebabkan kawanan burung meniru pola global yang kompleks (Alam, et al., 2014).

Dalam PSO, setiap individu dalam *swarm* disebut partikel, berperilaku sebagai agen di lingkungan yang sangat terdesentralisasi dan cerdas. Setiap partikel dalam *swarm* berkontribusi pada lingkungan untuk mengikuti pola sederhana yaitu bekerjasama dan berkomunikasi dengan partikel lain dalam *swarm.* Sebuah perilaku kolektif global yang kompleks muncul dalam *swarm.* Perilaku tersebut dimanfaatkan untuk menyelesaikan permasalahan optimasi yang kompleks. Desentralisasi yang tinggi, kerjasama antar partikel, dan implementasi yang sederhana membuat PSO dapat digunakan untuk menyelesaikan permasalahan optimasi secara efisien (Alam, et al., 2014).

PSO memiliki tiga komponen utama diantaranya: partikel, komponen kognitif dan komponen sosial, serta kecepatan partikel. Pada ruang permasalahan yang membutuhkan solusi optimal sementara terdapat lebih dari satu kemungkinan solusi, setiap partikel merepresentasikan solusi penyelesaian. Pembelajaran partikel terdiri dari dua faktor yaitu pengalaman partikel (disebut *cognitive learning*) dan kombinasi pembelajaran dari keseluruhan *swarm* (disebut *social learning*). *Cognitive learning* direpresentasikan sebagai *personal best (pBest)* yaitu posisi terbaik yang pernah dicapai sebuah partikel sedangkan *social learning* direpresentasikan sebagai *global best (gBest)* yaitu posisi terbaik dari keseluruhan partikel dalam *swarm. Swarm* mengarahkan partikel menggunakan parameter *gBest. Cognitive learning* dan *social learning* digunakan untuk menghitung kecepatan partikel untuk posisi selanjutnya (Alam, et al., 2014).

2.5.1 Komponen *Particle Swarm Optimization*

Komponen algoritma PSO adalah sebagai berikut (Tuegeh, et al., 2009):

a. *Swarm*
   Swarm merupakan jumlah partikel dalam populasi pada suatu algoritma. Ukuran *swarm* bergantung pada seberapa kompleks masalah yang dihadapi. Secara umum, ukuran *swarm* pada algoritma PSO cenderung lebih kecil jika dibandingkan dengan algoritma evolusioner yang lain dalam mencari solusi terbaik.
b. Partikel
   Partikel merupakan individu dalam suatu *swarm* yang merepresentasikan solusi penyelesaian masalah. Setiap partikel memiliki posisi dan kecepatan yang ditentukan oleh representasi solusi pada saat itu.
c. *Personal Best (pBest)*

Personal best merupakan posisi terbaik yang pernah dicapai partikel dengan membandingkan *fitness* pada posisi partikel sekarang dengan sebelumnya. *Personal best* dipersiapkan untuk mendapatkan solusi terbaik.

d. *Global Best (gBest)*
Global best merupakan posisi terbaik partikel yang diperoleh dengan membandingakan nilai *fitness* terbaik dari keseluruhan partikel dalam *swarm.*

e. Kecepatan *(velocity)*
Velocity merupakan vektor yang menentukan arah perpindahan posisi sebuah partikel. Perubahan *velocity* dilakukan setiap iterasi dengan tujuan memperbaiki posisi partikel semula.

f. Bobot inersia *(inertia weight)*
Parameter bobot inersia digunakan untuk mengontrol dampak dari perubahan *velocity* yang diberikan oleh partikel. Selain itu, penggunaan bobot inersia pada beberapa penelitian menunjukkan adanya peningkatkan performansi. Secara umum, bobot inersia (w) dihitung menggunakan Persamaan 2.1 sebagai berikut:

$$w = w_{max} - \frac{w_{max} - w_{min}}{Iterasi_{max}} \; x \; Iterasi \tag{2.1}$$

Keterangan:

w_{max} : Bobot inersia maksimal
w_{min} : Bobot inersia minimal
$Iterasi_{max}$: Iterasi maksimal
$Iterasi$: Iterasi sekarang

g. Koefisien Akselerasi
Koefisien akselerasi merupakan faktor pengontrol sejauh mana partikel berpindah dalam satu iterasi. Secara umum nilai koefisien akselerasi c_1 dan c_2 adalah sama yaitu dalam rentang 0 sampai 4. Namun demikian, nilai tersebut dapat ditentukan sendiri untuk setiap penelitian berbeda.

2.6 *K-Means*

Algoritma *clustering K-Means* dikembangkan oleh MacQueen pada tahun 1976. Algoritma ini merupakan algoritma *clustering unsupervised* yang menghasilkan sejumlah *disjoint* dan *cluster* (non-hirarki). Prosedur *K-Means* cenderung sederhana dan mudah dalam mengelompokkan sebuah *dataset* ke dalam sejumlah *cluster* (Santhanam, et al., 2015).

Algoritma *K-Means* memilih secara acak sejumlah *k* objek data sebagai representasi dari *k* pusat *cluster.* Langkah selanjutnya adalah menempatkan dan mengasosiasikan setiap data dengan pusat *cluster* terdekat dihitung berdasarkan kedekatan setiap data dengan pusat *cluster* menggunakan teori pengukuran jarak seperti *Euclidean.* Setelah semua data didistribusikan, pusat *cluster* dihitung kembali. Proses tersebut diulang sampai tidak ada perubahan pada *k* pusat *cluster.*

Langkah-langkah algoritma *K-Means* secara keseluruhan dijelaskan sebagai berikut (Santhanam, et al., 2015).

1. Memilih *k* data secara random sebagai *k* pusat *cluster*
2. Untuk setiap data pada *dataset*, lakukan langkah berikut:
   a. Menghitung jarak setiap data dengan setiap *cluster*
   b. Kelompokkan setiap data ke dalam *cluster* yang memiliki jarak terdekat.
3. Mengulangi langkah 2 sampai tidak terdapat lagi perubahan pusat *cluster* dari satu *cluster* ke *cluster* lainnya. Pada kondisi ini, *cluster* sudah stabil dan proses *clustering* selesai
4. Pemilihan partisi awal sangat mempengaruhi hasil akhir sebuah *cluster,* meliputi: jarak *inter cluster,* jarak *intra cluster,* dan kohesi.

2.7 *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM)

Hybrid Particle Swarm Optimization dan *K-Means* merupakan algoritma hasil penggabungan antara *Particle Swarm Optimization* dan *K-Means*. Algoritma ini dilakukan secara sekuensial artinya algoritma PSO dijalankan terlebih dahulu sampai pada kondisi berhenti kemudian dilanjutkan dengan algoritma *K-Means.*

K-Means merupakan algoritma *clustering* yang sering digunakan karena mudah diimplementasikan dan efisien dalam hal waktu eksekusi. Namun, algoritma ini memiliki kelemahan yaitu sensitif terhadap pemilihan pusat *cluster* awal sehingga solusi yang dihasilkan *K-Means* cenderung berada pada daerah optimum lokal (Rana, et al., 2010).

Pada penelitian ini, PSO digunakan untuk menginisialisasi pusat *cluster* awal *K-Means.* Hal tersebut bertujuan bahwa pusat *cluster* awal yang digunakan pada algoritma *K-Means* nantinya merupakan pusat *cluster* yang paling mendekati optimum. Sehingga, jika pemilihan pusat *cluster* awal tepat maka algoritma *K-Means* dapat bekerja dengan baik dalam menyempurnakan proses *clustering* data (Rana et al., 2010).

2.7.1 Penerapan *Particle Swarm Optimization*

PSO digunakan untuk mengoptimasi pusat *cluster* awal algoritma *K-Means.* Optimasi bertujuan untuk mendapatkan pusat *cluster* yang paling optimum berdasarkan nilai *cost* setiap partikel. Langkah-langkah algoritma PSO untuk *clustering* data yang diterapkan pada penelitian ini adalah sebagai berikut (Cui, et al., 2005):

1. Inisialisasi partikel.
   Panjang partikel adalah *d x k* (*d* = jumlah dimensi dan *k* = jumlah *cluster*) sehingga data dengan 6 dimensi jika dikelompokkan ke dalam 2 *cluster* memiliki panjang 12 sel, dengan 6 sel pertama merepresentasikan pusat *cluster* pertama dan 6 sel kedua merepresentasikan pusat *cluster* kedua. Pusat-pusat *cluster* pada penelitian ini diperolah secara acak dari koleksi data UKT Proprosional yang telah dinormalisasi.
2. Untuk setiap partikel, lakukan langkah berikut:

a. Mengelompokkan setiap objek data ke dalam *cluster* yang memiliki jarak terdekat menggunakan teori *Euclidean* yang dirumuskan pada Persamaan 2.2 berikut:

$$D(i,j) = \sqrt{(x_{1j} - y_{1j})^2 + (x_{2i} - y_{2j})^2 + \cdots + (x_{ki} - y_{kj})^2} \quad (2.2)$$

Keterangan:

$D(i,j)$	: Jarak data ke-*i* terhadap pusat *cluster j*
x_{ki}	: Data i pada atribut data ke-*k*
y_{kj}	: Titik pusat ke-*j* pada atribut data ke-*k*

b. Menghitung nilai *cost* menggunakan Persamaan 2.3 berikut:

$$f = \frac{\sum_{j=1}^{N_c} \{ \frac{\sum_{j=1}^{p_i} d(o_i, m_{ij})}{p_i} \}}{N_c} \quad (2.3)$$

Keterangan:

C_i	: *Cluster* ke-*i*
N_c	: Jumlah *cluster*
p_i	: Jumlah *data* pada *cluster* C_i
o_i	: Pusat *cluster* C_i
m_{ij}	: Data ke-*j* dan merupakan anggota dari *cluster i*
$d(o_i, m_{ij})$	: Jarak antara o_i dan m_{ij}

c. *Update* kecepatan dan posisi partikel menggunakan Persamaan 2.4 dan Persamaan 2.5 berikut:

$$v_{id} = w * v_{id} + c_1 * rand_1(p_{id} - x_{id}) + c_2 * rand_2(p_{gd} - x_{id}) \quad (2.4)$$

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

Keterangan:

v_{id}	: Kecepatan partikel ke-*i* dimensi ke-*d*
w	: Bobot inersia
c_1 *dan* c_2	: Koefisien akselerasi 1 dan koefisien akselerasi 2
$rand_1$ *dan* $rand_2$	: Nilai acak 1 dan nilai acak 2 (interval 0 - 1)
p_{id}	: *Personal best* partikel ke-*i* dimensi ke-*d*
p_{gd}	: *Global best* partikel ke-*g* dimensi ke-*d*
x_{id}	: Posisi partikel ke-*i* dimensi ke-*d*

d. *Update personal best* dengan ketentuan sebagai berikut:

$$p_i(t+1) = \begin{cases} p_i(t) & f(x_i(t+1)) \leq f(x_i(t)) \\ x_i(t+1) & f(x_i(t+1)) > f(x_i(t)) \end{cases}$$

Keterangan:

$p_i(t+1)$	: *Personal best* pada iterasi $t+1$
$x_i(t+1)$	: *Partikel* pada iterasi $t+1$
$f(x_i(t+1))$	: Nilai *cost* partikel pada iterasi $t+1$

e. *Update global best.* Proses *updat global best* adalah mencari nilai *personal best* yang memiliki nilai *cost* terbaik

3. Ulangi langkah ke 2 sampai kriteria kondisi berhenti terpenuhi (maksimal iterasi).

2.7.2 Penerapan *K-Means*

Penerapan *K-Means* digunakan untuk memperbaiki hasil *clustering* lebih lanjut. Inisialisasi pusat *cluster* awal algoritma *K-Means* diperoleh dari pusat *cluster* paling optimum yang telah dioptimasi terlebih dahulu menggunakan algoritma PSO. Langkah-langkah algoritma *K-Means* yang diterapkan pada penelitian ini adalah sebagai berikut (Cui, et al., 2005):

1. Inisialisasi pusat *cluster (centroid)* awal.
2. Mengalokasi setiap data ke dalam *cluster* yang memiliki jarak terdekat. Kedekatan objek data dengan pusat *cluster* dihitung menggunakan teori *Euclidean* yang dirumuskan pada Persamaan (2.2).
3. Menghitung pusat *cluster* baru dengan keanggotaan *cluster* sekarang. Perhitungan pusat *cluster* baru dihitung berdasarkan nilai jarak setiap objek data dengan pusat *cluster* yang dirumuskan pada Persamaan (2.6) berikut:

$$c_j = \frac{1}{n_j} \sum_{\forall d_j \in S_j} d_j \tag{2.6}$$

Keterangan:

S_j : *Cluster*
c_j : Pusat *cluster* baru
n_j : Jumlah data pada *cluster* S_j
d_j : Nilai jarak data pada *cluster* S_j

4. Ulangi langkah ke 2 (pengelompokkan data berdasarkan pusat *cluster* baru) dan langkah ke 3 sampai tingkat konvergensi terpenuhi.

2.8 *Min-Max Normalization*

Normalisasi atau *normalization* merupakan proses transformasi untuk menskalakan atribut data numerik menjadi lebih kecil yaitu antara rentang 1.0 sampai dengan 0.0 atau 0.0 sampai dengan 1.0. Terdapat beberapa metode atau teknik yang dapat diterapkan untuk menormalisasi data, diantaranya: *Min-max Normalization, Z-Score Normalization,* dan *Normalization by Decimal Scalling*. Dalam penelitian ini, penulis menggunakan *Min-max Normalization* sebagai metode normalisasi data. *Min-max Normalization* akan memetakan sebuah nilai v dari atribut A menjadi v' ke dalam rentang $[new\_min_A, new\_max_A]$. Rumus *Min-max Normalization* dijelaskan pada Persamaan (2.7) adalah sebagai berikut (Junaedi, et al., 2011):

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{2.7}$$

Keterangan:

v' : Nilai data yang sudah dinormalisasi

v : Nilai data yang belum dinormalisasi
max_A : Nilai minimum data dari atribut ke- A
min_A : Nilai minimum data dari atribut ke- A
new_max_A : Nilai maksimum data baru dari atribut ke- A (diasumsikan 1)
new_min_A : Nilai minimum data baru dari atribut ke- A (diasumsikan 0)

2.9 *Silhouette Coefficient*

Silhouette Coefficient merupakan salah satu metode intrinsik yang digunakan untuk mengukur kualitas *clustering.* Secara umum, metode intrinsik mengevaluasi hasil *clustering* dengan memeriksa seberapa baik sebuah objek data terpisah dengan objek data lain pada *cluster* berbeda dan seberapa kompak sebuah objek dengan objek lain bergabung dalam satu *cluster* yang sama. Perhitungan pada metode intrinsik memanfaatkan kelebihan dari matriks kesamaan antar objek data dalam *dataset* (Han, et al., 2012).

Langkah-langkah mengukur kualitas sebuah *cluster* menggunakan metode *Silhouette Coefficient* adalah sebagai berikut (Han, et al., 2012):

a. Hitung jarak rata-rata data ke - *o* dengan semua data yang berada dalam satu *cluster.*

$$a(o) = \frac{\sum o' \in C_i, o \neq o' dist(o, o')}{|C_i| - 1} \quad (2.8)$$

Keterangan:

o : Data ke-*o* pada *cluster i*
o' : Data lain pada *cluster i* selain data ke-*o*
C_i : *Cluster i*
$|C_i|$: Jumlah data pada *cluster i*
$dist(o, o')$: Jarak data *o* dengan data o'

b. Hitung jarak rata-rata jarak data ke - *o* dengan semua data yang berada pada *cluster* lain kemudian diambil nilai paling minimum.

$$b(o) = \min_{C_j: 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum o' \in C_j \, dist(o, o')}{|C_j|} \right\} \quad (2.9)$$

c. Hitung nilai S*ilhouette Coeficient.*

$$s(o) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}} \quad (2.10)$$

Nilai *Silhouette Coefficient* berada pada interval -1 sampai dengan 1. Nilai $a(o)$ merepresentasikan "compactness" setiap objek data dalam satu *cluster* yang sama sehingga semakin kecil nilai $a(o)$ maka semakin baik. Sedangkan nilai $b(o)$ merepresentasikan "separation" antar objek data dalam *cluster* berbeda sehingga semakin besar nilai $b(o)$ maka semakin baik. Oleh karenanya, ketika nilai *Silhouette Coefficient* mendekati 1, maka pengelompokkan data ke dalam suatu *cluster* semakin baik atau tepat sebaliknya jika nilai *Silhouette Coefficient* mendekati -1, maka pengelompokkan data semakin buruk karena jarak objek data

terhadap semua data pada *cluster* berbeda lebih dekat dibandingkan jarak objek data terhadap semua data pada satu *cluster* yang sama (Han, et al., 2012).

Untuk mengetahui nilai *fitness* setiap *cluster* dapat dilakukan dengan menghitung rata-rata nilai *Silhouette Coefficient* semua objek data dalam satu *cluster*. Sedangkan untuk mengetahui kualitas dari hasil *clustering* dapat dilakukan dengan menghitung rata-rata nilai *Silhouette Coefficient* semua objek data pada *dataset*. Metode *Silhouette Coefficient* maupun metode intrinsik lainnya dapat pula digunakan untuk menentukan jumlah *cluster* optimal yaitu dengan mengganti variasi jumlah *cluster* (Han, et al., 2012).

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan penelitian, teknik pengumpulan data, algoritma yang digunakan meliputi pemilihan bahasa pemrograman dan batasan implementasi, spesifikasi kebutuhan sistem meliputi kebutuhan perangkat keras dan kebutuhan perangkat lunak, serta rancangan pengujian algoritma.

3.1 Tahapan Penelitian

Skripsi dengan judul **"*Hybrid Particle Swarm Optimization* Dan *K-Means* Untuk *Clustering* Data Penentuan UKT Proporsional"** ini dapat digolongkan sebagai penelitian implementatif dengan pendekatan perancangan *(design).* Penelitian ini akan menghasilkan sebuah purwarupa *(prototype)* berupa perangkat lunak *(software)* yang dapat digunakan untuk *clustering* data penentuan UKT Proporsional. Tahapan penelitian yang utuh adalah sebagai berikut:

1. Analisis
   Tahap ini menjelaskan tentang formulasi permasalahan meliputi deskripsi dan batasan masalah, kebutuhan data sebagai objek penelitian, serta contoh solusi penyelesaian dari pemasalahan yang dihadapi.
2. Perancangan
   Tahap ini menjelaskan tentang siklus penyelesaian *clustering* data UKT Proporsional menggunakan algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* serta (HPSOKM) perhitungan manual berdasarkan contoh kasus yang akan diselesaikan.
3. Pengujian
   Tahap ini menjelaskan tentang skenario pengujian sistem diantaranya: pengujian parameter algoritma PSO yang menghasilkan solusi terbaik, pengujian jumlah *cluster* untuk mengetahui jumlah *cluster* optimal, serta pengujian perbandingan algoritma untuk mengetahui peningkatan kualitas solusi yang dihasilkan sistem dengan hanya menggunakan algoritma *K-Means* dan sistem yang menerapkan algoritma PSO terlebih dahulu untuk mengoptimasi pusat *cluster* awal pada algoritma *K-means.*

3.2 Teknik Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan data penentuan UKT Proporsional tahun 2014 dari jalur SNMPTN dan SBMPTN Jurusan Teknik Informatika PTIIK UB yang terdiri atas 6 atribut meliputi: penghasilan orang tua (gaji), rekening listrik, rekening telepon, rekening PAM atau PDAM, rekening pajak bumi dan bangunan (PBB), rekening pajak kendaraan bermotor (PKB) dengan total keseluruhan 347 data (SNMPTN = 180 data dan SBMPTN = 167 data). Data tersebut diperoleh dari sub bagian keuangan rektorat Lt.5 Universitas Brawijaya setelah sebelumnya dilakukan wawancara terhadap pakar terkait variabel yang digunakan sebagai pertimbangan dalam menetapkan kategori UKT Proporsional.

3.3 Algoritma yang Digunakan

Proses *clustering* data UKT Proporsional menggunakan algoritma *Hybrid* PSO dan *K-Means,* selanjutnya kualitas hasil *clustering* dievaluasi menggunakan metode *Silhouette Coefficient*. Implementasi algoritma menggunakan bahasa pemrograman C#. Bahasa pemrograman C# dipilih karena fleksibel, *powerful*, efisien, memiliki *memory management* yang baik, serta menyediakan berbagai *tools* untuk mengatur tampilan secara *customizable*. Selain itu, fokus algoritma PSO terletak pada kompleksitas komputasi sehingga diharapkan solusi yang dihasilkan adalah yang terbaik dengan tetap memperhatikan waktu komputasi yang efisien. Batasan yang digunakan selama proses implementasi dan pengujian algoritma adalah sebagai berikut:

1. Perangkat lunak yang dibangun menggunakan pendekatan *desktop base application*
2. *Input* sistem berupa data UKT Proporsional dalam format .xls
3. Atribut data untuk proses *clustering* adalah dinamis sedangkan untuk evaluasi kualitas hasil *clustering* adalah sejumlah 6 atribut
4. Dalam satu kali proses *clustering* hanya untuk satu *sheet* data
5. *Output* sistem berupa data UKT Proprosional yang telah dikelompokkan, ditampilkan dalam bentuk tabel *datagridview* dan format *file* .pdf

3.4 Kebutuhan Sistem

Kebutuhan sistem merupakan tahapan untuk mengidentifikasi spesifikasi perangkat keras dan perangkat lunak yang diperlukan agar sistem yang dibangun dapat berjalan dengan baik selama proses implementasi dan pengujian. Spesifikasi perangkat keras meliputi: Processor AMD Athlon(tm) II Neo K345 Dual-Core Processor 1.40 GHz, RAM 4.00 GB, Hardisk kapasitas 297 GB sedangkan spesifikasi perangkas lunak meliputi: Sistem operasi Windows 7 Home Premium 64-bit, Microsoft Office Excel 2013, XAMPP, Database MySQL, Microsoft Visual C# 2010 Express.

3.5 Pengujian Algoritma

Pengujian dan pembahasan pada penelitian ini terdiri dari dua bagian yaitu pengujian terkait parameter algoritma *Particle Swarm Optimization* (PSO) yang digunakan untuk mengoptimasi pusat *cluster* dan pengujian terkait hasil *clustering* data. Skenario pengujian yang dilakukan antara lain sebagai berikut:

a. Pengujian interval kecepatan partikel
b. Pengujian pengaruh *random injection*
c. Pengujian parameter PSO
d. Pengujian jumlah *cluster*
e. Pengujian perbandingan algoritma

3.5.1 Pengujian Interval Kecepatan Partikel

Pengujian interval kecepatan partikel bertujuan untuk mengetahui interval kecepatan partikel yang sesuai sehingga dapat menghasilkan solusi penyelesaian yang optimum. Besar interval kecepatan yang diuji coba pada penelitian ini adalah 50% sampai dengan 0,005% dari interval posisi. Parameter yang digunakan pada pengujian interval kecepatan partikel adalah sebagai berikut:

a. Bobot inersia max = 0,9
b. Bobot inersia min = 0,4
c. Koefisien akselerasi 1 = 2
d. Koefisien akselerasi 2 = 2
e. Ukuran *swarm* = 10
f. Jumlah iterasi = 1000
g. Jumlah *cluster* = 3

Tabel 3.1 Rancangan Pengujian Interval Kecepatan Partikel

Interval Kecepatan		Uji coba ke-	Nilai *Cost* Terbaik	Evaluasi 1	Evaluasi 2
w_{max}	w_{min}				
-5.10^{-1}	5.10^{-1}				
-5.10^{-2}	5.10^{-2}				
-5.10^{-3}	5.10^{-3}				
-5.10^{-4}	5.10^{-4}				
-5.10^{-5}	5.10^{-5}				

3.5.2 Pengujian Pengaruh *Random Injection*

Random injection (RI) merupakan salah satu metode untuk penanganan konvergensi dini pada algoritma optimasi salah satunya adalah *Particle Swarm Optimization* (PSO). Konvergensi dini terjadi jika dalam satu populasi *(swarm)* terdapat beberapa individu (partikel) yang bernilai sama sebelum mencapai titik optimum yang diinginkan. Metode ini diawali dengan memilih *n* partikel dalam *swarm* selanjutnya sejumlah partikel tersebut dibangkitkan secara acak seperti halnya pada proses inisialisasi partikel (Mahmudy, et al., 2014b).

Pada penelitian ini, *random injection* dilakukan per 10 iterasi dengan jumlah partikel yang diganti adalah 20% dari ukuran *swarm. Random injection* diuji coba pada keseluruhan interval kecepatan partikel dengan tujuan mengetahui pengaruhnya terhadap kualitas solusi yang dihasilkan oleh setiap interval kecepatan partikel. Parameter yang digunakan pada pengujian pengaruh *random injection* adalah sebagai berikut:

a. Bobot inersia max = 0,9
b. Bobot inersia min = 0,4
c. Koefisien akselerasi 1 = 2
d. Koefisien akselerasi 2 = 2
e. Ukuran *swarm* = 50
f. Jumlah iterasi = 1000
g. Jumlah *cluster* = 3

Tabel 3.2 Rancangan Pengujian Pengaruh *Random Injection*

Iterasi ke-	Nilai *Cost* Terbaik		Rata-rata Nilai *Cost*	
	Dengan RI	Tanpa RI	Dengan RI	Tanpa RI
1 s.d 1000				
Rata-rata				

3.5.3 Pengujian Parameter PSO

Pengujian ini dilakukan untuk mengetahui pengaruh parameter PSO terhadap hasil *clustering* data. Karena tidak terdapat metode yang pasti untuk menentukan parameter PSO, maka evaluasi dilakukan dengan melakukan iji coba:

1. Uji coba untuk mencari kombinasi bobot inersia yang optimal
2. Uji coba untuk mencari kombinasi koefisien akselerasi yang optimal
3. Uji coba untuk menentukan ukuran *swarm* yang optimal
4. Uji coba untuk menentukan jumlah iterasi yang optimal

3.5.3.1 Pengujian Bobot Inersia

Uji coba bobot inersia dilakukan untuk mengetahui kombinasi bobot inersia maksimal dan bobot inersia minimal yang tepat guna memperoleh pusat-pusat *cluster* yang paling optimal. Sebelum melakukan uji coba, terlebih dahulu dilakukan plot nilai *cost* untuk mengetahui pada iterasi berapa partikel mulai konvergen dan hasil yang diperoleh adalah 800 iterasi. Bobot inersia maksimal dan minimal diuji coba pada rentang 0 sampai dengan 1 yang ditunjukkan pada Tabel 3.3. Parameter yang digunakan pada uji coba bobot inersia adalah sebagai berikut:

a. Bobot inersia max = 0 - 1
b. Bobot inersia min = 0 - 1
c. Koefisien akselerasi 1 = 2
d. Koefisien akselerasi 2 = 2
e. Ukuran *swarm* = 10
f. Jumlah iterasi = 800
g. Jumlah *cluster* = 3

Tabel 3.3 Rancangan Pengujian Bobot Inersia

Bobot inersia		Rata-rata Nilai *Cost* Terbaik
w_{max}	w_{min}	
0,9	0,2	
	0,3	
	0,4	
0,8	0,2	
	0,3	
	0,4	
0,7	0,2	
	0,3	
	0,4	

3.5.3.2 Pengujian Koefisien Akselerasi

Pengujian koefisien dilakukan untuk mengetahui kombinasi akselerasi 1 dan koefisien akselerasi 2 terbaik guna memperoleh pusat-pusat *cluster* yang paling optimal. Sebelum melakukan uji coba, terlebih dahulu dilakukan plot nilai *cost* untuk mengetahui pada iterasi berapa partikel mulai konvergen dan hasil yang diperoleh adalah 600 iterasi. Koefisien akselerasi diuji coba pada rentang 1 sampai dengan 2 yang ditunjukkan pada Tabel 3.4. Parameter yang digunakan pada uji coba koefisien akselerasi adalah sebagai berikut:

a. Bobot inersia max = w_{max} terbaik pada pengujian bobot inersia
b. Bobot inersia min = w_{min} terbaik pada pengujian bobot inersia
c. Koefisien akselerasi 1 = 1 – 2
d. Koefisien akselerasi 2 = 1 – 2
e. Ukuran *swarm* = 10
f. Jumlah iterasi = 600
g. Jumlah *cluster* = 3

Tabel 3.4 Rancangan Pengujian Koefisien Akselerasi

Koefisien Akselerasi		Rata-rata Nilai *Cost* Terbaik
c_1	c_2	
1	1	
	1,5	
	2	
1,5	1	
	1,5	
	2	
2	1	
	1,5	
	2	

3.5.3.3 Pengujian Jumlah Iterasi

Pengujian jumlah iterasi dilakukan untuk mengetahui jumlah iterasi yang tepat guna memperoleh pusat-pusat *cluster* yang paling optimal. Jumlah iterasi yang diuji coba adalah kelipatan 100 yang ditunjukkan pada Tabel 3.5. Parameter yang digunakan pada uji coba jumlah iterasi adalah sebagai berikut:

a. Bobot inersia max = w_{max} terbaik pada pengujian bobot inersia
b. Bobot inersia min = w_{min}terbaik pada pengujian bobot inersia
c. Koefisien akselerasi 1 = c_1 terbaik pada pengujian koefisien akselerasi
d. Koefisien akselerasi 2 = $c_2$terbaik pada pengujian koefisien akselerasi
e. Ukuran *swarm* = 10
f. Jumlah iterasi = 100 – 1000
g. Jumlah *cluster* = 3

Tabel 3.5 Rancangan Pengujian Jumlah Iterasi

Uji coba ke	Nilai *Cost* Terbaik				
	Iterasi = 100	Iterasi = ..	Iterasi = ..	Iterasi = ..	Iterasi = 1000
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
Rata-rata					

3.5.3.4 Pengujian Ukuran *Swarm*

Pengujian ukuran *swarm* dilakukan untuk mengetahui jumlah partikel yang dibutuhkan guna memperoleh pusat-pusat *cluster* yang paling optimal. Algoritma PSO membutuhkan jumlah populasi lebih sedikit dibandingkan dengan algoritma evolusioner lain yaitu antara 20 sampai dengan 50 populasi (Teugeh, 2007). Ukuran *swarm* yang diuji coba adalah kelipatan 10 yang ditunjukkan pada Tabel 3.6. Parameter yang digunakan pada uji coba ukuran *swarm* adalah sebagai berikut:

a. Bobot inersia max = w_{max} terbaik pada pengujian bobot inersia
b. Bobot inersia min = w_{min} terbaik pada pengujian bobot inersia
c. Koefisien akselerasi 1 = c_1 terbaik pada pengujian koefisien akselerasi
d. Koefisien akselerasi 2 = c_2 terbaik pada pengujian koefisien akselerasi
e. Ukuran *swarm* = 10 – 100
a. Jumlah iterasi = jumlah terbaik pada pengujian jumlah iterasi
f. Jumlah *cluster* = 3

Tabel 3.6 Rancangan Pengujian Ukuran S*warm*

Uji coba ke	Nilai *Cost* Terbaik				
	Partikel = 10	Partikel = ..	Partikel = ..	Partikel = ..	Partikel = 100
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
Rata-rata					

3.5.4 Pengujian Jumlah *Cluster*

Pengujian jumlah *cluster* dilakukan untuk mengetahui jumlah *cluster* dengan tingkat validasi data tertinggi pada permasalahan *clustering* data UKT Proporsional. Nilai kualitas sebuah *cluster* dievaluasi menggunakan metode analisis *cluster Silhouette Coefficient*. Jumlah *cluster* yang diuji coba adalah sebanyak 2, 3, 4, 5, dan 6 *cluster* yang ditunjukkan pada Tabel 3.7. Parameter yang digunakan pada uji coba jumlah *cluster* adalah sebagai berikut:

b. Bobot inersia max = w_{max} terbaik pada pengujian bobot inersia
c. Bobot inersia min = w_{min} terbaik pada pengujian bobot inersia
d. Koefisien akselerasi 1 = c_1 terbaik pada pengujian koefisien akselerasi
e. Koefisien akselerasi 2 = c_2 terbaik pada pengujian koefisien akselerasi
f. Ukuran *swarm* = jumlah terbaik pada pengujian ukuran *swarm*
g. Jumlah iterasi = jumlah terbaik pada pengujian jumlah iterasi
h. Jumlah *cluster* = 2 – 6

Tabel 3.7 Rancangan Pengujian Jumlah *Cluster*

Jumlah *Cluster*	Uji coba ke	Max Iterasi	Nilai *Silhouette Coefficient*	
			Interval -1 s.d 1	Dalam (%)
	1			
	2			
	3			
	4			
	5			
	6			
	7			
	8			
	9			
	10			
	Rata-rata			

3.5.5 Pengujian Perbandingan Algoritma

Pengujian perbandingan algoritma dilakukan untuk mengetahui kinerja algoritma HPSOKM dan *K-Means*. Selain itu, pengujian perbandingan algoritma juga bertujuan sebagai pembuktian apakah algoritma HPSOKM memiliki kinerja lebih baik dibandingkan algoritma *K-Means* dalam menyelesaikan permasalahan *clustering* data. Kualitas hasil *clustering* dievaluasi menggunakan metode analisis *cluster Silhouette Coefficient*. Jumlah *cluster* yang di uji coba adalah sebanyak *cluster* terbaik pada pengujian jumlah *cluster.* Parameter yang digunakan pada uji coba perbandingan algoritma adalah sebagai berikut:

a. Bobot inersia max = w_{max} terbaik pada pengujian bobot inersia
b. Bobot inersia min = w_{min}terbaik pada pengujian bobot inersia
c. Koefisien akselerasi 1 = c_1 terbaik pada pengujian koefisien akselerasi
d. Koefisien akselerasi 2 = $c_2$terbaik pada pengujian koefisien akselerasi
e. Ukuran *swarm* = jumlah terbaik pada pengujian ukuran *swarm*

f. Jumlah iterasi = jumlah terbaik pada pengujian jumlah iterasi
g. Jumlah *cluster* = jumlah *cluster* terbaik pada pengujian jumlah *cluster*

Tabel 3.8 Rancangan Pengujian Perbandingan Algoritma

	Waktu Komputasi	Interval -1 s.d 1	Dalam %
K-Means			
HPSOKM			

BAB 4 PERANCANGAN

Bab ini menjelaskan formulasi permasalahan, siklus algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM), siklus penyelesaian *clustering* data penentuan UKT Proporsional menggunakan algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM), perhitungan manual, serta perancangan antar muka.

4.1 Formulasi Permasalahan

Permasalahan yang diselesaikan adalah *clustering* data dengan jumlah *cluster* ditentukan oleh masukan pengguna sistem. Hasil pengelompokkan kemudian dievaluasi menggunakan metode intrinsik untuk mengetahui kualitas hasil *clustering.* Nilai kualitas *clustering* tersebut mengindikasikan berapa nilai ketepatan pengelompokkan jika data dikelompokkan ke dalam *k cluster.*

Solusi penyelesaian dari hasil optimasi adalah pusat *cluster* optimum, diperoleh berdasarkan partikel dengan nilai *cost* terbaik. Nilai *cost* pada penelitian ini bertujuan untuk meminimalkan jarak data dengan pusat *cluster* sehingga semakin kecil nilai *cost* maka peluang partikel ditetapkan sebagai solusi penyelesaian semakin besar. Partikel sebagai representasi penyelesaian memiliki panjang *d x k* (*d* = jumlah dimensi dan *k* = jumlah *cluster*).

Masukan sistem antara lain data UKT Proporsional dalam format .xls dan parameter perhitungan meliputi: jumlah *cluster,* jumlah iterasi, ukuran *swarm*, bobot maksimal (w_{max}), bobot inersia minimal (w_{min}), koefisien akselerasi 1 (c_1), dan koefisien akselerasi 2 (c_2). Selanjutnya masuk pada proses *clustering* yang terdiri dari 2 tahap. Tahap pertama adalah proses optimasi pusat *cluster* menggunakan algoritma PSO. Hasil daripada optimasi tersebut merupakan pusat-pusat *cluster* optimum. Tahap kedua adalah proses *clustering* menggunakan algoritma *K-Means.* Inisialisasi pusat *cluster* awal pada algoritma *K-Means* diperoleh dari hasil optimasi pusat *cluster* menggunakan algoritma PSO. Selanjutnya masuk pada proses evaluasi kualitas *cluster* menggunakan metode *Silhouette Coefficient.* Hasil evaluasi kualitas *cluster* tersebut berupa nilai *fitness* setiap *cluster* dan nilai kualitas *clustering.* Keluaran sistem antara lain jumlah *cluster,* pusat-pusat *cluster,* hasil pengelompokkan data UKT Proporsional dan nilai *Silhouette Coefficient*.

Adapun sampel data UKT Proporsional yang digunakan pada penelitian ini berjumlah 10 *record* sebagai berikut: (data lengkap disertakan pada Lampiran 2)

Tabel 4.1 Data UKT Proporsional

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air
1	1874000	0	152900	0	0	0
2	5000000	14124	2202000	393793	0	6616
3	10000000	50000	1999500	498019	50000	0
4	1330000	0	123500	27000	0	0
5	1200000	0	174000	80000	0	0

6	13500000	45644	392500	595216	745995	52090
7	5000000	57574	218800	400000	172000	0
8	2949800	125000	0	134989	0	65784
9	10000000	404500	1758500	406000	503315	220000
10	4900000	0	288900	0	0	0

Atribut data terpilih telah dikonsultasikan dengan pakar terkait variabel-variabel yang digunakan sebagai pertimbangan dalam menentukan kategori UKT Proporsional. Keterangan setiap atribut data pada Tabel 4.1 adalah sebagai berikut:

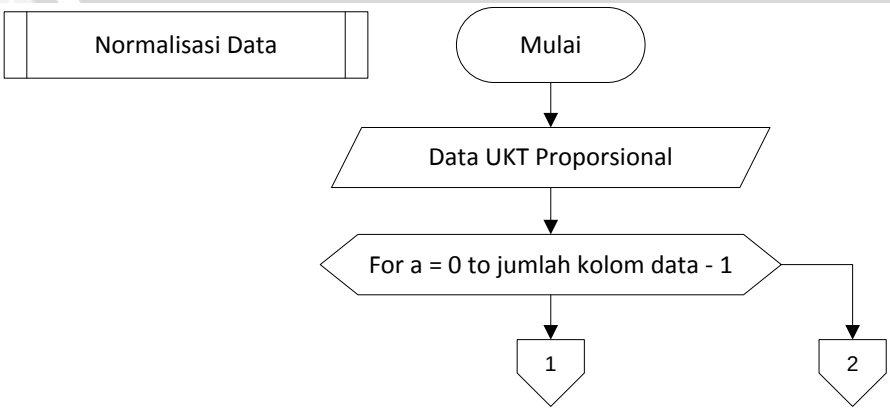
1. Gaji : menerangkan penghasilan orang tua/wali mahasiswa (integer)
2. PBB : menerangkan rekening pajak bumi dan bangunan (integer)
3. PKB : menerangkan rekening pajak kendaraan bermontor (integer)
4. Listrik : menerangkan rata-rata rekening listrik bulanan (integer)
5. Telepon : menerangkan rata-rata rekening telepon bulanan (integer)
6. Air : menerangkan rata-rata rekening PAM/PDAM bulanan (integer)

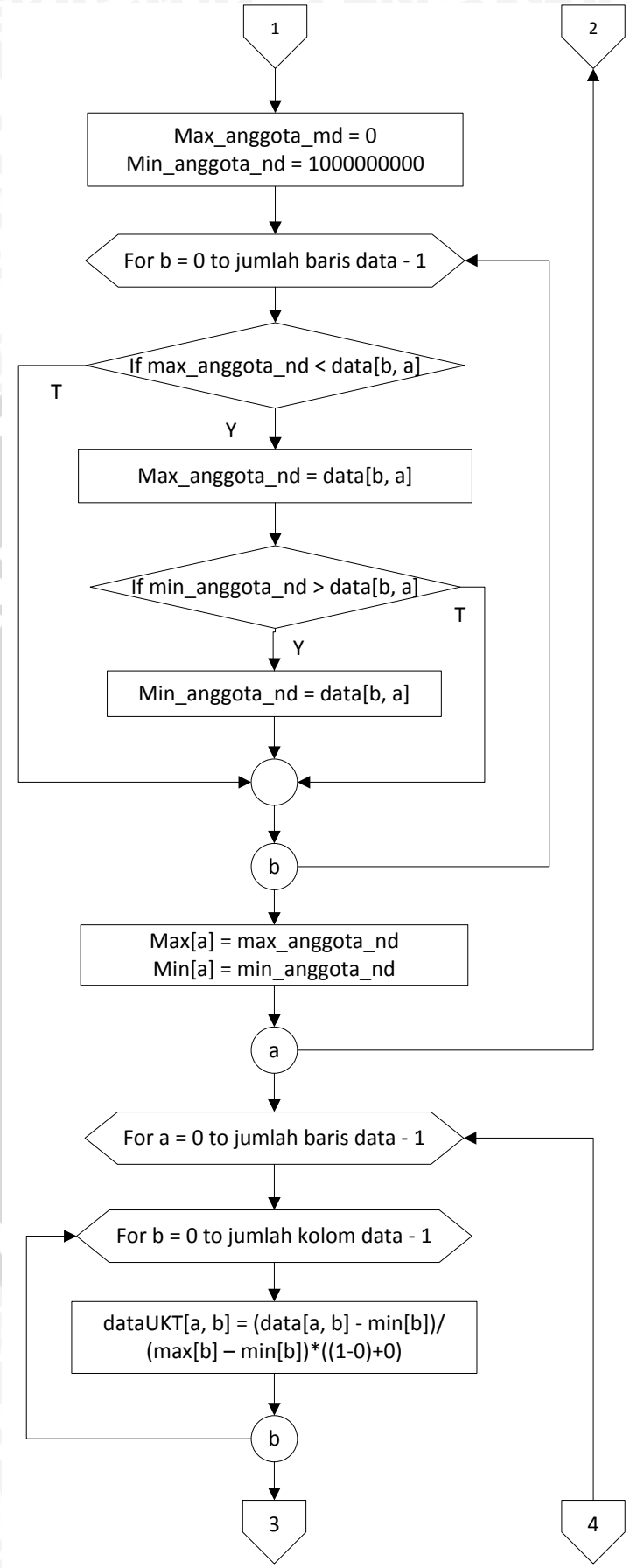
4.2 Siklus Algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* (HSPOKM)

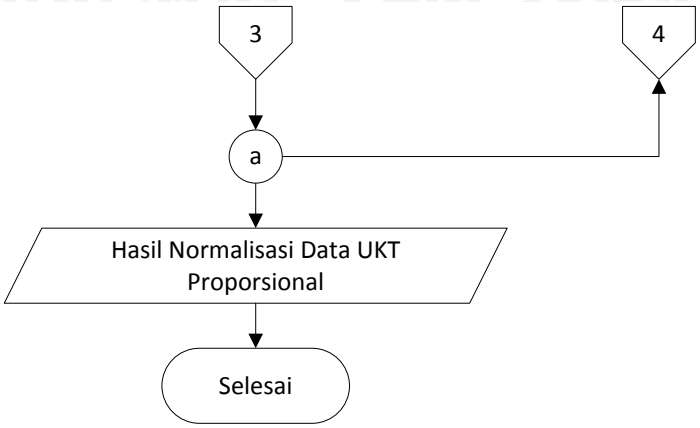
Siklus algoritma HPSOKM merupakan urutan penyelesaian masalah menggunakan algoritma *Particle Swarm Optimization* dan *K-Means* secara sekuensial. Pada siklus algoritma yang akan dijelaskan pada sub bab berikutnya, selain proses optimasi pusat *cluster* menggunakan algoritma PSO dan proses *clustering* menggunakan algoritma *K-Means,* telah ditambahkan proses normalisasi data menggunakan metode *Min-Max Normalization* dan proses analisis kualitas *cluster* menggunakan metode *Silhouette Coefficient*

4.2.1 Proses Normalisasi Data

Normalisasi data merupakan proses transformasi data dimana nilai atribut data diskalakan menjadi lebih kecil antara rentang 0 sampai dengan 1. Pada penelitian ini, proses normalisasi data menggunakan metode *Min-Max Normalization.* Diagram alir proses normalisasi data ditunjukkan pada Gambar 4.1 berikut sebagai:




1
2
Max_anggota_md = 0
Min_anggota_nd = 1000000000
For b = 0 to jumlah baris data - 1
If max_anggota_nd < data[b, a]
T
Y
Max_anggota_nd = data[b, a]
If min_anggota_nd > data[b, a]
T
Y
Min_anggota_nd = data[b, a]
b
Max[a] = max_anggota_nd
Min[a] = min_anggota_nd
a
For a = 0 to jumlah baris data - 1
For b = 0 to jumlah kolom data - 1
dataUKT[a, b] = (data[a, b] - min[b])/
(max[b] – min[b])*((1-0)+0)
b
3
4



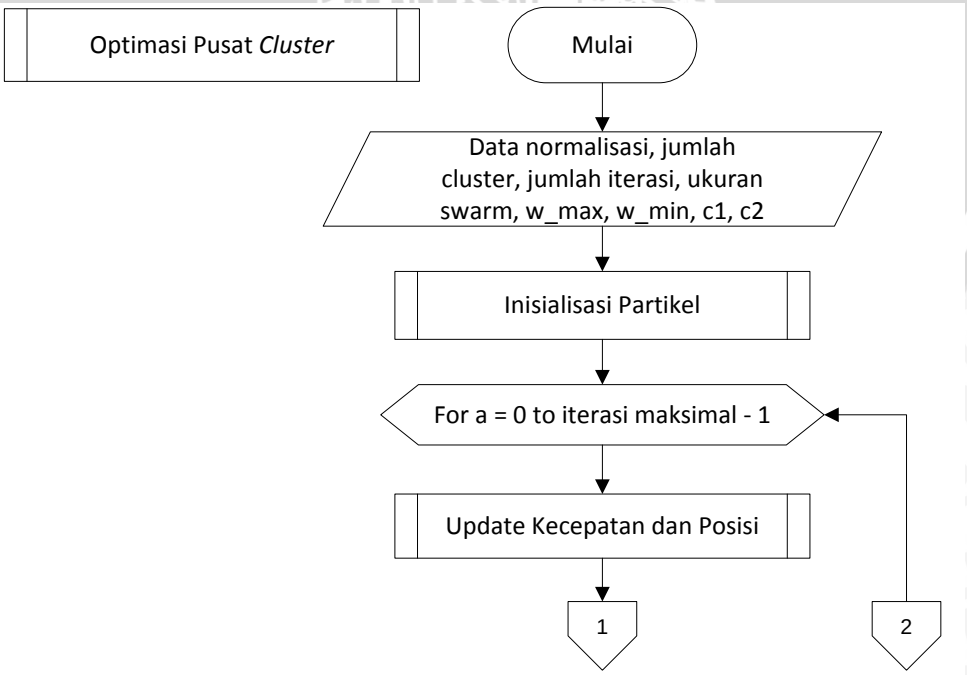
Gambar 4.1 Diagram Alir Proses Normalisasi Data

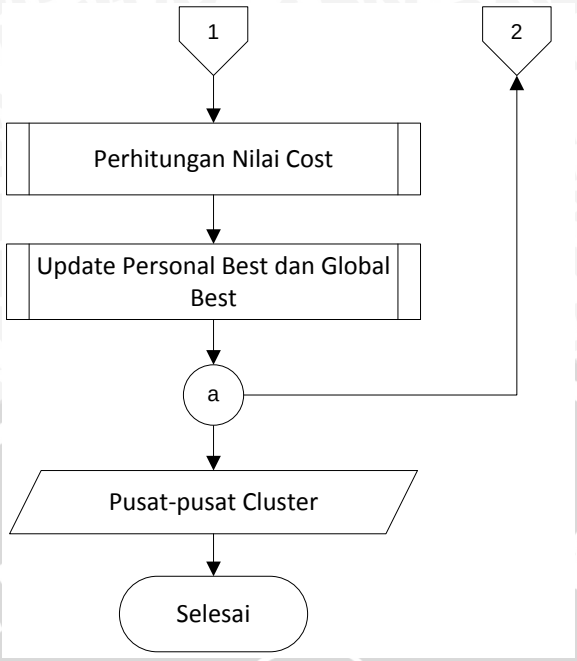
Langkah-langkah normalisasi data menggunakan metode *Min-Max Normalization* berdasarkan Gambar 4.1 adalah sebagai berikut:

1. Sistem menerima masukan berupa data UKT Proporsional
2. Mencari nilai maksimal (max[a]) dan minimal (min[a]) untuk setiap atribut data
3. Menghitung nilai normalisasi menggunakan Persamaan 2.7 untuk setiap data
4. Keluaran sistem adalah data UKT Proporsional yang telah ternormalisasi

4.2.2 Proses Optimasi Pusat *Cluster* Menggunakan Algoritma PSO

Optimasi pusat *cluster* awal menggunakan algortima PSO bertujuan untuk memperoleh pusat *cluster* optimum yang digunakan sebagai inisialisasi pusat *cluster* awal pada algoritma *K-Means.* Pemilihan pusat *cluster* optimum berdasarkan nilai *cost* yang dihasilkan oleh partikel sebagai representasi solusi. Diagram alir proses optimasi pusat *cluster* menggunakan algoritma PSO ditunjukkan pada Gambar 4.2 sebagai berikut:





Gambar 4.2 Diagram Alir Proses Optimasi Pusat *Cluster* Menggunakan PSO

Langkah-langkah optimasi pusat *cluster* menggunakan algoritma PSO Berdasarkan Gambar 4.2 adalah sebagai berikut:

1. Sistem menerima masukan berupa data UKT Proporsional yang telah ternormalisasi dan parameter PSO meliputi: jumlah *cluster,* jumlah iterasi, ukuran *swarm,* bobot inersia maksimal (w_{max}), bobot inersia minimal (w_{min}), koefisien akselerasi 1 (c_1), dan koefisien akselerasi 2 (c_2).
2. Untuk iterasi 1 sampai iterasi maksimal, lakukan langkah 4 sampai langkah 6
3. Melakukan proses inisialisasi partikel
4. Melakukan *update* kecepatan dan posisi partikel
5. Melakukan perhitungan nilai *cost*
6. Melakukan *update personal best* dan *global best*
7. Keluaran sistem adalah *global best* dengan nilai *cost* terbaik yaitu pusat-pusat *cluster* paling optimum

4.2.2.1 Inisialisasi Partikel

Inisialisasi partikel pada penelitian ini menggunakan pengkodean *real code*. Partikel memiliki panjang *d x k* (*d* = jumlah dimensi dan *k* = jumlah *cluster*) sehingga sebagai contoh data dengan 6 dimensi yang akan dikelompkkan ke dalam 2 *cluster* memiliki panjang 12 sel dengan 6 sel pertama merepresentasikan pusat *cluster* pertama dan 6 sel kedua merepresentasikan pusat *cluster* kedua. Skema inisialisasi partikel yang ditunjukkan pada Gambar 4.3 sebagai berikut:

$d_{i,j}$	$d_{i,j+1}$	$d_{i,j+2}$	$d_{i,j+3}$	$d_{i,j+4}$	$d_{i,j+5}$	$d_{i+1,j}$	$d_{i+1,j+1}$	$d_{i+1,j+2}$	$d_{i+1,j+3}$
$d_{i+1,j+4}$	$d_{i+1,j+5}$								

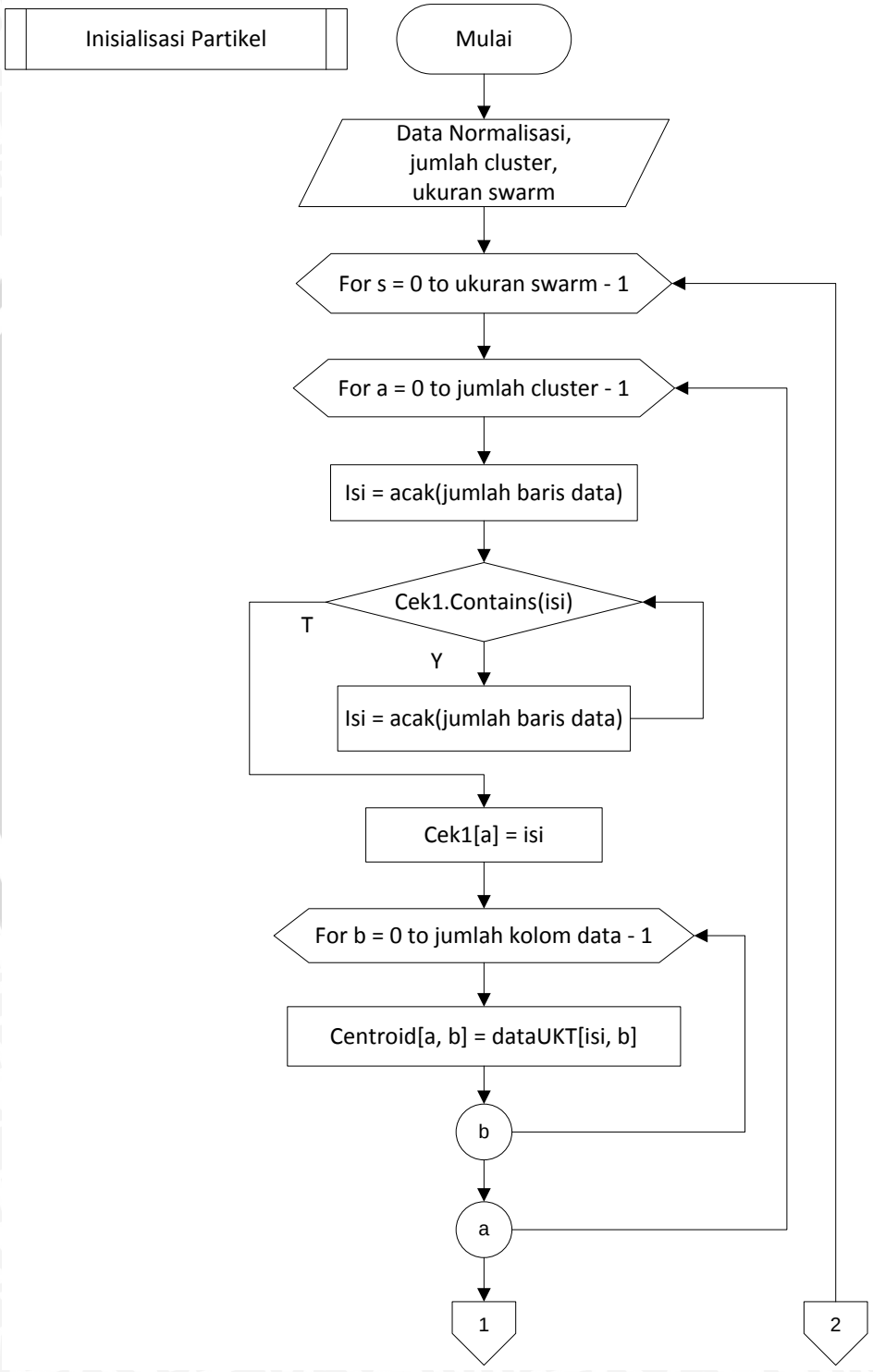
Gambar 4.3 Skema Inisialisasi Partikel

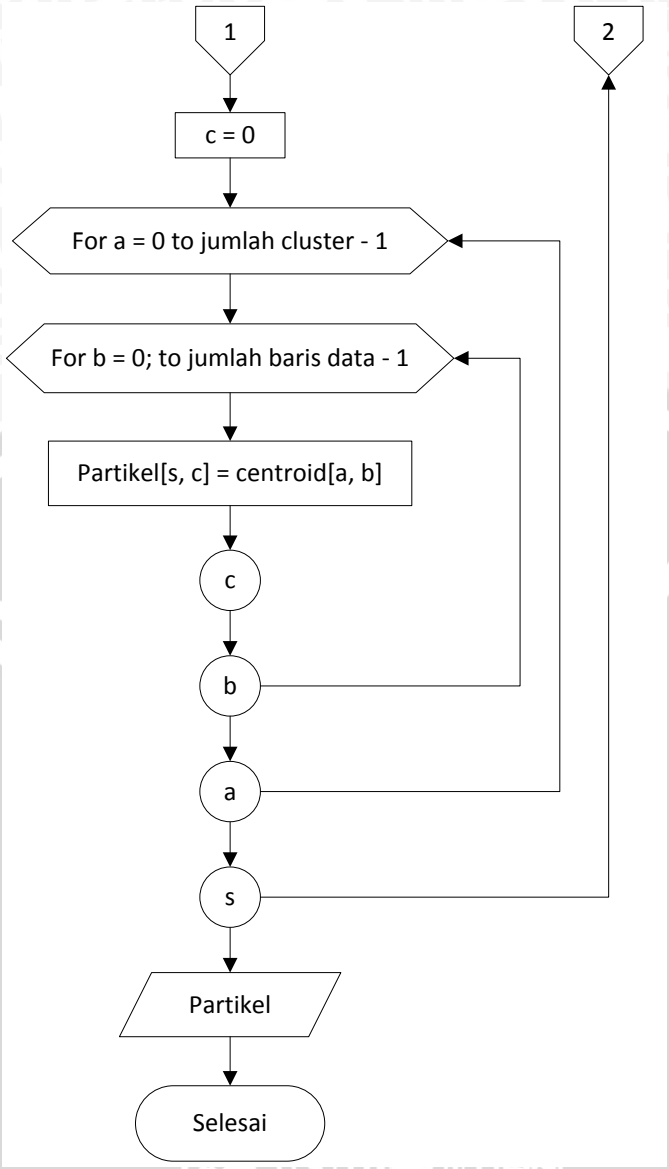
Keterangan:

i : Data ke – *i*

j : Atribut data ke – *j*

Diagram alir proses inisialisasi partikel PSO ditunjukkan pada Gambar 4.4 sebagai berikut:





Gambar 4.4 Diagram Alir Proses Inisialisasi Partikel

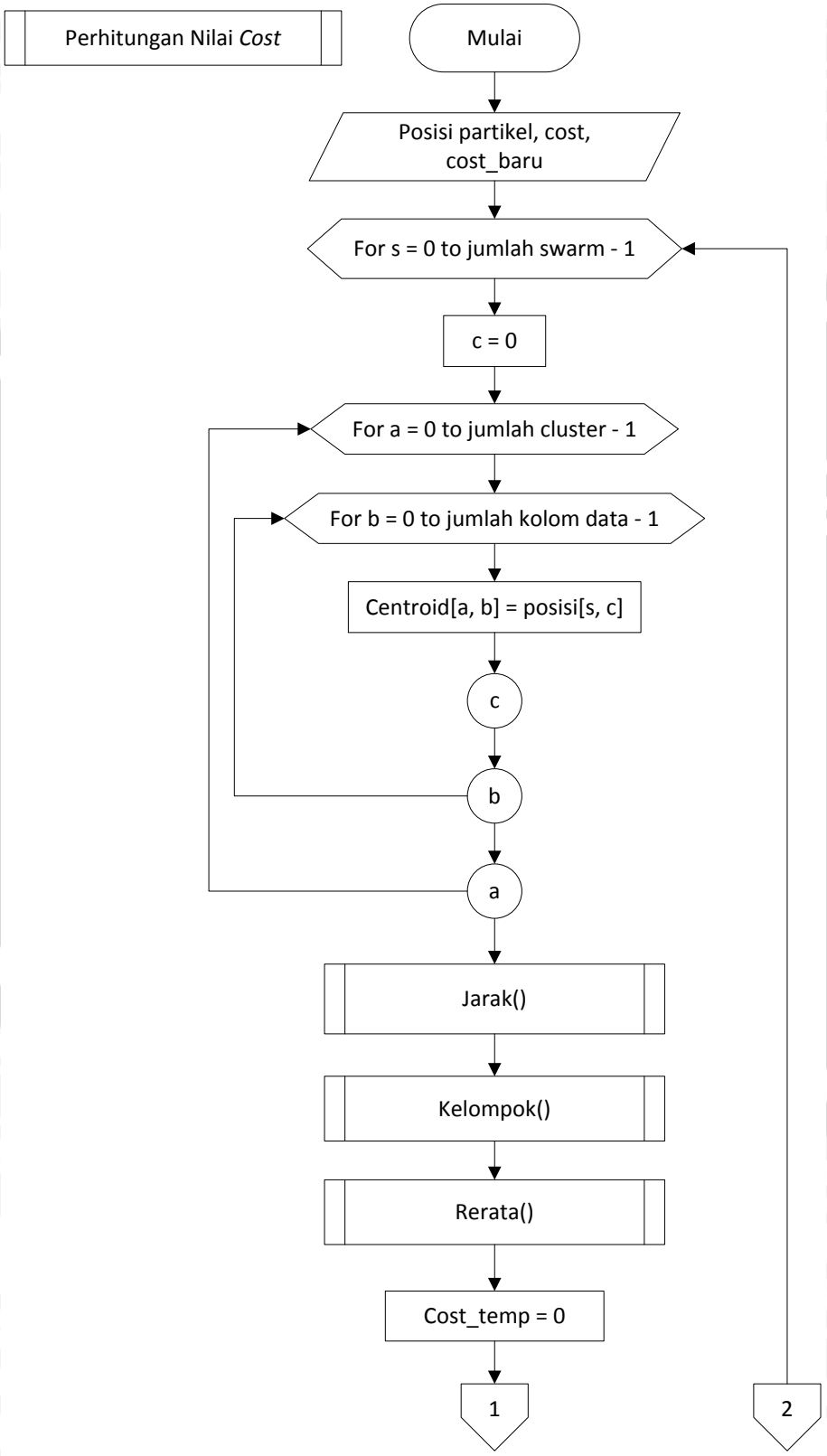
Langkah-langkah inisialisasi partikel pada algoritma PSO berdasarkan Gambar 4.4 adalah sebagai berikut:

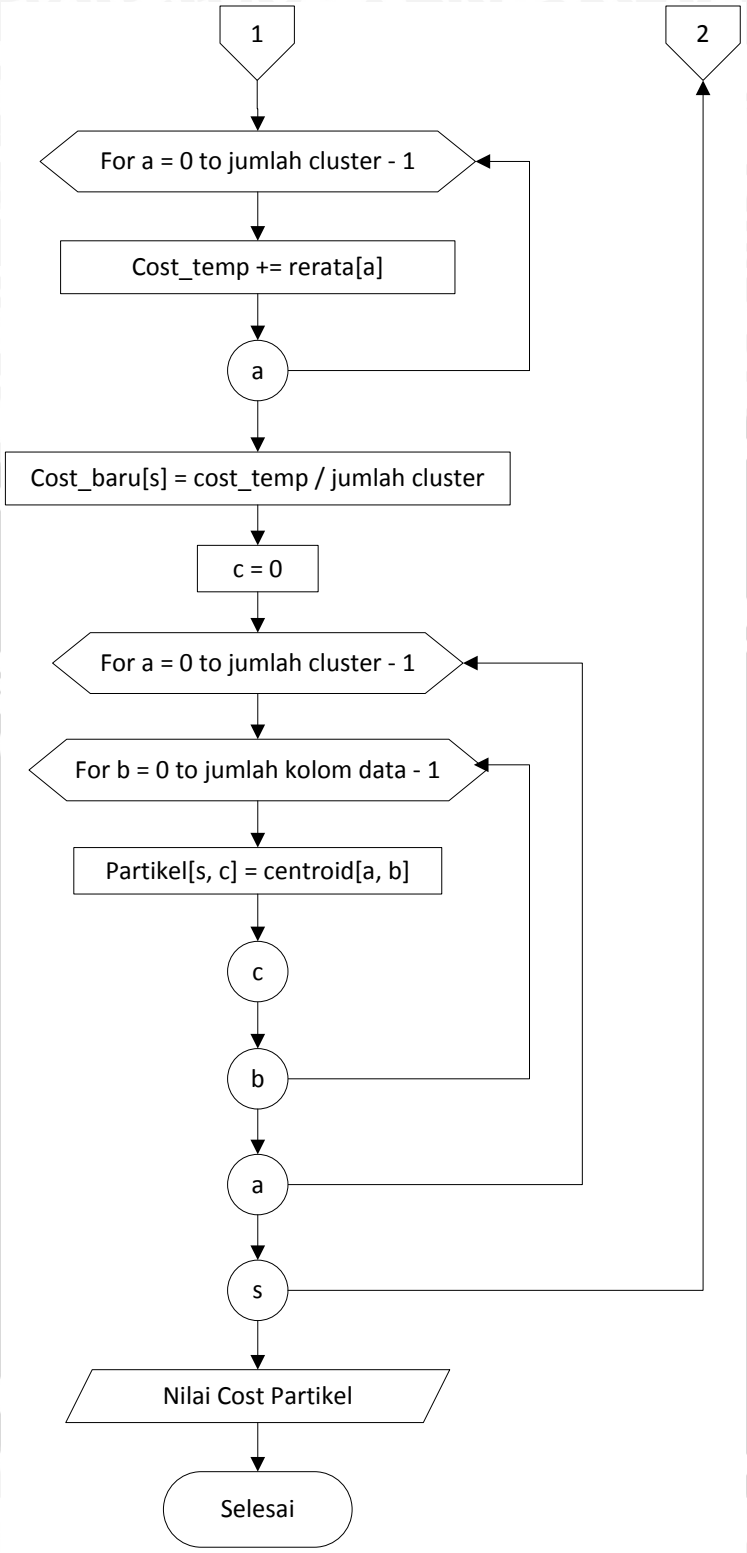
1. Sistem menerima masukan berupa data UKT Proporsional yang telah ternormalisasi, jumlah *cluster,* dan ukuran *swarm*
2. Untuk partikel 1 sampai jumlah *swarm,* lakukan langkah 3 sampai langkah 5
3. Untuk *cluster* 1 sampai jumlah *cluster,* lakukan langkah 4 sampai langkah 5
4. Inisialisasi pusat *cluster* ke-*i* secara acak diperoleh dari data UKT Proporsional yang telah ternormalisasi
5. Melakukan penggabungan setiap pusat *cluster* menjadi satu partikel
6. Keluaran sistem adalah partikel PSO sebanyak jumlah *swarm*

4.2.2.2 Perhitungan Nilai *Cost*

Setiap partikel akan menghasilkan nilai *cost* untuk menetukan seberapa baik partikel tersebut keluar sebagai solusi akhir penyelesaian masalah. Pada penelitian

ini, nilai *cost* bertujuan untuk meminimalkan jarak antara data dengan pusat *cluster* sehingga semakin kecil nilai *cost* maka semakin dekat pula jarak antara data dengan pusat *cluster.* Diagram alir perhitungan nilai *cost* ditunjukkan pada Gambar 4.5 sebagai berikut:





Gambar 4.5 Diagram Alir Proses Perhitungan Nilai *Cost*

Langkah-langkah menghitung nilai *cost* setiap partikel PSO berdasarkan Gambar 4.5 adalah sebagai berikut:

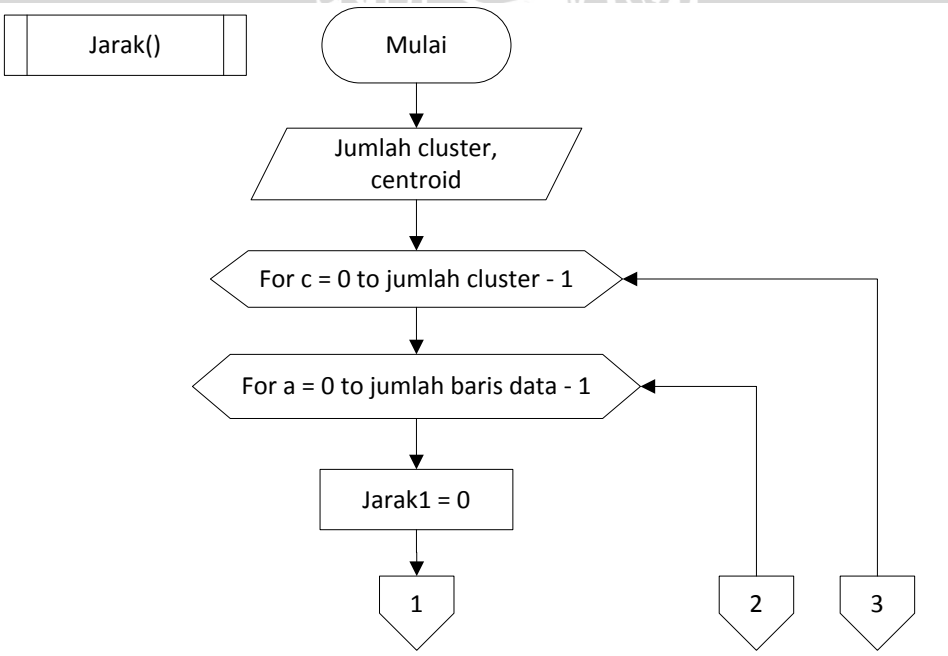
1. Sistem menerima masukan berupa nilai posisi partikel, nilai *cost,* dan nilai *cost* baru. Nilai *cost* baru merupakan nilai *cost* partikel pada iterasi ke-*x* sedangkan nilai *cost* baru merupakan nilai *cost* partikel pada iterasi ke-*x+1*
2. Untuk partikel 1 sampai jumlah *swarm*, lakukan langkah 3 sampai langkah 9
3. Melakukan konversi atau pemecahan dari posisi partikel menjadi pusat *cluster*
4. Untuk *cluster* 1 sampai jumlah *cluster*, lakukan langkah 4 sampai langkah 7
5. Hitung jarak setiap data dengan setiap pusat *cluster*
6. Kelompokkan setiap data ke dalam *cluster* yang memiliki jarak terdekat
7. Hitung rerata setiap *cluster*
8. Hitung nilai *cost* setiap partikel
9. Melakukan penggabungan dari setiap pusat *cluster* menjadi satu partikel
10. Keluaran sistem adalah nilai *cost* setiap partikel

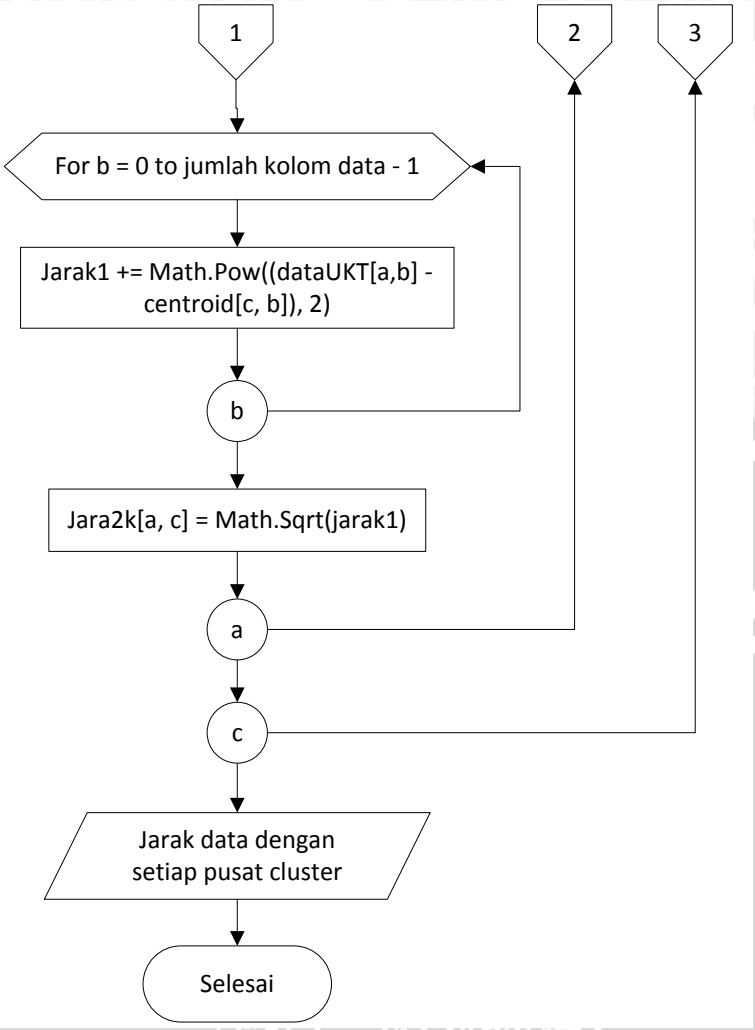
Proses perhitungan nilai *cost* untuk setiap partikel memerlukan penjelasan lebih rinci terkait beberapa sub fungsi sebagai berikut:

1. Jarak yaitu menghitung jarak setiap data terhadap masing-masing pusat *cluster.* Perhitungan jarak pada penelitian ini menggunakan teori *Euclidean*
2. Kelompok yaitu mengelompokkan setiap data ke dalam *cluster* yang memiliki jarak terdekat sekaligus menghitung anggota setiap *cluster*
3. Rerata yaitu menghitung nilai rata-rata jarak setiap *cluster*

Selanjutnya setelah menghitung jarak, mengelompokkan data, dan menghitung rerata, nilai *cost* setiap partikel diperoleh dengan melakukan rata-rata dari nilai rerata setiap *cluster.* Partikel terbaik merupakan partikel yang memiliki nilai *cost* terkecil karena dengan demikian jarak data dengan pusat *cluster* juga semakin kecil.

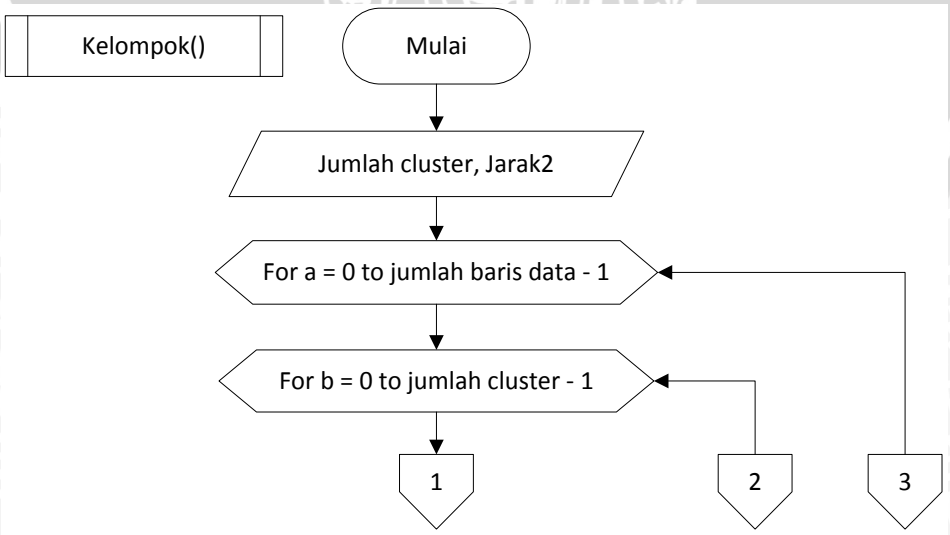
Diagram alir sub fungsi jarak ditunjukkan pada Gambar 4.6 sebagai berikut:

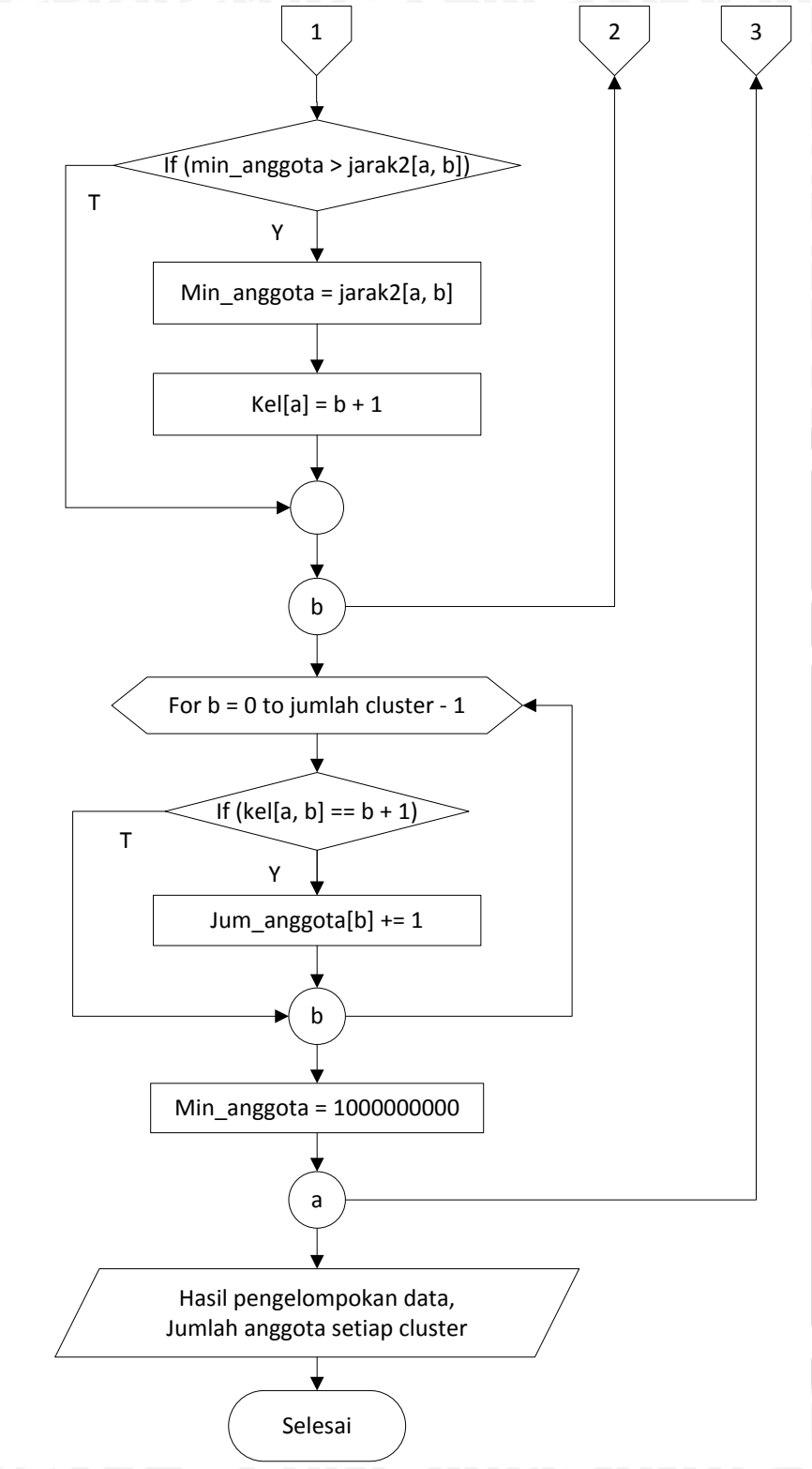




Gambar 4.6 Diagram Alir Perhitungan Jarak Data Dengan Pusat *Cluster*

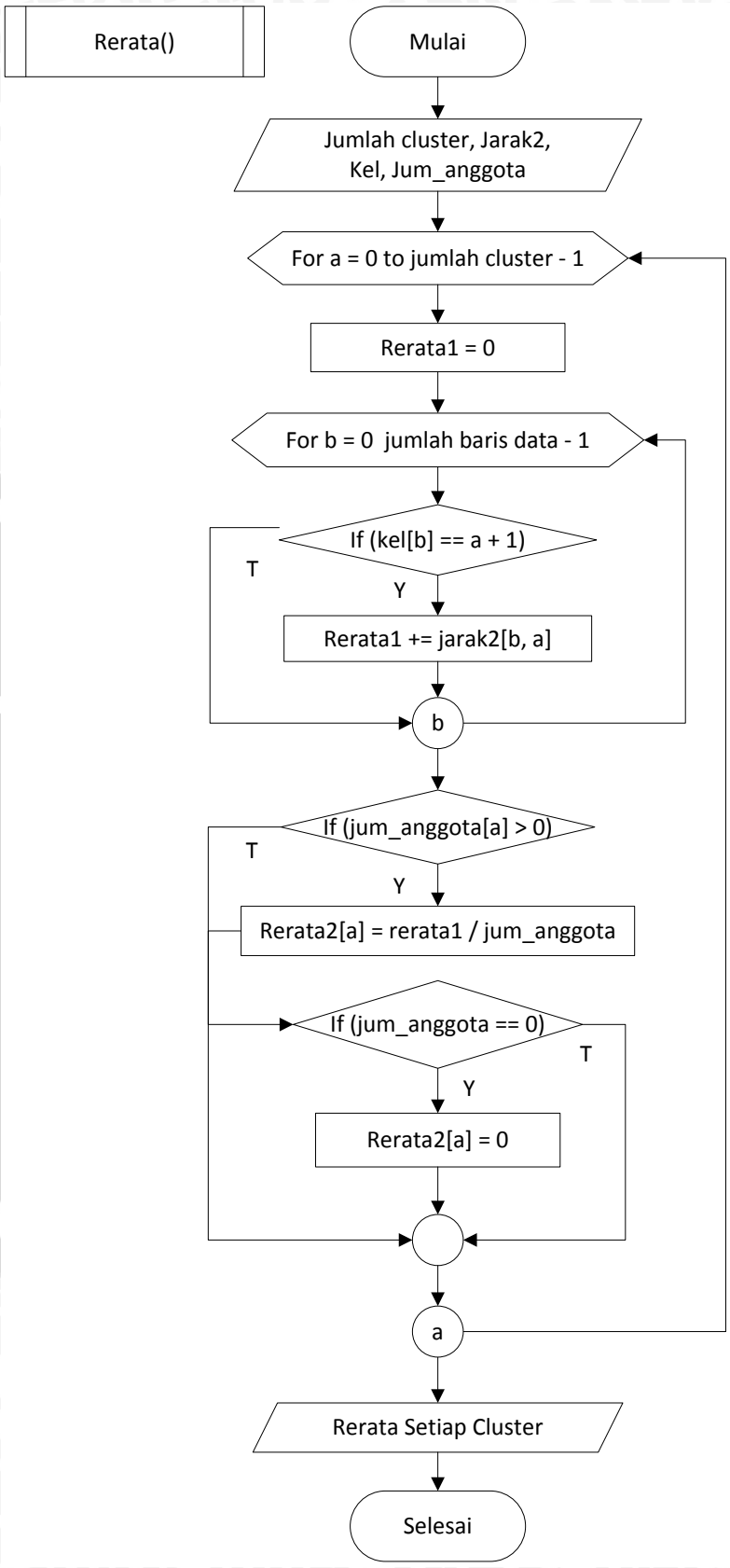
Diagram alir sub fungsi kelompok ditunjukkan pada Gambar 4.7 sebagai berikut:





Gambar 4.7 Diagram Alir Pengelompokkan Data

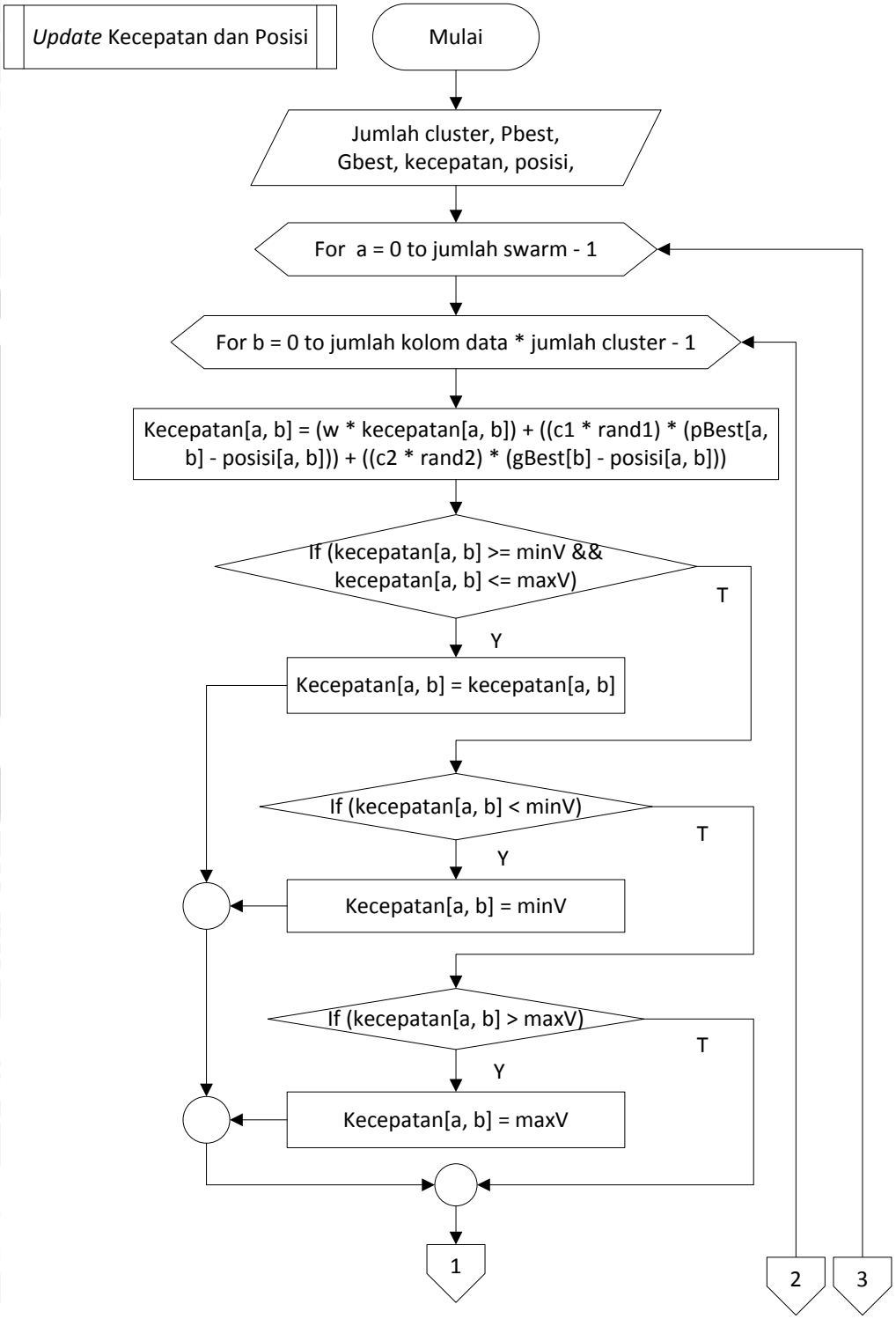
Diagram alir sub fungsi rerata ditunjukkan pada Gambar 4.8 sebagai berikut:

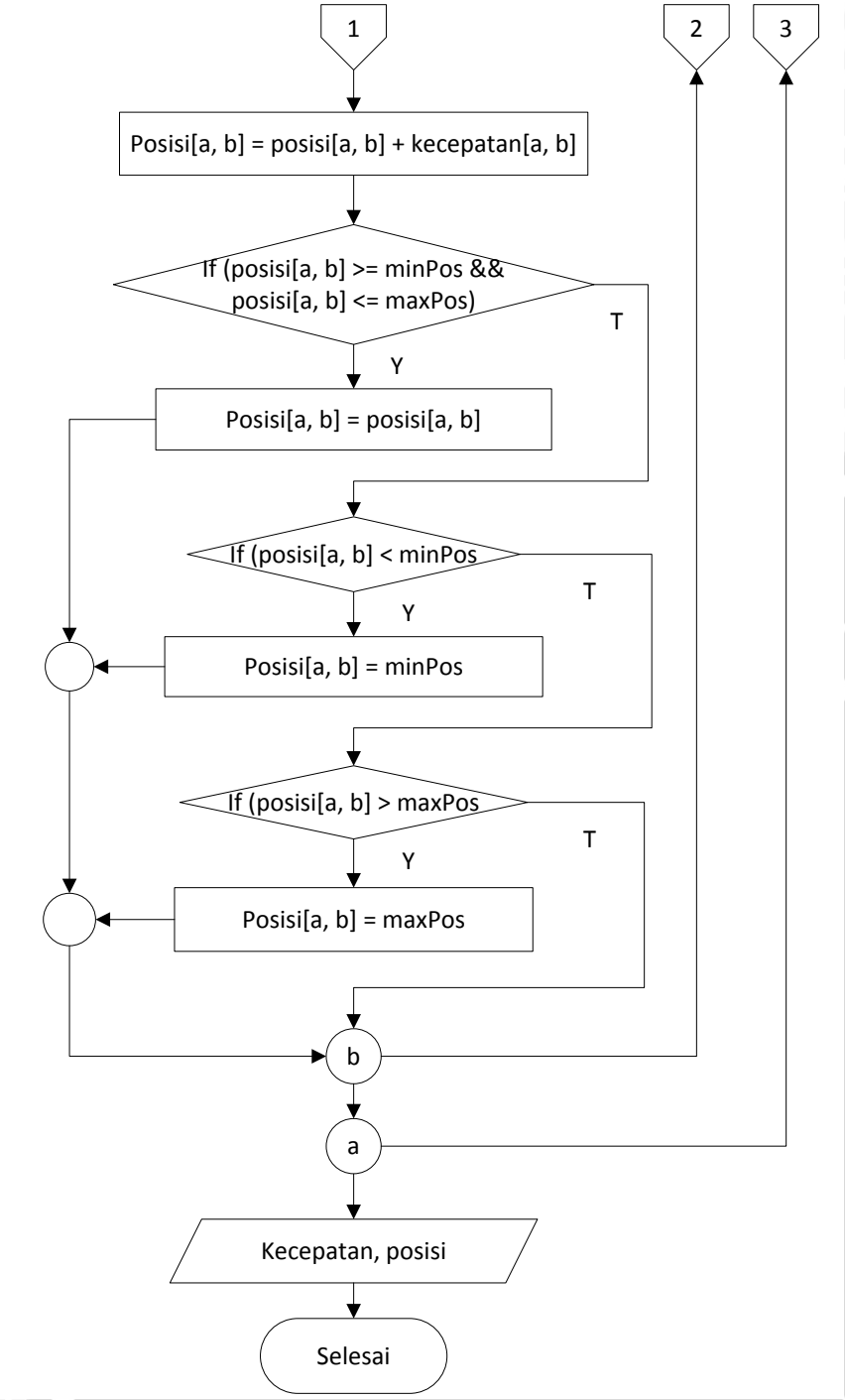


Gambar 4.8 Diagram Alir Perhitungan Rerata *Cluster*

4.2.2.3 *Update* Kecepatan dan Posisi

Pada tahap inisialisasi, nilai kecepatan adalah 0 dan nilai posisi adalah sama seperti pada tahap inisialisasi partikel. Pada iterasi selanjutnya, nilai kecepatan dan posisi bertutut-turut dihitung berdasarkan Persamaan 2.4 dan Persamaan 2.5 Diagram alir *update* kecapatan dan posisi ditunjukkan pada Gambar 4.9 sebagai berikut:





Gambar 4.9 Diagram Alir Proses *Update* Kecepatan dan Posisi

Langkah-langkah *update* kecepatan dan posisi partikel berdasarkan Gambar 4.9 adalah sebagai berikut:

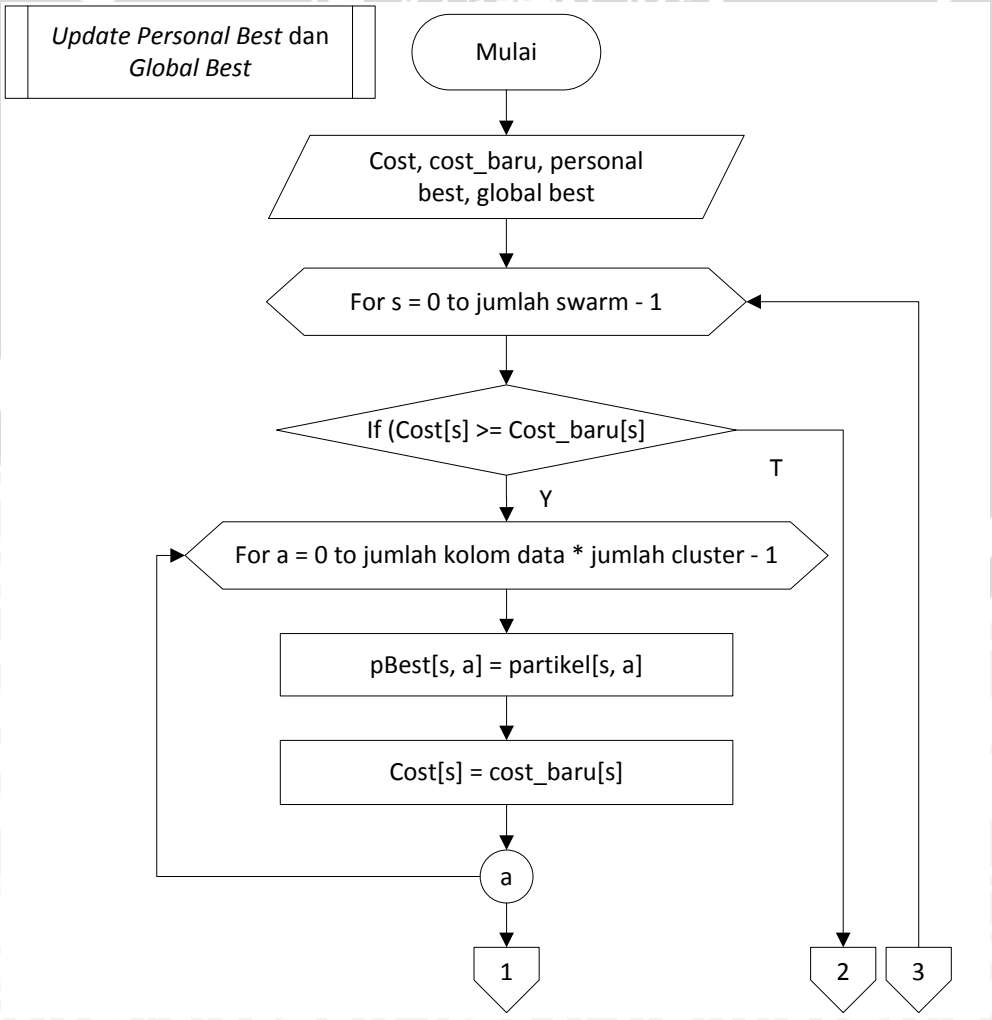
1. Sistem menerima masukan berupa, *personal best (pBest)*, *global bes*t *(gBest)*, kecepatan, dan posisi partikel
2. Melakukan *update* kecepatan menggunakan Persamaan 2.4. Aturan *update* kecepatan adalah sebagai berikut: jika nilai kecepatan berada pada interval minimal kecepatan (minKec) sampai dengan maksimal kecepatan (maxKec)

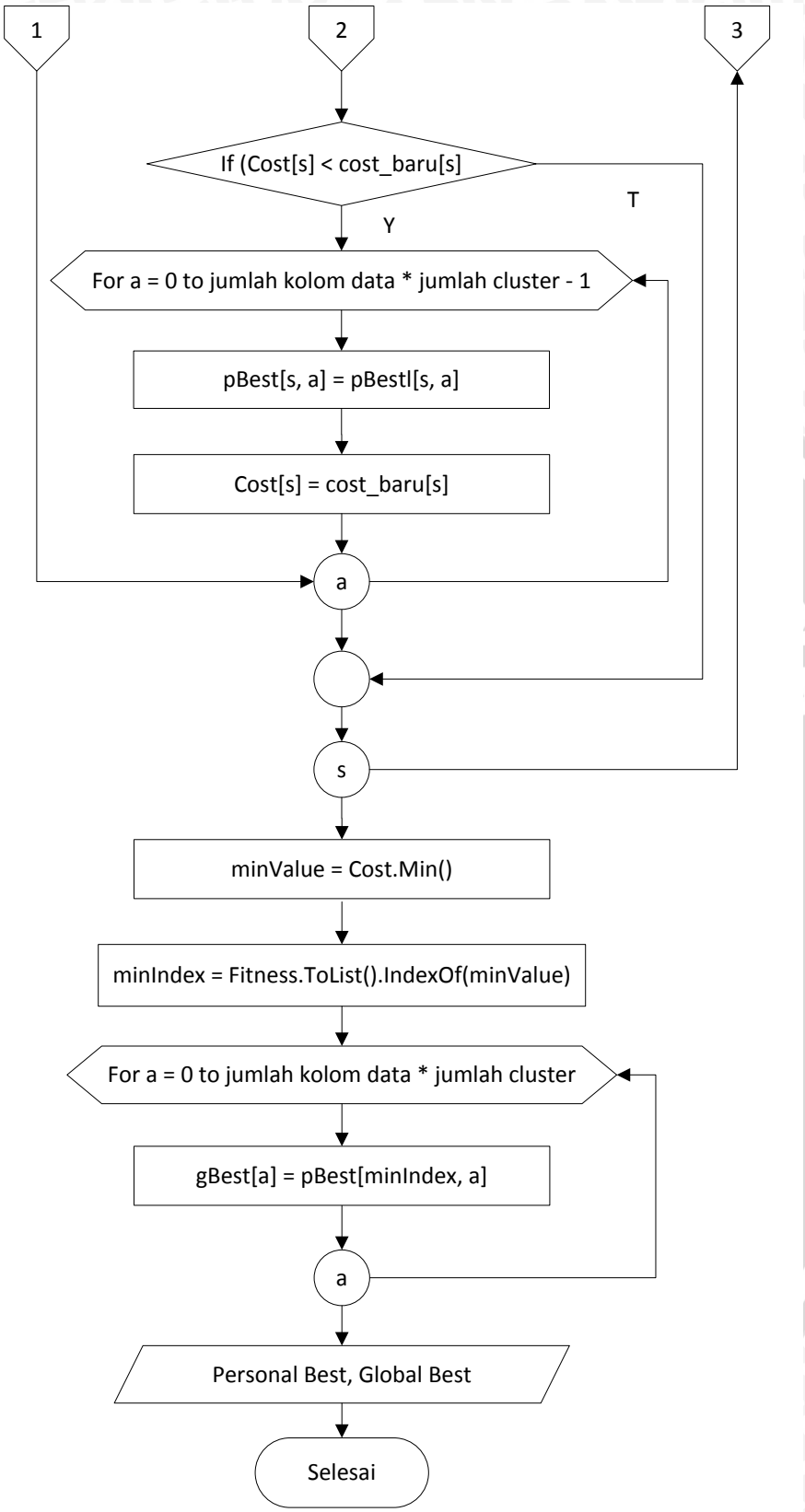
maka nilai kecepatan tetap, jika nilai kecepatan kurang dari minKec maka nilai kecepatan diset minKec, dan jika nilai kecepatan lebih dari maxKec maka nilai kecepatan diset maxKec

3. Melakukan *update* update posisi menggunakan Persamaan 2.5. Aturan *update* posisi adalah sebagai berikut: jika nilai posisi berada pada interval minimal posisi (minPos) sampai dengan maksimal posisi (maxPos) maka nilai posisi tetap, jika nilai posisi kurang dari minPos maka nilai posisi diset minPos, dan jika nilai posisi lebih dari maxPos maka nilai posisi diset maxPos
4. Keluaran sistem adalah kecepatan dan posisi yang telah diperbarui

4.2.2.4 *Update Personal Best* dan *Global Best*

Pada tahap inisialisasi, nilai *personal best (pBest)* ditentukan berdasarkan nilai partikel awal sedangkan *global best (gBest)* ditentukan berdasarkan nilai *personal best* terbaik yaitu *personal best* dengan nilai *cost* terkecil. Pada iterasi selanjutnya, nilai *personal best* diperoleh dengan membandingkan nilai *cost personal best* sekarang dengan nilai *personal best* sebelum sedangkan nilai *global best* seperti halnya pada tahap inisialisasi, diperoleh dari nilai *personal best* baru dengan nilai *cost* terkecil. Diagram alir *update personal best* dan *global best* ditunjukkan pada Gambar 4.10 sebagai berikut:





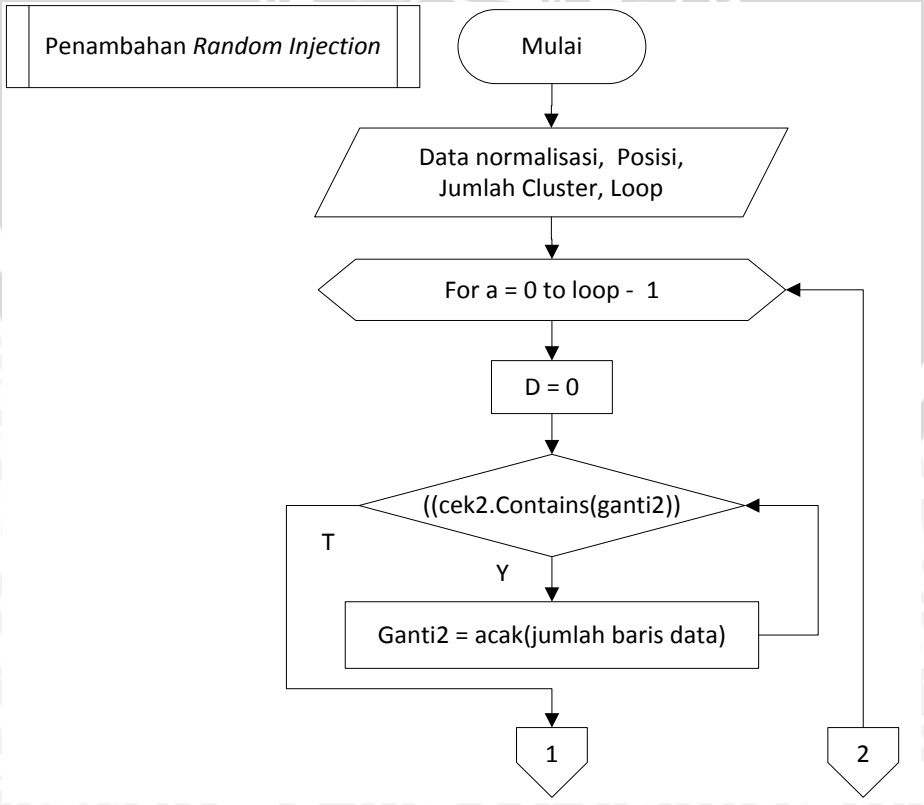
Gambar 4.10 Diagram Alir Proses *Update Personal Best* dan *Global Best*

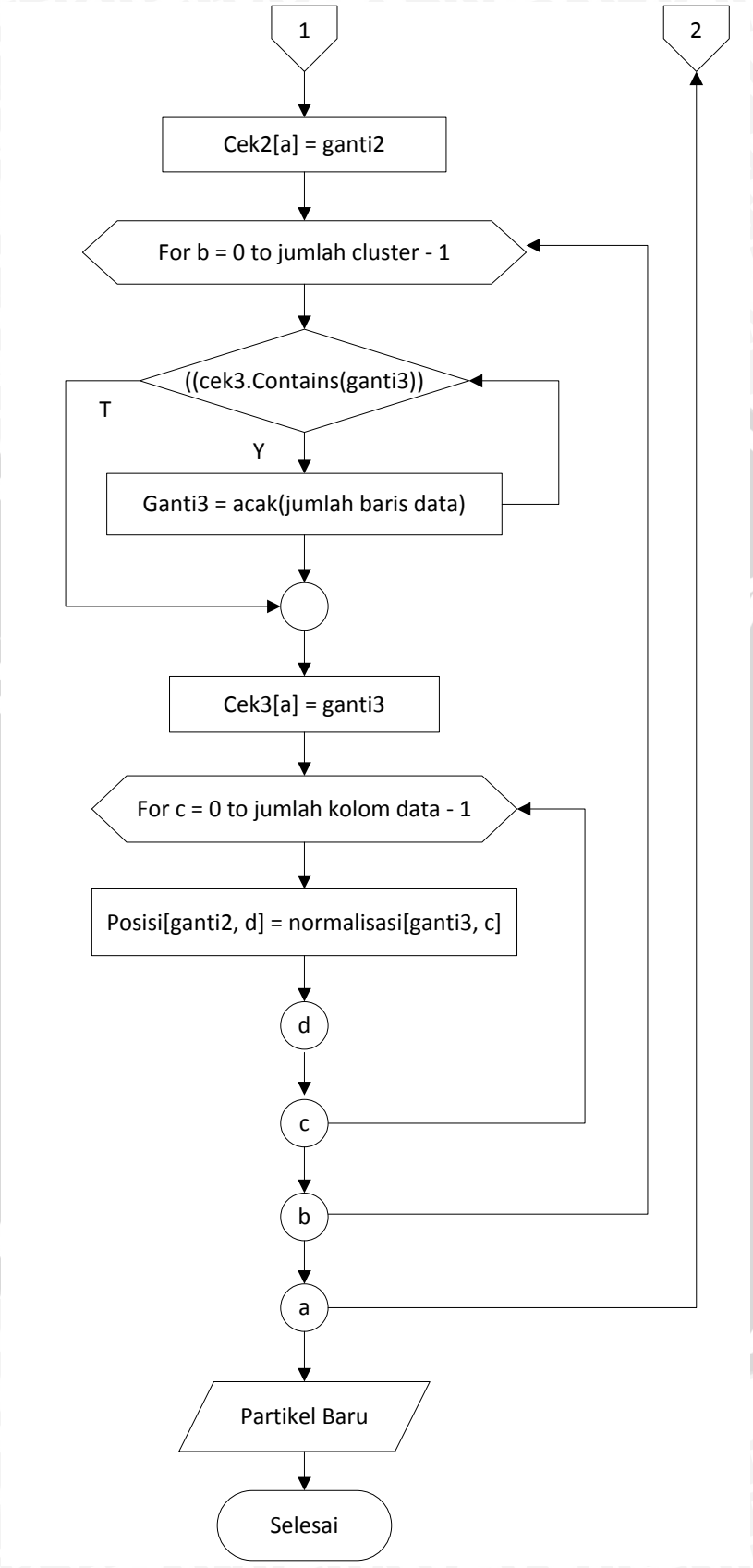
Langkah-langkah *update personal best* dan *global best* berdasarkan Gambar 4.10 adalah sebagai berikut:

1. Sistem menerima masukan berupa nilai posisi partikel, nilai *cost,* nilai *cost* baru, *pesonal best (pBest),* dan *global best (gBest).* Nilai *cost* baru merupakan nilai *cost* partikel pada iterasi ke x sedangkan nilai *cost* baru merupakan nilai *cost* partikel pada iterasi ke x+1
2. Melakukan *update personal best* dengan ketentuan sebagai berikut: jika nilai *cost* sebelum lebih dari atau sama dengan nilai *cost* sekarang maka *pBest* adalah partikel sekarang namun jika nilai *cost* sebelum kurang dari nilai *cost* sekarang maka *pBest* adalah tetap
3. Melakukan *update global best* dengan ketentuan sebagai berikut: nilai *global best* diinisialisasi berdasarkan nilai indeks dari nilai *cost* terkecil. Nilai indeks tersebut selanjutnya digunakan untuk mendeklarasikan indeks baris pada *array pBest* sehingga dengan kata lain nilai *global best* merupakan nilai *personal best* dengan nilai *cost* terkecil
4. Keluaran sistem adalah nilai *personal best* dan *global best* yang telah diperbarui

4.2.2.5 *Random Injection*

Penambahan *Random Injection* bertujuan untuk menangani konvergensi dini pada hasil optimasi menggunakan algoritma PSO. Mekanisme ini diadopsi dari algoritma genetika yang diimplementasikan oleh Mahmudy et al, (2014b) untuk optimasi *part type selection* dan *machine loading problem* pada permasalahan pada *flexible manufacturing system* (FMS). *Random injection* dilakukan setiap 10 kali iterasi dengan jumlah partikel yang diganti adalah sebanyak 20% dari ukuran *swarm.* Diagram alir proses penambahan *random injection* ditunjukkan pada Gambar 4.11 sebagai berikut:





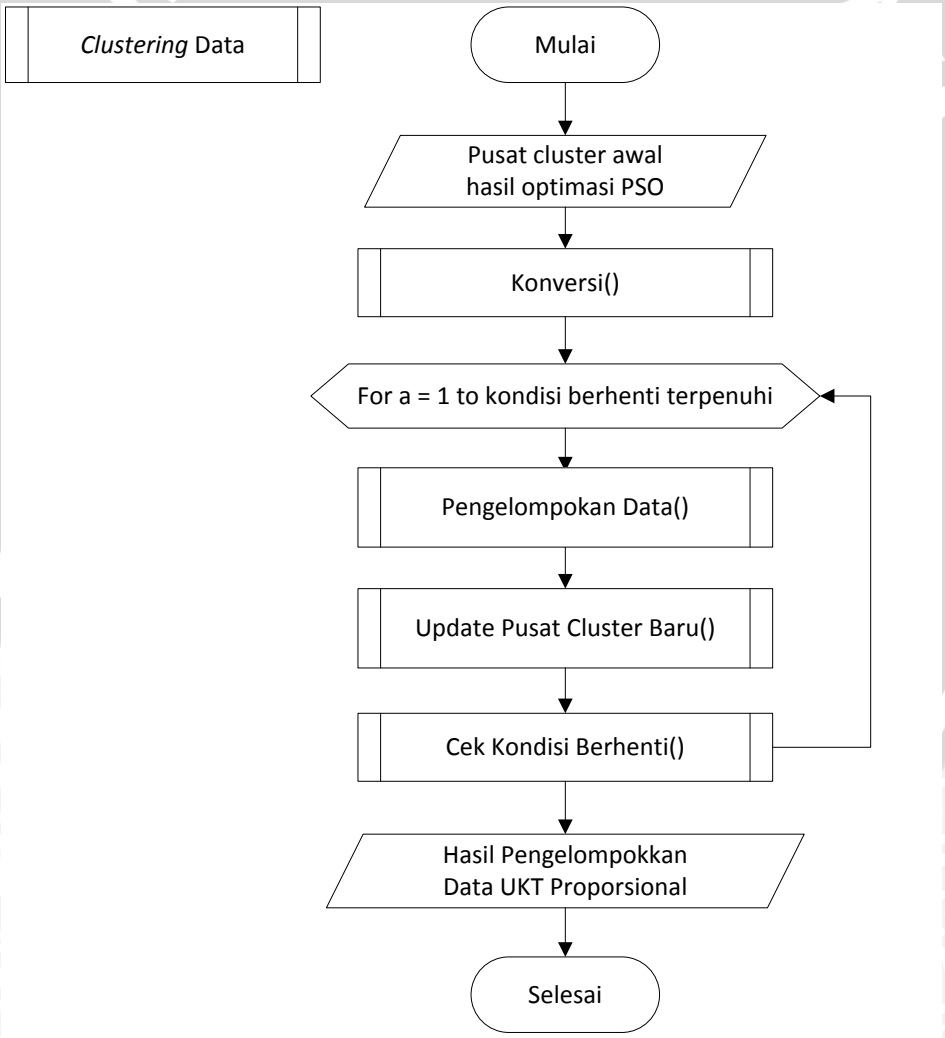
Gambar 4.11 Diagram Alir Proses *Random Injection*

Langkah-langkah penambahan *random injection* berdasarkan Gambar 4.11 adalah sebagai berikut:

1. Sistem menerima masukan berupa data UKT Proporsional yang telah ternormalisasi, posisi partikel, dan *loop*. *Loop* merupakan jumlah partikel yang diganti yaitu 20% dari ukuran *swarm.*
2. Mengganti sejumlah partikel secara acak dengan partikel baru. Proses pergantian tersebut sama halnya ketika proses inisialisasi partikel yaitu menggunakan data UKT Proporsional yang telah ternormalisasi
3. Keluaran sistem adalah partikel baru

4.2.3 Proses *Clustering* Data Menggunakan Algoritma *K-Means*

Clustering data menggunakan algoritma *K-Means* bertujuan untuk menyempurnakan proses pengelompokan data lebih lanjut. Diagram alir proses *clustering* data menggunakan algoritma *K-Means* ditunjukkan pada Gambar 4.12 sebagai berikut:



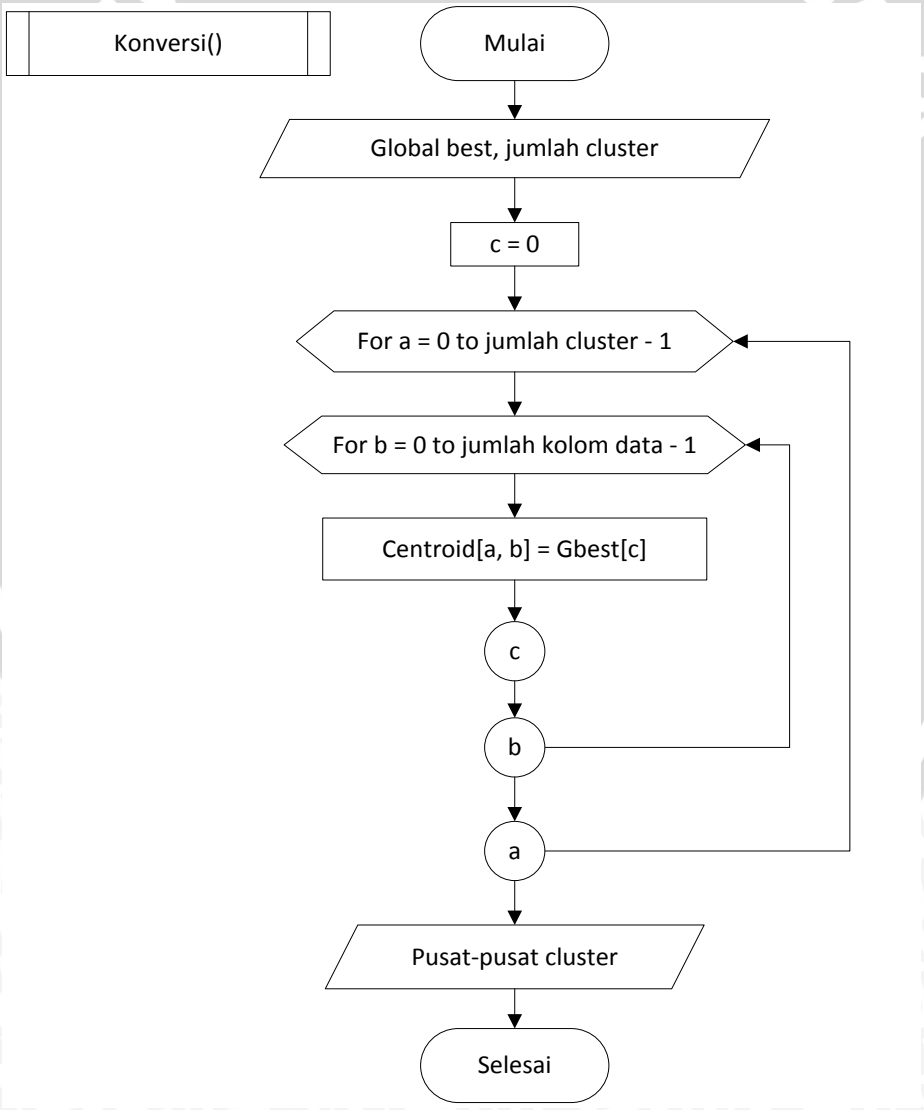
Gambar 4.12 Diagram Alir Proses *Clustering* Data Menggunakan *K-Means*

Langkah-langkah *clustering* data menggunakan algoritma *K-Means* berdasarkan Gambar 4.12 adalah sebagai berikut:

1. Sistem menerima masukan berupa *global best* hasi optimasi algoritma PSO. *Global best* tersebut selanjutnya dikonversi menjadi pusat-pusat *cluster*
2. Mengelompokkan setiap data ke dalam *cluster* yang memiliki jarak terdekat
3. Melakukan *update* pusat *cluster* baru
4. Melakukan cek kondisi berhenti. Jika kondisi berhenti terpenuhi maka iterasi berhenti jika tidak maka kembali ke langkah 2
5. Keluaran sistem adalah data UKT Proporsional yang telah dikelompokkan

4.2.3.1 Konversi *Global Best* Ke *Pusat Cluster*

Konversi *global best* ke pusat *cluster* bertujuan untuk menguraikan pusat-pusat *cluster* optimum dari hasil optimasi menggunakan algoritma PSO menjadi pusat-pusat *cluster* awal untuk algoritma *K-Means.* Diagram alir proses konversi *global best* ke pusat *cluster* dintujukkan pada Gambar 4.13 sebagai berikut:



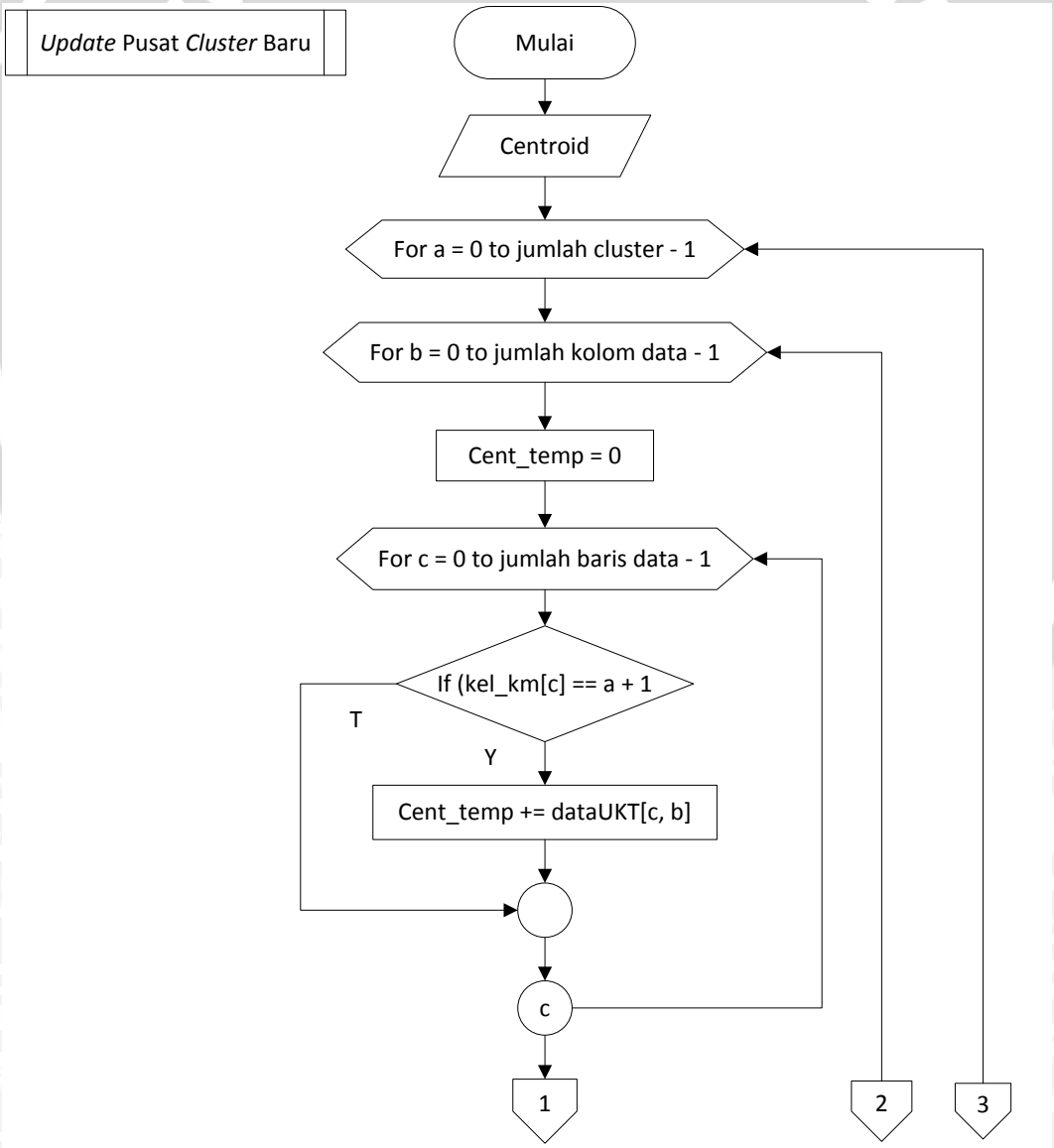
Gambar 4.13 Diagram Alir Konversi *Global Best* ke Pusat *Cluster*

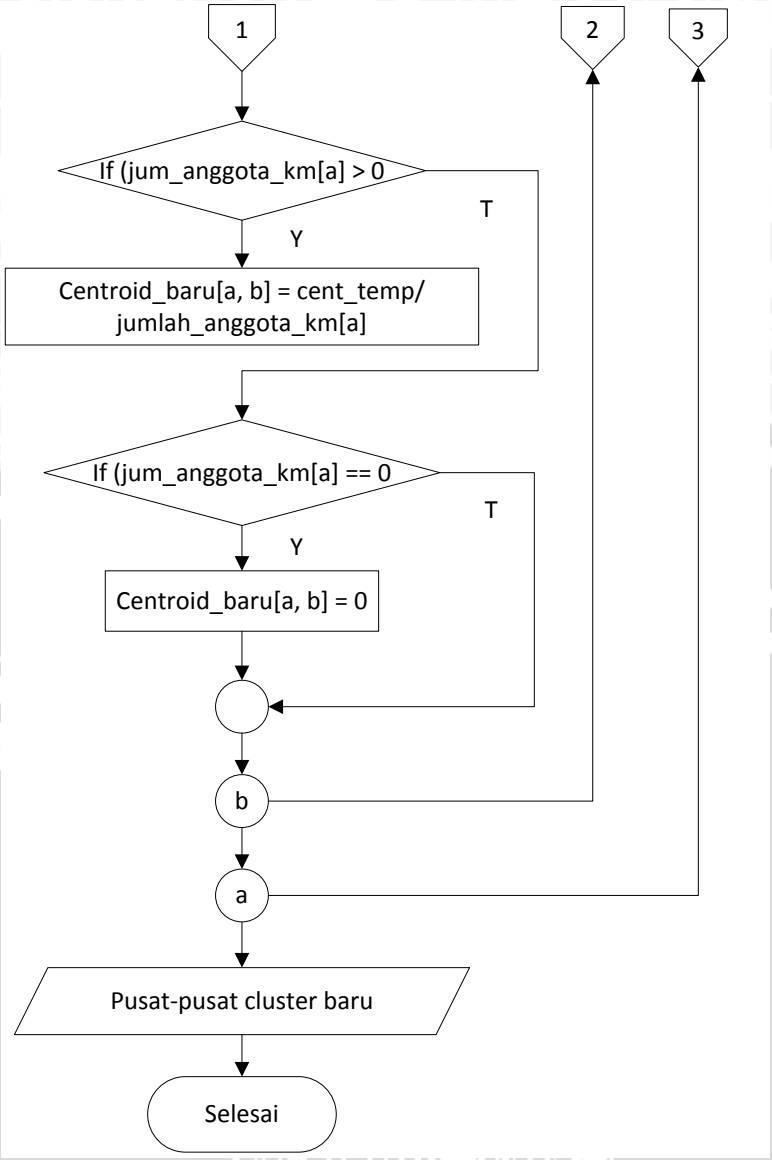
4.2.3.2 Pengelompokkan Data

Pengelompokkan data bertujuan untuk mengelompokkan setiap data ke dalam *cluster* yang memiliki jarak terdekat. Pada penelitian ini, penilaian kedekatan data dengan pusat *cluster* dihitung menggunakan teori pengukuran jarak *Euclidean.* Langkah-langkah pengelompokkan data terdiri dari dua tahap yaitu: mengukur jarak data dengan pusat *cluster* dan mengelompokkan setiap data ke dalam *cluster* terdekat. Diagram alir proses mengukur jarak dan proses mengelompokkan data berturut-turut ditunjukan pada Gambar 4.6 dan Gambar 4.7 pada sub bab sebelumnya.

4.2.3.3 *Update* Pusat *Cluster* Baru

Update pusat *cluster* dilakukan setiap iterasi dengan menggunakan Persamaan 2.6 yaitu menghitung rata-rata setiap nilai dimensi pusat *cluster*. Langkah ini bertujuan agar pusat *cluster* menjadi semakin lebih baik. Diagram alir *update* pusat *cluster* ditunjukkan pada Gambar 4.14 sebagai berikut:





Gambar 4.14 Diagram Alir Proses *Update* Pusat *Cluster* Baru

4.2.3.4 Cek Kondisi Berhenti 1

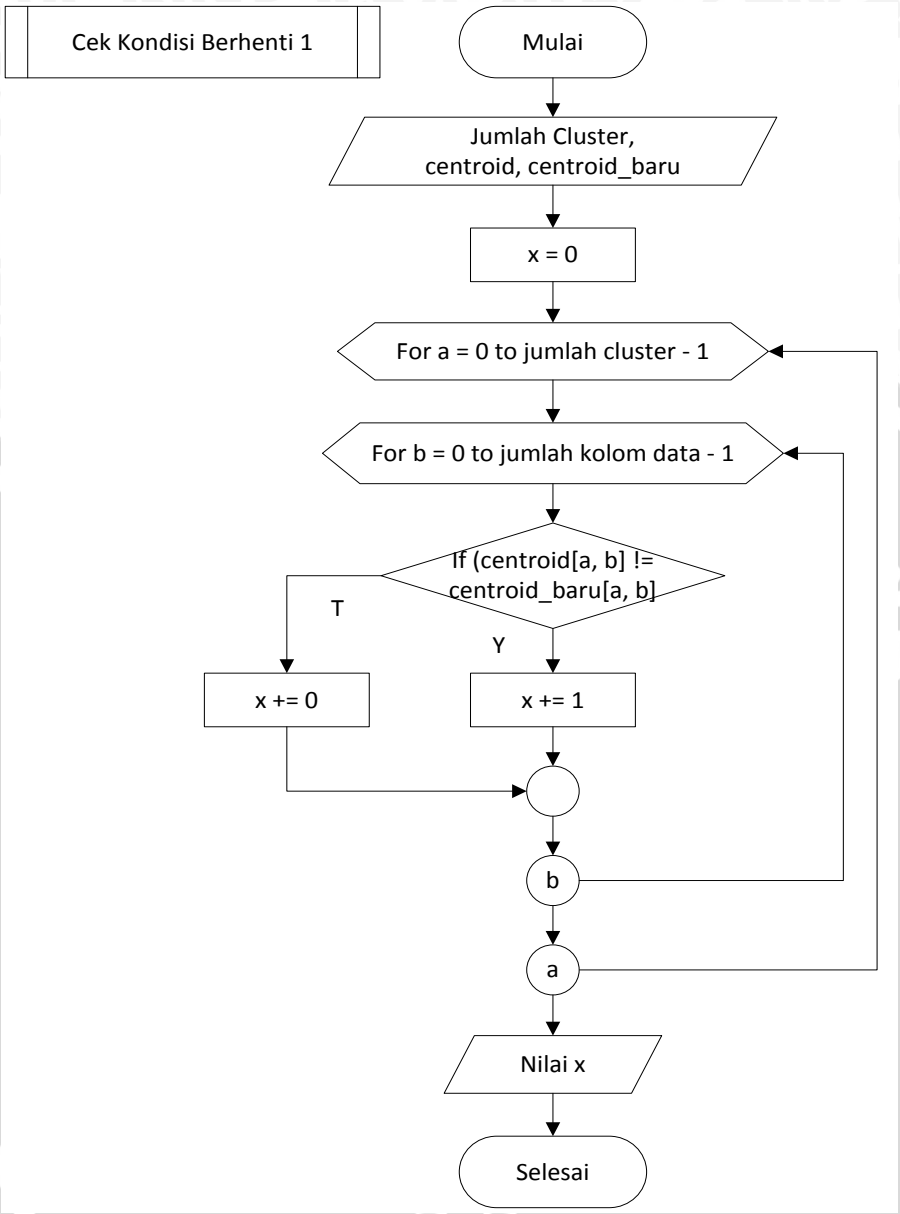
Alternatif kondisi pertama untuk penghentian iterasi pengelompokkan data menggunakan algoritma *K-Means* adalah ketika pusat *cluster* tidak mengalami perubahan sehingga anggota setiap *cluster* juga tidak berubah. Secara matematis, fungsi obyektif untuk alternatif kondisi berhenti pertama pada algoritma *K-Means* adalah sebagai berikut:

$$J = \sum_{i=1}^{c} D(v_i, v_i')$$

Keterangan:

J	: Total jarak antar pusat *cluster*
c	: Jumlah *cluster*
$D(v_i, v_i')$	: Jarak antara pusat *cluster* ke-*i* dengan pusat *cluster* ke-*i'*

Diagram alir alternatif kondisi berhenti pertama algoritma *K-Means* ditunjukkan pada Gambar 4.15 sebagai berikut:



Gambar 4.15 Diagram Alir Proses Cek Kondisi Berhenti 1

4.2.3.5 Cek Kondisi Berhenti 2

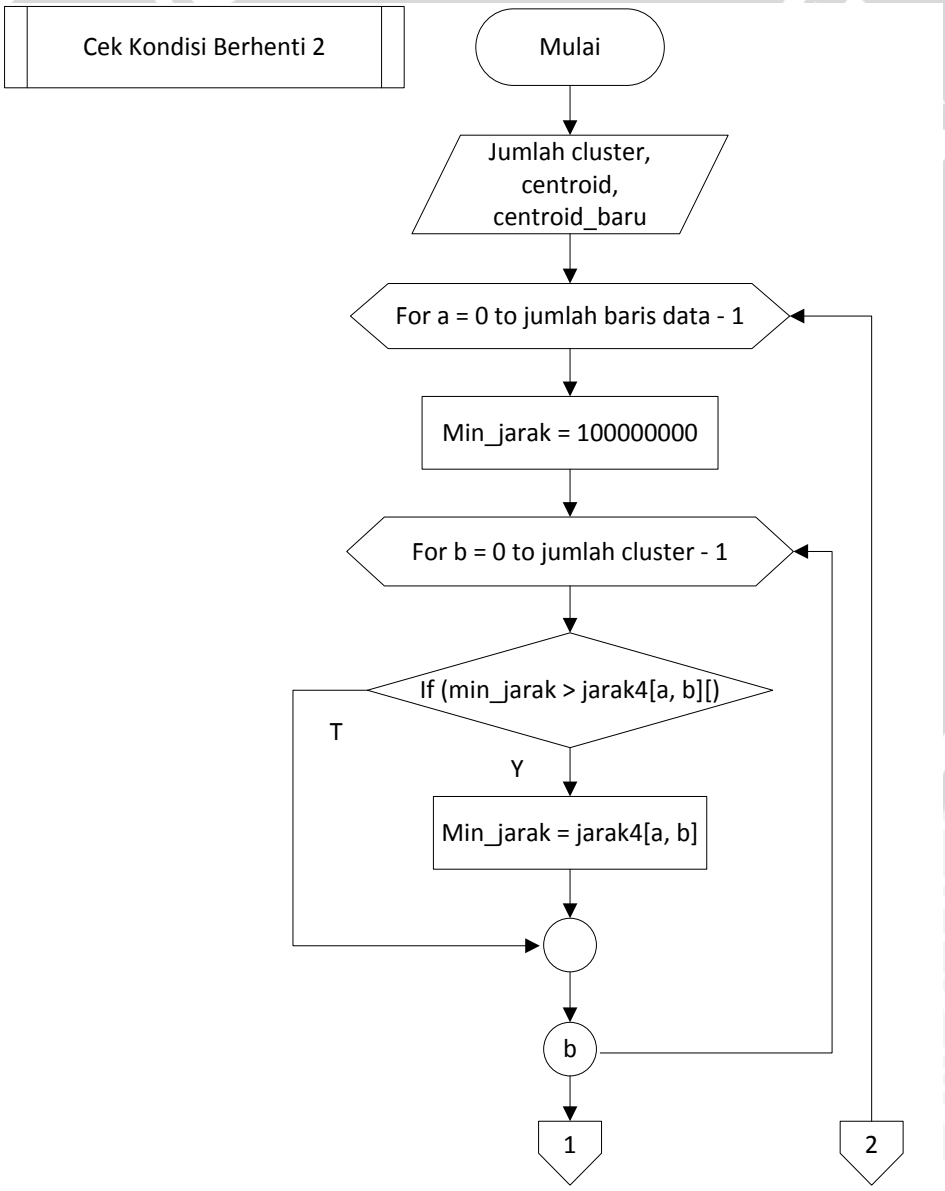
Alternatif kondisi kedua untuk penghentian iterasi pengelompokkan data menggunakan algoritma *K-Means* adalah ketika selisih jarak minimal yang dibentuk antara objek data dan pusat *cluster* bernilai kurang dari *threshold*. Nilai *threshold* pada penelitian ini telah ditentukan sebesar 10^{-8}. Secara matematis, fungsi obyektif untuk alternatif kondisi berhenti kedua pada algoritma *K-Means* adalah sebagai berikut:

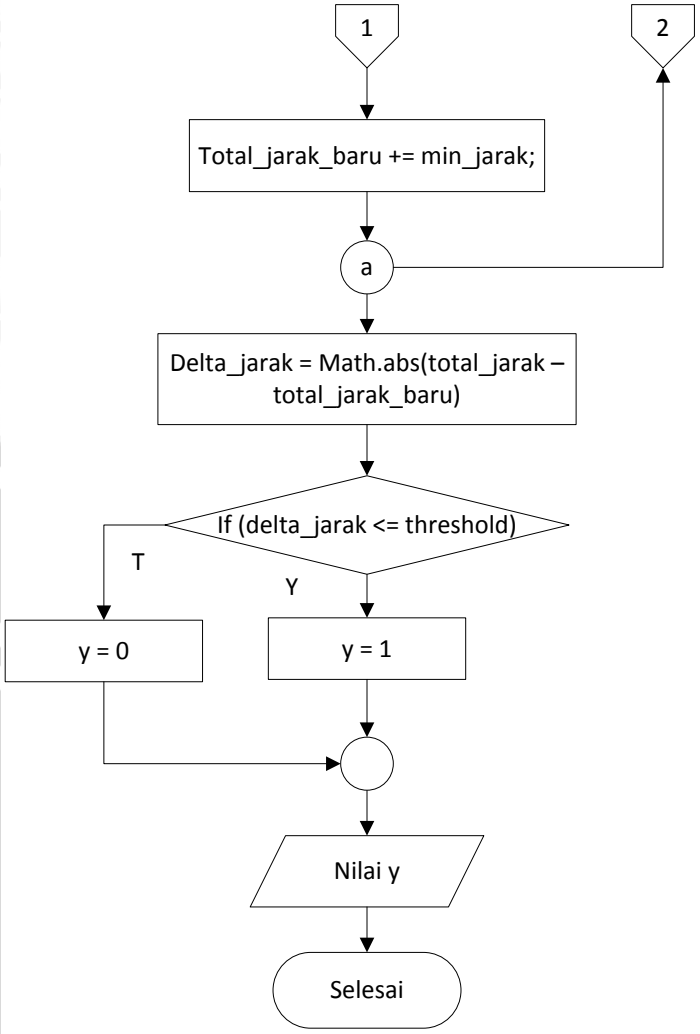
$$J = \sum_{k=1}^{N} \sum_{i=1}^{c} a_{ik} D(x_k, v_i)^2$$

Keterangan:

J	: Total jarak minimal antara data dengan pusat *cluster*
N	: Jumlah data
c	: Jumlah *cluster*
a_{ik}	: Keanggotaan data ke-*k* pada *cluster* ke-*i*
x_k	: Data ke-*k*
v_i	: Nilai pusat *cluster* ke-*i*

a_{ik} merupakan derajat keanggotaan data terhadap pusat *cluster* yang memiliki nilai 0 dan 1. a_{ik} bernilai 1 jika sebuah data merupakan anggota suatu *cluster* dan sebaliknya bernilai 0 jika sebuah data bukan merupakan anggota suatu *cluster.* Hal tersebut dapat pula diinterpretasikan sebagai penjumlahan jarak minimal data dengan pusat *cluster.* Diagram alir alternatif kondisi berhenti kedua algoritma *K-Means* ditunjukkan pada Gambar 4.16 sebagai berikut:





Gambar 4.16 Diagram Alir Proses Cek Kondisi Berhenti 2

4.2.3.6 Cek Kondisi Berhenti 3

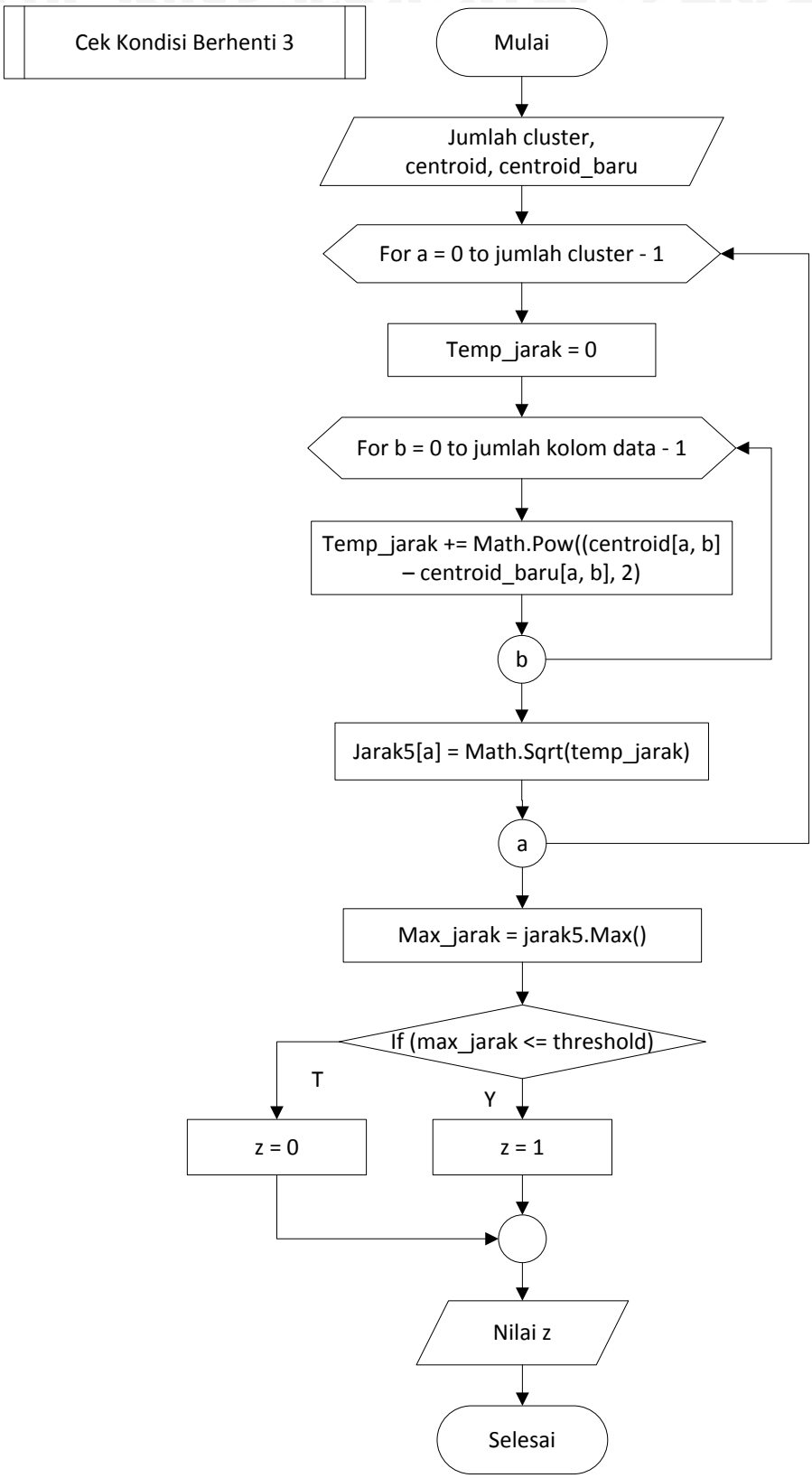
Alternatif kondisi ketiga untuk penghentian iterasi pengelompokkan data menggunakan algoritma *K-Means* adalah ketika jarak maksimal antara perubahan pusat *cluster* bernilai kurang dari *threshold.* Perbedaan mendasar dari Alternatif kondisi ketiga dan pertama terletak pada deklarasi jarak minimal. Pada kondisi pertama jarak minimal adalah 0 sedangkan pada kondisi ketiga adalah bernilai kurang dari *threshold* (10^{-8}). Secara matematis, fungsi obyektif untuk alternatif kondisi berhenti ketiga pada algoritma *K-Means* adalah sebagai berikut:

$$J = max\ (\sum_{i=1}^{c} D(v_i, v_i'))$$

Keterangan:

J	: Total jarak antar pusat *cluster*
c	: Jumlah *cluster*
$D(v_i, v_i')$	: Jarak antara pusat *cluster* ke-*i* dengan pusat *cluster* ke-*i'*

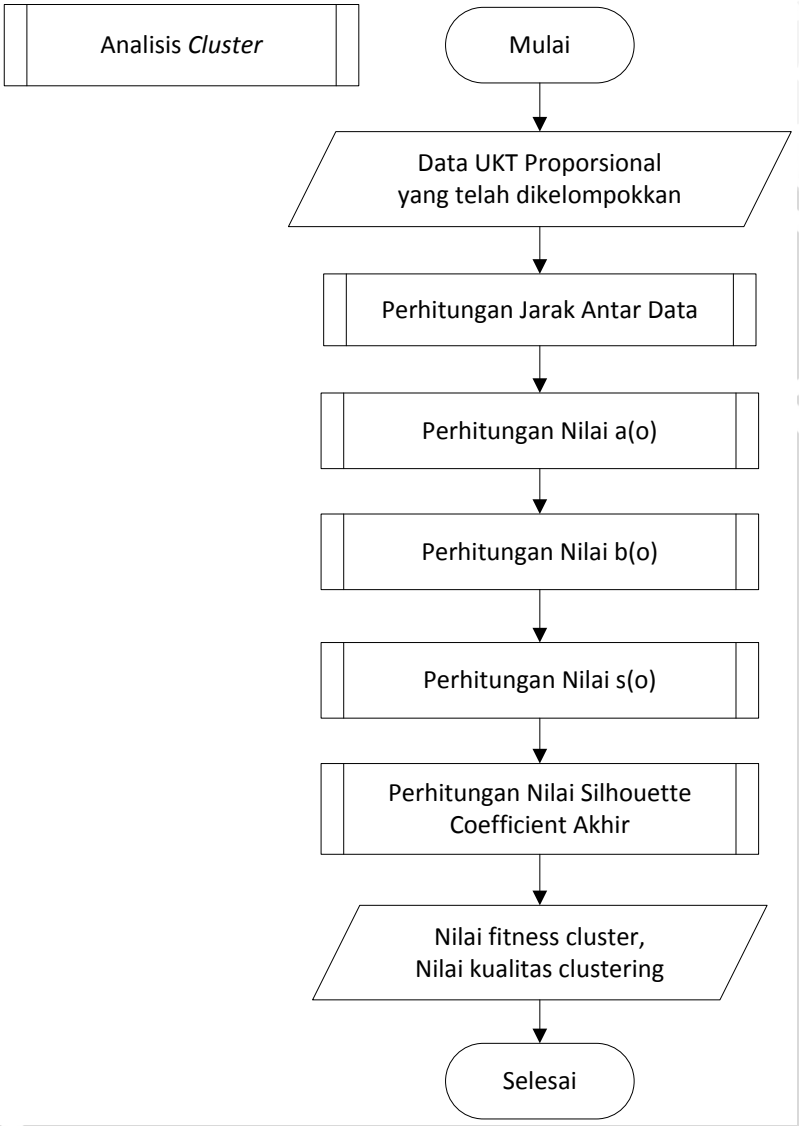
Diagram alir alternatif kondisi berhenti ketiga algoritma *K-Means* ditunjukkan pada Gambar 4.17 sebagai berikut:



Gambar 4.17 Diagram Alir Proses Cek Kondisi Berhenti 3

4.2.4 Proses Analisis *Cluster* Menggunakan *Silhouette Coefficient*

Analisis *cluster* bertujuan untuk mengetahui kualitas hasil *clustering* yang dihasilkan dari proses *clustering* data menggunakan metode *Hybrid* PSO dan *K-Means* (HPSOKM). Diagram alir proses analisis *cluster* menggunakan metode *Silhouette Coefficient* ditunjukkan pada Gambar 4.18 sebagai berikut:



Gambar 4.18 Diagram Alir Proses Analisis *Cluster Silhouette Coefficient*

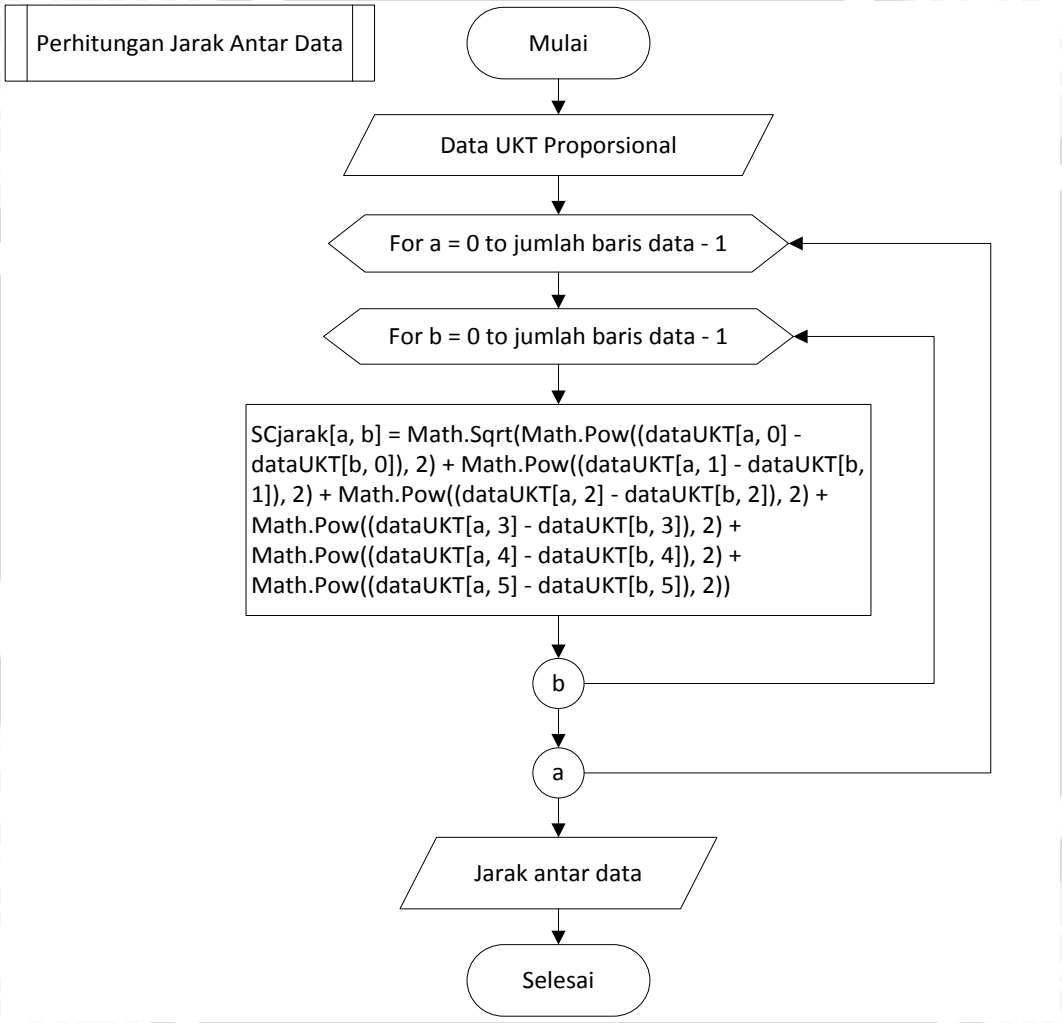
Langkah-langkah analisis *cluster* menggunakan *Silhouette Coefficient* berdasarkan Gambar 4.18 adalah sebagai berikut:

1. Sistem menerima masukan berupa data UKT Proporsional yang telah dikelompokkan
2. Menghitung nilai $a(o)$ yaitu jarak antar data dalam satu *cluster* sama
3. Menghitung nilai $b(o)$ yaitu jarak antar data pada *cluster* berbeda
4. Menghitung nilai $s(o)$ yaitu nilai *Silhouette Coefficient* setiap data
5. Menghitung nilai *Silhouette Coefficient* akhir (kualitas hasil *clustering*)

6. Keluaran sistem adalah nilai *fitness* setiap *cluster,* nilai rata-rata *fitness cluster*, dan nilai kualitas *clustering.* Nilai yang digunakan untuk mengevaluasi hasil *cluster* adalah nilai kualitas *clustering*

4.2.4.1 Perhitungan Jarak Antar Data

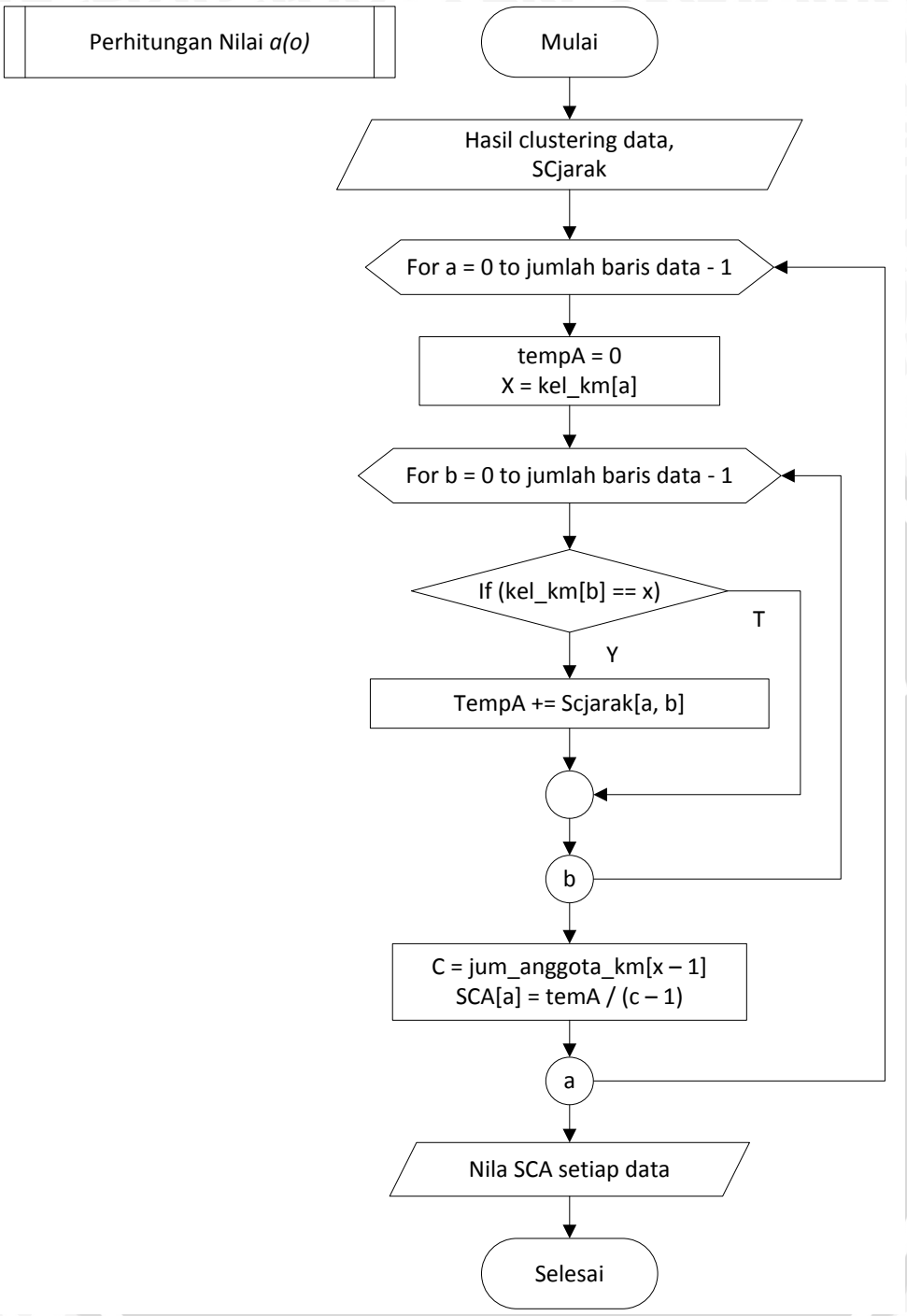
Perhitungan jarak antar data bertujuan untuk menghitung jarak antar data dalam satu *dataset* menggunakan teori pengukuran jarak *Euclidean*. Selain itu, langkah perhitungan jarak ini untuk memudahkan perhitungan nilai $a(o)$ dan $b(o)$ setiap data. Diagram alir proses perhitungan jarak antar data ditunjukkan pada Gambar 4.19 sebagai berikut:



Gambar 4.19 Diagram Alir Proses Perhitungan Jarak Antar Data

4.2.4.2 Perhitungan Nilai $a(o)$

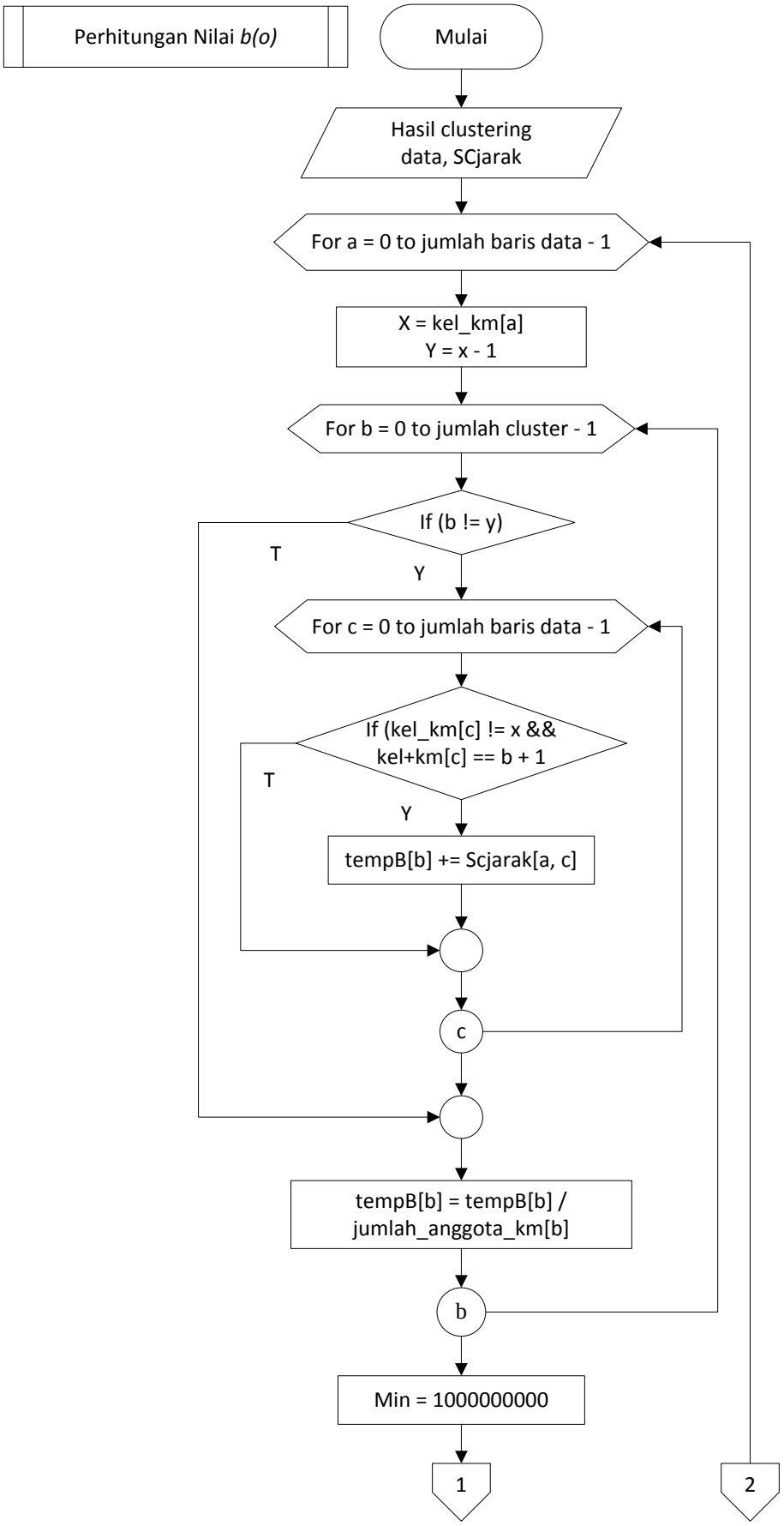
Perhitungan nilai $a(o)$ bertujuan untuk menghitung jarak antar data dalam satu *cluster* yang sama. Semakin kecil nilai $a(o)$ maka semakin baik karena mengindikasikan bahwa jarak antar data dalam satu *cluster* sama semakin dekat dan sebaliknya. Diagram alir proses perhitungan nilai $a(o)$ ditunjukkan pada Gambar 4.20 sebagai berikut:

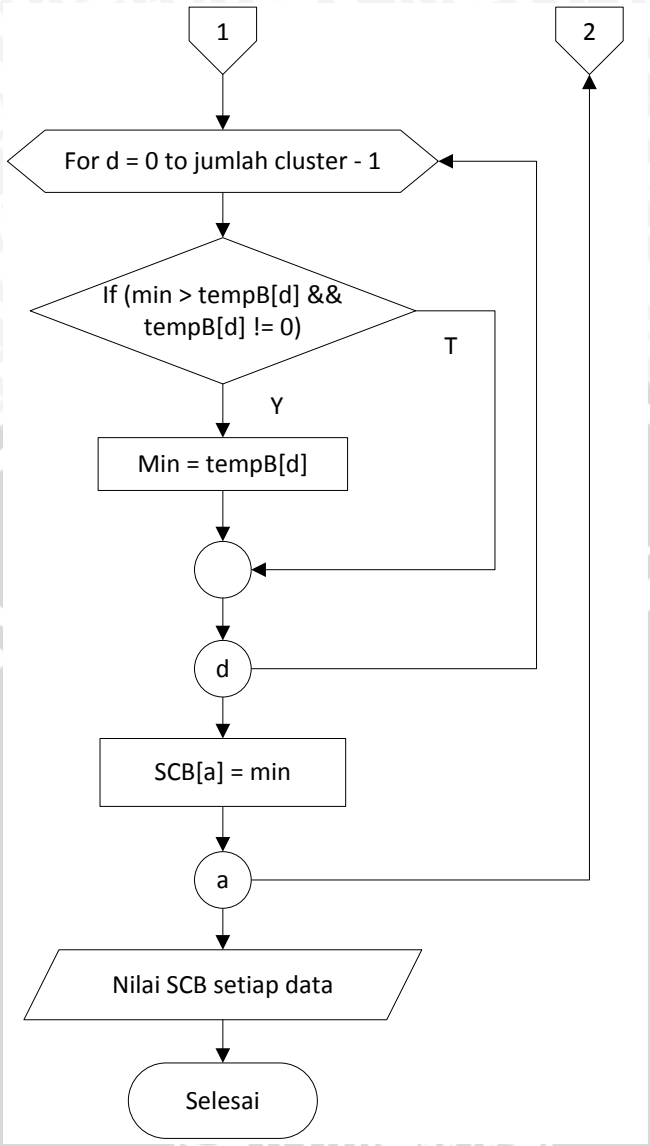


Gambar 4. 20 Diagram Alir Proses Perhitungan Nilai $a(o)$

4.2.4.3 Perhitungan Nilai $b(o)$

Perhitungan nilai $b(o)$ bertujuan untuk menghitung jarak antar data pada *cluster* berbeda. Semakin besar nilai $b(o)$ maka semakin baik karena mengindikasikan bahwa jarak antar data pada *cluster* berbeda semakin jauh dan sebaliknya. Diagram alir proses perhitungan nilai $b(o)$ ditunjukkan pada Gambar 4.21 sebagai berikut:

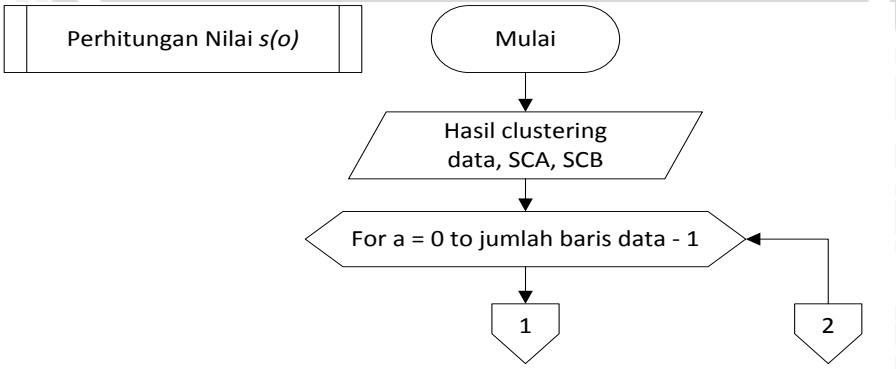

Perhitungan Nilai b(o)
Mulai
Hasil clustering data, SCjarak
For a = 0 to jumlah baris data - 1
X = kel_km[a]
Y = x - 1
For b = 0 to jumlah cluster - 1
If (b != y)
T
Y
For c = 0 to jumlah baris data - 1
If (kel_km[c] != x && kel+km[c] == b + 1
T
Y
tempB[b] += Scjarak[a, c]
c
tempB[b] = tempB[b] / jumlah_anggota_km[b]
b
Min = 1000000000
1
2

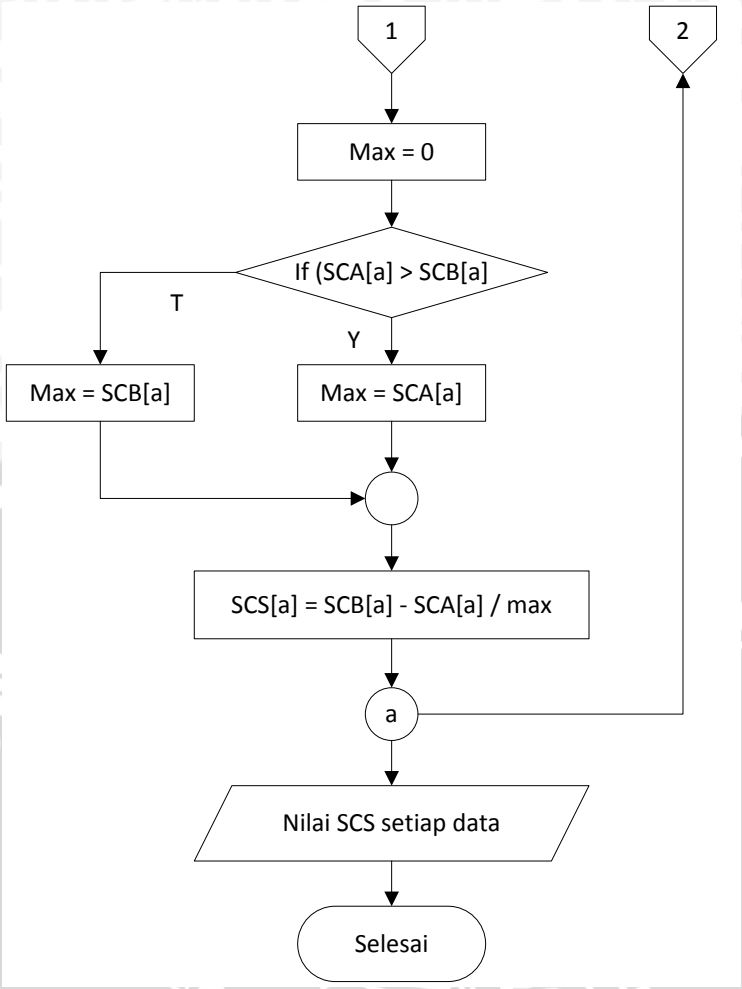


Gambar 4.21 Diagram Alir Proses Perhitungan Nilai $b(o)$

4.2.4.4 Perhitungan Nilai $s(o)$

Perhitungan nilai $s(o)$ bertujuan untuk menghitung nilai *Silhouette Coefficient* dari setiap data menggunakan Persamaan 2.10. Diagram alir proses perhitungan nilai $s(o)$ ditunjukkan pada Gambar 4.22 sebagai berikut:

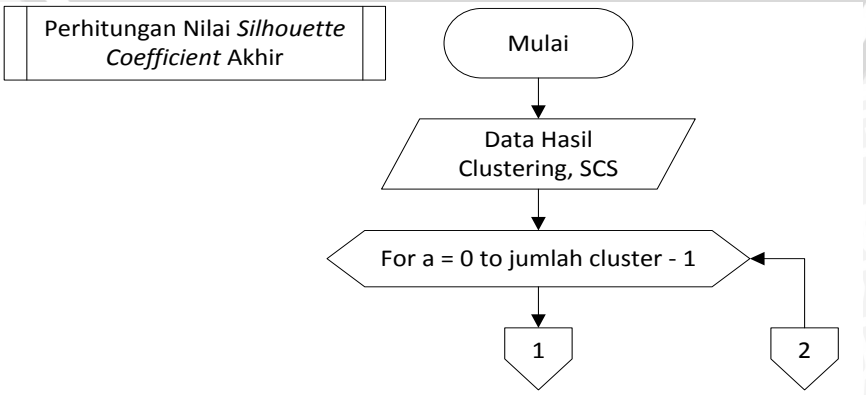


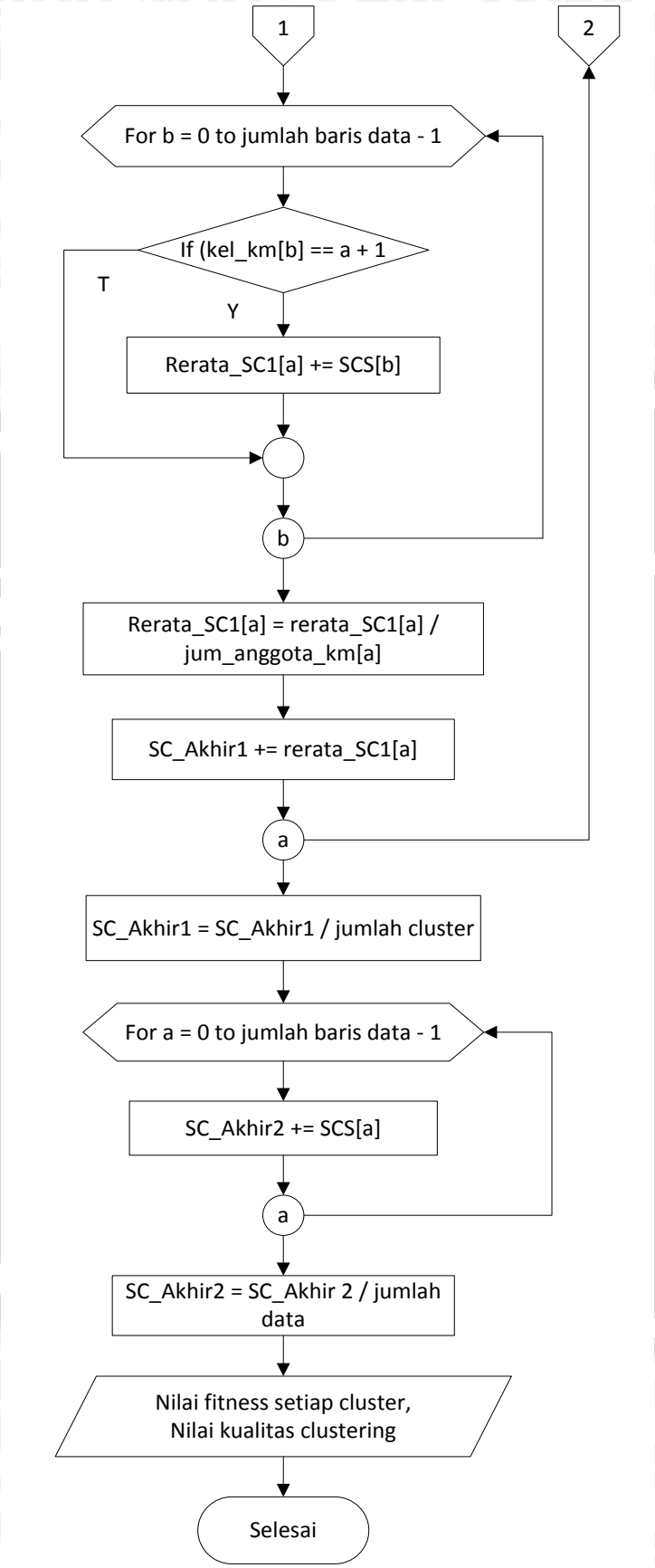


Gambar 4.22 Diagram Alir Proses Perhitungan Nilai $s(o)$

4.2.4.5 Perhitungan Nilai *Silhouette Coefficient* Akhir

Perhitungan nilai *Silhouette Coefficient* akhir bertujuan untuk mengevaluasi kualitas hasil *clustering* data UKT Proporsional. Pada penelitian ini, metode *Silhouette Coefficient* digunakan untuk menghitung nilai *fitness* setiap *cluster* dan kualitas *clustering.* Nilai kualitas *clustering* berada pada interval -1 sampai dengan 1. Diagram alir proses perhitungan nilai *Silhouette Coefficient* akhir ditunjukkan pada Gambar 4.23 sebagai berikut:

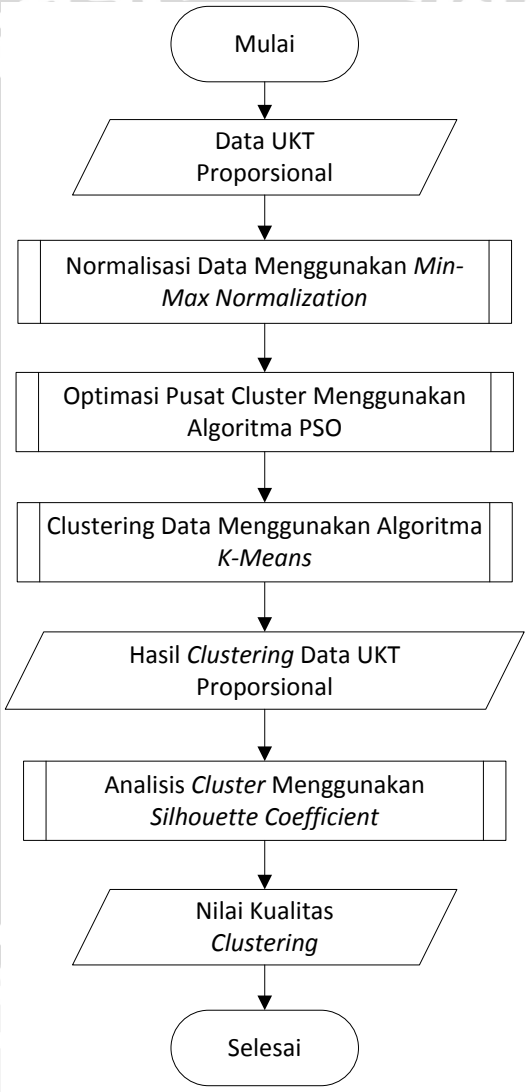




Gambar 4.23 Diagram Alir Perhitungan Nilai *Silhouette Coefficient* Akhir

4.3 Siklus Penyelesaian Masalah Menggunakan Algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM)

Siklus algoritma HPSOKM yang telah diuraikan pada Sub bab 4.1 selanjutnya disederhanakan menjadi perhitungan manual untuk memudahkan pemahaman tentang penyelesaian *clustering* data UKT Proporsional sebelum diimplementasikan ke dalam kode program. Siklus penyelesaian masalah diawali dengan perhitungan normalisasi data menggunakan metode *Min-Max Normalization*, perhitungan optimasi pusat *cluster* menggunakan algoritma PSO, perhitungan *clustering* data menggunakan algoritma *K-Means,* dan perhitungan evaluasi kualitas *clustering* menggunakan metode *Silhouette Coefficient.* Diagram alir proses sistem *clustering* data penentuan UKT Proporsional secara keseluruhan ditunjukkan pada Gambar 4.24 sebagai berikut:



Gambar 4.24 Diagram Alir Sistem *Clustering* Data UKT Proporsional

Perhitungan manual pada penelitian ini menggunakan sampel data UKT Proporsional sebanyak 10 *record* (ditampilkan pada Sub bab 4.1). Berikut inisialisasi awal yang digunakan dalam perhitungan manual:

- Jumlah *cluster* : 3
- Jumlah iterasi : 5
- Ukuran *swarm* : 5
- Bobot inersia maksimal : 0.9
- Bobot inersia minimal : 0.4
- Koefisien akselerasi 1 & 2 : 2
- Max & Min kecepatan : 0.005 & -0.005
- Max & Min Posisi : 1 & 0

4.3.1 Perhitungan Normalisasi Data

Langkah-langkah normalisasi data menggunakan metode *Min-Max Normalization* dihitung berdasarkan Persamaan 2.7 adalah sebagai berikut:

Langkah 1: menghitung nilai maksimal dan minimal setiap atribut. Nilai maksimal dan minimal setiap atribut data UKT Proporsional ditunjukkan pada Tabel 4.2 sebagai berikut:

Tabel 4.2 Nilai Maksimal dan Minimal Setiap Atribut

	Gaji	PBB	PKB	Listrik	Telepon	Air
Max	13500000	404500	2202000	595216	745995	220000
Min	1200000	0	0	0	0	0

Langkah 2: menghitung nilai normalisasi data menggunakan metode *Min - Max Normalization.* Sebagai contoh menghitung nilai normalisasi data ke-1 untuk semua atribut data

$$d_{1,1} = \frac{1874000 - 1200000}{13500000 - 1200000}(1 - 0) + 0 = 0.054797$$

$$d_{1,2} = \frac{0 - 0}{404500 - 0}(1 - 0) + 0 = 0$$

$$d_{1,3} = \frac{152900 - 0}{2202000 - 0}(1 - 0) + 0 = 0.069437$$

$$d_{1,4} = \frac{0 - 0}{595216 - 0}(1 - 0) + 0 = 0$$

$$d_{1,5} = \frac{0 - 0}{745995 - 0}(1 - 0) + 0 = 0$$

$$d_{1,6} = \frac{0 - 0}{220000 - 0}(1 - 0) + 0 = 0$$

Hasil perhitungan normalisasi secara keseluruhan ditunjukkan pada Tabel 4.3 sebagai berikut:

Tabel 4.3 Normalisasi Data UKT Proporsional

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air
1	0,054797	0	0,069437	0	0	0
2	0,308943	0,034917	1	0,661597	0	0,030073

3	0,715447	0,123609	0,908038	0,836703	0,067025	0
4	0,010569	0	0,056085	0,045362	0	0
5	0	0	0,079019	0,134405	0	0
6	1	0,112841	0,178247	1	1	0,236773
7	0,308943	0,142334	0,099364	0,672025	0,230565	0
8	0,14226	0,309023	0	0,22679	0	0,299018
9	0,715447	1	0,798592	0,682105	0,67469	1
10	0,300813	0	0,131199	0	0	0

4.3.2 Perhitungan Inisialisasi Partikel

Partikel diinisialisasi dengan memilih sejumlah *k* data secara acak dari data normalisasi yang diasumsikan sebagai pusat *cluster* awal. Sebagai contoh, telah ditentukan ukuran *swarm* sebanyak 5 partikel dan jumlah *cluster* 3 sehingga inisialisasi partikel adalah sebagai berikut:

Tabel 4.4 Perhitungan Inisialisasi Partikel

Partikel ke-	Nilai Partikel					
1	0,308943	0,034917	1	0,661597	0	0,030073
	0,308943	0,142334	0,099364	0,672025	0,230565	0
	0,715447	1	0,798592	0,682105	0,67469	1
2	0,300813	0	0,131199	0	0	0
	0	0	0,079019	0,134405	0	0
	0,308943	0,034917	1	0,661597	0	0,030073
3	0,010569	0	0,056085	0,045362	0	0
	0,715447	0,123609	0,908038	0,836703	0,067025	0
	0,054797	0	0,069437	0	0	0
4	0,715447	0,123609	0,908038	0,836703	0,067025	0
	0,14226	0,309023	0	0,22679	0	0,299018
	0	0	0,079019	0,134405	0	0
5	1	0,112841	0,178247	1	1	0,236773
	0,010569	0	0,056085	0,045362	0	0
	0,054797	0	0,069437	0	0	0

4.3.3 Perhitungan Nilai *Cost Iterasi 0*

Nilai *cost* dihitung menggunakan 3 tahapan yaitu menghitung jarak data dengan pusat *cluster,* menggelompokkan data, dan menghitung rerata. Nilai *cost* selanjutnya dihitung dengan merata-rata nilai rerata setiap *cluster.* Langkah-langkah menghitung nilai *cost* adalah sebagai berikut:

Langkah 1: menghitung jarak data dengan pusat *cluster* dan mengelompokkan data ke dalam *cluster* terdekat. Sebagai contoh, menghitung nilai *cost* partikel ke-1

Jarak data ke-1 terhadap pusat *cluster* ke-1

$$\text{Jarak } 1 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}308943)^2 + (0 - 0{,}034917)^2 + (0{,}069437 - 1)^2 + \\ (0 - 0{,}661597)^2 + (0 - 0)^2 + (0 - 0{,}030073)^2\end{matrix}} = 1{,}170629$$

Jarak data ke-1 terhadap pusat *cluster* ke-2

$$\text{Jarak } 2 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}308943)^2 + (0 - 0{,}142334)^2 + (0{,}069437 - 0{,}099364)^2 + \\ (0 - 0{,}672025)^2 + (0 - 0{,}230565)^2 + (0 - 0)^2\end{matrix}} = 0{,}768455$$

Jarak data ke-1 terhadap pusat *cluster* ke-3

$$\text{Jarak } 3 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}715447)^2 + (0 - 1)^2 + (0{,}069437 - 0{,}798592)^2 + \\ (0 - 0{,}682105)^2 + (0 - 0{,}67469)^2 + (0 - 1)^2\end{matrix}} = 1{,}971953$$

Hasil perhitungan jarak data dengan pusat *cluster* partikel 1 dan pengelompokkan data ditunjukkan pada Tabel 4.5 sebagai berikut:

Tabel 4.5 Perhitungan Jarak Dan Pengelompokkan Data Iterasi 0

Partikel ke-	Data ke-	Jarak 1	Jarak 2	Jarak 3	*Cluster*
1	1	1,170629	0,768455	1,971953	2
	2	0	0,936406	1,591723	1
	3	0,466506	0,934566	1,474176	1
	4	1,166992	0,746342	1,977073	2
	5	1,106213	0,676989	1,945464	2
	6	1,521891	1,113693	1,429073	2
	7	0,936406	0	1,608448	2
	8	1,168035	0,637376	1,611718	2
	9	1,591723	1,608448	0	3
	10	1,09303	0,725339	1,880906	2

Langkah 2: menghitung rerata setiap *cluster.* Sebagai contoh, menghitung nilai rerata setiap *cluster* terhadap partikel ke-1

$$\text{Rerata } 1 = \frac{0 + 0{,}466506}{2} = 0{,}233253$$

$$\text{Rerata } 2 = \frac{0{,}768455 + 0{,}746342 + 0{,}676989 + 1{,}113693 + 0 + 0{,}637376 + 0{,}725339}{7} = 0{,}666885$$

$$\text{Rerata } 3 = \frac{0}{1} = 0$$

Langkah 3: menghitung nilai *cost.* Sebagai contoh, menghitung nilai *cost* partikel ke-1

$$\text{Cost} = \frac{0{,}233253 + 0{,}666885 + 0}{3} = 0{,}300046$$

Hasil perhitungan nilai *cost* iterasi ke-0 secara keseluruhan ditunjukkan pada Tabel 4.6 sebagai berikut:

Tabel 4.6 Perhitungan Nilai *Cost* Iterasi 0

Partikel ke-	Nilai *Cost*
1	0,300046
2	0,390608
3	0,418899
4	0,419898
5	0,506414

4.3.4 Perhitungan Inisialisasi *Personal Best* dan *Global Best*

Pada tahap inisialisasi nilai *personal best* adalah sama seperti nilai partikel pada tahap inisialisasi partikel PSO. Sedangkan nilai *global best* adalah nilai *personal best* terbaik yang memiliki nilai *cost* terkecil. Nilai *personal best* dan *global best* pada iterasi pertama ditunjukkan berturut-turut pada Tabel 4.7 dan Tabel 4.8 sebagai berikut:

Tabel 4.7 Perhitungan Inisialisasi *Personal Best*

Pbest ke-	Nilai *Personal Best*						*Cost*
1	0,308943	0,034917	1	0,661597	0	0,030073	0,300046
	0,308943	0,142334	0,099364	0,672025	0,230565	0	
	0,715447	1	0,798592	0,682105	0,67469	1	
2	0,300813	0	0,131199	0	0	0	0,390608
	0	0	0,079019	0,134405	0	0	
	0,308943	0,034917	1	0,661597	0	0,030073	
3	0,010569	0	0,056085	0,045362	0	0	0,418899
	0,715447	0,123609	0,908038	0,836703	0,067025	0	
	0,054797	0	0,069437	0	0	0	
4	0,715447	0,123609	0,908038	0,836703	0,067025	0	0,419898
	0,14226	0,309023	0	0,22679	0	0,299018	
	0	0	0,079019	0,134405	0	0	
5	1	0,112841	0,178247	1	1	0,236773	0,506414
	0,010569	0	0,056085	0,045362	0	0	
	0,054797	0	0,069437	0	0	0	

Tabel 4.8 Perhitungan Inisialisasi *Global Best*

Pbest ke-	Nilai *Global Best*						*Cost*
1	0,308943	0,034917	1	0,661597	0	0,030073	0,300046
	0,308943	0,142334	0,099364	0,672025	0,230565	0	
	0,715447	1	0,798592	0,682105	0,67469	1	

4.3.5 Perhitungan *Update* Kecepatan dan Posisi

Proses *update* kecepatan dan posisi partikel pada iterasi ke-1 dan selanjutnya dihitung berdasarkan Persamaan 2.4 dan Persamaan 2.5 dengan

menggunakan hasil konversi kecepatan dan posisi. Langkah-langkah menghitung nilai kecepatan dan posisi partikel adalah sebagai berikut:

Langkah 1: menghitung perubahan bobot inersia menggunakan Persamaan 2.1

$$\text{Bobot inersia (w)} = 0{,}9 - \left(\frac{0{,}9 - 0{,}4}{5}\right) * 1 = 0{,}8$$

Langkah 2: menghitung nilai kecepatan partikel menggunakan Persamaan 2.4. Sebagai contoh, menghitung nilai kecepatan partikel ke-1 dimensi ke-1

$$\text{Kecepatan} = (0{,}8 * 0) + (2 * 0{,}653051) * (0{,}308943 - 0{,}308943) + (2 * 0{,}462105) * (0{,}308943 - 0{,}308943) = 0$$

Hasil perhitungan kecepatan partikel secara keseluruhan ditunjukkan pada Tabel 4.9 sebagai berikut:

Tabel 4.9 Perhitungan *Update* Kecepatan Partikel

Kecepatan ke-	Nilai Kecepatan Partikel					
1	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
2	0,007514	0,032271	0,802954	0,611454	0	0,027793
	0,285528	0,131546	0,018803	0,496873	0,21309	0
	0,375695	0,891939	-0,18614	0,018954	0,623554	0,896416
3	0,27576	0,032271	0,872375	0,56953	0	0,027793
	-0,37569	0,017305	-0,74738	-0,1522	0,151145	0
	0,610579	0,924209	0,673892	0,630408	0,623554	0,924209
4	-0,37569	-0,08197	0,084992	-0,16183	-0,06194	0,027793
	0,15405	-0,15406	0,091833	0,41149	0,21309	-0,27636
	0,661223	0,924209	0,665036	0,50619	0,623554	0,924209
5	-0,63868	-0,07202	0,759472	-0,31276	-0,92421	-0,19103
	0,27576	0,131546	0,039999	0,579168	0,21309	0
	0,610579	0,924209	0,673892	0,630408	0,623554	0,924209

Langkah 3: konversi kecepatan partikel. Konversi kecepatan partikel ditunjukkan pada Tabel 4.10 sebagai berikut:

Tabel 4.10 Perhitungan Konversi Kecepatan Partikel Iterasi 1

Kecepatan ke-	Nilai Kecepatan Partikel					
1	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
2	0,005	0,005	0,005	0,005	0	0,005
	0,005	0,005	0,005	0,005	0,005	0
	0,005	0,005	-0,005	0,005	0,005	0,005
3	0,005	0,005	0,005	0,005	0	0,005
	-0,005	0,005	-0,005	-0,005	0,005	0
	0,005	0,005	0,005	0,005	0,005	0,005

4	-0,005	-0,005	0,005	-0,005	-0,005	0,005
	0,005	-0,005	0,005	0,005	0,005	-0,005
	0,005	0,005	0,005	0,005	0,005	0,005
5	-0,005	-0,005	0,005	-0,005	-0,005	-0,005
	0,005	0,005	0,005	0,005	0,005	0
	0,005	0,005	0,005	0,005	0,005	0,005

Langkah 4: menghitung nilai posisi partikel berdasarkan Persamaan 2.5. Sebagai contoh, menghitung nilai posisi partikel ke-1 dimensi ke-1

$$\text{Posisi} = 0 + 0{,}308943 = 0{,}308943$$

Hasil perhitungan posisi partikel secara keseluruhan ditunjukkan pada Tabel 4.11 sebagai berikut:

Tabel 4.11 Perhitungan *Update* Posisi Partikel

Posisi ke-	Nilai Posisi Partikel					
1	0,308943	0,034917	1	0,661597	0	0,030073
	0,308943	0,142334	0,099364	0,672025	0,230565	0
	0,715447	1	0,798592	0,682105	0,67469	1
2	0,305813	0,005	0,136199	0,005	0	0,005
	0,005	0,005	0,084019	0,139405	0,005	0
	0,313943	0,039917	0,995	0,666597	0,005	0,035073
3	0,015569	0,005	0,061085	0,050362	0	0,005
	0,710447	0,128609	0,903038	0,831703	0,072025	0
	0,059797	0,005	0,074437	0,005	0,005	0,005
4	0,710447	0,118609	0,913038	0,831703	0,062025	0,005
	0,14726	0,304023	0,005	0,23179	0,005	0,294018
	0,005	0,005	0,084019	0,139405	0,005	0,005
5	0,995	0,107841	0,183247	0,995	0,995	0,231773
	0,015569	0,005	0,061085	0,050362	0,005	0
	0,059797	0,005	0,074437	0,005	0,005	0,005

Langkah 5: konversi posisi partikel. Konversi posisi partikel ditunjukkan pada Tabel 4.12 sebagai berikut:

Tabel 4.12 Perhitungan Konversi Posisi Partikel Iterasi 1

Posisi ke-	Nilai Posisi Partikel					
1	0,308943	0,034917	1	0,661597	0	0,030073
	0,308943	0,142334	0,099364	0,672025	0,230565	0
	0,715447	1	0,798592	0,682105	0,67469	1
2	0,305813	0,005	0,136199	0,005	0	0,005
	0,005	0,005	0,084019	0,139405	0,005	0
	0,313943	0,039917	0,995	0,666597	0,005	0,035073
3	0,015569	0,005	0,061085	0,050362	0	0,005
	0,710447	0,128609	0,903038	0,831703	0,072025	0

	0,059797	0,005	0,074437	0,005	0,005	0,005
4	0,710447	0,118609	0,913038	0,831703	0,062025	0,005
	0,14726	0,304023	0,005	0,23179	0,005	0,294018
	0,005	0,005	0,084019	0,139405	0,005	0,005
5	0,995	0,107841	0,183247	0,995	0,995	0,231773
	0,015569	0,005	0,061085	0,050362	0,005	0
	0,059797	0,005	0,074437	0,005	0,005	0,005

4.3.6 Perhitungan Nilai *Cost* Iterasi 1

Nilai *cost* pada iterasi ke-1 dihitung berdasarkan nilai posisi partikel yang telah diperoleh pada iterasi ke-0. Langkah-langkah menghitung nilai *cost* adalah sebagai berikut:

Langkah 1: menghitung jarak data dengan pusat *cluster* dan mengelompokkan data ke dalam *cluster* terdekat. Sebagai contoh, menghitung nilai *cost* partikel ke-2

Jarak data ke-1 terhadap pusat *cluster* ke-1

$$\text{Jarak } 1 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}305813)^2 + (0 - 0{,}005)^2 + (0{,}069437 - 0{,}136199)^2 + \\ (0 - 0{,}005)^2 + (0 - 0)^2 + (0 - 0{,}005)^2\end{matrix}}$$
$$= 0{,}259887$$

Jarak data ke-1 terhadap pusat *cluster* ke-2

$$\text{Jarak } 2 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}005)^2 + (0 - 0{,}005)^2 + (0{,}069437 - 0{,}084019)^2 + \\ (0 - 0{,}139405)^2 + (0 - 0{,}005)^2 + (0 - 0)^2\end{matrix}} = 0{,}148916$$

Jarak data ke-1 terhadap pusat *cluster* ke-3

$$\text{Jarak } 3 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}313943)^2 + (0 - 0{,}039917)^2 + (0{,}069437 - 0{,}995)^2 + \\ (0 - 0{,}666597)^2 + (0 - 0{,}005)^2 + (0 - 0{,}035073)^2\end{matrix}}$$
$$= 1{,}170907$$

Hasil perhitungan jarak data dengan pusat *cluster* partikel 2 dan pengelompokkan data ditunjukkan pada Tabel 4.13 sebagai berikut:

Tabel 4.13 Perhitungan Jarak Dan Pengelompokkan Data Iterasi 1

Partikel ke-	Data ke-	Jarak 1	Jarak 2	Jarak 3	*Cluster*
2	1	0,259887	0,148916	1,170907	2
	2	1,085728	1,09814	0,012247	3
	3	1,214023	1,299185	0,458023	3
	4	0,308652	0,098516	1,167209	2
	5	0,337026	0,01118	1,106191	2
	6	1,593436	1,672496	1,511603	3
	7	0,719954	0,667863	0,929955	2
	8	0,522846	0,464115	1,164062	2
	9	1,870907	1,936074	1,581566	3
	10	0,01118	0,330477	1,092485	1

Langkah 2: menghitung rerata setiap *cluster.* Sebagai contoh, menghitung nilai rerata setiap *cluster* terhadap partikel ke-2

$$\text{Rerata } 1 = \frac{0{,}01118}{1} = 0{,}01118$$

$$\text{Rerata } 2 = \frac{0{,}148916 + 0{,}098516 + 0{,}01118 + 0{,}667863 + 0{,}464115}{5} = 0{,}278118$$

$$\text{Rerata } 3 = \frac{0{,}012247 + 0{,}458023 + 1{,}511603 + 1{,}581566}{4} = 0{,}89086$$

Langkah 3: menghitung nilai *cost.* Sebagai contoh, menghitung nilai *cost* partikel ke-2

$$\text{Cost} = \frac{0{,}01118 + 0{,}278118 + 0{,}89086}{3} = 0{,}393386$$

Hasil perhitungan nilai *cost* iterasi ke-1 secara keseluruhan ditunjukkan pada Tabel 4.14 sebagai berikut:

Tabel 4.14 Perhitungan Nilai *Cost* Iterasi 1

Partikel ke-	Nilai *Cost*
1	0,300046
2	0,393386
3	0,418892
4	0,422680
5	0,506754

4.3.7 Perhitungan *Update Personal Best* Dan *Global Best*

Proses *update personal best* pada iterasi ke-1 dan selanjutnya diperoleh dengan membandingkan nilai *cost personal best* sekarang dan nilai *cost personal best* sebelum sedangkan nilai *global best* seperti halnya pada iterasi ke-1 diperoleh dari nilai *personal best* baru yang memiliki nilai *cost* terkecil. Langkah-langkah *update personal best* dan *global best* adalah sebagai berikut:

Langkah 1: membandingkan nilai *cost personal best* sebelum dengan nilai *cost personal best* sekarang. Nilai *personal best* yang baru diperoleh berdasarkan nilai *cost* yang lebih kecil

Partikel ke-	Nilai *Cost* sebelum	Nilai *Cost* Sekarang
1	0,300046	0,300046
2	0,390608	0,393386
3	0,418899	0,418892
4	0,419898	0,422680
5	0,506414	0,506754

Sehingga, pada iterasi ke-1 nilai *personal best* baru adalah sebagai berikut:

Tabel 4.15 Perhitungan *Update Personal Best*

Pbest ke-	Nilai *Personal Best*						*Cost*
1	0,308943	0,034917	1	0,661597	0	0,030073	0,300046

	0,308943	0,142334	0,099364	0,672025	0,230565	0	
	0,715447	1	0,798592	0,682105	0,67469	1	
2	0,300813	0	0,131199	0	0	0	0,390608
	0	0	0,079019	0,134405	0	0	
	0,308943	0,034917	1	0,661597	0	0,030073	
3	0,015569	0,005	0,061085	0,050362	0	0,005	0,418892
	0,710447	0,128609	0,903038	0,831703	0,072025	0	
	0,059797	0,005	0,074437	0,005	0,005	0,005	
4	0,715447	0,123609	0,908038	0,836703	0,067025	0	0,419898
	0,14226	0,309023	0	0,22679	0	0,299018	
	0	0	0,079019	0,134405	0	0	
5	1	0,112841	0,178247	1	1	0,236773	0,506414
	0,010569	0	0,056085	0,045362	0	0	
	0,054797	0	0,069437	0	0	0	

Langkah 2: menentukan nilai *personal best* terbaik yaitu *personal best* dengan nilai *cost* terkecil untuk selanjutnya ditetapkan sebagai *global best* baru

Tabel 4.16 Perhitungan *Update Global Best*

Pbest ke-	Nilai *Global Best*						*Cost*
1	0,308943	0,034917	1	0,661597	0	0,030073	0,300046
	0,308943	0,142334	0,099364	0,672025	0,230565	0	
	0,715447	1	0,798592	0,682105	0,67469	1	

4.3.8 Perhitungan Konversi *Global Best* Ke Pusat *Cluster*

Konversi *global best* adalah menguraikan *global best* yang diperoleh sampai dengan iterasi maksimal ke dalam pusat *cluster* untuk selanjutnya dijadikan sebagai inisialisasi pusat *cluster* awal algoritma *K-Means.* Dalam perhitungan ini, sampai dengan iterasi 3 *global best* diperoleh sebagai berikut:

Pbest ke-	Nilai *Global Best*						*Cost*
1	0,308943	0,034917	1	0,661597	0	0,030073	0,300046
	0,308943	0,142334	0,099364	0,672025	0,230565	0	
	0,715447	1	0,798592	0,682105	0,67469	1	

Sehingga, jika *global best* dikonversi menjadi pusat *cluster* adalah sebagai berikut:

Tabel 4.17 Perhitungan Konversi *Global Best* Ke Pusat *Cluster*

Pusat *cluster* ke-	Nilai Pusat *Cluster*					
1	0,308943	0,034917	1	0,661597	0	0,030073
2	0,308943	0,142334	0,099364	0,672025	0,230565	0
3	0,715447	1	0,798592	0,682105	0,67469	1

4.3.9 Perhitungan Pengelompokkan Data

Proses pengelompkkan data terdiri dari dua tahapan yaitu menghitung jarak data dengan pusat-pusat *cluster* dan mengelompokkan setiap data kedalam *cluster* yang memiliki jarak terdekat. Langkah-langkah pengelompokkan data adalah sebagai berikut:

Langkah 1: menghitung jarak data dengan pusat-pusat *cluster*
Jarak data ke-1 terhadap pusat *cluster* ke-1

$$\text{Jarak } 1 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}308943)^2 + (0 - 0{,}34917)^2 + (0{,}069437 - 1)^2 + \\ (0 - 0{,}66159)^2 + (0 - 0)^2 + (0 - 0{,}030073)^2\end{matrix}} = 1{,}170629$$

Jarak data ke-1 terhadap pusat *cluster* ke-2

$$\text{Jarak } 2 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}308943)^2 + (0 - 0{,}142334)^2 + (0{,}069437 - 0{,}099364)^2 + \\ (0 - 0{,}672025)^2 + (0 - 0{,}230565)^2 + (0 - 0)^2\end{matrix}} = 0{,}74845458$$

Jarak data ke-1 terhadap pusat *cluster* ke-3

$$\text{Jarak } 3 = \sqrt{\begin{matrix}(0{,}054797 - 0{,}715447)^2 + (0 - 1)^2 + (0{,}069437 - 0{,}798592)^2 + \\ (0 - 0{,}682105)^2 + (0 - 0{,}67469)^2 + (0 - 1)^2\end{matrix}} = 1{,},971953$$

Hasil perhitungan jarak data dengan pusat-pusat *cluster* ditunjukkan pada Tabel 4.18 sebagai berikut:

Tabel 4.18 Perhitungan Jarak Data Algoritma *K-Means*

Data ke-	Jarak 1	Jarak 2	Jarak 3
1	1,170629	0,76845458	1,971953
2	0	0,93640604	1,591723
3	0,466506	0,93456614	1,474176
4	1,166992	0,74634164	1,977073
5	1,106213	0,67698883	1,945464
6	1,521891	1,1136929	1,429073
7	0,936406	0	1,608448
8	1,168035	0,63737588	1,611718
9	1,591723	1,60844811	0
10	1,09303	0,72533853	1,880906

Langkah 2: mengelompokkan data ke dalam *cluster* terdekat
Pengelompokan data berdasarkan jarak paling minimal antara data dengan pusat *cluster.* Hasil pengelompokkan data ditunjukkan pada Tabel 4.19 sebagai berikut:

Tabel 4.19 Pengelompokkan Data Algoritma *K-Means*

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air	*Cluster*
1	0,054797	0	0,069437	0	0	0	2
2	0,308943	0,034917	1	0,661597	0	0,030073	1
3	0,715447	0,123609	0,908038	0,836703	0,067025	0	1

4	0,010569	0	0,056085	0,045362	0	0	2
5	0	0	0,079019	0,134405	0	0	2
6	1	0,112841	0,178247	1	1	0,236773	2
7	0,308943	0,142334	0,099364	0,672025	0,230565	0	2
8	0,14226	0,309023	0	0,22679	0	0,299018	2
9	0,715447	1	0,798592	0,682105	0,67469	1	3
10	0,300813	0	0,131199	0	0	0	2

Tabel 4.20 Jumlah Dan Anggota Setiap *Cluster*

Cluster ke	Jumlah Anggota	Anggota *Cluster*
1	2	2, 3
2	7	1, 4, 5, 6, 7, 8, 10
3	1	9

4.3.10 Perhitungan *Update* Pusat *Cluster* Baru

Proses *update* pusat *cluster* dilakukan pada setiap iterasi sampai dengan kondisi berhenti iterasi terpenuhi. Pusat *cluster* baru dihitung berdasarkan Persamaan 2.6 yaitu dengan merata-rata nilai atribut setiap data dalam satu *cluster.* Langkah-langkah menghitung pusat *cluster* baru adalah sebagai berikut:

Langkah 1: menghitung rata-rata nilai atribut dalam satu *cluster.* Sebagai contoh, menghitung pusat *cluster* ke-1 baru

$$\text{Atribut ke} - 1 = \frac{0{,}308943 + 0{,}715447}{2} = 0{,}512195$$

$$\text{Atribut ke} - 2 = \frac{0{,}034917 + 0{,}123609}{2} = 0{,}079263$$

$$\text{Atribut ke} - 3 = \frac{1 + 0{,}908038}{2} = 0{,}954019$$

$$\text{Atribut ke} - 4 = \frac{0{,}661597 + 0{,}836703}{2} = 0{,}74915$$

$$\text{Atribut ke} - 5 = \frac{0 + 0{,}067025}{2} = 0{,}033512$$

$$\text{Atribut ke} - 6 = \frac{0{,}030073 + 0}{2} = 0{,}015036$$

Hasil perhitungan pusat *cluster* baru secara keseluruhan berturut-turut ditunjukkan pada Tabel 4.21, Tabel 4.22, dan Tabel 4.23 sebagai berikut:

Tabel 4.21 Perhitungan Pusat *Cluster* 1 Baru

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air
2	0,308943	0,034917	1	0,661597	0	0,030073
3	0,715447	0,123609	0,908038	0,836703	0,067025	0
Pusat *Cluster* 1	0,512195	0,079263	0,954019	0,74915	0,033512	0,015036

Tabel 4.22 Perhitungan Pusat *Cluster* 2 Baru

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air

1	0,054797	0	0,069437	0	0	0
4	0,010569	0	0,056085	0,045362	0	0
5	0	0	0,079019	0,134405	0	0
6	1	0,112841	0,178247	1	1	0,236773
7	0,308943	0,142334	0,099364	0,672025	0,230565	0
8	0,14226	0,309023	0	0,22679	0	0,299018
10	0,300813	0	0,131199	0	0	0
Pusat *Cluster* 2	0,259626	0,0806	0,087622	0,29694	0,175795	0,076542

Tabel 4.23 Perhitungan Pusat *Cluster* 3 Baru

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air
9	0,715447	1	0,798592	0,682105	0,67469	1
Pusat *Cluster* 3	0,715447	1	0,798592	0,682105	0,67469	1

4.3.11 Perhitungan Cek Kondisi Berhenti 1

Kondisi berhenti pertama terpenuhi jika tidak terdapat perubahan nilai pada pusat-pusat *cluster* sehingga anggota daripada masing-masing *cluster* juga tidak berubah. Sebagai contoh, melakukan cek perubahan *cluster* pada iterasi ke-1 proses *clustering* menggunakan algoritma *K-Means*

Tabel 4.24 Cek Perubahan Pusat *Cluster* 1

	Gaji	PBB	PKB	Listrik	Telepon	Air
Sebelum	0,308943	0,034917	1	0,661597	0	0,030073
Sesudah	0,512195	0,079263	0,954019	0,74915	0,033512	0,015036

Tabel 4.25 Cek Perubahan Pusat *Cluster* 2

	Gaji	PBB	PKB	Listrik	Telepon	Air
Sebelum	0,308943	0,142334	0,099364	0,672025	0,230565	0
Sesudah	0,259626	0,0806	0,087622	0,29694	0,175795	0,076542

Tabel 4.26 Cek Perubahan Pusat *Cluster* 3

	Gaji	PBB	PKB	Listrik	Telepon	Air
Sebelum	0,715447	1	0,798592	0,682105	0,67469	1
Sesudah	0,715447	1	0,798592	0,682105	0,67469	1

Perubahan nilai *cluster* terjadi pada pusat *cluster* 1 dan pusat *cluster* 2 sehingga iterasi proses *clustering* data dilanjutkan. Pada perhitugan manual sebenarnya, iterasi berhenti pada iterasi ke-2 dengan pusat-pusat *cluster* sebagai berikut:

Tabel 4.27 Pusat *Cluster* Akhir Algoritma *K-Means*

Pusat *Cluster*	Gaji	PBB	PKB	Listrik	Telepon	Air
1	0,512195	0,079263	0,954019	0,74915	0,033512	0,015036
2	0,259626	0,0806	0,087622	0,29694	0,175795	0,076542
3	0,715447	1	0,798592	0,682105	0,67469	1

4.3.12 Perhitungan Cek Kondisi Berhenti 2

Kondisi berhenti kedua terpenuhi jika selisih jarak minimal yang dibentuk antara objek data dan pusat *cluster* bernilai kurang dari *threshold*. Nilai *threshold* pada penelitian ini telah ditentukan sebesar 10^{-8}. Sebagai contoh, menghitung perubahan jarak data dengan pusat *cluster* pada iterasi ke-1 dan iterasi ke-2 proses *clustering* menggunakan algoritma *K-Means*

Langkah 1: menghitung jarak minimal setiap data terhadap pusat *cluster*

Data ke-	Iterasi 1			Iterasi 2		
	Jarak 1	Jarak 2	Jarak 3	Jarak 1	Jarak 2	Jarak 3
1	1,170629	0,768455	1,971953	1,249222	0,416795	1,971953
2	0	0,936406	1,591723	0,233253	1,001494	1,591723
3	0,466506	0,934566	1,474176	0,233253	1,091668	1,474176
4	1,166992	0,746342	1,977073	1,249345	0,411794	1,977073
5	1,106213	0,676989	1,945464	1,188912	0,370346	1,945464
6	1,521891	1,113693	1,429073	1,373716	1,325401	1,429073
7	0,936406	0	1,608448	0,90594	0,394877	1,608448
8	1,168035	0,637376	1,611718	1,205994	0,398686	1,611718
9	1,591723	1,608448	0	1,516239	1,67588	0
10	1,09303	0,725339	1,880906	1,136034	0,367461	1,880906

Jarak data dengan pusat *cluster* pada iterasi 1

$$\begin{aligned}\text{Jarak} = {} & (0 \times 1{,}170629 + 1 \times 0{,}768455 + 0 \times 1{,}971953) \\ & + (1 \times 0 + 0 \times 0{,}936406 + 0 \times 1{,}591723) \\ & + (1 \times 0{,}466506 + 0 \times 0{,}934566 + 0 \times 1{,}474176) \\ & + (0 \times 1{,}166992 + 1 \times 0{,}746342 + 0 \times 1{,}977073) \\ & + (0 \times 1{,}106213 + 1 \times 0{,}676989 + 0 \times 1{,}945464) \\ & + (0 \times 1{,}521891 + 1 \times 1{,}113693 + 0 \times 1{,}429073) \\ & + (0 \times 0{,}936406 + 1 \times 0 + 0 \times 1{,}608448) \\ & + (0 \times 1{,}168035 + 1 \times 0{,}637376 + 0 \times 1{,}611718) \\ & + (0 \times 1{,}591723 + 0 \times 1{,}608448 + 1 \times 0) \\ & + (0 \times 1{,}093030 + 1 \times 0{,}725339 + 0 \times 1{,}880906) = 5{,}134698\end{aligned}$$

Jarak data dengan pusat *cluster* pada iterasi 2

$$\begin{aligned}\text{Jarak} = {} & (0 \times 1{,}249222 + 1 \times 0{,}416795 + 0 \times 1{,}971953) \\ & + (1 \times 0{,}233253 + 0 \times 1{,}001494 + 0 \times 1{,}591723) \\ & + (1 \times 0{,}233253 + 0 \times 1{,}091668 + 0 \times 1{,}474176) \\ & + (0 \times 1{,}249345 + 1 \times 0{,}411794 + 0 \times 1{,}977073) \\ & + (0 \times 1{,}188912 + 1 \times 0{,}370346 + 0 \times 1{,}945464) \\ & + (0 \times 1{,}373716 + 1 \times 1{,}325401 + 0 \times 1{,}429073) \\ & + (0 \times 0{,}90594 + 1 \times 0{,}394877 + 0 \times 1{,}608448) \\ & + (0 \times 1{,}205994 + 1 \times 0{,}398686 + 0 \times 1{,}611718) \\ & + (0 \times 1{,}516239 + 0 \times 1{,}67588 + 1 \times 0) \\ & + (0 \times 1{,}136034 + 1 \times 0{,}367461 + 0 \times 1{,}880906) = 4{,}151866\end{aligned}$$

Langkah 2: menghitung selisih jarak (langkah 1) pada iterasi ke - i dengan iterasi ke - i+1. Jika hasil selisih kurang dari nilai *threshold* maka iterasi berhenti dan sebaliknya

$$\text{J} = |5{,}134698 - 4{,}151866| = 0{,}982832$$

Karena nilai J masih lebih dari nilai *threshold,* maka iterasi dilanjutkan.

4.3.13 Perhitungan Cek Kondisi Berhenti 3

Kondisi berhenti ketiga terpenuhi jika jarak maksimal antara perubahan pusat *cluster* bernilai kurang dari *threshold.* Nilai *threshold* pada penelitian ini telah ditentukan sebesar 10^{-8}. Sebagai contoh, menghitung perubahan *cluster* awal dengan pusat *cluster* pada iterasi ke-1 proses *clustering* menggunakan algoritma *K-Means*

Langkah 1: menghitung jarak perubahan pusat *cluster*

Pusat-pusat *cluster* awal

Centroid 1	0,308943	0,034917	1	0,661597	0	0,030073
Centroid 2	0,308943	0,142334	0,099364	0,672025	0,230565	0
Centroid 3	0,715447	1	0,798592	0,682105	0,67469	1

Pusat-pusat *cluster* pada iterasi 1

Centroid 1	0,512195	0,079263	0,954019	0,74915	0,033512	0,015036
Centroid 2	0,259626	0,0806	0,087622	0,29694	0,175795	0,076542
Centroid 3	0,715447	1	0,798592	0,682105	0,67469	1

Jarak pusat cluster 1

$$= \sqrt{\begin{matrix}(0{,}308943 - 0{,}512195)^2 + (0{,}034917 - 0{,}079263)^2 + (1 - 0{,}954019)^2 + \\ (0{,}661597 - 0{,}74915)^2 + (0 - 0{,}033512)^2 + (0{,}033512 - 0{,}015036)^2\end{matrix}} = 0{,}233253$$

Jarak pusat cluster 2

$$= \sqrt{\begin{matrix}(0{,}308943 - 0{,}259626)^2 + (0{,}142334 - 0{,}0806)^2 + (0{,}099364 - 0{,}087622)^2 + \\ (0{,}672025 - 0{,}29694)^2 + (0{,}230565 - 0{,}175795)^2 + (1 - 0{,}076542)^2\end{matrix}}$$

$$= 0{,}394877$$

$$\text{Jarak pusat cluster 3} = \sqrt{\begin{matrix}(0{,}715447 - 0{,}715447)^2 + (1 - 1)^2 + (0{,}798592 - 0{,}7985929)^2 + \\ (0{,}682105 - 0{,}682105)^2 + (0{,}67469 - 0{,}67469)^2 + (1 - 1)^2\end{matrix}}$$

$$= 0$$

Langkah 2: membandingkan jarak maksimal pusat *cluster* dengan *threshold.* Jika jarak maksimal tersebut bernilai kurang dari *threshold* maka iterasi berhenti dan sebaliknya

$$\text{J max} = 0{,}394877$$

Karena nilai J masih lebih dari nilai *threshold,* maka iterasi dilanjutkan.

Hasil akhir pengelompokan data UKT Proporsional menggunakan algoritma HPSOKM ditunjukkan pada Tabel 4.28 sebagai berikut:

Tabel 4.28 Hasil Pengelompokan Data UKT Proporsional

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air	*Cluster*
1	0,054797	0	0,069437	0	0	0	2
2	0,308943	0,034917	1	0,661597	0	0,030073	1
3	0,715447	0,123609	0,908038	0,836703	0,067025	0	1

4	0,010569	0	0,056085	0,045362	0	0	2
5	0	0	0,079019	0,134405	0	0	2
6	1	0,112841	0,178247	1	1	0,236773	2
7	0,308943	0,142334	0,099364	0,672025	0,230565	0	2
8	0,14226	0,309023	0	0,22679	0	0,299018	2
9	0,715447	1	0,798592	0,682105	0,67469	1	3
10	0,300813	0	0,131199	0	0	0	2

4.3.14 Perhitungan Nilai $a(o)$

Perhitungan nilai $a(o)$ merupakan tahap pertama proses analisis *cluster* menggunakan metode *Silhouette Coefficient* yaitu dengan menghitung jarak setiap data terhadap data lain dalam satu *cluster* yang sama. Langkah-langkah menghitung nilai $a(o)$ adalah sebagai berikut:

Langkah 1: menghitung nilai data dengan data lain *cluster* sama. Sebagai contoh, menghitung nilai $a(1)$ anggota *cluster* 1

$$\text{a}(1,4) = \sqrt{\begin{matrix}(0{,}054797-0{,}010569)^2+(0-0)^2+(0{,}069437-0{,}056085)^2+\\(0-0{,}045362)^2+(0-0)^2+(0-0)^2\end{matrix}} = 0{,}064746$$

$$\text{a}(1,5) = \sqrt{\begin{matrix}(0{,}054797-0)^2+(0-0)^2+(0{,}069437-0{,}079019)^2+\\(0-0{,}134405)^2+(0-0)^2+(0-0)^2\end{matrix}} = 0{,}145642$$

$$\text{a}(1,6) = \sqrt{\begin{matrix}(0{,}054797-1)^2+(0-0{,}112841)^2+(0{,}069437-0{,}178247)^2+\\(0-1)^2+(0-1)^2+(0-0{,}236773)^2\end{matrix}} = 1{,}724541$$

$$\text{a}(1,7) = \sqrt{\begin{matrix}(0{,}054797-0{,}308943)^2+(0-0{,}142334)^2+\begin{pmatrix}0{,}069437\\-0{,}099364\end{pmatrix}^2+\\(0-0{,}672025)^2+(0-0{,}230565)^2+(0-0)^2\end{matrix}} = 0{,}768455$$

$$\text{a}(1,8) = \sqrt{\begin{matrix}(0{,}054797-0{,}14226)^2+(0-0{,}309023)^2+(0{,}069437-0)^2+\\(0-0{,}22679)^2+(0-0)^2+(0-0{,}299018)^2\end{matrix}} = 0{,}498881$$

$$\text{a}(1,10) = \sqrt{\begin{matrix}(0{,}054797-0{,}300813)^2+(0-0)^2+(0{,}069437-0{,}131199)^2+\\(0-0)^2+(0-0)^2+(0-0)^2\end{matrix}} = 0{,}25365$$

Langkah 2: menghitung nilai $a(o)$. Nilai $a(o)$ diperoleh dengan merata-rata nilai jarak data 1 dengan data lain dalam satu *cluster* yang sama

$$\text{a}(1) = \frac{0{,}064746+0{,}145642+1{,}724541+0{,}768455+0{,}498881+0{,}25365}{6} = 0{,}575944$$

4.3.15 Perhitungan Nilai $b(o)$

Perhitungan nilai $b(o)$ merupakan tahap kedua proses analisis *cluster* menggunakan metode *Silhouette Coefficient* yaitu dengan menghitung jarak setiap data terhadap data lain berbeda *cluster*. Langkah-langkah menghitung nilai $b(o)$ adalah sebagai berikut:

Langkah 1: menghitung nilai data dengan data lain berbeda *cluster.* Sebagai contoh, menghitung nilai $b(1)$ anggota *cluster* 1 terhadap data pada *cluster* 2 dan 3

$$b(1,2) = \sqrt{\begin{matrix}(0{,}054797 - 0{,}308943)^2 + (0 - 0{,}034917)^2 + (0{,}069437 - 1)^2 + \\ (0 - 0{,}661597)^2 + (0 - 0)^2 + (0 - 0{,}030073)^2\end{matrix}} = 1{,}170629$$

$$b(1,3) = \sqrt{\begin{matrix}(0{,}054797 - 0{,}715447)^2 + (0 - 0{,}123609)^2 + \left(\frac{0{,}069437 -}{0{,}908038}\right)^2 + \\ (0 - 0{,}836703)^2 + (0 - 0{,}067025)^2 + (0 - 0)^2\end{matrix}} = 1{,}363655$$

$$a(1,9) = \sqrt{\begin{matrix}(0{,}054797 - 0{,}300813)^2 + (0 - 0)^2 + (0{,}069437 - 0{,}131199)^2 + \\ (0 - 0)^2 + (0 - 0)^2 + (0 - 0)^2\end{matrix}} = 1{,}971953$$

Langkah 2: merata-rata nilai jarak untuk masing-masing *cluster*

$$\text{Cluster } 1 = \frac{1{,}170629 + 1{,}363655}{2} = 1{,}267142$$

$$\text{Cluster } 3 = \frac{1{,}971953}{1} = 1{,}971953$$

Langkah 3: menghitung nilai $b(o)$. Nilai $b(o)$ diperoleh berdasarkan nilai terkecil dari rata-rata nilai jarak masing-masing *cluster*

$$b(1) = \text{Min}(1{,}267142: 1{,}971953) = 1{,}267142$$

4.3.16 Perhitungan Nilai $s(o)$

Perhitungan nilai $s(o)$ merupakan tahap ketiga proses analisis *cluster* menggunakan metode *Silhouette Coefficient* berdasarkan Persamaan 2.10. Sebagai contoh menghitung nilai $s(1)$ pada data pertama.

$$a(1) = 0{,}575944$$

$$b(1) = 1{,}267142$$

$$s(1) = \frac{1{,}26714 - 0{,}575944}{\text{Max}(1{,}26714\ ;\ 0{,}575944)} = 0{,}545477706$$

Hasil perhitungan nilai $s(o)$ *Silhouette Coefficient* secara keseluruhan ditunjukkan pada Tabel 4.29 sebagai berikut:

Tabel 4.29 Perhitungan Nilai *Silhouette Coefficient*

Data ke-	$a(o)$	$b(o)$	Max$[a(o): b(o)]$	$s(o)$
1	0,575944	1,2671419	1,26714186	0,545477706
2	0,466506	1,1661709	1,166170875	0,599967867
3	0,466506	1,2477239	1,24772387	0,626114532
4	0,569928	1,2669926	1,266992555	0,550172508
5	0,56653	1,2073485	1,207348513	0,53076493
6	1,566007	1,3868151	1,566007196	-0,114426114
7	0,778032	0,9354861	0,935486085	0,168312525
8	0,695485	1,2269315	1,226931467	0,433150635
9	0	1,5329495	1,532949511	1
10	0,623944	1,157916	1,157916049	0,461149615

4.3.17 Perhitungan Nilai *Silhouette Coefficient* Akhir

Perhitungan nilai *Silhouette Coefficient* akhir terdiri dari dua nilai. Nilai pertama adalah nilai *fitness* masing-masing *cluster* dan nilai kedua adalah nilai kualitas hasil *clustering.* Langkah-langkah menghitung nilai *Silhouette Coefficient* akhir adalah sebagai berikut:

Langkah 1: menghitung nilai *fitness* masing-masing *cluster*

$$\text{Fitness cluster } 1 = \frac{0{,}599967867 + 0{,}626114532}{2} = 0{,}613041199$$

$$\text{Fitness cluster } 2 = \frac{0{,}545477706 + 0{,}550172508 + 0{,}53076493 + (-0{,}114426114) + 0{,}168312525 + 0{,}433150635 + 0{,}461149615}{7} = 0{,}367800258$$

$$\text{Fitness cluster } 3 = \frac{1}{1} = 1$$

$$\text{Rata} - \text{rata fitness cluster } = \frac{0{,}613041199 + 0{,}367800258 + 1}{3} = 0{,}660280486$$

Langkah 2: menghitung nilai kualitas *clustering*

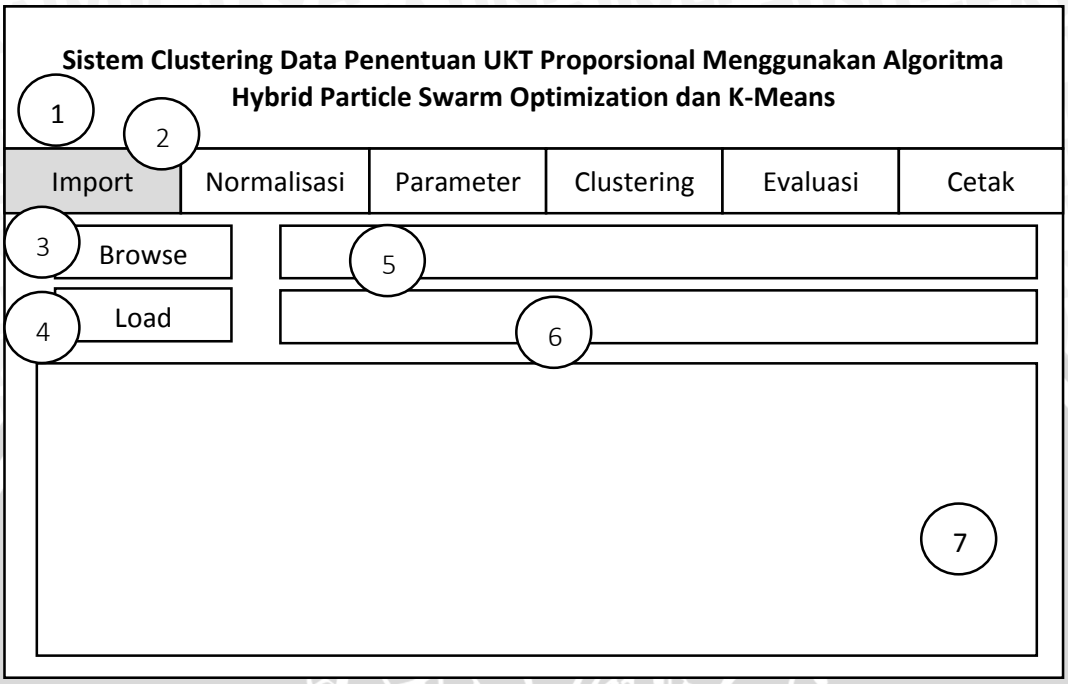
$$\text{Kualitas Clustering} = \frac{0{,}545477706 + 0{,}599967867 + 0{,}626114532 + 0{,}550172508 + 0{,}53076493 + (-0{,}114426114) + 0{,}168312525 + 0{,}433150635 + 1 + 0{,}461149615}{10} = 0{,}48006842$$

4.4 Perancangan Antar Muka

Perancangan antar muka bertujuan untuk mengambarkan keadaan sebenarnya dari implementasi sistem *clustering* data UKT Proporsional yang akan dibangun. Implementasi terdiri dari 6 halaman yaitu *import* data, normalisasi data, parameter, *clustering hybrid* PSO dan *K-Means*, evaluasi, serta cetak hasil.

4.4.1 Perancangan Halaman *Import* Data

Halaman *import* data merupakan halaman untuk melakukan *import* data UKT Proporsional berupa *file* dengan format *file* .xls. Rancangan antarmuka halaman *import* data ditunjukkan pada Gambar 4.25 sebagai berikut:



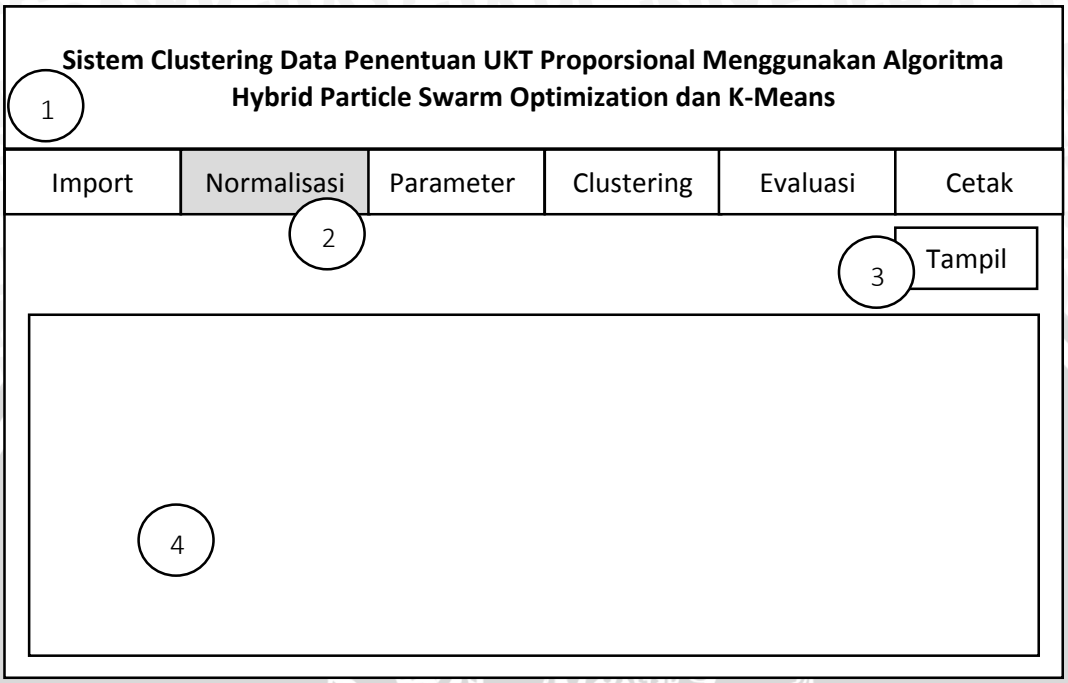
Gambar 4.25 Rancangan Halaman *Import* Data

Keterangan rancangan antarmuka halaman *import* data pada Gambar 4.25 adalah sebagai berikut:

1. *Header* aplikasi
2. Tab menu aplikasi, tab menu berwarna abu-abu menandakan bahwa menu *import* data sedang aktif
3. *Button* untuk *browse file* dengan format .xls
4. *Button* untuk *load* data yang telah di-*import*
5. *Textbox* untuk menampilkan lokasi *file* yang akan di-*load*
6. *Textbox* untuk memasukkan *sheet* data pada file .xls yang akan diproses
7. *Datagridview* untuk menampilkan data yang berhasil di-*load* dalam tabel

4.4.2 Perancangan Halaman Normalisasi Data

Halaman normalisasi data merupakan halaman untuk menampilkan hasil normalisasi data UKT Proporsional yang telah diimport menggunakan rumus pada metode *Min-Max Normalization.* Rancangan antarmuka halaman normalisasi data ditunjukkan pada Gambar 4.26 sebagai berikut:



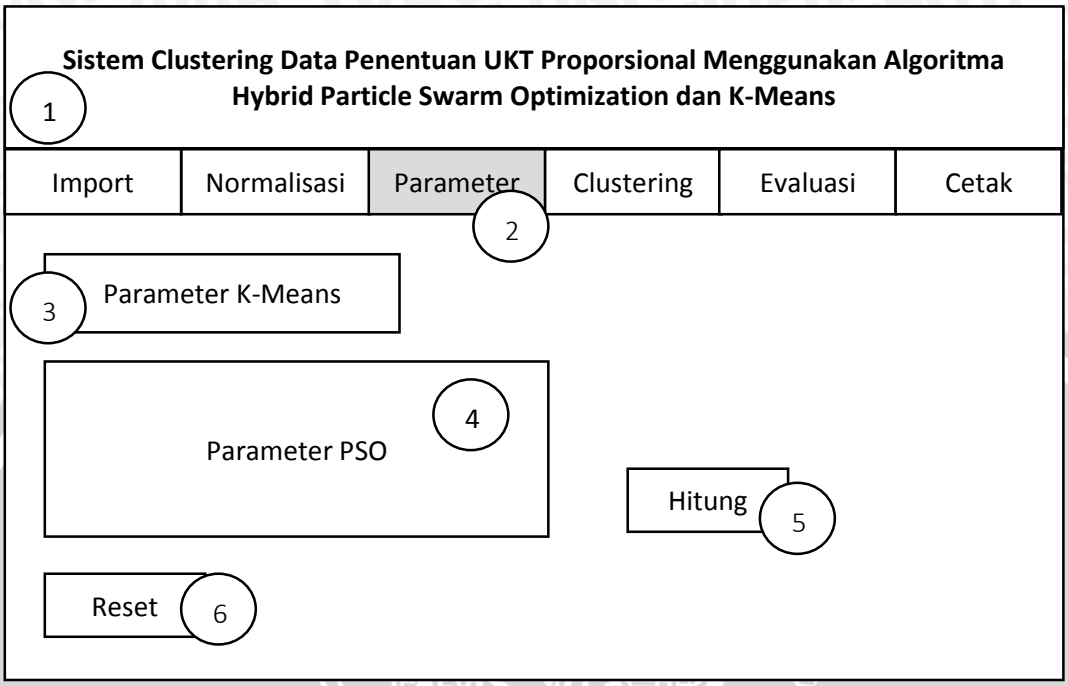
Gambar 4.26 Rancangan Halaman Normalisasi Data

Keterangan rancangan antarmuka halaman normalisasi data pada Gambar 4.26 adalah sebagai berikut:

1. *Header* aplikasi
2. Tab menu aplikasi, tab menu berwarna abu-abu menandakan bahwa menu normalisasi data sedang aktif
3. *Button* untuk menampilkan hasil normalisasi data
4. *Datagridview* untuk menampilkan data hasil normalisasi dalam tabel

4.4.3 Perancangan Halaman Parameter

Halaman paramter merupakan halaman untuk memasukkan parameter sebelum proses perhitungan menggunakan algoritma *Hybrid* PSO dan *K-Means*. Rancangan antarmuka halaman parameter ditunjukkan pada Gambar 4.27 sebagai berikut:



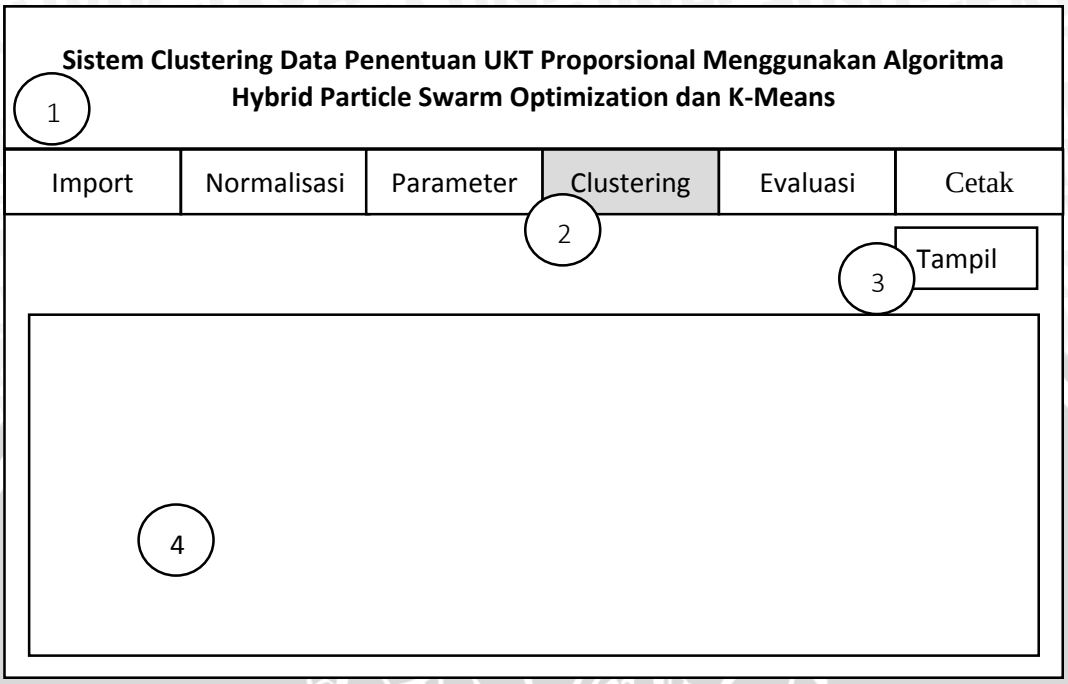
Gambar 4.27 Rancangan Halaman Parameter

Keterangan rancangan antarmuka halaman parameter pada Gambar 4.27 adalah sebagai berikut:

1. *Header* aplikasi
2. Tab menu aplikasi, tab menu berwarna abu-abu menandakan bahwa menu peremeter sedang aktif
3. *Textbox* untuk memasukkan parameter algoritma *K-Means* yaitu jumlah *cluster*
4. *Textbox* untuk memasukkan parameter algoritma PSO yaitu jumlah iterasi, ukuran *swarm,* bobot inersia maks, bobot inersia min, koefisien akselerasi 1, dan koefisien akselerasi 2
5. *Button* untuk memulaia proses *clustering* data UKT Proporsional menggunakan algoritma *Hybrid* PSO dan *K-Means*
6. *Button* untuk me-*reset* semua *textbox* parameter agar kosong kembali

4.4.4 Perancangan Halaman *Clustering* HPSOKM

Halaman *clustering* HPSOKM merupakan halaman untuk menampilkan hasil *clustering* dalam bentuk tabel agar mudah dibaca. Rancangan antarmuka halaman *clustering* HPSOKM ditunjukkan pada Gambar 4.28 sebagai berikut:



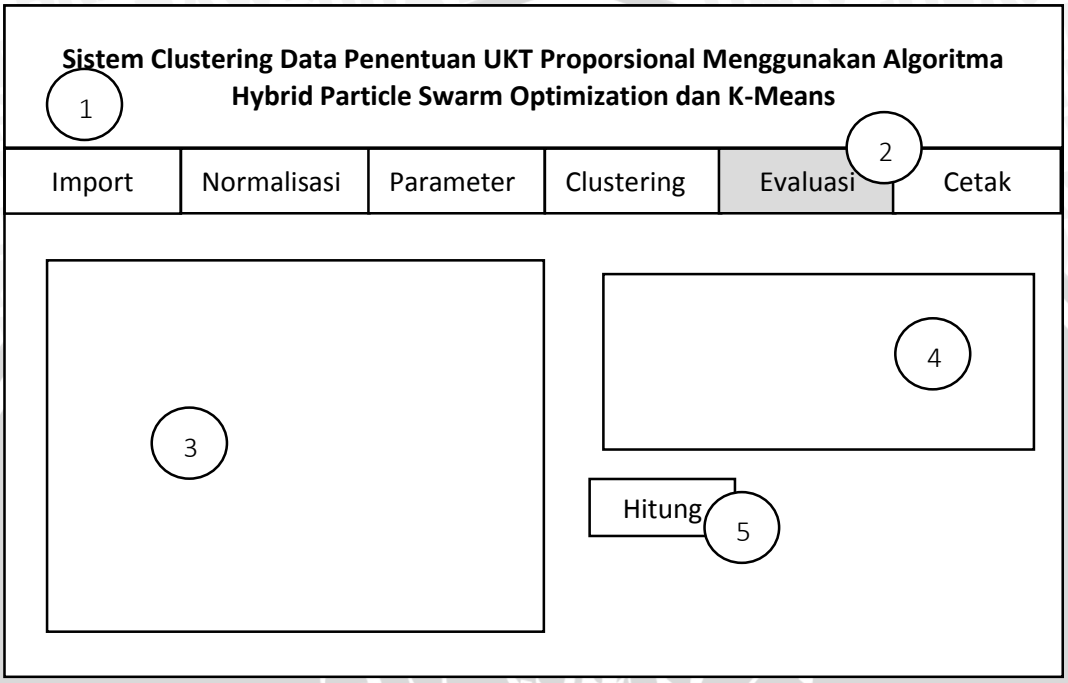
Gambar 4.28 Rancangan Halaman *Clustering* HPSOKM

Keterangan rancangan antarmuka halaman *clustering* HPSOKM pada Gambar 4.28 adalah sebagai berikut:

1. *Header* aplikasi
2. Tab menu aplikasi, tab menu berwarna abu-abu menandakan bahwa menu *clustering Hybrid* PSO dan *K-Means* sedang aktif
3. *Button* untuk menampilkan hasil *clustering* data UKT Proporsional
4. *Datagridview* untuk menampilkan data *clustering* data UKT Proporsional

4.4.5 Perancangan Halaman Evaluasi Kualitas *Clustering*

Halaman evaluasi kualitas *clustering* merupakan halaman untuk menampilkan hasil perhitungan kualitas *clustering* menggunakan algoritma *Hybrid* PSO dan *K-Means.* Evaluasi tersebut dihitung menggunakan metode intrinsik *Silhouette Coefficient.* Rancangan antarmuka halaman evaluasi kualitas *clustering* ditunjukkan pada Gambar 4.29 sebagai berikut:



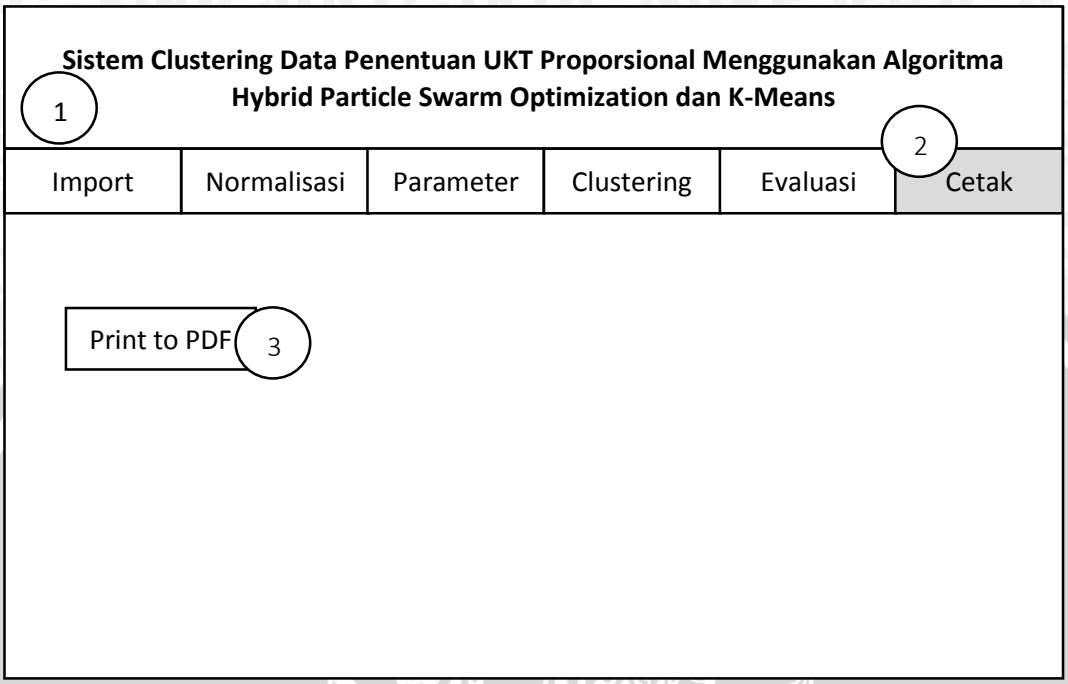
Gambar 4.29 Rancangan Halaman Evaluasi Kualitas *Clustering*

Keterangan rancangan antarmuka halaman evaluasi kualitas *clustering* pada Gambar 4.29 adalah sebagai berikut:

1. *Header* aplikasi
2. Tab menu aplikasi, tab menu berwarna abu-abu menandakan bahwa menu evaluasi kualitas *clustering* sedang aktif
3. *Datagridview* untuk menampilkan nilai *Silhouette Coefficient* setiap objek data
4. *Listbox* untuk menampilkan nilai indeks *Silhouette Coefficient* yaitu nilai *fitness* setiap *cluster,* rata-rata nilai *fitness cluster,* serta nilai kualitas *clustering*
5. *Button* untuk memulai proses evaluasi kualitas *clustering*

4.4.6 Perancangan Halaman Cetak Hasil

Halaman cetak hasil merupakan halaman untuk mencetak hasil data yang telah di kelompokkan ke dalam beberapa *cluster* dalam format .pdf. Rancangan antarmuka halaman evaluasi kualitas *clustering* ditunjukkan pada Gambar 4.30 sebagai berikut:



Gambar 4.30 Rancangan Halaman Cetak Hasil

Keterangan rancangan antarmuka halaman cetak hasil pada Gambar 4.30 adalah sebagai berikut:

1. *Header* aplikasi
2. Tab menu aplikasi, tab menu berwarna abu-abu menandakan bahwa menu cetak hasil sedang aktif
3. *Button* untuk memulai cetak hasil *clustering* data ke dalam format .pdf

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi sistem berdasarkan analisis kebutuhan dan proses perancangan sistem yang telah dibuat. Pembahasan pada bab ini terdiri dari penjelasan implementasi algoritma HPSOKM untuk penyelesaian *clustering* data UKT Proporsional, implementasi metode *Silhouette Coefficient* untuk evaluasi kualitas hasil *clustering* serta implementasi antarmuka sistem.

5.1 Implementasi Algoritma *Hybrid Particle Swarm Optimization* dan *K-Means*

Implementasi algoritma merupakan hasil perancangan sistem *clustering* data UKT Proporsional menggunakan algoritma HPSOKM ke dalam kode program. Sistem *clustering* data ini terdiri dari 12 proses meliputi proses normalisasi data, proses inisialisasi partikel, proses inisialisasi *personal best* dan *global best*, proses inisialisasi kecepatan dan posisi partikel, proses menghitung nilai *cost,* proses *update personal best* dan *global best,* proses *update* kecepatan dan posisi partikel, proses konversi *global best* menjadi pusat *cluster* awal algoritma *K-Means*, proses menghitung jarak, proses pengelompokkan data, proses menghitung pusat *cluster* baru, dan proses cek kondisi berhenti. Sedangkan sistem evaluasi hasil *clustering* terdiri dari 4 proses meliputi proses menghitung jarak antar data, proses menghitung nilai $a(o)$, proses menghitung nilai $b(o)$, proses menghitung nilai $s(o)$ dan proses menghitung nilai *Silhouette Coefficient* akhir.

5.1.1 Implementasi Normalisasi Data

Implementasi normalisasi data merupakan proses awal sebelum proses utama *clustering* data. Proses ini bertujuan untuk menyamakan rentang nilai setiap atribut agar berada pada interval 0 sampai dengan 1. Implementasi normalisasi data ditunjukkan pada Kode Program 5.1 sebagai berikut:

Kode Program 5.1 Implementasi Normalisasi Data

```
1.  public static void Normalisasi(){
2.      int a, b;
3.      double[] max = new double[data.GetLength(1)];
4.      double[] min = new double[data.GetLength(1)];
5.      normalisasi = new double[dataUKT.GetLength(0),
6.  dataUKT.GetLength(1)];
7.      for (a = 0; a < data.GetLength(1); a++){
8.          max_anggota_nd = 0;
9.          min_anggota_nd = 1000000000;
10.         for (b = 0; b < data.GetLength(0); b++){
11.             if (max_anggota_nd < data[b, a]){
12.                 max_anggota_nd = data[b, a];
13.             }
14.             if (min_anggota_nd > data[b, a]){
15.                 min_anggota_nd = data[b, a];
16.             }
17.         }
18.                 max[a] = max_anggota_nd;
19.                 min[a] = min_anggota_nd;
```

```
20.         }
21.         for (a = 0; a < data.GetLength(0); a++){
22.             for (b = 0; b < data.GetLength(1); b++){
23.                 dataUKT[a, b] = (data[a, b] - min[b]) / (max[b] -
24.     min[b]) * ((1 - 0) + 0);
25.                 normalisasi[a, b] = dataUKT[a, b];
26.             }
27.         }
28.     }
```

Penjelasan Kode Program 5.1 tentang implementasi normalisasi data adalah sebagai berikut:

Baris 7 s.d 20 : Proses mencari nilai maksimal dan nilai minimal setiap atribut data

Baris 21 s.d 27 : Proses menghitung nilai normalisasi setiap data menggunakan Persamaan 2.7

5.1.2 Implementasi Inisialisasi Partikel

Implementasi inisialisasi partikel merupakan proses untuk menginisialisasi partikel PSO. Proses ini diawali dengan inisialisasi pusat *cluster* sebanyak jumlah *cluster* secara random dari hasil normalisasi data. Langkah selanjutnya adalah menghitung nilai *cost* kemudian menggabungkan beberapa pusat *cluster* tersebut menjadi satu partikel. Proses tersebut diulang sampai dengan batas ukuran *swarm* terpenuhi. Implementasi inisialisasi partikel ditunjukkan pada Kode Program 5.2 sebagai berikut:

Kode Program 5.2 Implementasi Inisialisasi Partikel

```
1.  public static void Inisialisasi(int jum_cluster, int swarm){
2.      Random autoRand = new Random();
3.      int s, a, b, c = 0;
4.      int isi = 0;
5.      double fit_temp;
6.      int[] cek1 = new int[jum_cluster];
7.      centroid = new double[jum_cluster, dataUKT.GetLength(1)];
8.      cost = new double[swarm];
9.      partikel = new double[swarm, dataUKT.GetLength(1) *
10.     jum_cluster];
11.
12.     int p = 1;
13.     for (s = 0; s < swarm; s++){
14.         Array.Clear(cek1, 0, cek1.Length);
15.         for (a = 0; a < jum_cluster; a++) {
16.             isi = autoRand.Next(dataUKT.GetLength(0));
17.             while (cek1.Contains(isi)){
18.                 isi = autoRand.Next(dataUKT.GetLength(0));
19.             }
20.             cek1[a] = isi;
21.             for (b = 0; b < dataUKT.GetLength(1); b++) {
22.                 centroid[a, b] = dataUKT[isi, b];
23.             }
24.         }
25.         Jarak(jum_cluster);
26.         Kelompok(jum_cluster);
27.         Rerata(jum_cluster);
28.         fit_temp = 0;
29.         for (a = 0; a < jum_cluster; a++) {
30.             fit_temp += rerata2[a];
31.         }
```

```
32.             cost[s] = fit_temp / jum_cluster;
33.             c = 0;
34.             for (a = 0; a < jum_cluster; a++) {
35.                 for (b = 0; b < dataUKT.GetLength(1); b++) {
36.                     partikel[s, c] = centroid[a, b];
37.                     c++;
38.                 }
39.             }
40.             p++;
41.         }
42.     }
```

Penjelasan Kode Program 5.2 tentang implementasi inisialisasi partikel adalah sebagai berikut:

Baris 13 s.d 41	:	Proses membangkitkan partikel sebanyak ukuran *swarm*
Baris 14 s.d 24	:	Proses inisialisasi pusat *cluster* sebanyak jumlah partikel
Baris 17 s.d 19	:	Kondisi pengulanggan agar dalam satu partikel tidak terdapat pusat *cluster* dari data yang sama
Baris 25	:	Pemanggilan fungsi jarak
Baris 26	:	Pemanggilan fungsi kelompok
Baris 27	:	Pemanggilan fungsi rerata
Baris 28 s.d 32	:	Proses menghitung nilai *cost* partikel
Baris 33 s.d 39	:	Proses penggabungan setiap pusat *cluster* (baris 14 s.d 24) menjadi satu partikel

5.1.3 Implementasi Hitung *Cost*

Implementasi hitung *cost* merupakan proses untuk menghitung nilai *cost* setiap partikel. Langkah menghitung *cost* terdiri dari 3 proses yaitu menghitung jarak antara data dengan pusat *cluster*, mengelompokkan data ke dalam *cluster* terdekat, dan menghitung rerata setiap *cluster*. Nilai *cost* diperoleh dengan menjumlahkan nilai rerata setiap *cluster* dan membaginya dengan jumlah *cluster.* Langkah selanjutnya adalah menggabungkan beberapa pusat *cluster* tersebut menjadi satu partikel dalam satu *swarm.* Implementasi hitung *cost* ditunjukkan pada Kode Program 5.3 sebagai berikut:

Kode Program 5.3 Implementasi Hitung *Cost*

```
1.  public static void HitungCost(int jum_cluster, int swarm){
2.      int a, b, c, s;
3.      double fit_temp;
4.      cost_baru = new double[swarm];
5.      int p = 1;
6.      for (s = 0; s < swarm; s++){
7.          c = 0;
8.          for (a = 0; a < jum_cluster; a++){
9.              for (b = 0; b < dataUKT.GetLength(1); b++){
10.                 centroid[a, b] = posisi[s, c];
11.                 c++;
12.             }
13.         }
14.         Jarak(jum_cluster);
15.         Kelompok(jum_cluster);
16.         Rerata(jum_cluster);
17.
18.         fit_temp = 0;
19.         for (a = 0; a < jum_cluster; a++){
```

```
20.                 fit_temp += rerata2[a];
21.             }
22.             cost_baru[s] = fit_temp / jum_cluster;
23.             c = 0;
24.             for (a = 0; a < jum_cluster; a++){
25.                 for (b = 0; b < dataUKT.GetLength(1); b++){
26.                     partikel[s, c] = centroid[a, b];
27.                     c++;
28.                 }
29.             }
30.             p++;
31.         }
32.     }
```

Penjelasan Kode Program 5.3 tentang implementasi perhitung nilai *cost* partikel adalah sebagai berikut:

Baris 6 s.d 31	:	Proses menghitung partikel sebanyak ukuran *swarm*
Baris 7 s.d 13	:	Proses mengguraikan satu posisi partikel menjadi beberapa pusat *cluster*
Baris 14	:	Pemanggilan fungsi jarak
Baris 15	:	Pemanggilan fungsi kelompok
Baris 16	:	Pemanggilan fungsi rerata
Baris 18 s.d 22	:	Proses menghitung nilai *cost* partikel
Baris 23 sd 29	:	Proses penggabungan kembali dari beberapa pusat *cluster* menjadi satu partikel

5.1.3.1 Implementasi Menghitung Jarak Pada PSO

Implementasi menghitung jarak merupakan proses pertama pada serangkaian proses menghitung nilai *cost* partikel. Proses ini bertujuan untuk menghitung jarak data dengan pusat *cluster* menggunakan rumus perhitungan jarak *Euclidean.* Implementasi menghitung jarak pada PSO ditunjukkan pada Kode Program 5.4 sebagai berikut:

Kode Program 5.4 Implementasi Menghitung Jarak Pada PSO

```
1.  public static void Jarak(int jum_cluster){
2.      int a, b, c;
3.      double jarak1 = 0;
4.      jarak2 = new double[dataUKT.GetLength(0), jum_cluster];
5.      for (c = 0; c < jum_cluster; c++){
6.          for (a = 0; a < dataUKT.GetLength(0); a++){
7.              jarak1 = 0;
8.              for (b = 0; b < dataUKT.GetLength(1); b++){
9.                  jarak1 += Math.Pow((dataUKT[a, b] - centroid[c,
10. b]), 2);
11.             }
12.             jarak2[a, c] = Math.Sqrt(jarak1);
13.         }
14.     }
15. }
```

Penjelasan Kode Program 5.4 tentang implementasi menghitung jarak pada PSO menggunakan teori *Euclidean* adalah sebagai berikut:

Baris 5 s.d 14 : Penggulangan menghitung jarak setiap data terhadap masing-masing pusat *cluster.* Perhitungan jarak menggunakan Persamaan 2.2

5.1.3.2 Implementasi Mengelompokkan Data Pada PSO

Implementasi mengelompokkan data merupakan proses kedua pada serangkaian proses menghitung nilai *cost* partikel. Proses ini bertujuan untuk mengelompokkan setiap objek data ke dalam *cluster* yang memiliki jarak terdekat. Implementasi menghitung jarak pada PSO ditunjukkan pada Kode Program 5.5 sebagai berikut:

Kode Program 5.5 Implementasi Mengelompokkan Data Pada PSO

```
1.  public static void Kelompok(int jum_cluster){
2.      int a, b;
3.      double min_anggota_pso = 1000000000;
4.      jum_anggota_pso = new int[jum_cluster];
5.      kel_pso = new int[dataUKT.GetLength(0)];
6.
7.      Array.Clear(jum_anggota_pso, 0, jum_anggota_pso.Length);
8.      for (a = 0; a < dataUKT.GetLength(0); a++){
9.          for (b = 0; b < jum_cluster; b++){
10.             if (min_anggota_pso > jarak2[a, b]){
11.                 min_anggota_pso = jarak2[a, b];
12.                 kel_pso[a] = b + 1;
13.             }
14.         }
15.         for (b = 0; b < jum_cluster; b++){
16.             if (kel_pso[a] == b + 1){
17.                         jum_anggota_pso[b] += 1;
18.             }
19.         }
20.         min_anggota_pso = 1000000000;
21.     }
22. }
```

Penjelasan Kode Program 5.5 tentang implementasi mengelompokkan data pada PSO adalah sebagai berikut:

Baris 8 s.d 21 : Penggulangan untuk mengelompokkan setiap data
Baris 9 s.d 14 : Proses mengelompokkan data ke dalam *cluster* yang memiliki jarak terdekat
Baris 15 s.d 19 : Proses menghitung anggota setiap *cluster*

5.1.3.3 Implementasi Menghitung Rerata Pada PSO

Implementasi menghitung rerata merupakan proses ketiga pada serangkaian proses menghitung nilai *cost* partikel. Proses ini bertujuan untuk menghitung jarak setiap data terhadap pusat *cluster* dan membaginya dengan jumlah anggota setiap *cluster.* Implementasi menghitung rerata pada PSO ditunjukkan pada Kode Program 5.6 sebagai berikut:

Kode Program 5.6 Implementasi Menghitung Rerata Pada PSO

```
1.  public static void Rerata(int jum_cluster) {
2.      int a, b;
3.      double rerata1 = 0;
4.      rerata2 = new double[jum_cluster];
```

```
5.
6.      Array.Clear(rerata2, 0, rerata2.Length);
7.      for (a = 0; a < jum_cluster; a++) {
8.          rerata1 = 0;
9.          for (b = 0; b < dataUKT.GetLength(0); b++) {
10.             if (kel_pso[b] == a + 1) {
11.                 rerata1 += jarak2[b, a];
12.             }
13.         }
14.         if (jum_anggota_pso[a] > 0) {
15.             rerata2[a] = rerata1 / jum_anggota_pso[a];
16.         }
17.         else {
18.             if (jum_anggota_pso[a] == 0) {
19.                 rerata2[a] = 0;
20.             }
21.         }
22.     }
23. }
```

Penjelasan Kode Program 5.6 tentang implementasi menghitung rerata setiap *cluster* adalah sebagai berikut:

Baris 8 s.d 13 : Proses menghitung jarak keseluruhan data setiap *cluster*
Baris 14 s.d 21 : Proses menghitung rerata setiap *cluster* yaitu total jarak (baris 8 s.d 13) dibagi dengan jumlah anggota

5.1.4 Implementasi Inisialisasi Kecepatan dan Posisi Partikel

Implementasi inisialisasi kecepatan dan posisi partikel merupakan proses untuk menginisialisasi nilai kecepatan dan posisi partikel. Pada proses ini atau iterasi pertama, nilai kecepatan awal adalah 0 sedangkan nilai posisi adalah sama seperti pada tahap inisialisasi partikel. Implementasi inisialisasi kecepatan dan posisi partikel ditunjukkan pada Kode Program 5.7 sebagai berikut:

Kode Program 5.7 Implementasi Inisialisasi Kecepatan dan Posisi Partikel

```
1.  public static void InisialisasiKecPos(int jum_cluster, int swarm)
2.  {
3.      int a, b;
4.      kecepatan = new double[swarm, dataUKT.GetLength(1) *
5.  jum_cluster];
6.      posisi = new double[swarm, dataUKT.GetLength(1) *
7.  jum_cluster];
8.
9.      for (a = 0; a < swarm; a++) {
10.         for (b = 0; b < dataUKT.GetLength(1) * jum_cluster; b++)
11. {
12.             kecepatan[a, b] = 0;
13.             posisi[a, b] = partikel[a, b];
14.         }
15.     }
16. }
```

Penjelasan Kode Program 5.7 tentang implementasi kecepatan dan posisi partikel adalah sebagai berikut:

Baris 9 s.d 15 : Proses membangkitkan kecepatan dan posisi partikel sebanyak ukuran *swarm*
Baris 12 : Inisialisasi kecepatan awal adalah 0

Baris 13 : Inisialisasi posisi awal adalah partikel itu sendiri (sama seperti halnya inisialisasi partikel)

5.1.5 Implementasi Inisialisasi *Personal Best* dan *Global Best*

Implementasi inisialisasi *personal best* dan *global best* merupakan proses untuk menginisialisasi nilai *personal best* dan *global best*. Pada proses ini atau iterasi pertama, nilai *personal best* diperoleh dari inisialisasi partikel sementara nilai *global best* diperoleh dari *personal best* berdasarkan nilai *cost* terkecil. Proses implementasi inisialisasi *personal best* dan *global best* ditunjukkan pada Kode Program 5.8 sebagai berikut:

Kode Program 5.8 Implementasi Inisialisasi *pBest* dan *gBest*

```
1.  public static void InisialisasiPGBest(int jum_cluster, int
2.  swarm){
3.      int a, b;
4.      pbest = new double[swarm, dataUKT.GetLength(1) *
5.  jum_cluster];
6.      gbest = new double[dataUKT.GetLength(1) * jum_cluster];
7.      for (a = 0; a < swarm; a++){
8.          for (b = 0; b < dataUKT.GetLength(1) * jum_cluster; b++){
9.              pbest[a, b] = partikel[a, b];
10.         }
11.     }
12.     minValue = cost.Min();
13.     minIndex = cost.ToList().IndexOf(minValue);
14.     for (a = 0; a < dataUKT.GetLength(1) * jum_cluster; a++){
15.         gbest[a] = pbest[minIndex, a];
16.     }
17. }
```

Penjelasan Kode Program 5.8 tentang implementasi inisialisasi *personal best* dan *global best* adalah sebagai berikut:

Baris 17 s.d 11 : Proses inisialisasi *personal best* diperoleh dari hasil inisialisasi partikel awal
Baris 12 : Proses mencari elemen array yaitu elemen dengan nilai terkecil pada array *cost*
Baris 13 : Proses mencari index data dari eleman array (baris 12)
Baris 14 s.d 16 : Proses inisialisasi *global best* diperoleh dari *personal best* dengan nilai *cost* terbaik

5.1.6 Implementasi *Update* Kecepatan dan Posisi Partikel

Implementasi *update* kecepatan dan posisi partikel merupakan proses untuk memperbarui nilai kecepatan dan posisi partikel. Nilai kecepatan baru dihitung berdasarkan rumus Persamaan 2.4 sedangkan nilai posisi baru dihitung berdasarkan rumus Persamaan 2.5. Batasan interval untuk nilai kecepatan adalah antara -0.005 sampai dengan 0.005 sedangkan batasan interval nilai posisi adalah antara 0 sampai dengan 1. Implementasi *update* kecepatan dan posisi partikel ditunjukkan pada Kode Program 5.9 sebagai berikut:

Kode Program 5.9 Implementasi *Update* Kecepatan dan Posisi Partikel

```
1.  public static void UpdateKecPos(int jum_cluster, int swarm,
2.  double w, int c1, int c2, double rand1, double rand2) {
3.      int a, b;
4.      for (a = 0; a < swarm; a++) {
5.          for (b = 0; b < dataUKT.GetLength(1) * jum_cluster; b++)
6.  {
7.              kecepatan[a, b] = (w * kecepatan[a, b]) + ((c1 *
8.  rand1) * (pbest[a, b] - posisi[a, b])) + ((c2 * rand2) *
9.  (gbest[b] - posisi[a, b]));
10.             if (kecepatan[a, b] >= -0.005 && kecepatan[a, b] <=
11. 0.005) {
12.                 kecepatan[a, b] = kecepatan[a, b];
13.             }
14.             else {
15.                 if (kecepatan[a, b] < -0.005) {
16.                     kecepatan[a, b] = -0.005;
17.                 }
18.                 else {
19.                     if (kecepatan[a, b] > 0.005) {
20.                         kecepatan[a, b] = 0.005;
21.                     }
22.                 }
23.             }
24.             posisi[a, b] = posisi[a, b] + kecepatan[a, b];
25.                 if (posisi[a, b] >= 0 && posisi[a, b] <= 1) {
26.                     posisi[a, b] = posisi[a, b];
27.                 }
28.                 else {
29.                     if (posisi[a, b] < 0) {
30.                         posisi[a, b] = 0;
31.                     }
32.                     else {
33.                         if (posisi[a, b] > 1) {
34.                             posisi[a, b] = 1;
35.                         }
36.                     }
37.                 }
38.             }
39.         }
40.     }
```

Penjelasan Kode Program 5.9 tentang implementasi *update* kecepatan dan posisi partikel adalah sebagai berikut:

Baris 4 s.d 39	:	Penggulanagan proses *update* kecepatan dan posisi partikel sebanyak ukuran *swarm*
Baris 7 s.d 9	:	Proses *update* kecepatan menggunakan Persamaan 2.4
Baris 10 s.d 23	:	Proses konversi nilai kecepatan partikel (lihat ketentuan konversi kecepatan partikel)
Baris 24	:	Proses *update* posisi menggunakan Persamaan 2.5
Baris 25 s.d 36	:	Proses konversi nilai posisi partikel (lihat ketentuan
	:	konversi posisi partikel)

5.1.7 Implementasi *Update Personal Best* dan *Global Best*

Implementasi *update personal best* dan *global best* merupakan proses untuk memperbarui nilai *personal best* dan *global best.* Nilai *personal best* baru diperoleh berdasarkan nilai *cost* terbaik yang pernah dicapai tiap-tiap partikel

sampai dengan iterasi tertentu. Sedangkan nilai *global best* diperoleh dari nilai *personal best* baru yang memiliki nilai *cost* terkecil. Implementasi *update personal best* dan *global best* ditunjukkan pada Kode Program 5.10 sebagai berikut:

Kode Program 5.10 Implementasi *Update pBest* dan *gBest*

```
1.  public static void UpdatePGBest(int jum_cluster, int swarm) {
2.      int s, a;
3.      for (s = 0; s < swarm; s++) {
4.          if (cost[s] >= cost_baru[s]) {
5.              for (a = 0; a < dataUKT.GetLength(1) * jum_cluster;
6.  a++) {
7.                  pbest[s, a] = partikel[s, a];
8.                  cost[s] = cost_baru[s];
9.              }
10.         }
11.         else {
12.             if (cost[s] < cost_baru[s]) {
13.                 for (a = 0; a < dataUKT.GetLength(1) *
14.  jum_cluster; a++) {
15.                     pbest[s, a] = pbest[s, a];
16.                     cost[s] = cost[s];
17.                 }
18.             }
19.         }
20.     }
21.     minValue = cost.Min();
22.     minIndex = cost.ToList().IndexOf(minValue);
23.     for (a = 0; a < dataUKT.GetLength(1) * jum_cluster; a++) {
24.         gbest[a] = pbest[minIndex, a];
25.     }
26. }
```

Penjelasan Kode Program 5.10 tentang implementasi *update personal best* dan *global best* adalah sebagai berikut:

Baris 3 s.d 20	:	Penggulangan proses *update personal best* sebanyak ukuran *swarm*
Baris 4 dan 12	:	Proses membandingkan nilai *cost persobal best* sebelum dengan nilai *cost* sekarang
Baris 21	:	Proses mencari elemen array dengan nilai terkecil pada array *cost*
Baris 22	:	Proses mencari index data dari eleman array (baris 21)
Baris 23 s.d 25	:	Proses inisialisasi *global best* diperoleh dari *personal best* dengan nilai *cost* terbaik

5.1.8 Implementasi Penambahan *Random Injection*

Implementasi *random injection* merupakan proses untuk mengganti partikel sejumlah 20% dari ukuran *swarm* dengan partikel baru untuk setiap 10 kali iterasi. Partikel baru tersebut dibangkitkan secara acak seperti halnya pada proses inisialisasi partikel. Implementasi *random injection* ditunjukkan pada Kode Program 5.11 sebagai berikut:

Kode Program 5.11 Implementasi Penambahan *Random Injection*

```
1.  public static void RandomInjection(int jum_cluster, int loop, int
2.  swarm){
```

```
3.      Random autoRand = new Random();
4.      int a, b, c, d;
5.      int ganti2 = 0;
6.      int ganti3 = 0;
7.      int[] cek2 = new int[loop];
8.      int[] cek3 = new int[loop];
9.      Array.Clear(cek2, 0, cek2.Length);
10.     Array.Clear(cek3, 0, cek3.Length);
11.
12.     for (a = 0; a < loop; a++) {
13.         d = 0;
14.         while (cek2.Contains(ganti2)) {
15.             ganti2 = autoRand.Next(posisi.GetLength(0));
16.         }
17.         cek2[a] = ganti2;
18.         for (b = 0; b < jum_cluster; b++) {
19.             while (cek3.Contains(ganti3)) {
20.                 ganti3 = autoRand.Next(normalisasi.GetLength(0));
21.             }
22.             cek3[a] = ganti3;
23.             for (c = 0; c < dataUKT.GetLength(1); c++) {
24.                 posisi[ganti2, d] = normalisasi[ganti3, c];
25.                 d++;
26.             }
27.         }
28.     }
29. }
```

Penjelasan Kode Program 5.11 tentang implementasi penambahan *random injection* adalah sebagai berikut:

Baris 14 s.d 17 : Kondisi pengulanggan agar partikel yang diganti selalu berbeda

Baris 19 s.d 22 : Kondisi pengulanggan agar data normalisasi sebagai pengganti selalu berbeda

Baris 23 s.d 26 : Proses pergantian partikel dengan data normalisasi baru

5.1.9 Implementasi Konversi *Global Best* Ke Pusat *Cluster* Awal

Implementasi konversi *global best* ke pusat *cluster* awal merupakan proses untuk mengkonversi nilai *global best* yang telah diperoleh pada perhitungan *clustering* PSO ke dalam pusat *cluster* awal algoritma *K-Means.* Proses ini sekaligus akan mendekonversi partikel *global best* menjadi pusat *cluster* secara terpisah. Implementasi konversi *global best* ke pusat *cluster* awal *K-Means* ditunjukkan pada Kode Program 5.12 sebagai berikut:

Kode Program 5.12 Implementasi Konversi *gBest* Ke Pusat *Cluster* Awal

```
1.  public static void Konversi(int jum_cluster) {
2.      int a, b, c;
3.      c = 0;
4.      for (a = 0; a < jum_cluster; a++) {
5.          for (b = 0; b < dataUKT.GetLength(1); b++) {
6.              centroid[a, b] = gbest[c];
7.              c++;
8.          }
9.      }
10. }
```

Penjelasan Kode Program 5.12 tentang implementasi konversi *global best* menjadi beberapa pusat *cluster* adalah sebagai berikut:

Baris 4 s.d 9 : Proses penguraian dari hasil optimasi PSO *(global best)* menjadi beberapa pusat *cluster*

5.1.10 Implementasi Menghitung Jarak Pada *K-Means*

Implementasi menghitung jarak merupakan proses untuk menghitung jarak data dengan pusat *cluster* menggunakan teori menghitung jarak *Euclidean.* Implementasi menghitung jarak pada *K-Means* ditunjukkan pada Kode Program 5.13 sebagai berikut:

Kode Program 5.13 Implementasi Menghitung Jarak Pada *K-Means*

```
1.  public static void HitungJarak(int jum_cluster, double[,]
2.  centroid){
3.      double jarak3;
4.      int c, a, b;
5.      jarak4 = new double[dataUKT.GetLength(0), jum_cluster];
6.      for (c = 0; c < jum_cluster; c++){
7.          for (a = 0; a < dataUKT.GetLength(0); a++){
8.              jarak3 = 0;
9.              for (b = 0; b < dataUKT.GetLength(1); b++){
10.                 jarak3 += Math.Pow((dataUKT[a, b] - centroid[c,
11. b]), 2);
12.             }
13.             jarak4[a, c] = Math.Sqrt(jarak3);
14.         }
15.     }
16. }
```

Penjelasan Kode Program 2.13 tentang implementasi menghitung jarak data terhadap pusat *cluster* pada *K-Means* adalah sebagai berikut:

Baris 6 s.d 15 : Penggulangan menghitung jarak setiap data terhadap masing-masing pusat *cluster.* Perhitungan jarak menggunakan Persamaan 2.2

5.1.11 Implementasi Mengelompokkan Data Pada *K-Means*

Implementasi mengelompokkan data merupakan proses untuk mengelompokkan setiap objek data ke dalam *cluster* yang memiliki jarak terdekat. Implementasi mengelompokkan data pada *K-Means* ditunjukkan pada Kode Program 5.14 sebagai berikut:

Kode Program 5.14 Implementasi Mengelompokkan Data Pada *K-Means*

```
1.  public static void KelompokData(int jum_cluster){
2.      int a, b;
3.      double min_anggota_km = 1000000000;
4.      kel_km = new int[dataUKT.GetLength(0)];
5.      jum_anggota_km = new int[jum_cluster];
6.
7.      for (a = 0; a < dataUKT.GetLength(0); a++){
8.          for (b = 0; b < jum_cluster; b++){
9.              if (min_anggota_km > jarak4[a, b]){
10.                 min_anggota_km = jarak4[a, b];
11.                 kel_km[a] = b + 1;
12.             }
```

```
13.             }
14.             for (b = 0; b < jum_cluster; b++){
15.                 if (kel_km[a] == b + 1) {
16.                     jum_anggota_km[b] += 1;
17.                 }
18.             }
19.         min_anggota_km = 1000000000;
20.         }
21.     }
```

Penjelasan Kode Program 5.14 tentang implementasi mengelompokkan data ke dalam *cluster* terdekat pada *K-Means* adalah sebagai berikut:

Baris 7 s.d 20 : Penggulangan untuk mengelompokkan setiap data
Baris 8 s.d 13 : Proses mengelompokkan data ke dalam *cluster* yang memiliki jarak terdekat
Baris 14 s.d 18 : Proses menghitung anggota setiap *cluster*

5.1.12 Implementasi Menghitung Pusat *Cluster* Baru

Implementasi menghitung pusat *cluster* baru merupakan proses untuk memperbarui pusat *cluster* baru pada algoritma *K-Means.* Proses ini dihitung berdasarkan rumus pada Persamaan 2.6 yaitu menjumlahkan rata-rata nilai untuk setiap atribut kemudian membaginya dengan jumlah anggota pada *cluster* pada masing-masing *cluster.* Implementasi menghitung pusat *cluster* baru ditunjukkan pada Kode Program 5.15 sebagai berikut:

Kode Program 5.15 Implementasi Menghitung Pusat *Cluster* Baru

```
1.  public static void HitungCentroidBaru(int jum_cluster) {
2.      int a, b, c;
3.      double cent_temp;
4.      centroid_baru = new double[jum_cluster,
5.  dataUKT.GetLength(1)];
6.      for (a = 0; a < jum_cluster; a++) {
7.          for (b = 0; b < dataUKT.GetLength(1); b++) {
8.              cent_temp = 0;
9.              for (c = 0; c < dataUKT.GetLength(0); c++) {
10.                 if (kel_km[c] == a + 1) {
11.                     cent_temp += dataUKT[c, b];
12.                 }
13.             }
14.             if (jum_anggota_km[a] > 0) {
15.                 centroid_baru[a, b] = cent_temp /
16. jum_anggota_km[a];
17.             }
18.             else {
19.                 if (jum_anggota_km[a] == 0) {
20.                     centroid_baru[a, b] = 0;
21.                     }
22.                 }
23.             }
24.         }
25.     }
```

Penjelasan Kode Program 5.15 tentang implementasi menghitung pusat *cluster* baru adlaah sebagai berikut:

Baris 6 s.d 24 : Pengulanggan *update* pusat *cluster* untuk setiap *cluster*

Baris 7 s.d 13 : Menghitung nilai setiap atribut untuk data yang berada pada satu *cluster* yang sama

Baris 14 s.d 22 : Menghitung rata-rata (baris 7 s.d 13) untuk setiap atribut

5.1.13 Implementasi Cek Kondisi Berhenti 1

Implementasi cek kondisi berhenti 1 merupakan proses untuk memeriksa kelanjutan iterasi pada proses *clustering*. Kondisi berhenti 1 pada proses *clustering* menggunakan algoritma *K-Means* adalah ketika pusat *cluster* tidak mengalami perubahan sehingga anggota setiap *cluster* juga tidak berubah. Implementasi cek kondisi berhenti ditunjukkan pada Kode Program 5.16 sebagai berikut:

Kode Program 5.16 Implementasi Cek Kondisi Berhenti 1

```
1.  public static void CekBerhenti1(int jum_cluster) {
2.      int a, b;
3.      x = 0;
4.      for (a = 0; a < jum_cluster; a++) {
5.          for (b = 0; b < dataUKT.GetLength(1); b++) {
6.              if (centroid[a, b] != centroid_baru[a, b]) {
7.                  x += 1;
8.              }
9.              else {
10.                 x += 0;
11.             }
12.         }
13.     }
14. }
```

Penjelasan Kode Program 5.16 tentang implementasi cek kondisi berhenti 1 pada algoritma *K-Means* adalah sebagai berikut:

Baris 3 s.d 13 : Proses cek perubahan pusat *cluster*. Nilai akhir variabel x bernilai > 0 jika masih terdapat perubahan pusat *cluster* sebaliknya bernilai 0 jika tidak sudah terdapat perubahan pusat *cluster*

5.1.14 Implementasi Cek Kondisi Berhenti 2

Implementasi cek kondisi berhenti 2 merupakan proses untuk memeriksa kelanjutan iterasi pada proses *clustering*. Kondisi berhenti 2 pada proses *clustering* menggunakan algoritma *K-Means* adalah ketika selisih jarak minimal yang dibentuk antara objek data dan pusat *cluster* bernilai kurang dari sama dengan *threshold* Implementasi cek kondisi berhenti ditunjukkan pada Kode Program 5.17 sebagai berikut:

Kode Program 5.17 Implementasi Cek Kondisi Berhenti 2

```
1.  public static void CekBerhenti2(int jum_cluster, double
2.  threshold) {
3.      double min_jarak;
4.      total_jarak_baru = 0;
5.      for (a = 0; a < dataUKT.GetLength(0); a++) {
6.          min_jarak = 1000000000;
7.          for (b = 0; b < jum_cluster; b++) {
8.              if (min_jarak > jarak4[a, b])
9.                  min_jarak = jarak4[a, b];
10.         }
```

```
11.             total_jarak_baru += min_jarak;
12.         }
13.
14.         delta_jarak = Math.Abs(total_jarak - total_jarak_baru);
15.         if (delta_jarak <= threshold) {
16.             y = 1;
17.         }
18.            else {
19.             y = 0;
20.         }
21.         total_jarak = total_jarak_baru;
22.     }
```

Penjelasan Kode Program 5.17 tentang implementasi cek kondisi berhenti 2 pada algoritma *K-Means* adalah sebagai berikut:

Baris 5 s.d 12 : Proses menghitung jarak minimal setiap data dengan pusat *cluster*

Baris 14 : Menghitung selisih jarak minimal pada iterasi ke-*i* dengan : iterasi ke-*i*+1

Baris 15 s.d 21 : Proses cek selisih jarak minimal. Variabel y bernilai 0 jika selisih jarak minimal lebih dari *threshold* sebaliknya bernilai 1 jika selisih jarak minimal kurang dari sama dengan *threshold*

5.1.15 Implementasi Cek Kondisi Berhenti 3

Implementasi cek kondisi berhenti 3 merupakan proses untuk memeriksa kelanjutan iterasi pada proses *clustering*. Kondisi berhenti 3 pada proses *clustering* menggunakan algoritma *K-Means* adalah ketika jarak maksimal antara perubahan pusat *cluster* bernilai kurang dari *threshold*. Implementasi cek kondisi berhenti ditunjukkan pada Kode Program 5.18 sebagai berikut:

Kode Program 5.18 Implementasi Cek Kondisi Berhenti 3

```
1.  public static void CekBerhenti3(int jum_cluster, double
2.  threshold) {
3.      int a, b;
4.      double temp_jarak = 0;
5.      jarak5 = new double[jum_cluster];
6.      for (a = 0; a < jum_cluster; a++) {
7.          temp_jarak = 0;
8.          for (b = 0; b < dataUKT.GetLength(1); b++) {
9.              temp_jarak += Math.Pow((centroid[a, b] -
10. centroid_baru[a, b]), 2);
11.         }
12.         jarak5[a] = Math.Sqrt(temp_jarak);
13.     }
14.     double max_jarak = jarak5.Max();
15.        if (max_jarak <= threshold) {
16.         z = 1;
17.     }
18.     else {
19.         z = 0;
20.     }
21. }
```

Penjelasan Kode Program 5.18 tentang implementasi cek kondisi berhenti 3 pada algoritma *K-Means* adalah sebagai berikut:

Baris 6 s.d 13 : Proses menghitung jarak antara pusat *cluster* ke-*i* dengan pusat *cluster* ke-*i*+1
Baris 14 : Mencari jarak maksimal (baris 6 s.d 13)
Baris 15 s.d 20 : Proses cek jarak maksimal. Variabel z bernilai 0 jika selisih jarak maksimal lebih dari *threshold* sebaliknya bernilai 1 jika selisih jarak minimal kurang dari sama dengan *threshold*

5.2 Implementasi Metode *Silhouette Coefficient*

Implementasi metode *Silhouette Coefficient* merupakan hasil perancangan evaluasi kualitas *cluster* menggunakan metode *Silhouette Coefficient* ke dalam kode program. Evaluasi kualitas *cluster* ini terdiri dari 5 proses meliputi proses menghitung jarak antar data, proses menghitung nilai $a(o)$, proses menghitung nilai $b(o)$, proses menghitung nilai $s(o)$ dan proses menghitung nilai *Silhouette Coefficient* akhir.

5.2.1 Implementasi Menghitung Jarak Antar Data

Implementasi menghitung jarak antar data pada *Silhouette Coefficient* merupakan proses untuk menghitung jarak antar data dalam satu *dataset*. Proses ini bertujuan untuk memudahkan menghitung jarak data baik dalam satu *cluster* maupun berbeda *cluster.* Dalam implementasi ini, telah ditentukan jumlah atribut data adalah 6. Implementasi menghitung jarak antar data ditunjukkan pada Kode Program 5.19 sebagai berikut:

Kode Program 5.19 Implementasi Menghitung Jarak Antar Data

```
1.  public static void HitungJarak(){
2.      int a, b;
3.      SCjarak = new double[dataUKT.GetLength(0),
4.  dataUKT.GetLength(0)];
5.      for (a = 0; a < dataUKT.GetLength(0); a++){
6.          for (b = 0; b < dataUKT.GetLength(0); b++){
7.              SCjarak[a, b] = Math.Sqrt(Math.Pow((dataUKT[a, 0] -
8.  dataUKT[b, 0]), 2) + Math.Pow((dataUKT[a, 1] - dataUKT[b, 1]), 2)
9.  + Math.Pow((dataUKT[a, 2] - dataUKT[b, 2]), 2) +
10. Math.Pow((dataUKT[a, 3] - dataUKT[b, 3]), 2) +
11. Math.Pow((dataUKT[a, 4] - dataUKT[b, 4]), 2) +
12. Math.Pow((dataUKT[a, 5] - dataUKT[b, 5]), 2));
13.         }
14.     }
15. }
```

Penjelasan Kode Program 5.19 tentang implementasi menghitung jarak antar data adalah sebagai berikut:

Baris 5 s.d 14 : Menghitung jarak antar data dalam satu *dataset.* Perhitungan jarak menggunakan Persamaan 2.2

5.2.2 Implementasi Menghitung Nilai $a(o)$

Implementasi menghitung nilai $a(o)$ merupakan proses pertama pada serangkaian proses menghitung nilai *Silhouette Coefficient* setiap objek data. Proses ini bertujuan untuk menghitung jarak antar setiap objek data dalam satu

cluster yang sama. Implementasi menghitung nilai $a(o)$ ditunjukkan pada Kode Program 5.20 sebagai berikut:

Kode Program 5.20 Implementasi Menghitung Nilai $a(o)$

```
1.  public static void HitungSCA(){
2.      int a, b, c, x;
3.      double tempA;
4.      SCA = new double[dataUKT.GetLength(0)];
5.      for (a = 0; a < dataUKT.GetLength(0); a++){
6.          tempA = 0;
7.          x = kel_km[a];
8.          for (b = 0; b < dataUKT.GetLength(0); b++){
9.              if (kel_km[b] == x){
10.                 tempA += SCjarak[a, b];
11.             }
12.         }
13.     c = jum_anggota_km[x - 1];
14.     SCA[a] = tempA / (c - 1);
15.         if (double.IsNaN(SCA[a])){
16.             SCA[a] = 0;
17.         }
18.     }
19. }
```

Penjelasan Kode Program 5.20 tentang implementasi menghitung nilai $a(o)$ pada *Silhouette Coefficient* adalah sebagai berikut:

Baris 5 s.d 12	:	Menghitung jarak antar data dalam satu *cluster* yang sama
Baris 13 s.d 17	:	Kondisi jika sebuah *cluster* memiliki jumlah anggota <= 1 maka nilai $a(o)$ pada data bersangkutan di *set* 0

5.2.3 Implementasi Menghitung Nilai $b(o)$

Implementasi menghitung nilai $b(o)$ merupakan proses kedua pada serangkaian proses menghitung nilai *Silhouette Coefficient* setiap objek data. Proses ini bertujuan untuk menghitung jarak setiap objek data pada *cluster* berbeda. Implementasi menghitung nilai $b(o)$ ditunjukkan pada Kode Program 5.21 sebagai berikut:

Kode Program 5.21 Implementasi Menghitung Nilai $b(o)$

```
1.  public static void HitungSCB(int jum_cluster) {
2.      int a, b, c, d, x, y;
3.      double min;
4.      SCB = new double[dataUKT.GetLength(0)];
5.      tempB = new double[dataUKT.GetLength(0)];
6.      for (a = 0; a < dataUKT.GetLength(0); a++) {
7.          Array.Clear(tempB, 0, tempB.Length);
8.          x = kel_km[a];
9.          y = x - 1;
10.         for (b = 0; b < jum_cluster; b++) {
11.             if (b != y) {
12.                 for (c = 0; c < dataUKT.GetLength(0); c++) {
13.                     if (kel_km[c] != x && kel_km[c] == b + 1) {
14.                     tempB[b] += SCjarak[a, c];
15.                     }
16.                 }
17.             }
18.             tempB[b] = tempB[b] / jum_anggota_km[b];
19.         }
20.         min = 1000000000;
```

```
21.             for (d = 0; d < jum_cluster; d++) {
22.                 if (min > tempB[d] && tempB[d] != 0) {
23.                     min = tempB[d];
24.                 }
25.             }
26.             SCB[a] = min;
27.         }
28.     }
```

Penjelasan Kode Program 5.21 tentang implementasi menghitung nilai $b(o)$ pada *Silhouette Coefficient* adalah sebagai berikut:

Baris 6 s.d 19 : Menghitung jarak antar data pada *cluster* berbeda
Baris 20 s.d 27 : Mencari nilai jarak minimal diantara beberapa *cluster* kemudian menentukan nilai $b(o)$ setiap data

5.2.4 Implementasi Menghitung Nilai $s(o)$

Implementasi menghitung nilai $s(o)$ merupakan proses ketiga pada serangkaian proses menghitung nilai *Silhouette Coefficient* setiap objek data. Proses ini bertujuan untuk menghtiung nilai *Silhouette Coefficient* setiap objek data berdasarkan rumus Persamaan 2.10. Implementasi menghitung nilai $s(o)$ ditunjukkan pada Kode Program 5.22 sebagai berikut:

Kode Program 5.22 Implementasi Menghitung Nilai $s(o)$

```
1.  public static void HitungSCS() {
2.      int a;
3.      double max;
4.      SCS = new double[dataUKT.GetLength(0)];
5.      for (a = 0; a < dataUKT.GetLength(0); a++) {
6.          max = 0;
7.          if (SCA[a] > SCB[a]) {
8.              max = SCA[a];
9.          }
10.         else {
11.             max = SCB[a];
12.         }
13.         SCS[a] = (SCB[a] - SCA[a]) / max;
14.     }
15. }
```

Penjelasan Kode Program 5.22 tentang implementasi menghitung nilai $s(o)$ pada *Silhouette Coefficient* adalah sebagai berikut:

Baris 6 s.d 12 : Mencari nilai jarak maksimal antara nilai $a(o)$ dan nilai $b(o)$
Baris 5 s.d 14 : Menghitung nilai $s(o)$ setiap data

5.2.5 Implementasi Menghitung Nilai *Silhouette Coefficient* Akhir

Implementasi menghitung nilai *Silhouette Coefficient* akhir merupakan proses terakhir untuk menghitung nilai *Silhouette Coefficient* untuk satu kali proses *clustering* data. Proses ini memiliki 2 bagian yaitu nilai nilai *fitness cluster* dan nilai kualitas *clustering.* Nilai *fitness cluster* merupakan nilai yang dihasilkan setiap *cluster* sedangkan nilai kualitas *clustering* merupakan nilai yang dihasilkan keseluruhan data dalam satu *dataset*. Implementasi menghitung nilai *Silhouette Coefficient* ditunjukkan pada Kode Program 5.23 sebagai berikut:

Kode Program 5.23 Implementasi Menghitung Nilai *Silhouette Coefficient* Akhir

```
1.  public static void HitungSC_Akhir(int jum_cluster) {
2.      int a, b;
3.      rerata_SC1 = new double[jum_cluster];
4.      for (a = 0; a < jum_cluster; a++) {
5.          for (b = 0; b < dataUKT.GetLength(0); b++) {
6.              if (kel_km[b] == a + 1) {
7.                  rerata_SC1[a] += SCS[b];
8.              }
9.          }
10.         rerata_SC1[a] = rerata_SC1[a] / jum_anggota_km[a];
11.         SC_Akhir1 += rerata_SC1[a];
12.     }
13.     SC_Akhir1 = SC_Akhir1 / jum_cluster;
14.     for (a = 0; a < dataUKT.GetLength(0); a++) {
15.         SC_Akhir2 += SCS[a];
16.     }
17.     SC_Akhir2 = SC_Akhir2 / dataUKT.GetLength(0);
18. }
```

Penjelasan Kode Program 5.23 tentang implementasi menghitung nilai *Silhouette Coefficient* akhir adalah sebagai berikut:

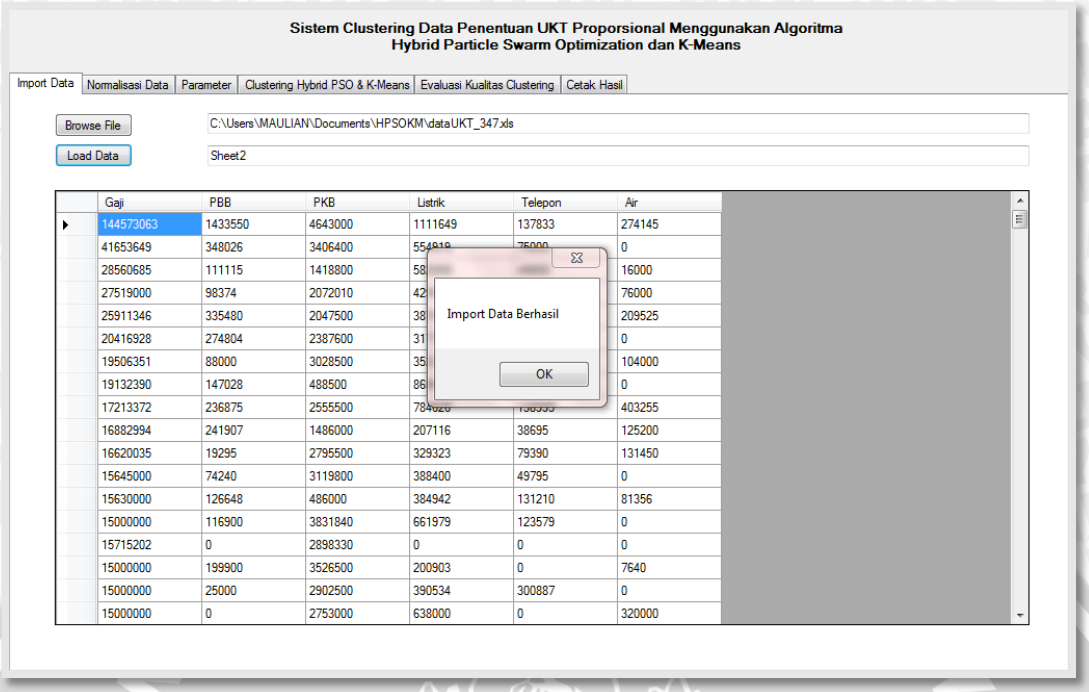
Baris 4 s.d 12 : Menghitung nilai *fitness* setiap *cluster*
Baris 4 s.d 13 : Menghitung rata-rata nilai *fitness cluster*
Baris 14 s.d 17 : Menghitung kualitas hasil *clustering*

5.3 Implementasi Antarmuka

Implementasi antarmuka merupakan hasil perancangan antarmuka yang telah dibuat sebelumnya. Antarmuka sistem *clustering* data penentuan UKT Proporsional terdiri dari 6 halaman utama meliputi halaman *import* data, halaman normalisasi data, halaman parameter, halaman *clustering* HPSOKM*,* halaman evaluasi kualitas *cluster,* serta halaman cetak hasil.

5.3.1 Implementasi Halaman *Import* Data

Implementasi halaman *import* data merupakan halaman utama pada sistem *clustering* data penentuan UKT Proprosional. Halaman ini berfungsi untuk menerima masukan *dataset* UKT Proprosional. Halaman ini juga berfungsi untuk me-*load* data dan menampilkan hasilnya pada *datagridview* dalam bentuk tabel. Implementasi halaman *import* data ditunjukkan pada Gambar 5.1 sebagai berikut:

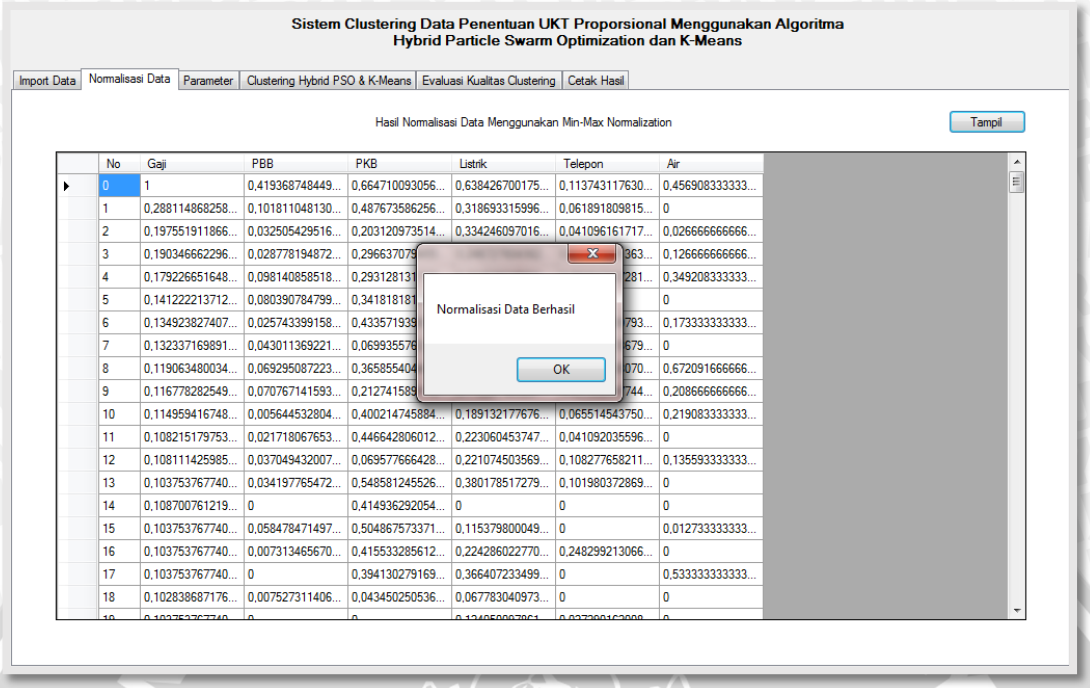


Gambar 5.1 Implementasi Halaman *Import* Data

Gambar 5.1 menjelaskan hasil implementasi halaman *import* data. Proses *import* data diawali dengan pengguna menekan tombol *browse file* bertujuan untuk memasukkan *file* berisi data UKT Proporsional dengan format .xls dari komputer pengguna. Lokasi *file* ditampilkan pada *textbox* pertama. Pada *textbox* kedua, pengguna diwajibkan untuk mengetikkan pada *sheet* berapa data bersangkutan akan diproses, sebagai contoh memproses data pada *sheet* kedua maka pengguna mengetik "Sheet2" (tanpa tanda petik ganda). Langkah selanjutnya, pengguna menekan tombol *load* data bertujuan untuk me-*copy* data dari *file* ke dalam *array* (kebutuhan komputasi) ketika proses *import* data selesai, sistem akan menampilkan *messagebox* bahwa *import* data berhasil. Data yang berhasil diproses oleh sistem ditampilkan dalam bentuk tabel pada *datagridview.*

5.3.2 Implementasi Halaman Normalisasi Data

Implementasi halaman normalisasi data merupakan halaman kedua pada sistem *clustering* data penentuan UKT Proprosional. Halaman ini berfungsi untuk melakukan proses normalisasi data sekaligus menampilkan hasilnya pada *datagridview* dalam bentuk tabel. Implementasi halaman normalisasi data ditunjukkan pada Gambar 5.2 sebagai berikut:

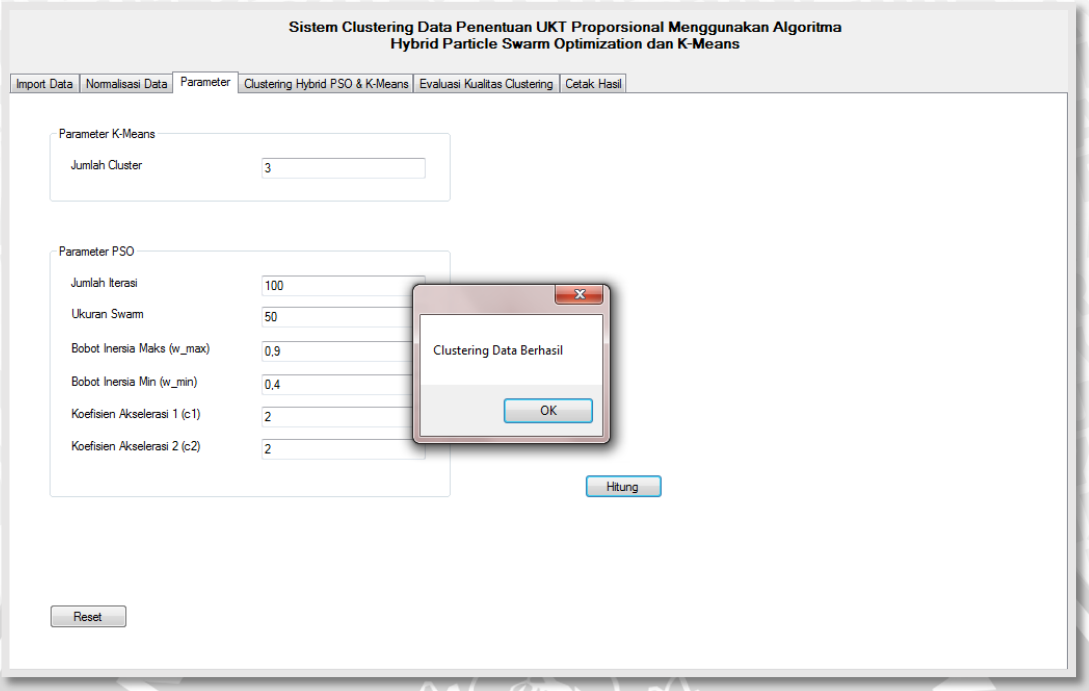


Gambar 5.2 Implementasi Halaman Normalisasi Data

Gambar 5.2 menjelaskan hasil implementasi halaman normalisasi data. Proses normalisasi data diawali dengan pengguna menekan tombol tampil yaitu sistem mulai melakukan perhitungan normlaisasi data menggunkana metode *Min-Max Normalization* sekaligus memasukkan hasil normalisasi data ke dalam basis data. Proses ini membutuhkan waktu beberapa saat tergantung pada jumlah data yang akan diproses sehingga semakin banyak data maka proses komputasi juga semakin lama. Ketika proses nomalisasi data selesai, sistem akan menampilkan *messagebox* bahwa normalisasi data berhasil. Hasil normalisasi data UKT Proporsional ditampilkan dalam bentuk tabel pada *datagridview.*

5.3.3 Implemenatsi Halaman Parameter

Implementasi halaman parameter merupakan halaman ketiga pada sistem *clustering* data penentuan UKT Proprosional. Halaman ini berfungsi untuk menerima masukan parameter yang diperlukan oleh sistem untuk melakukan *clustering* data penentuan UKT Proporsional. Parameter yang dibutuhkan antara lain jumlah *cluster,* jumlah iterasi, ukuran *swarm,* bobot inersia maksimal, bobot inersia minimal, koefisien akselerasi 1, dan koefisien akselerasi 2. Implementasi halaman parameter ditunjukkan pada Gambar 5.3 sebagai berikut:

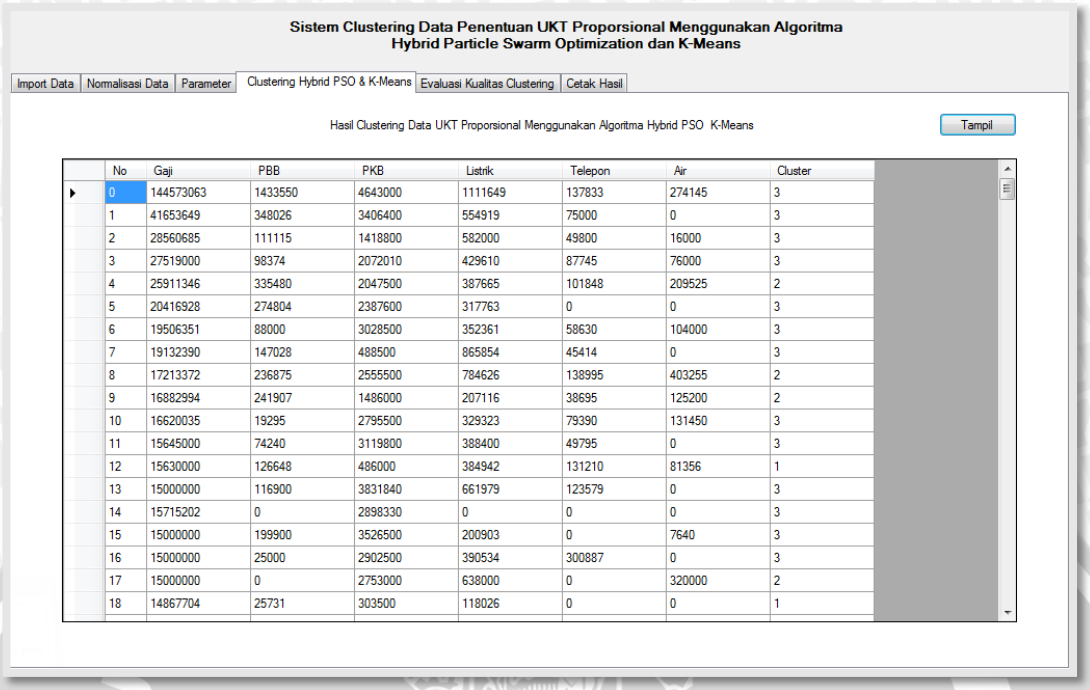


Gambar 5.3 Implementasi Halaman Parameter

Gambar 5.3 menjelaskan hasil implementasi halaman parameter. Pada halaman ini, pengguna diwajibkan mengisi sejumlah parameter yang dibutuhkan guna kelancaran proses *clustering* data. Parameter tersebut antara lain: jumlah *cluster* (bertipe data *integer*: 2, 3, 4, 5, 6, ..), jumlah iterasi (bertipe data *integer*: 10, 30, 50, 100, ..), ukuran *swarm* (bertipe data *integer*: 20, 40, 60, 100, ..), bobot inersia maksimal (bertipe data *double*: bilangan antara 0 sampai dengan 1), bobot inersia minimal (bertipe data *double*: bilangan antara 0 sampai dengan 1), koefisien akselerasi 1 (bertipe data *double*: 0.5, 1, 1.5, 2, ..), koefisien akselerasi 2 (bertipe data *double*: 0.5, 1, 1.5, 2, ..). Tanda yang digunakan untuk menyatakan pecahan adalah menggunakan koma (sebagai contoh: 0,5 bukan 0.5). Setelah semua *textbox* parameter diisi, pengguna menekan tombol hitung bertujuan untuk memulai proses *clustering* data. Namun, jika terdapat kesalahan dalam pengisian parameter, pengguna dapat menekan tombol *reset* untuk mengembalikan kondisi *default* semua *textbox*. Lamanya proses *clustering* data bergantung pada beberapa faktor diantaranya: jumlah data yang diproses, jumlah *cluster,* jumlah iterasi, dan jumlah partikel atau ukuran *swarm.* Hasil *clustering* data disimpan ke dalam basis data dan ketika proses *clustering* selesai, maka sistem akan menampilkan *messagebox* bahwa proses *clustering* data berhasil.

5.3.4 Implementasi Halaman *Clustering* HPSOKM

Implementasi halaman *clustering* HPSOKM merupakan halaman keempat pada sistem *clustering* data penentuan UKT Proprosional. Halaman ini berfungsi untuk menampilkan hasil *clustering* data UKTP Proporsional pada *datagridview* dalam bentuk tabel. Implementasi halaman *clustering* HPSOKM ditunjukkan pada Gambar 5.4 sebagai berikut:



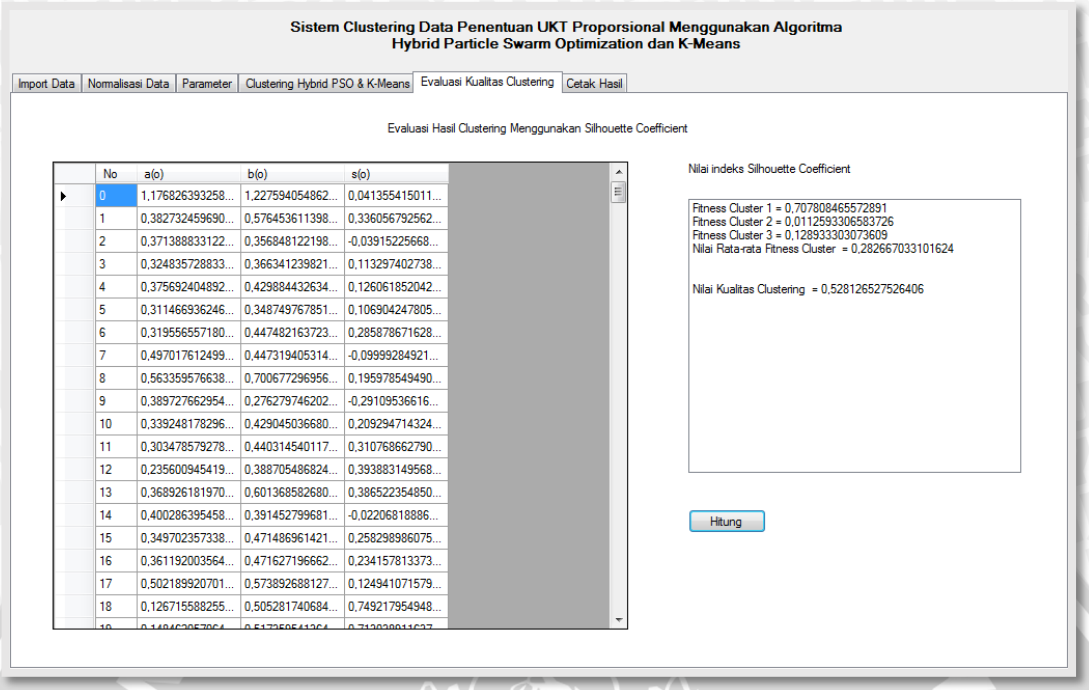
No	Gaji	PBB	PKB	Listrik	Telepon	Air	Cluster
0	144573063	1433550	4643000	1111649	137833	274145	3
1	41653649	348026	3406400	554919	75000	0	3
2	28560685	111115	1418800	582000	49800	16000	3
3	27519000	98374	2072010	429610	87745	76000	3
4	25911346	335480	2047500	387665	101848	209525	2
5	20416928	274804	2387600	317763	0	0	3
6	19506351	88000	3028500	352361	58630	104000	3
7	19132390	147028	488500	865854	45414	0	3
8	17213372	236875	2555500	784626	138995	403255	2
9	16882994	241907	1486000	207116	38695	125200	2
10	16620035	19295	2795500	329323	79390	131450	3
11	15645000	74240	3119800	388400	49795	0	3
12	15630000	126648	486000	384942	131210	81356	1
13	15000000	116900	3831840	661979	123579	0	3
14	15715202	0	2898330	0	0	0	3
15	15000000	199900	3526500	200903	0	7640	3
16	15000000	25000	2902500	390534	300887	0	3
17	15000000	0	2753000	638000	0	320000	2
18	14867704	25731	303500	118026	0	0	1

Gambar 5.4 Implementasi Halaman *Clustering Hybrid* HPSOKM

Gambar 5.4 menjelaskan hasil implementasi halaman *clustering Hybrid* HPSOKM. Pada halaman ini, sistem akan menampilkan hasil *clustering* data UKT Proporsional menggunakan algoritma HPSOKM dari basis data. Pengguna hanya perlu menekan tombol tampil dan secara data akan ditampilkan oleh sistem dalam bentuk tabel pada *datagridview. Index* data dimulai dari 0 (mengikuti aturan *array)* namun pada keadaan sebenarnya jumlah data adalah tetap. Ketika pengguna menekan *header* tertentu, maka tampilan data akan tersortir (*ascending* atau *descending*). Sebagai contoh, pengguna menekan *header cluster* maka tampilan data akan menjadi terurut berdasarkan *cluster* terkecil atau dari *cluster* terbesar.

5.3.5 Implementasi Halaman Evaluasi Kualitas *Clustering*

Implementasi halaman evaluasi kualitas *clustering* merupakan halaman kelima pada sistem *clustering* data penentuan UKT Proprosional. Halaman ini berfungsi untuk melakukan evaluasi tekait hasil *clustering* data menggunakan metode *Silhouette Coefficient*. Hasil evaluasi terdiri dari 3 bagian diantaranya: nilai *Silhouette Coefficient* setiap data, nilai *fitness* setiap *cluster*, dan nilai kualitas *clustering.* Implementasi halaman evaluasi kualitas *clustering* ditunjukkan pada Gambar 5.5 sebagai berikut:

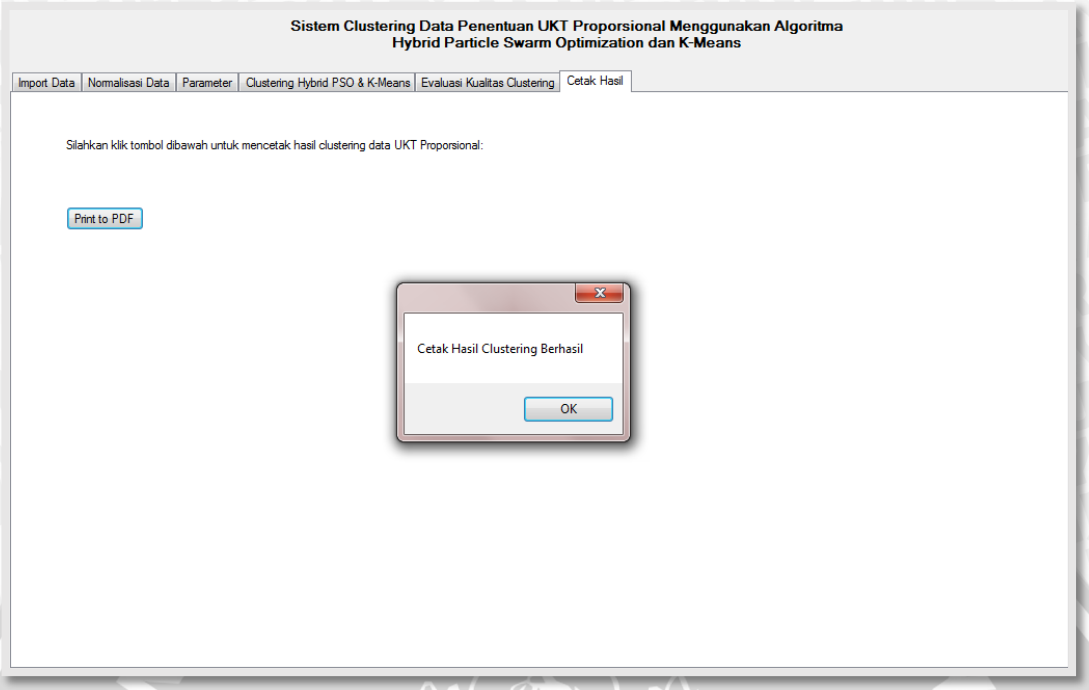


Gambar 5.5 Implementasi Halaman Evaluasi Kualitas *Clustering*

Gambar 5.5 menjelaskan hasil implementasi evaluasi kualitas *clustering.* Proses evaluasi diawali dengan pengguna menekan tombol hitung dan ketika proses tersebut selesai, maka sistem menampilkan *messagebox* bahwa proses evaluasi kualitas *clustering* selesai. Lamanya proses evaluasi bergantung pada beberapa faktor meliputi: jumlah data, jumlah *cluster* dan proses *insert* nilai *Silhouette Coefficient* ke dalam basis data. Selanjutnya, setelah proses evaluasi selesai dan *messagebox* muncul, pengguna diwajibkan menekan "OK" pada *messagebox* agar dapat melihat hasil evaluasi *clustering* data. Nilai *Silhouette Coefficient* setiap data (nilai $a(o)$ nilai $b(o)$, nilai $s(o)$) ditampilkan dalam bentuk tabel pada *datagridview.* Pada *listbox* sebelah kanan, sistem menampilkan nilai *fitness* setiap *cluster,* nilai rata-rata fitness *cluster,* dan nilai kualitas *clustering*.

5.3.6 Implementasi Halaman Cetak Hasil

Implementasi halaman cetak hasil merupakan halaman keenam pada sistem *clustering* data penentuan UKT Proprosional. Halaman ini berfungsi untuk mencetak hasil *clustering* data UKT Proporsional dalam format file .pdf. File .pdf tersebut terletak pada direktori *bin/debug*. Implementasi halaman cetak hasil ditunjukkan pada Gambar 5.6 sebagai berikut:



Gambar 5.6 Implementasi Halaman Cetak Hasil

Gambar 5.6 menjelaskan hasil implementasi halaman cetak hasil. Pada halaman ini, pengguna hanya perlu menekan tombol *print to pdf* untuk mencetak hasil *clustering* data selanjutnya ke dalam format *file* .pdf. Setelah proses cetak hasil selesai, maka sistem menampilkan *messagebox* bahwa proses cetak hasil *clustering* berhasil. Hasil pengelompokan data dalam format *file* .pdf, selanjutnya akan disimpan didalam sebuah direktori *HPSOKM/bin/debug/HasilClustering.pdf*. *File* tersebut akan selalu diperbarui untuk setiap kali proses *clustering* data UKT Proporsional sehingga tidak perlu dihapus secara manual karena tidak akan membentuk *file* baru.

BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab ini menjelaskan hasil pengujian dan pembahasan dari sistem *clustering* data penentuan UKT Proprosional menggunakan algoritma HPSOKM. Pengujian yang dilakukan meliputi: pengujian interval kecepatan partikel, pengujian pengaruh *random injection,* pengujian parameter PSO, pengujian jumlah *cluster,* dan pengujian perbandingan algoritma.

6.1 Pengujian Interval Kecepatan Partikel

Pengujian interval kecepatan partikel bertujuan untuk mengetahui interval kecepatan partikel yang sesuai sehingga dapat menghasilkan solusi penyelesaian yang optimum. Interval kecepatan dievaluasi berdasarkan kesesuaian penentuan pusat *cluster,* rata-rata nilai *cost* terbaik dan variasi partikel dalam *swarm*. PSO merupakan algoritma stokastis sehingga akan menghasilkan hasil berbeda setiap kali program dijalankan (Mahmudy, 2015), karenanya untuk memperoleh rata-rata nilai secara keseluruhan, dilakukan percobaan sebanyak 10 kali untuk setiap interval kecepatan partikel.

Evaluasi pertama (Evaluasi 1) yang harus terpenuhi adalah dalam satu partikel, nilai dimensi antar pusat *cluster* tidak boleh kembar identik. Jika pada satu atau dua dimesi pusat *cluster* nilainya sama, maka diperbolehkan asalkan tidak sama secara keseluruhan. Notasi X menyatakan kualitas solusi tidak memenuhi aturan penentuan pusat *cluster* dan sebaliknya notasi V menyatakan kualitas solusi memenuhi aturan penentuan pusat *cluster*. Evaluasi kedua (Evaluasi 2) yaitu memeriksa tingkat variasi partikel dalam *swarm.* Evaluasi kedua digunakan sebagai bahan pertimbangan setelah interval kecepatan memenuhi evaluasi pertama. Sebagai contoh, terdapat *dataset* yang akan dikelompokkan ke dalam tiga *cluster* (*cluster* A, *cluster* B, *cluster* C) dengan dua *cluster* (*cluster* A dan *cluster* B) memiliki titik pusat bernilai sama. Tanpa memperhatikan nilai *cost* yang dihasilkan partikel, sudah dapat dikatakan bahwa hasil pengelompokkan tidak akan berjalan optimal. Hal ini dikarenakan jarak data dengan *cluster* A dan *cluster* B adalah sama sehingga seolah-olah pengelompokkan hanya melibatkan dua *cluster.*

Pada percobaan ini, besar interval kecepatan adalah 50% sampai dengan 0,005% dari interval posisi partikel yang ditunjukkan pada Tabel 6.1 sampai Tabel 6.5 sebagai berikut:

Tabel 6.1 Hasil Pengujian Interval Kecepatan Partikel 50%

Interval Kecepatan		Uji coba ke-	Nilai *Cost* Terbaik	Evaluasi 1	Evaluasi 2
Min	Max				
-5.10^{-1}	5.10^{-1}	1	0,086383153	X	10%
		2	0,087774396	X	10%
		3	0,083951376	X	10%
		4	0,083944753	X	10%
		5	0,083952669	X	10%
		6	0,083951427	X	10%

		7	0,083938941	X	10%
		8	0,087774396	X	10%
		9	0,083904715	X	10%
		10	0,087774396	X	10%
Rata-rata			0,085335022	X	10%

Tabel 6.2 Hasil Pengujian Interval Kecepatan Partikel 5%

Interval Kecepatan		Uji coba ke-	Nilai *Cost* Terbaik	Evaluasi 1	Evaluasi 2
Min	Max				
-5.10^{-2}	5.10^{-2}	1	0,083951357	X	10%
		2	0,085280738	X	10%
		3	0,083951357	X	30%
		4	0,083951544	X	10%
		5	0,081374122	V	10%
		6	0,087774396	X	10%
		7	0,081467206	V	10%
		8	0,081212198	V	10%
		9	0,087774396	X	10%
		10	0,081302369	V	10%
Rata-rata			0,083803968	X	12%

Tabel 6.3 Hasil Pengujian Interval Kecepatan Partikel 0,5%

Interval Kecepatan		Uji coba ke-	Nilai *Cost* Terbaik	Evaluasi 1	Evaluasi 2
Min	Max				
-5.10^{-3}	5.10^{-3}	1	0,108010239	V	30%
		2	0,109610842	V	10%
		3	0,095518829	V	10%
		4	0,112545618	V	20%
		5	0,093495344	V	10%
		6	0,096139423	V	10%
		7	0,087986477	V	10%
		8	0,106552981	V	20%
		9	0,121375839	V	10%
		10	0,091601895	V	20%
Rata-rata			0,102283749	V	15%

Tabel 6.4 Hasil Pengujian Interval Kecepatan Partikel 0,05%

Interval Kecepatan		Uji coba ke-	Nilai *Cost* Terbaik	Evaluasi 1	Evaluasi 2
Min	Max				
-5.10^{-4}	5.10^{-4}	1	0,130124988	V	60%
		2	0,148211338	V	70%
		3	0,155979443	V	100%
		4	0,151870871	V	80%
		5	0,124364353	V	50%
		6	0,184002323	V	80%
		7	0,147559691	V	60%

		8	0,184514383	V	80%
		9	0,163348113	V	70%
		10	0,137683281	V	70%
Rata-rata			0,152765879	V	72%

Tabel 6.5 Hasil Pengujian Interval Kecepatan Partikel 0,005%

Interval Kecepatan		Uji coba ke-	Nilai *Cost* Terbaik	Evaluasi 1	Evaluasi 2
Min	Max				
-5.10^{-5}	5.10^{-5}	1	0,181921942	V	100%
		2	0,127818923	V	100%
		3	0,176692291	V	100%
		4	0,182693023	V	100%
		5	0,204295257	V	100%
		6	0,189049339	V	100%
		7	0,198576004	V	100%
		8	0,153045126	V	100%
		9	0,155265485	V	100%
		10	0,175692636	V	100%
Rata-rata			0,174505003	V	100%

Berdasarkan hasil pengujian interval kecepatan partikel diatas, maka diperoleh rata-rata nilai *cost* terbaik untuk setiap interval kecepatan partikel yang ditunjukkan pada Tabel 6.6 sebagai berikut:

Tabel 6.6 Hasil Pengujian Interval Kecepatan Partikel

Interval Kecepatan		Rata-rata *Cost* Terbaik	Rata-rata Evaluasi 1	Rata-rata Evaluasi 2
Min	Max			
-5.10^{-1}	5.10^{-1}	0,085335022	X	10%
-5.10^{-2}	5.10^{-2}	0,083803968	X	12%
-5.10^{-3}	5.10^{-3}	0,102283749	V	15%
-5.10^{-4}	5.10^{-4}	0,152765879	V	72%
-5.10^{-5}	5.10^{-5}	0,174505003	V	100%

Kecepatan partikel berpengaruh pada kinerja PSO dalam menemukan solusi optimum. Jika kecepatan terlalu tinggi, partikel akan berpindah tidak menentu dan PSO akan terlalu cepat dalam memutuskan suatu partikel sebagai *global best* atau solusi penyelesaian tanpa cukup mengeksploitasi dimensi ruang pencarian. Dampaknya, partikel-partikel tersebut akan keluar dari batas ruang dimensi pencarian dan kemungkinan terjebak pada lokal optimum semakin meningkat. Sebaliknya, jika kecepatan terlalu kecil maka pergerakan partikel akan terbatas sehingga menyebabkan partikel akan sulit untuk mencapai konvergensi (Jordehi., et al, 2013).

Berdasarkan hasil pengujian interval kecepatan partikel pada Tabel 6.6, maka diperoleh interval kecepatan terbaik adalah interval dengan kecepatan maksimal 0,005 dan kecepatan minimal -0,005. Interval tersebut memenuhi aturan penentuan pusat *cluster* ditunjukkan dengan nilai evaluasi 1 terpenuhi

sekaligus memiliki rata-rata nilai *cost* yang cukup rendah. Namun, interval dengan kecepatan maksimal 0,005 dan kecepatan minimal -0,005 tersebut memiliki nilai evaluasi 2 hanya 15% yang berarti tingkat konvergensi partikel dalam *swarm* cukup tinggi. Pada pengujian selanjutnya akan dibahas tentang metode untuk penanganan konvergensi dini.

6.2 Pengujian Pengaruh *Random Injection*

Random injection merupakan salah satu metode untuk penanganan konvergensi dini. Pengujian *random injection* (RI) bertujuan untuk mengetahui pengaruh penerapan metode ini terhadap kualitas solusi penyelesaian. Hasil pengaruh *random injection* dievaluasi berdasarkan rata-rata nilai *cost* terbaik. Berbeda dengan pengujian sebelumnya, pada pengujian ini untuk mengetahui pengaruh *random injection,* program dijalankan hanya satu kali dengan jumlah iterasi adalah 1000. Pada setiap iterasi kelipatan 50 akan dicatat nilai *cost* terbaik dan rata-rata nilai *cost.* Langkah tersebut berlaku untuk percobaan dengan menerapkan *random injection* maupun yang tidak dan diterapkan pada variasi interval kecepatan yang telah disebutkan pada pengujian kecepatan partikel . Hasil pengujian pengaruh *random injection* yang ditunjukkan pada Tabel 6.17 sampai Tabel 6.11 sebagai berikut:

Tabel 6.7 Hasil Pengujian Pengaruh *Random Injection* Interval 50%

Iterasi ke-	Tanpa *Random Injection*		Dengan *Random Injection*	
	Cost Terbaik	Rata-rata *Cost*	*Cost* Terbaik	Rata-rata *Cost*
10	0,087774396	0,110241769	0,116668442	0,167225644
20	0,087774396	0,089632201	0,087042276	0,096728044
30	0,087774396	0,088408414	0,084444455	0,106130220
40	0,087774396	0,087774396	0,083980731	0,098444559
50	0,087774396	0,087774396	0,083980731	0,095960900
100	0,087774396	0,087774396	0,083974204	0,091382660
150	0,087774396	0,087774396	0,083955136	0,089745481
200	0,087774396	0,087774396	0,083955136	0,089745481
250	0,087774396	0,087774396	0,083955136	0,088666487
300	0,087774396	0,087774396	0,083952178	0,087118623
350	0,087774396	0,087774396	0,083950261	0,086462972
400	0,087774396	0,087774396	0,083950223	0,088447742
450	0,087774396	0,087774396	0,083950221	0,086842718
500	0,087774396	0,087774396	0,083950190	0,088560486
550	0,087774396	0,087774396	0,083949611	0,088727398
600	0,087774396	0,087774396	0,083948168	0,087903652
650	0,087774396	0,087774396	0,083921131	0,090391778
700	0,087774396	0,087774396	0,083887361	0,086409705
750	0,087774396	0,087774396	0,083885466	0,087157768
800	0,087774396	0,087774396	0,083885457	0,086760903
850	0,087774396	0,087774396	0,083885355	0,088074967
900	0,087774396	0,087774396	0,083885354	0,086037914

950	0,087774396	0,087774396	0,083885351	0,085294950
1000	0,087774396	0,087774396	0,083885351	0,089647688
	Rata-rata			
	0,087774396	0,088814363	0,085446997	0,093244531

Tabel 6.8 Hasil Pengujian Pengaruh *Random Injection* Interval 5%

Iterasi ke-	Tanpa *Random Injection*		Dengan *Random Injection*	
	Cost Terbaik	Rata-rata *Cost*	*Cost* Terbaik	Rata-rata *Cost*
10	0,087352496	0,112879662	0,109473078	0,168383057
20	0,084082391	0,111805674	0,088580837	0,138631796
30	0,084082391	0,099920158	0,085972649	0,102652132
40	0,084038547	0,098330158	0,084112765	0,112283288
50	0,083964067	0,087217652	0,084112765	0,100971909
100	0,083963027	0,086561764	0,083783044	0,105139182
150	0,083953155	0,093137977	0,083770139	0,096899049
200	0,083952411	0,087163282	0,083755492	0,096163908
250	0,083951497	0,084133413	0,083755492	0,094232755
300	0,083951354	0,084302802	0,083753381	0,090266422
350	0,083951353	0,088069201	0,083753381	0,092653013
400	0,083951353	0,083951369	0,083753023	0,094482760
450	0,083951353	0,087981680	0,083749643	0,096618247
500	0,083951353	0,083951353	0,083749643	0,087231078
550	0,083951353	0,084038969	0,083749607	0,086314506
600	0,083951353	0,085353205	0,083748985	0,089260671
650	0,083951353	0,088069295	0,083748758	0,090841672
700	0,083951353	0,084214200	0,083748738	0,095286543
750	0,083951353	0,083951353	0,083748667	0,090329682
800	0,083951353	0,085703669	0,083748664	0,092668275
850	0,083951353	0,083951353	0,083748663	0,089851531
900	0,083951353	0,083951353	0,083748652	0,086180832
950	0,083951353	0,083951353	0,083748623	0,090117170
1000	0,083951353	0,083951353	0,083748608	0,087969039
	Rata-rata			
	0,084108761	0,089022594	0,085148471	0,098976188

Tabel 6.9 Hasil Pengujian Pengaruh *Random Injection* Interval 0,5%

Iterasi ke-	Tanpa *Random Injection*		Dengan *Random Injection*	
	Cost Terbaik	Rata-rata *Cost*	*Cost* Terbaik	Rata-rata *Cost*
10	0,131619589	0,214334377	0,141642942	0,227670471
20	0,118215540	0,198596780	0,119779888	0,222556862
30	0,112749846	0,182003504	0,117476540	0,220244776
40	0,109689651	0,172051822	0,116318103	0,214943029
50	0,107738089	0,163371977	0,110215506	0,206190335
100	0,105388759	0,126282419	0,097176869	0,178664586
150	0,097040183	0,107507735	0,094872023	0,173906244
200	0,096720897	0,102626881	0,094834312	0,169258443

250	0,096704984	0,102159386	0,091411596	0,177657913
300	0,096700395	0,098903666	0,088415686	0,195074674
350	0,096592676	0,098411288	0,088303955	0,171858183
400	0,096591193	0,098765513	0,088294678	0,175766176
450	0,096589026	0,098534356	0,088273455	0,168287011
500	0,096587839	0,098654103	0,088102297	0,159412697
550	0,096587834	0,098648209	0,087952292	0,168886629
600	0,096587805	0,098206957	0,087945643	0,172696634
650	0,096587805	0,098274477	0,087938123	0,161333990
700	0,096587805	0,098776539	0,087934047	0,167743122
750	0,096587805	0,098288164	0,087929361	0,157021729
800	0,096587805	0,098274050	0,087929361	0,144840680
850	0,096587805	0,098276994	0,087928274	0,164508290
900	0,096587805	0,098257715	0,087928273	0,160234206
950	0,096587805	0,098475450	0,087719727	0,174477536
1000	0,096587805	0,098326250	0,087677291	0,175649260
	Rata-rata			
	0,101033615	0,118583692	0,09600001	0,179536812

Tabel 6.10 Hasil Pengujian Pengaruh *Random Injection* Interval 0,05%

Iterasi ke-	Tanpa *Random Injection*		Dengan *Random Injection*	
	Cost Terbaik	Rata-rata *Cost*	*Cost* Terbaik	Rata-rata *Cost*
10	0,162549597	0,239244277	0,175361608	0,248793288
20	0,159267705	0,238548803	0,145820291	0,241407260
30	0,158147412	0,236569634	0,136233343	0,232665977
40	0,158082687	0,234560483	0,122266866	0,227725398
50	0,150307544	0,232446729	0,122231454	0,224650751
100	0,134070948	0,227888482	0,122231454	0,226189357
150	0,134070948	0,221624307	0,121967024	0,238694747
200	0,119145841	0,212860392	0,121965396	0,237644698
250	0,118381179	0,201035300	0,121965396	0,242810063
300	0,118346204	0,194791374	0,121965396	0,237918721
350	0,118284995	0,190357638	0,121965396	0,240419209
400	0,118225871	0,185690933	0,121965396	0,228262935
450	0,114382826	0,177992662	0,105750760	0,233525222
500	0,114355057	0,175098970	0,105750760	0,209536255
550	0,114354763	0,172590147	0,105750760	0,219054376
600	0,114354716	0,169522620	0,105750130	0,226736245
650	0,114344368	0,164856388	0,105750093	0,227053680
700	0,114339842	0,160516801	0,105041352	0,221932416
750	0,114339361	0,158097103	0,105033904	0,220216753
800	0,114339350	0,156361692	0,105033904	0,225722422
850	0,114339347	0,154316755	0,104113536	0,226560963
900	0,114178013	0,152990119	0,104113402	0,225863327
950	0,114150983	0,151169216	0,104113402	0,242524647

1000	0,114146278	0,149343422	0,104113402	0,231527495
	Rata-rata			
	0,12585441	0,189936427	0,117343934	0,230726509

Tabel 6.11 Hasil Pengujian Pengaruh *Random Injection* Interval 0,005%

Iterasi ke-	Tanpa *Random Injection*		Dengan *Random Injection*	
	Cost Terbaik	Rata-rata *Cost*	*Cost* Terbaik	Rata-rata *Cost*
10	0,154096749	0,236831367	0,177534729	0,238905728
20	0,154096749	0,236930730	0,167362938	0,231424287
30	0,154096749	0,236694794	0,167362938	0,230289178
40	0,154096749	0,236501727	0,154690158	0,224968295
50	0,154096749	0,236246799	0,154619419	0,228057461
100	0,153460971	0,235122256	0,119020165	0,243121467
150	0,152750117	0,233414573	0,119020165	0,241275335
200	0,151618657	0,232347412	0,119019428	0,238417619
250	0,149214147	0,231651320	0,119019341	0,234790905
300	0,147324871	0,230707150	0,119019341	0,238139082
350	0,143438417	0,229923977	0,119019341	0,249162835
400	0,142230597	0,229235465	0,119019341	0,248993243
450	0,139568284	0,228241613	0,119019341	0,243447019
500	0,137713504	0,227499925	0,119019341	0,245392215
550	0,136876880	0,226620214	0,119019341	0,237760528
600	0,134874338	0,225702705	0,119019341	0,226615360
650	0,134186518	0,225139326	0,119019341	0,243067426
700	0,132567387	0,224613215	0,109669898	0,223148665
750	0,132485917	0,223334818	0,109669898	0,224288143
800	0,131481725	0,222545417	0,109669898	0,234166000
850	0,131481725	0,221965091	0,109669898	0,233980089
900	0,130181703	0,221269950	0,109669898	0,235033871
950	0,128710448	0,220353609	0,109669898	0,239538195
1000	0,128214477	0,219627334	0,109669898	0,237429098
	Rata-rata			
	0,142036018	0,228855033	0,125728887	0,236308835

Berdasarkan hasil pengujian pengaruh *random injection* diatas, maka diperoleh rata-rata nilai *cost* terbaik untuk setiap interval kecepatan partikel yang ditunjukkan pada Tabel 6.12 sebagai berikut:

Tabel 6.12 Hasil Pengujian Pengaruh *Random Injection*

Interval Kecepatan		Rata-rata Nilai *Cost* Terbaik		Evaluasi 2		Evaluasi 1
V min	V max	Dengan RI	Tanpa RI	Dengan RI	Tanpa RI	
-5.10^{-1}	5.10^{-1}	0,083885351	0,087774396	100%	2%	X
-5.10^{-2}	5.10^{-2}	0,083748608	0,083951353	100%	4%	X
-5.10^{-3}	5.10^{-3}	0,087677238	0,096587805	100%	22%	V
-5.10^{-4}	5.10^{-4}	0,104113402	0,114146278	100%	54%	V
-5.10^{-5}	5.10^{-5}	0,109669898	0,128214477	100%	100%	V

Penerapan *random injection* untuk menangani konvergensi dini pada beberapa interval kecepatan terbukti menghasilkan partikel lebih variatif dan nilai *cost* lebih baik ditunjukkan dengan rata-rata nilai *cost* lebih kecil dibandingkan tanpa menerapkan *random injection*. Hasil pengujian ini sesuai dengan penelitian yang dilakukan oleh Mahmudy, et al (2013) yang mengungkapkan bahwa *random injection* terbukti efektif untuk menghindari konvergensi sekaligus mendapatkan solusi yang lebih baik. Pada interval kecepatan 0,05% dan 0,005%, mulai pada iterasi ke-200 dan ke-250, nilai *cost* terbaik selalu sama atau konvergen. Hal tersebut dikarenakan nilai *cost* yang diperoleh merupakan nilai *cost* paling optimum yang dapat dicapai partikel jika menerapkan kedua interval kecepatan tersebut. Oleh sebab itu, ketika dilakukan *random injection*, partikel baru yang diinisialisasi dari data normalisasi cenderung menghasilkan nilai *cost* lebih besar sehingga akan selalu tergantikan.

Berdasarkan pada pengujian interval kecepatan partikel dan pengujian pengaruh *random injection* maka interval kecepatan optimum adalah interval dengan kecepatan maksimal 0,005 dan minimal -0,005. Hal ini disebabkan interval tersebut telah memenuhi aturan penentuan pusat *cluster* dan memiliki rata-rata nilai *cost* terkecil pada pengujian pengaruh *random injection*. Interval 50% dan 5% meskipun memiliki rata-rata nilai *cost* lebih kecil dibandingkan interval 0,5% namun kedua interval tersebut tidak memenuhi aturan penentuan pusat *cluster* meskipun telah dilakukan *random injection*. Sedangkan, untuk interval 0,05% dan 0,005% meskipun memenuhi aturan penentuan pusat *cluster* namun pada pengujian pengaruh *random injection* memiliki rata-rata nilai *cost* lebih besar dibandingkan interval kecepatan 0,5%.

6.3 Pengujian Parameter PSO

Pengujian parameter PSO bertujuan untuk mengetahui pengaruh parameter PSO dalam melakukan *clustering* data. Parameter yang diuji coba adalah kombinasi bobot inersia, kombinasi koefisien akselerasi, jumlah itersi, serta ukuran *swarm.*

6.3.1 Pengujian Bobot Inersia

Pengujian bobot inersia bertujuan untuk mengetahui kombinasi bobot inersia maksimal (w_{max}) dan bobot inersia minimal (w_{min}) yang tepat guna memperoleh pusat-pusat *cluster* yang optimal. Nilai bobot inersia maksimal yang digunakan adalah 0.9, 0.8, dan 0.7 sedangkan nilai bobot inersia minimal yang digunakan adalah 0.2, 0.3, dan 0.4. Kombinasi bobot inersia maksimal dan minimal dievaluasi berdasarkan rata-rata nilai *cost* terbaik. PSO merupakan algoritma stokastis sehingga akan menghasilkan hasil berbeda setiap kali program dijalankan (Mahmudy, 2015), karenanya untuk memperoleh rata-rata nilai secara keseluruhan, dilakukan percobaan sebanyak 10 kali untuk setiap kombinasi bobot inersia. Hasil pengujian kombinasi bobot inersia ditunjukkan pada Tabel 6.13 sampai Tabel 6.15 sebagai berikut:

Tabel 6.13 Hasil Pengujian Bobot Inersia Maksimal = 0,9

Uji coba ke-	Nilai *Cost* Terbaik		
	w_{min}= 0,2	w_{min}= 0,3	w_{min}= 0,4
1	0,087434735	0,098854693	0,092359830
2	0,097353175	0,096854692	0,112749801
3	0,104721785	0,096392254	0,091777302
4	0,108571516	0,098538543	0,096856495
5	0,099078127	0,098254080	0,096881373
6	0,099096695	0,104089975	0,091458912
7	0,090905756	0,087786212	0,098421298
8	0,120505264	0,101894820	0,111636465
9	0,091444276	0,105889765	0,096144479
10	0,094500295	0,102041905	0,086434718
Rata-rata	0,099361162	0,099059694	0,097472067

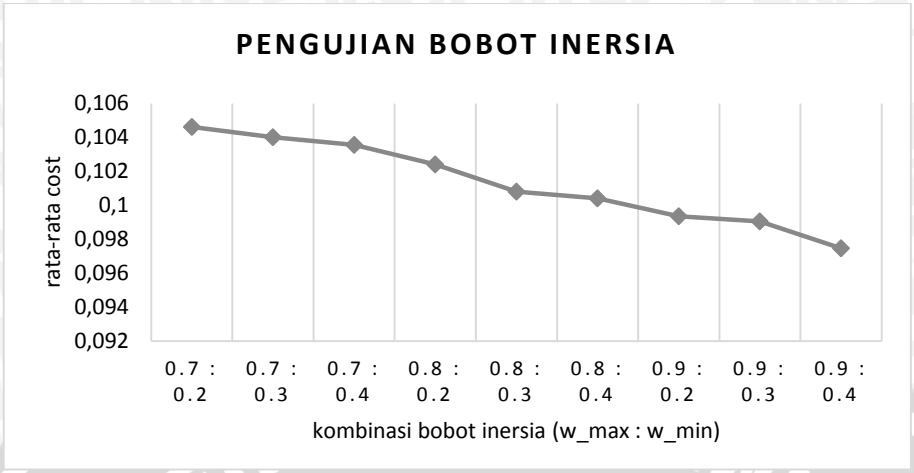
Tabel 6.14 Hasil Pengujian Bobot Inersia Maksimal = 0,8

Uji coba ke-	Nilai *Cost* Terbaik		
	w_{min}= 0,2	w_{min}= 0,3	w_{min}= 0,4
1	0,095813567	0,102577822	0,112988261
2	0,102769593	0,105908646	0,100109837
3	0,112966198	0,089523822	0,097595293
4	0,089485313	0,100144730	0,105749518
5	0,108378126	0,103217084	0,102752691
6	0,111068129	0,098362746	0,090048926
7	0,095441465	0,099182066	0,101324254
8	0,099628915	0,106220742	0,099477749
9	0,098700542	0,097969569	0,096123066
10	0,109765483	0,104933909	0,097916950
Rata-rata	0,102401733	0,100804114	0,100408655

Tabel 6. 15 Hasil Pengujian Bobot Inersia Maksimal = 0,7

Uji coba ke-	Nilai *Cost*		
	w_{min}= 0,2	w_{min}= 0,3	w_{min}= 0,4
1	0,095316498	0,109380318	0,105480112
2	0,103077555	0,112539089	0,110075608
3	0,102990006	0,101884516	0,093981091
4	0,104261794	0,101614446	0,106007215
5	0,106057103	0,105986599	0,106455380
6	0,106640468	0,104666185	0,108376509
7	0,094181718	0,108343366	0,094765196
8	0,106609259	0,109902803	0,093670812
9	0,114678493	0,093916081	0,107960685
10	0,112403122	0,091841972	0,108832324
Rata-rata	0,104621602	0,104007538	0,103560493

Grafik hasil pengujian kombinasi bobot inersia maksimal dan bobot inersia minimal ditunjukkan pada Gambar 6.1 sebagai berikut:



Gambar 6.1 Grafik Hasil Pengujian Bobot Inersia

Bobot inersia merupakan mekanisme untuk mengontrol daya eksplorasi dan eksploitasi partikel. Nilai bobot inersia yang besar meningkatkan daya eksplorasi sehingga keragaman partikel dalam *swarm* meningkat sedangkan nilai bobot inerisa yang kecil meningkatkan daya eksploitasi namun rentan terhadap hilangnya daya eksplorasi partikel (Engelbrecht, 2007). Eksplorasi cenderung memperluas ruang pencarian secara global sebaliknya eksploitasi cenderung berfokus pada pencarian solusi pada optimum lokal. Oleh karena keseimbangan antara kedua hal tersebut sangat menentukan PSO dalam menemukan solusi optimum, maka pemilihan inisialisasi nilai bobot inersia yang tepat akan berbeda, bergantung pada karakteristik permasalahan yang dihadapi (Shi, et al 1998).

Berdasarkan grafik hasil pengujian bobot inersia pada Gambar 6.1 maka diperoleh kombinasi bobot inersia terbaik adalah dengan menerapkan bobot inersia maksimal 0,9 dan bobot inersia minimal 0,4. Semakin besar kombinasi nilai bobot inersia (w_{max} dan w_{min}) maka nilai bobot inersia yang dihasilkan juga semakin besar (dihitung berdasarkan Persamaan 2.1). Nilai bobot inersia yang besar akan meningkatkan daya eksplorasi sehingga keragaman partikel meningkat. Kondisi ini memungkinkan partikel untuk menjelajah wilayah baru dan tidak lagi berfokus pada pencarian solusi pada daerah optimum lokal dimana pada daerah tersebut belum dapat dipastikan merupakan daerah pencarian yang menghasilkan sebuah solusi optimum. Hal ini dapat dilihat pada grafik pengujian bobot inersia bahwa semakin besar kombinasi bobot inersia maka semakin kecil nilai *cost* yang dihasilkan. Sehingga dapat disimpulkan bahwa dengan bobot inersia yang besar, pada permasalahan ini partikel-partikel PSO cenderung melakukan eksplorasi guna menemukan solusi optimum pada ruang pencarian.

6.3.2 Pengujian Koefisien Akselerasi

Pengujian koefisien dilakukan untuk mengetahui kombinasi akselerasi 1 (c_1) dan koefisien akselerasi 2 (c_1) terbaik guna memperoleh pusat-pusat *cluster* yang paling optimal. Nilai koefisien akselerasi 1 dan 2 yang digunakan adalah 1,

1.5, dan 2. Kombinasi koefisien akselerasi 1 dan 2 optimum dievaluasi berdasarkan rata-rata nilai *cost* terbaik. PSO merupakan algoritma stokastis sehingga akan menghasilkan hasil berbeda setiap kali program dijalankan (Mahmudy, 2015), karenanya untuk memperoleh rata-rata nilai secara keseluruhan, dilakukan percobaan sebanyak 10 kali untuk setiap kombinasi koefisien akselerasi. Hasil pengujian koefisien akselerasi ditunjukkan pada Tabel 6.16 sampai Tabel 6.18 sebagai berikut:

Tabel 6.16 Hasil Pengujian Koefisien Akselerasi 1 = 1

Uji coba ke-	Nilai *Cost* Terbaik		
	c_2 = 1	c_2 = 1,5	c_2 = 2
1	0,112166088	0,105745868	0,099309824
2	0,110055808	0,104853041	0,101645056
3	0,110276059	0,098547062	0,113084246
4	0,104095833	0,10795755	0,097278767
5	0,13082968	0,102775581	0,09888716
6	0,115226377	0,119407121	0,092378583
7	0,120045906	0,123845291	0,124076908
8	0,121259575	0,117094115	0,122345093
9	0,124197516	0,120445138	0,106225962
10	0,113962842	0,121625593	0,107928648
Rata-rata	0,116211568	0,112229636	0,106316025

Tabel 6.17 Hasil Pengujian Koefisien Akselerasi 1 = 1,5

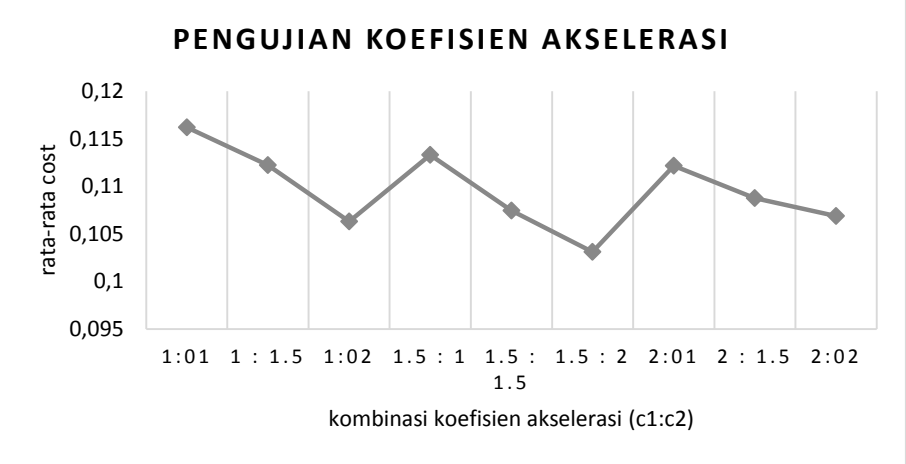
Uji coba ke-	Nilai *Cost* Terbaik		
	c_2 = 1	c_2 = 1,5	c_2 = 2
1	0,109657566	0,107427346	0,095935194
2	0,108327445	0,09679458	0,106420371
3	0,11820464	0,105362425	0,102279194
4	0,109837247	0,098221709	0,103333259
5	0,116458033	0,119545363	0,095933664
6	0,118316638	0,106347258	0,107173668
7	0,109318822	0,098963846	0,09448209
8	0,118180141	0,118185339	0,118131027
9	0,111968001	0,119363212	0,100099217
10	0,112944976	0,104238868	0,107454671
Rata-rata	0,113321351	0,107444995	0,103124236

Tabel 6.18 Hasil Pengujian Koefisien Akselerasi 1 = 2

Uji coba ke-	Nilai *Cost* Terbaik		
	c_2= 1	c_2= 1,5	c_2 = 2
1	0,090568239	0,106103206	0,10600927
2	0,122053557	0,097723687	0,102995119
3	0,115321766	0,117866389	0,110006019
4	0,10096197	0,108037592	0,101973793
5	0,10350077	0,105481788	0,099380429

6	0,126172355	0,10957592	0,108401268
7	0,116119127	0,103401269	0,102114527
8	0,129612866	0,107394755	0,112210699
9	0,111267257	0,123150731	0,12074505
10	0,10610222	0,108940295	0,105097004
Rata-rata	0,112168013	0,108767563	0,106893318

Grafik hasil pengujian kombinasi koefisien akselerasi 1 dan koefisien akselerasi 2 ditunjukkan pada Gambar 6.2 sebagai berikut:



Gambar 6.2 Grafik Hasil Pengujian Koefisien Akselerasi

Koefisien akselerasi bertugas mengontrol pergerakan partikel pada ruang pencarian. Secara umum, nilai koefisien akselerasi adalah sama dan bersifat statis. Nilai c_1 dan c_2 yang besar menyebabkan pergerakan partikel menempati posisi baru yang relatif lebih jauh sehingga kemampuan eksplorasi partikel menjadi lebih baik namun rentan menyimpang dari batas ruang pencarian. Sebaliknya, nilai c_1 dan c_2 yang kecil menyebabkan pergerakan partikel menjadi terbatas sehingga memungkinkan terperangkap pada pencarian optimum lokal. Kondisi lain adalah ketika nilai c_1 lebih besar dibandingkan c_2 maka pergerakan partikel cenderung mengarah pada posisi terbaik dari setiap partikel itu sendiri atau *personal best* dan sebaliknya jika nilai c_2 lebih besar dibandingkan nilai c_1 maka pergerakan partikel cenderung mengarah pada posisi terbaik partikel secara keseluruhan atau *global best* (Ahmed, et al., 2012). Sehingga, pemilihan nilai koefisien akselerasi dapat bervariasi disebabkan bobot untuk setiap partikel *personal best* dan *global best* akan berbeda satu sama lain bergantung pada karakteristik permasalahan yang dihadapi (Valle, et al., 2008).

Berdasarkan grafik hasil pengujian koefisien akselerasi pada Gambar 6.2 diperoleh kombinasi koefisien akselerasi terbaik adalah dengan menerapkan c_1 bernilai 1.5 dan c_2 bernilai 2. Karakteristik permasalahan berdasarkan hasil pengujian menunjukkan bahwa perubahan nilai *cost* cenderung dipengaruhi oleh koefisien akselerasi 2 yaitu semakin besar nilai c_2 maka semakin kecil nilai *cost* yang dihasilkan. Namun, hal tersebut tidak berlaku pada perubahan nilai koefisien akselerasi 1. Sebagai contoh, ketika nilai c_2 bernilai 1.5, semakin besar nilai c_1 tidak selalu menghasilkan nilai *cost* semakin kecil, hal tersebut berulang ketika c_2

bernilai 2. Sehingga dapat disimpulkan bahwa pada permasalahan ini, kemampuan pencarian solusi optimum oleh partikel-partikel PSO lebih dipengaruhi oleh posisi terbaik partikel secara keseluruhan atau *global best*.

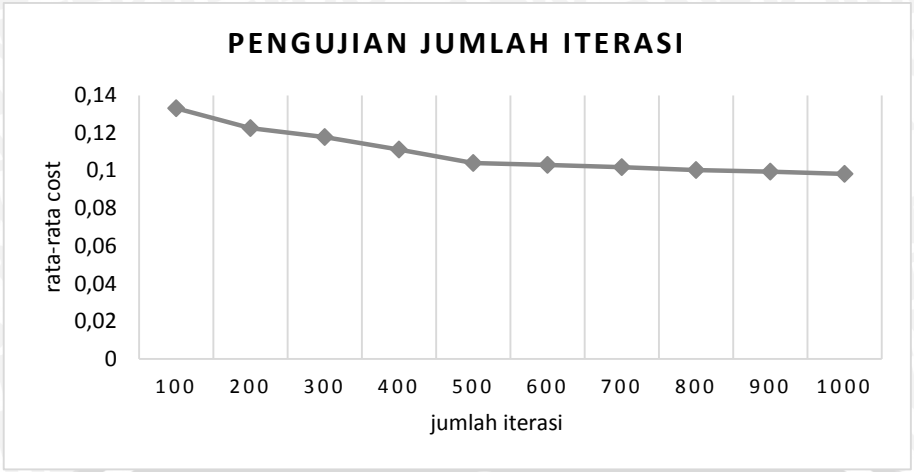
6.3.3 Pengujian Jumlah Iterasi

Pengujian jumlah iterasi dilakukan untuk mengetahui jumlah iterasi yang tepat guna memperoleh pusat-pusat *cluster* yang paling optimal. Jumlah iterasi optimum dievaluasi berdasarkan rata-rata nilai *cost* terbaik. Jumlah iterasi yang diguanakan adalah kelipatan 100. PSO merupakan algoritma stokastis sehingga akan menghasilkan hasil berbeda setiap kali program dijalankan (Mahmudy, 2015), karenanya untuk memperoleh rata-rata nilai secara keseluruhan, dilakukan percobaan sebanyak 10 kali untuk setiap jumlah iterasi. Hasil pengujian jumlah iterasi ditunjukkan pada Tabel 6.19 sebagai berikut:

Tabel 6.19 Hasil Pengujian Jumlah Iterasi

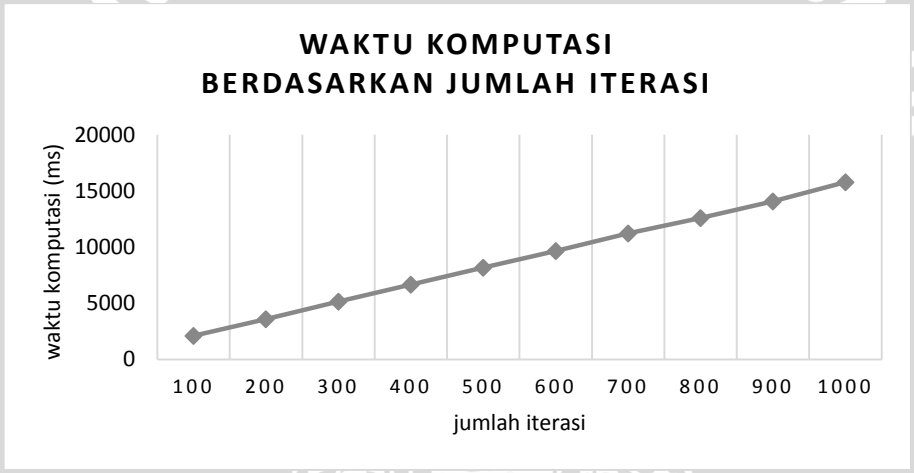
Uji coba ke-	Nilai *Cost* Terbaik				
	Iterasi = 100	Iterasi = 200	Iterasi = 300	Iterasi = 400	Iterasi = 500
1	0,144014916	0,136915204	0,093253116	0,113273362	0,105243012
2	0,152692532	0,118192111	0,095393769	0,114684229	0,089841215
3	0,134403391	0,135177327	0,135249066	0,108957953	0,105753303
4	0,112496367	0,121268778	0,124857863	0,105642425	0,123418388
5	0,123872477	0,123634743	0,119742174	0,113226549	0,116464769
6	0,159297571	0,141092858	0,139153835	0,119753696	0,096499612
7	0,117816885	0,127735439	0,122353847	0,111291802	0,103230908
8	0,130166994	0,106705426	0,119151482	0,094981475	0,097621876
9	0,133276566	0,114931124	0,108014193	0,112306996	0,102210315
10	0,122431358	0,100478414	0,121352507	0,117574177	0,100306154
Rata-rata	0,133046906	0,122613142	0,117852185	0,111169266	0,104058955
Uji coba ke-	Nilai *Cost* Terbaik				
	Iterasi = 600	Iterasi = 700	Iterasi = 800	Iterasi = 900	Iterasi = 1000
1	0,096336235	0,105553275	0,098590635	0,098888506	0,102561352
2	0,096973995	0,104798465	0,107206402	0,095026908	0,099863858
3	0,103870042	0,100134029	0,105136269	0,100649754	0,088549259
4	0,099504083	0,103625395	0,093762751	0,107574989	0,097926746
5	0,095970132	0,106416812	0,106354775	0,087877386	0,089491972
6	0,111646353	0,103235718	0,099612359	0,102384145	0,098139201
7	0,105875962	0,103957609	0,097321339	0,094618468	0,106915817
8	0,101843229	0,097304289	0,095508261	0,111872689	0,102787211
9	0,113046143	0,099306478	0,093115155	0,095187713	0,099172302
10	0,105350788	0,09468856	0,105861305	0,101406699	0,097577439
Rata-rata	0,103041696	0,101902063	0,100246925	0,099548726	0,098298516

Grafik hasil pengujian pertambahan jumlah iterasi terhadap rata-rata nilai *cost* ditunjukan pada Gambar 6.3 sebagai berikut:



Gambar 6.3 Grafik Hasil Pengujian Jumlah Iterasi

Grafik hasil pengujian jumlah iterasi terhadap waktu komputasi ditunjukan pada Gambar 6.4 sebagai berikut:



Gambar 6.4 Grafik Waktu Komputasi Berdasarkan Jumlah Iterasi

Berdasarkan grafik hasil pengujian jumlah iterasi pada Gambar 6.3 diperoleh jumlah iterasi terbaik dengan rata-rata nilai *cost* terkecil adalah 1000 iterasi. Jumlah iterasi yang dibutuhkan untuk memperoleh solusi terbaik bergantung pada jenis permasalahan yang dihadapi. Jumlah iterasi yang terlalu sedikit menyebabkan pencarian solusi penyelesaian menjadi prematur atau berhenti sebelum solusi optimum didapatkan. Sebaliknya, jumlah iterasi yang terlalu banyak menyebabkan kompleksitas komputasi semakin meningkat walaupun sebenarnya langkah tersebut tidak diperlukan (Engelbrecht, 2007). Hal ini ditunjukkan pada Gambar 6.4 dimana semakin besar jumlah iterasi maka waktu yang dibutuhkan untuk menyelesaikan komputasi juga semakin lama. Oleh karena itu, direkomendasikan menggunakan jumlah iterasi sebanyak 500 iterasi karena mulai pada iterasi ke-500, perubahan nilai *cost* tidak terlalu signifikan dan waktu komputasi yang dibutuhkan lebih efisien. Hal tersebut disebabkan partikel melakukan pencarian pada wilayah yang sama namun tidak menemukan solusi

yang signifikan lebih baik dari solusi yang diperoleh pada iterasi sebelumnya sehingga nilai *cost* yang dihasilkan relatif sama.

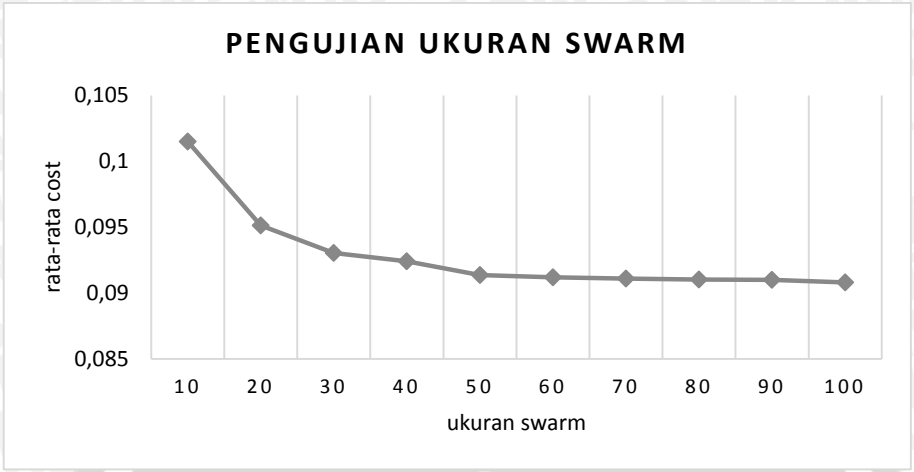
6.3.4 Pengujian Ukuran *Swarm*

Pengujian ukuran swarm dilakukan untuk mengetahui jumlah populasi yang dibutuhkan guna memperoleh pusat-pusat cluster yang paling optimal. Jumlah partikel dalam swarm yang diguanakan adalah kelipatan 10. Ukuran swarm optimum dievaluasi berdasarkan rata-rata nilai cost terbaik. PSO merupakan algoritma stokastis sehingga akan menghasilkan hasil berbeda setiap kali program dijalankan (Mahmudy, 2015), karenanya untuk memperoleh rata-rata nilai secara keseluruhan, dilakukan percobaan sebanyak 10 kali untuk setiap ukuran swarm. Hasil pengujian ukuran swarm ditunjukkan pada Tabel 6.20 sebagai berikut:

Tabel 6.20 Hasil Pengujian Ukuran *Swarm*

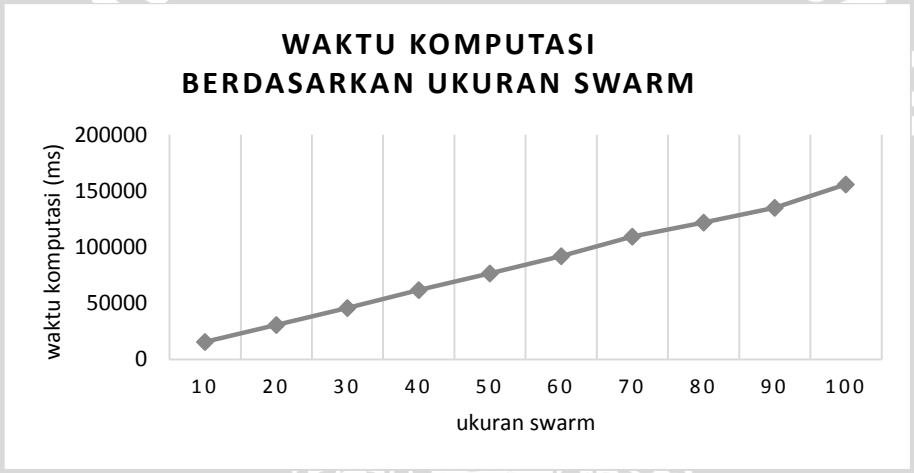
Uji coba ke-	Nilai *Cost* Terbaik				
	Partikel = 10	Partikel = 20	Partikel = 30	Partikel = 40	Partikel = 50
1	0,093073784	0,097585406	0,096923272	0,08959441	0,086833018
2	0,13073497	0,09140841	0,093052513	0,09421285	0,09085483
3	0,096926072	0,091612539	0,094350826	0,094377346	0,088782083
4	0,089927966	0,103383094	0,095827576	0,093956538	0,084392601
5	0,097929366	0,089957157	0,089387131	0,094977591	0,090562036
6	0,086271537	0,093146965	0,095571713	0,093391617	0,093377197
7	0,122665455	0,093050658	0,096261335	0,083235797	0,085247945
8	0,097428949	0,105240485	0,087178728	0,093803972	0,094737719
9	0,096949544	0,094838832	0,091117388	0,090946583	0,100179992
10	0,103003743	0,091136412	0,090894644	0,095557332	0,098681064
Rata-rata	0,101491139	0,095135996	0,093056513	0,092405404	0,091364849
Uji coba ke-	Nilai *Cost* Terbaik				
	Partikel = 60	Partikel = 70	Partikel = 80	Partikel = 90	Partikel = 100
1	0,094136162	0,093283994	0,09012458	0,096670384	0,089606415
2	0,098038082	0,093677393	0,083178554	0,092666434	0,085302332
3	0,08876467	0,09400939	0,084192664	0,086097725	0,088124632
4	0,093451197	0,090048142	0,087652988	0,090438077	0,093062455
5	0,086416027	0,098601467	0,106192542	0,093337422	0,094739927
6	0,083774764	0,083835467	0,088298336	0,098817516	0,092916943
7	0,093856767	0,083933081	0,089562372	0,09362937	0,094770485
8	0,091221395	0,092907478	0,094432846	0,085015804	0,084924605
9	0,093630711	0,092948344	0,097161711	0,08711139	0,0887693
10	0,08860619	0,087723833	0,089627314	0,086378967	0,095844147
Rata-rata	0,091189597	0,091096859	0,091042391	0,091016309	0,090806124

Grafik hasil pengujian pertambahan ukuran *swam* terhadap rata-rata nilai *cost* ditunjukkan pada Gambar 6.5 sebagai berikut:



Gambar 6.5 Grafik Hasil Pengujian Ukuran *Swarm*

Grafik hasil pengujian ukuran *swarm* terhadap waktu komputasi ditunjukan pada Gambar 6.6 sebagai berikut:



Gambar 6.6 Grafik Waktu Komputasi Berdasarkan Ukuran *Swarm*

Berdasarkan hasil pengujian ukuran *swam* pada Gambar 6.5 diperoleh jumlah partikel terbaik dengan rata-rata nilai *cost* terkecil adalah 100 partikel. Semakin besar jumlah partikel maka semakin tinggi tingkat keragaman partikel dalam *swarm* sehingga menyebabkan skema inisialisasi partikel juga semakin baik. Ukuran *swarm* yang besar memungkinkan partikel PSO untuk mengeksplorasi ruang pencarian semakin luas sehingga peluang untuk mendapatkan solusi penyelesaian optimum juga semakin besar. Namun demikian, semakin besar jumlah partikel menyebabkan waktu komputasi yang dibutuhkan juga semakin kompleks (Engelbrecht, 2007). Hal tersebut ditunjukkan pada Gambar 6.6 dimana semakin besar ukuran *swarm* maka waktu yang dibutuhkan untuk menyelesaikan komputasi juga semakin lama. Oleh karena itu, direkomendasikan menggunakan ukuran *swarm* sebanyak 50 partikel karena dengan jumlah partikel sebanyak 50, partikel-partikel PSO sudah mampu menghasilkan nilai *cost* yang cukup rendah dengan waktu komputasi yang lebih efisien.

6.4 Pengujian Jumlah *Cluster*

Pengujian jumlah *cluster* dilakukan untuk mengetahui jumlah *cluster* dengan tingkat validasi data tertinggi pada penyelesaian *clustering* data UKT Proporsional. Nilai kualitas sebuah *cluster* dievaluasi menggunakan metode analisis *cluster Silhouette Coefficient*. Jumlah *cluster* yang diuji coba sebanyak 2, 3, 4, 5, dan 6. Untuk memperoleh nilai secara keseluruhan, dilakukan percobaan sebanyak 10 kali untuk setiap jumlah *cluster*. Hasil pengujian masing-masing *cluster* ditunjukkan pada Tabel 6.21 sampai Tabel 6.25 sebagai berikut:

Tabel 6.21 Hasil Pengujian Jumlah *Cluster* = 2

Uji coba ke-	Maks Iterasi	Nilai *Silhouette Coefficient*
1	7	0,524514799
2	7	0,524514799
3	7	0,524514799
4	7	0,524514799
5	7	0,524514799
6	7	0,524514799
7	7	0,524514799
8	7	0,524514799
9	7	0,524514799
10	7	0,524514799
Rata-rata		0,524514799

Tabel 6.22 Hasil Pengujian Jumlah *Cluster* = 3

Uji coba ke-	Maks Iterasi	Nilai *Silhouette Coefficient*
1	21	0,528126528
2	23	0,528126528
3	21	0,528126528
4	21	0,528126528
5	21	0,528126528
6	18	0,528126528
7	22	0,528126528
8	21	0,528126528
9	24	0,528126528
10	22	0,528126528
Rata-rata		0,528126528

Tabel 6.23 Hasil Pengujian Jumlah *Cluster* = 4

Uji coba ke-	Maks Iterasi	Nilai *Silhouette Coefficient*
1	20	0,516205915
2	22	0,524181357
3	22	0,524982474
4	19	0,524982474
5	18	0,524982474
6	16	0,470573676

7	16	0,470573676
8	19	0,470780668
9	16	0,469736094
10	15	0,470573676
Rata-rata		0,496757248

Tabel 6.24 Hasil Pengujian Jumlah *Cluster* = 5

Uji coba ke-	Maks Iterasi	Nilai *Silhouette Coefficient*
1	19	0,376453986
2	28	0,411864916
3	25	0,349604086
4	16	0,440611611
5	17	0,497435505
6	17	0,440611611
7	21	0,52305412
8	24	0,394181511
9	21	0,506071029
10	27	0,392769498
Rata-rata		0,433265787

Tabel 6.25 Hasil Pengujian Jumlah *Cluster* = 6

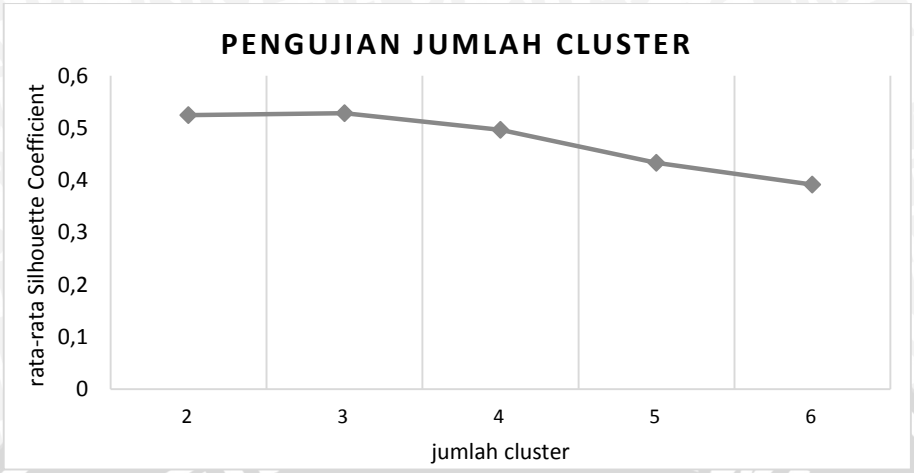
Uji coba ke-	Maks Iterasi	Nilai *Silhouette Coefficient*
1	27	0,402996257
2	26	0,374267234
3	31	0,416572379
4	27	0,402096679
5	33	0,378340103
6	17	0,350148139
7	31	0,416572379
8	31	0,416572379
9	34	0,404843959
10	20	0,358186896
Rata-rata		0,392059640

Berdasarkan hasil pengujian jumlah *cluster* diatas, maka diperoleh rata-rata nilai *Silhouette Coefficient* setiap jumlah *cluster* yang ditunjukkan pada Tabel 6.26 sebagai berikut:

Tabel 6.26 Hasil Pengujian Jumlah *Cluster*

Jumlah *Cluster*	Nilai *Silhouette Coefficient*	
	Interval -1 s.d 1	Dalam %
2	0,524514799	76,23%
3	0,528126528	76,41%
4	0,496757248	74,84%
5	0,433265787	71,66%
6	0,392059640	69,6%

Grafik hasil pengujian jumlah *cluster* terhadap rata-rata nilai *Silhouette Coefficient* ditunjukkan pada Gambar 6.7 sebagai berikut:



Gambar 6.7 Grafik Hasil Pengujian Jumlah *Cluster*

Nilai *Silhouette Coefficient* berada pada rentang -1 sampai dengan 1. Ketika nilai *Silhouette Coefficient* mendekati 1, maka jarak antar data dalam satu *cluster* yang sama semakin dekat dan jarak data pada *cluster* berbeda semakin jauh. Sementara ketika nilai *Silhouette Coefficient* mendekati -1, maka jarak antar data dalam satu *cluster* yang sama semakin jauh dan jarak data dengan *cluster* berbeda semakin dekat (Han, et al., 2012). Pengelompokkan data ke dalam 3 *cluster* terbukti menghasilkan nilai *Silhouette Coefficient* tertinggi. Hal tersebut menunjukkan bahwa dengan mengelompokkan data UKT Proporsional ke dalam 3 *cluster* maka tingkat kemiripan data dalam satu *cluster* yang sama semakin tinggi dan tingkat kemiripan data pada *cluster* berbeda semakin rendah sehingga hasil pengelompokkan data semakin tepat.

6.5 Pengujian Perbandingan Algoritma

Pengujian perbandingan algoritma dilakukan untuk mengetahui kinerja masing-masing algoritma. Pengujian ini sekaligus sebagai langkah pembuktian apakah algoritma HPSOKM memiliki kinerja lebih baik dibandingkan algoritma *K-Means* dalam melakukan proses *clustering* data. Kinerja algoritma dievaluasi menggunakan metode *Silhouette Coefficient* yang memiliki rentang antara -1 sampai dengan 1. Hasil pengujian perbandingan masing-masing algoritma ditunjukkan pada Tabel 6.27 dan Tabel 6.28 sebagai berikut:

Tabel 6.27 Hasil Pengujian *Clustering* Algoritma *K-Means*

Uji coba ke-	Max Itreasi	Waktu Komputasi	Nilai *Silhouette Coefficient*	
			Interval -1 s.d 1	Dalam %
1	18	538 ms	0,528126528	76,41%
2	8	524 ms	0,489162918	74,46%
3	8	524 ms	0,525295197	76,26%
4	12	587 ms	0,502796957	75,14%
5	19	540 ms	0,528126528	76,41%

6	13	529 ms	0,487626765	74,38%
7	7	530 ms	0,503293325	75,16%
8	5	514 ms	0,533959392	76,7%
9	9	523 ms	0,501500018	75,08%
10	13	556 ms	0,505384965	75,27%
Rata-rata		536,5 ms	0,510527259	75,53%

Tabel 6.28 Hasil Pengujian *Clustering* Algoritma HPSOKM

Uji coba ke-	Max Itreasi	Waktu Komputasi	Nilai *Silhouette Coefficient*	
			Interval -1 s.d 1	Dalam %
1	22	151273 ms	0,528126528	76,41%
2	22	146285 ms	0,528126528	76,41%
3	22	151563 ms	0,528126528	76,41%
4	22	152796 ms	0,528126528	76,41%
5	22	150906 ms	0,528126528	76,41%
6	22	151474 ms	0,528126528	76,41%
7	22	156320 ms	0,528126528	76,41%
8	22	152988 ms	0,528126528	76,41%
9	22	156885 ms	0,528126528	76,41%
10	22	151906 ms	0,528126528	76,41%
Rata-rata		152239,6 ms	0,528126528	76,41%

Berdasarkan hasil pengujian masing-masing algoritma diatas, maka diperoleh rata-rata nilai *Silhouette Coefficient* kedua algoritma yang ditunjukkan pada Tabel 6.29 sebagai berikut:

Tabel 6.29 Hasil Pengujian Perbandingan Algoritma

	Waktu Komputasi	Interval -1 s.d 1	Dalam %
K-Means	536,5 ms	0,510527259	75,53%
HPSOKM	152239,6 ms	0,528126528	76,41%

Berdasarkan hasil pengujian perbandingan algoritma pada Tabel 6.29 maka diketahui algoritma HPSOKM memiliki nilai *Silhouette Coefficient* lebih tinggi dibandingkan algoritma *K-Means*. Hal tersebut mengindikasikan bahwa algoritma HPSOKM menghasilkan hasil pengelompokan lebih baik dibandingkan algoritma *K-Means* meskipun membutuhkan waktu komputasi yang lebih lama dengan peningkatan kualitas kurang dari 1%. Sebagai contoh, pada algoritma *K-Means* percobaan ke-2 dan ke-6, ketika pusat *cluster* kurang tepat maka nilai *Silhouette Coefficient* rendah sebaliknya pada pencobaan ke-8, ketika pusat *cluster* semakin tepat maka nilai *Silhouette Coefficient* tinggi bahkan lebih tinggi dibandingkan hasil *clustering* menggunakan algoritma HPSOKM. Kondisi ini menunjukkan bahwa kualitas hasil *clustering* algoritma *K-Means* sangat bergantung terhadap pemilihan pusat *cluster* awal. Sedangkan, pada pusat *cluster* yang telah dioptimasi terlebih dahulu (HPSOKM), kualitas hasil *clustering* menunjukkan rata-rata lebih baik dan stabil. Namun demikian, fokus penelitian ini lebih condong pada kualitas hasil *clustering* dibandingkan waktu komputasi sehingga dapat disimpulkan bahwa

algoritma HPSOKM memiliki kinerja yang lebih baik dibandingkan algoritma *K-Means* untuk menyelesaikan pengelompokkan data UKT Proporsional.

6.6 Pembahasan Hasil Pengujian

Sistem *clustering* data UKT Proporsional menggunakan algoritma HPSOKM dijalankan dengan menggunakan parameter optimum diperoleh dari serangkaian hasil pengujian, meliputi: jumlah *cluster* = 3, jumlah iterasi = 1000, ukuran *swarm* = 100, bobot inersia maksimum = 0.9, bobot inersia minimum = 0.4, koefisien akselerasi 1 = 1.5 dan koefisien akselerasi 2 = 2. Sistem tersebut menghasilkan solusi penyelesaian berupa pusat-pusat *cluster* optimum dengan nilai *cost* sebesar 0,0869899019227701. Representasi partikel *global best* ditunjukkan pada Tabel 6.30 sebagai berikut:

Tabel 6.30 Solusi Pusat *Cluster* Optimum

PC ke-	Nilai Pusat *Cluster*					
1	0,018412	0,007251	0,037795	0,036303	4,90E-06	0,010023
2	0	0,000573	0,023237	0,022177	0	0
3	0,005613	0,000883	0,019297	0,010516	0	1,42E-06

Keterangan:
PC : Pusat *Cluster*

Jumlah *cluster* optimum yang dihasilkan sistem menggunakan algoritma HPSOKM adalah 3 kategori. Kondisi ini tidak sesuai dengan kenyataan lapangan bahwa jumlah kategori penentuan UKT Proporsional berdasarkan kebijakan Universitas Brawijaya adalah 6 kategori. Hal tersebut dapat disebabkan oleh beberapa alasan. Pertama, penentuan jumlah kategori UKT Proporsional oleh Universitas Brawijaya kemungkinan besar tidak didahului dengan pendekatan *clustering* melainkan berdasarkan kebijakan yang diputuskan pada suatu waktu tertentu. Kedua, berbeda dengan sistem yang hanya menggunakan variabel utama, penentuan jumlah kategori UKT Proporsional pada kondisi sebenarnya juga menggunakan variabel pendukung seperti jumlah anggota keluarga yang menjadi tanggungan orang tua dimana analisis seperti ini dilakukan oleh panitia khusus yang ditunjuk oleh Universitas Brawijaya. Ketiga, sistem hanya melibatkan data penentuan UKT Proporsional dari Jurusan Teknik Informatika PTIIK UB sehingga hasil pengelompokkan kurang obyektif, disisi lain aturan penentuan UKT Proporsional secara keseluruhan untuk setiap fakultas/program studi adalah sama, hanya berbeda pada besar nominal untuk setiap kategori penentuan.

Terlepas dari beberapa alasan tersebut diatas, sistem yang dibangun bertujuan sebatas memberikan rekomendasi jumlah *cluster* atau kategori ideal berdasarkan pendekatan *clustering* yang dievaluasi menggunakan metode analisis *cluster Silhouette Coefficient.* Selanjutnya, dari hasil pengelompokan tersebut dapat dilakukan analisis lebih lanjut guna menemukan informasi tertentu sehingga dapat digunakan sebagai bahan pertimbangan dalam menentukan besar nominal UKT Proposional untuk setiap kategori.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan pembahasan dari implementasi algoritma *Hybrid Particle Swarm Optimization* dan *K-Means* (HPSOKM) untuk *clustering* data UKT Proprosional maka diperoleh kesimpulan sebagai berikut:

1. Algoritma HPSOKM dapat digunakan untuk mengelompokkan data UKT Proporsional ke dalam *k* kategori. Hal tersebut dilakukan dengan mendefinisikan partikel sebagai representasi penyelesaian. Panjang partikel adalah dimensi data dikali dengan jumlah *cluster* sehingga jika data dengan jumlah dimensi 6 dikelompokkan ke dalam 2 *cluster* maka 6 sel pertama merepresentasikan pusat *cluster* pertama dan 6 sel kedua merepresentasikan pusat *cluster* kedua. Selama proses optimasi, kecepatan dan posisi partikel selalu diperbarui. Solusi penyelesaian berupa pusat-pusat *cluster* optimum, merupakan posisi terbaik yang pernah dicapai sebuah partikel sampai dengan iterasi tertentu. Pusat *cluster* tersebut kemudian dijadikan sebagai inisialisasi pusat *cluster* awal pada algoritma *K-Means* untuk proses *clustering* lebih lanjut.
2. Penambahan jumlah *cluster* berpengaruh terhadap kualitas hasil *clustering.* Jumlah *cluster* terbaik berdasarkan pengujian jumlah *cluster* adalah sebanyak 3 *cluster* dengan nilai *Silhouette Coefficient* tertinggi yaitu 76,41%.
3. Algoritma HPSOKM terbukti lebih baik dalam melakukan pengelompokkan data UKT Proporsional dibandingkan dengan algoritma *K-Means.* Berdasarkan hasil pengujian perbandingan algoritma, peningkatan kualitas *clustering* HPSOKM lebih tinggi 0,88% dibandingkan algoritma *K-Means.* Selain itu, algoritma HPSOKM juga lebih stabil, terbukti dengan nilai *Silhouette Coefficient* yang relatif sama untuk 10 kali percobaan.

7.2 Saran

Penelitian tentang *clustering* menggunakan algoritma *Hybrid* PSO dan *K-Means* (HPSOKM) dapat dikembangkan dengan beberapa saran sebagai berikut:

1. Pada saat menentukan kecepatan partikel yang sesuai, dapat dilakukan metode perhitungan lebih lanjut agar interval kecepatan sesuai dengan karakteristik permasalahan sehingga PSO dapat menghasilkan solusi yang lebih baik.
2. Pada penelitian mendatang dapat ditambahkan seleksi fitur sehingga dapat diketahui atribut mana saja yang memiliki pengaruh lebih besar terhadap hasil pengelompokan. Sehingga diharapkan dengan menerapkan seleksi fitur, sistem dapat menghasilkan hasil pengelompokan lebih baik dengan waktu komputasi yang lebih efisien.
3. Data pengelompokan yang digunakan sebaiknya melibatkan keseluruhan data penentuan UKT Proporsional dari semua fakultas. Hal ini bertujuan agar hasil pengelompokan lebih obyektif, selain itu fakta bahwa aturan UKT Proporsional (jumlah kategori) adalah sama untuk semua fakultas/program studi dan hanya berbeda pada penentuan besar nominal untuk setiap kategori.

DAFTAR PUSTAKA

Afivi, R., 2005. Pengelompokan Selari Untuk Data Skala Besar dan Dimensional Tinggi Pada Aplikasi Perlombongan Data*. Proceedings of The Postgraduate Annual Research Seminar 2005.* Malaysia: University Teknologi Malaysia.

Ahmed, H., Glasgow, J. 2012. Swarm Intelligence: Concepts, Models, and Applications. Queen's University, Canada.

Alam, S., et al., 2014. Research on Particle Swarm Optimization Based Clustering: A Systematic Review of Literature and Techniques. *Swarm and Evolutionary Computation.*

Cui, X., Potok, T.E., 2005. Document Clustering Analysis Based on Hybrid PSO+K-Means Algorithm.

Engelbrecht, A.P., 2007. *Computational Intellegent.* Second Edition. West Sussex: John Wiley & Sons Ltd. ISBN: 978-0-470-03561-0.

Han, J., Kamber, M., Pie, J., 2012. *Data mining: Concept and Techniques.* Third Edition. Waltham: Elsevier. ISBN: 978-0-12-381479-1.

Irwansyah, E., Faisal, M., 2015. *Advanced Clustering Teori dan Aplikasi.* [e-book]. DeePublish. ISBN: 9786022805007. Tersedia di: Google Play Books <https://books.google.co.id/books?id=8y80BgAAQBAJ&printsec=frontcover&dq=advanced+clustering&hl=en&sa=X&ved=0CCQQ6AEwAWoVChMIgYPy7JqOxwIVgimUCh3IgQle#v=onepage&q=advanced%20clustering&f=false>. [Diakses pada: Selasa, 4 Agustus 2015].

Jordedi, A.R., Jasni, J., 2013. Parameter Selection In Particle Swarm Optimization. *Journal Of Experimental & Theoretical Artificial Intelligence.*

Junaedi, H, et al., 2011. Data Transformation Pada *Data mining*. *Prosiding Konferensi Nasional "Inovasi Dalam Desain dan Teknologi"- IdeaTech 201.* ISSN: 2809-1121.

Karegowda, A.G, et al., 2012. Genetic Algorithm Based Dimensionality Reduction for Improving Performance of K-Means Clustering: A Case Study for Categorization of Medical *Dataset*. *International Journal of Soft Computing 7(5): 249-255, 2012.* ISSN: 1816-9503.

Kusrini., Luthfi, E.T., 2009. *Algoritma Data mining*. Yogyakarta: Andi Offset. ISBN: 978-979-29-0809-1.

Mahmudy, W.F., et al., 2013. Optimization of Part Type Selection and Loading Problem with Alternative Production Plans in Flexible Manufacturing System Using Hybrid Genetic Algorithms- Part 2: Genetic Operators and Results. *5th International Conference on Knowledge and Smart Technology (KST).* University of South Australia, IEEE 978-1-4673-4853-9.

Mahmudy, W.F., 2014a. Optimasi Part Type Selection and Machine Loading Problems Pada FMS Menggunakan Metode Particle Swarm Optimization. *Konferensi Nasional Sistem Informasi (KNSI).* STMIK Dipanegara, Makassar, 27 Februari – 1 Maret pp.1718-1723.

Mahmudy, W.F., 2014a. Optimasi Part Type Selection and Machine Loading Problems Pada FMS Menggunakan Metode Particle Swarm Optimization. *Konferensi Nasional Sistem Informasi (KNSI).* STMIK Dipanegara, Makassar, 27 Februari – 1 Maret pp.1718-1723.

Mahmudy, W.F., et al., 2014b. Hybrid Genetic Algorithms for Part Type Selection and Machine Loading Problems with Alternative Production Plans in Flexible Manufacturing System. *ECTI Transaction on Computer and Information Technology (ECTI-CIT)*, Vol. 8, pp. 80-93.

Mahmudy, W.F., 2015. Improved Particle Swarm Optimization Untuk Menyelesaikan Permasalahan Part Type Selection dan Machine Loading Pada Flexible Manufacturing System (FMS). *Konferensi Nasional Sistem Informasi (KNSI).* Universitas Klabat, Airmadidi, Minahasa Utara, Sulawesi Utara, 26 – 28 Februari, pp. 1003 – 1008.

Van der Merwe, DW., Engelbrecht, AP., 2003. Data Clustering Using Particle Swram Optimization. University of Pretoria, IEEE 0-7803-7804-0/03.

Prawoto, H., 2015. *Wawancara Penentuan UKT Proporsional Universitas Brawijaya Tahun 2014.* Diwawancara oleh Maulian Eka Kusuma [langsung] Sub Bagian Keuangan Rektort Lt. 5 Unversitas Brawijaya Malang. Senin, 4 Mei 2015 pukul 10.30 WIB.

Puspitarini, M., 2013. *Bahas Uang Kuliah Tunggal, UB Ajak Mahasiswa.* [online] Tersedia di: < http://news.okezone.com/read/2013/05/10/373/804961/large> [Diakses pada: Senin, 11 Mei 2015].

Rana, S., Jasola, S., Kumar, R., 2010. A Hybrid Squential Approach For Data Clustering Using K-Means And Particle Swarm Optimization Algorithm. *International Journal of Engineering, Science and Technology,* vol. 2, no. 6, 2010, pp. 167 – 176.

Santhanam, T., Padmavathi, M.S., 2015. Application of K-Means and Genetic Algorithm for Dimension Reduction By Integrating SVM for Diabetes Diagnosis. *Procedia Computer Science 47 (1025) 76 – 83.*

Universitas Brawijaya, 2015a. *Jalur Masuk Universitas Brawijaya.* [online] Tersedia di: <http://selma.ub.ac.id/jalur-masuk/> [Diakses pada: Senin, 4 Mei 2015].

Universitas Brawijaya, 2015b. *Info Biaya Pendidikan Universitas Brawijaya.* [online] Tersedia di: <http://selma.ub.ac.id/info-biaya-pendidikan-20142015/> [Diakses pada: Senin, 4 Mei 2015].

Shi, Y., Eberhart, R. 1998. A Modifiend Particle Swarm Optimizer. *IEEE World Congress on Computational Intelligence*. IEEE 0-7803-4869-9/98.

Tuegeh, M., Soeprijanto., Purnomo, M.H., 2009. Modified Improved Particle Swarm Optimization for Optimal Generator Scheduling. *Seminar Nasional Aplikasi Teknologi Informasi 2009 (SNATI 2009).* Yogyakarta, 20 Juni 2009. ISSN: 1907-5022.

Valle, Y.D., Mohagheghi, S., 2008. Particle Swarm Optimization: Basic Concept, Variants and Application in Power System. *IEEE Transactions On Evolutionary Computation*, vol. 12, no. 2, April 2008.

LAMPIRAN 1
DAFTAR PERTANYAAN PAKAR

1. Apakah yang dimaksud UKT Proposional?
   Jawab:
   UKT Proporsional merupakan aturan penentuan biaya kuliah berdasarkan kemampuan mahasiswa mencakup SPP Proposional sekaligus Sumbangan Pengengembangan Fasilitas Pendidikan (SPFP) yang dibayarkan sekaligus untuk masa studi 8 semeter atau 4 tahun pada jenjang S1.
   Sumbangan Pengembangan Fasilitas Pendidikan (SPFP) dalam pengelolaannya dapat dipergunakan untuk pembayaran uang pangkal atau gedung, biaya praktikum, dan biaya administrasi pendukung lainnya.
   Istilah UKT Proporsional ditujukan untuk pembayaran biaya kuliah mahasiswa melalui jalur SBMPTN dan SNMPTN

2. Bagaimana untuk mahasiswa yang masa studinya lebih dari delapan semeter?
   Jawab:
   Pembayaran biaya kuliah untuk semester sembilan dan seterusnya dikenakan SPP Progresif yang besar persentasenya ditentukan oleh universitas

3. Sejak kapan diberlakukan UKT Proporsional?
   Jawab:
   UKT Proporsional diberlakukan sejak tahun akademik 2013/2014 berdasarkan Surat Keputusan Rektor Universitas Brawijaya Nomor 078/SK/2013 kemudian diganti dengan 276/SK/2013

4. Apakah tujuan dari pengelompokkan UKT Proporsional ke dalam beberapa kategori?
   Jawab:
   a. Menjalankan keputusan dari Direktorat Jenderal Pendidikan Tinggi (Ditjen Dikti)
   b. Besarnya biaya UKT Proporsional yang dibayarkan tepat sasaran yakni sesuai dengan kemampuan mahasiswa
   c. Memberikan kesempatan bagi mahasiswa yang kurang mampu untuk dapat melanjutkan kuliah

5. Faktor yang menjadi pertimbangan penentuan biaya UKT Proposional?
   Jawab:
   a. Penghasilan orang tua
   b. Rekening listrik
   c. Rekening telepon
   d. Rekening PAM/PDAM
   e. Rekening pajak bumi dan bangunan (PBB)
   f. Rekening pajak kendaraan mobil dan motor (PKB)

6. Dengan penentuan kategori berjumlah 6, apakah ada kemungkinan bahwa dimasa yang akan datang ada perubahan kategori?
   Jawab:
   Ya. Evaluasi terkait jumlah kategori penentuan UKT Proporsional dilakukan secara berkala. Hal tersebut dimaksudkan agar sebaran kategori bersifat normal artinya

tidak berkumpul pada satu kategori saja atau sebarannya tidak merata, sehingga jalannya administrasi di lingkungan perguruan tinggi dapat berlangsung lancar.

Tertanda,

Bapak Heri Prawoto
Kepala Sub Bagian Penerimaan Negara Bukan Pajak (PNBP)
Rektorat LT.5 Universitas Brawijaya

LAMPIRAN 2
HASIL CLUSTERING DATA UKT PROPORSIONAL

Data ke-	Gaji	PBB	PKB	Listrik	Telepon	Air	Cluster
1	144573063	1433550	4643000	1111649	137833	274145	1
2	41653649	348026	3406400	554919	75000	0	1
3	28560685	111115	1418800	582000	49800	16000	1
4	27519000	98374	2072010	429610	87745	76000	1
5	25911346	335480	2047500	387665	101848	209525	3
6	20416928	274804	2387600	317763	0	0	1
7	19506351	88000	3028500	352361	58630	104000	1
8	19132390	147028	488500	865854	45414	0	1
9	17213372	236875	2555500	784626	138995	403255	3
10	16882994	241907	1486000	207116	38695	125200	3
11	16620035	19295	2795500	329323	79390	131450	1
12	15645000	74240	3119800	388400	49795	0	1
13	15630000	126648	486000	384942	131210	81356	2
14	15000000	116900	3831840	661979	123579	0	1
15	15715202	0	2898330	0	0	0	1
16	15000000	199900	3526500	200903	0	7640	1
17	15000000	25000	2902500	390534	300887	0	1
18	15000000	0	2753000	638000	0	320000	3
19	14867704	25731	303500	118026	0	0	2
20	15000000	0	0	216000	33070	0	2
21	13532584	64779	515400	364210	0	0	2
22	12500000	847800	2159500	1700000	150000	25000	1
23	11209000	440760	2609500	858160	335742	56180	1
24	11074939	203872	2311000	205434	0	0	1
25	10729400	74805	2184000	995236	37070	0	1
26	11037403	1303032	0	451198	177535	62145	3
27	10202100	268887	1426200	381686	0	274670	3
28	10148005	38166	3336750	703885	10642	141460	1
29	10241407	70818	189500	200000	0	121900	2
30	9868152	36314	1176500	88068	49445	0	2
31	9860878	79000	175000	21903	0	0	2
32	9330498	175960	2966500	144815	0	0	1
33	9566100	114114	415500	118900	0	70600	2
34	9575979	177960	269000	200000	0	123300	2
35	9000000	454896	3368500	204260	314229	0	1
36	9177900	43080	430500	342498	256085	160670	3
37	9355553	0	238000	100000	0	0	2
38	8734726	105490	2741500	124020	270845	115200	1

39	8978719	13514	702200	53511	0	0	2
40	8738116	331224	256400	220000	0	186600	3
41	8606106	67689	420000	104947	0	0	2
42	8822160	0	152250	80789	34070	0	2
43	8176878	79265	2510000	76492	258606	43250	1
44	8336178	43100	586600	154343	0	0	2
45	8615472	0	358000	0	0	0	2
46	8022402	71472	2198000	21040	0	0	2
47	8072109	25354	689900	433002	0	121900	3
48	8000000	201000	194000	200000	0	0	2
49	7668076	97350	920500	101344	48642	64500	2
50	7717255	76837	385000	216089	0	0	2
51	7746057	66700	186500	84944	54585	139500	2
52	7500000	71940	2879500	352639	0	0	1
53	7313853	40500	1895500	280903	1211792	0	3
54	7167747	3418352	2634900	426054	0	63000	1
55	7426400	204503	215750	90896	43043	116096	2
56	7332187	9000	397000	100000	0	62820	2
57	7266163	13144	241400	195000	0	46000	2
58	7131496	171193	259800	178607	64134	116910	2
59	7000000	74676	475500	72386	38360	62500	2
60	7000000	17400	475500	142518	0	26010	2
61	6948465	53380	406500	135066	0	0	2
62	6800000	192568	395500	196147	41660	112890	2
63	6432107	417350	6985000	500000	76147	34090	1
64	6521646	55778	1672500	84256	0	0	2
65	6419800	290874	1727000	881485	0	0	1
66	6561200	19200	550500	100000	140000	0	2
67	6788000	0	171500	138182	0	0	2
68	6722197	0	478500	471560	277707	113500	3
69	6549400	14000	116000	194245	50000	0	2
70	6533635	7340	234500	110022	0	115500	2
71	6169700	86330	531500	100932	58435	0	2
72	6000000	450300	1243500	207336	0	0	2
73	6000000	127760	500000	84400	0	41500	2
74	6000000	39104	485400	64926	0	0	2
75	6020016	51553	195000	120000	0	0	2
76	5944919	49446	229000	103858	0	67500	2
77	5619000	243276	2442000	200000	274582	299443	3
78	5761400	71812	239000	161867	0	0	2
79	5700000	38400	245000	80638	0	0	2
80	5500000	431400	1783000	1741232	176880	110890	1
81	5860000	0	377500	0	0	0	2

82	5637013	9600	410500	150000	96500	0	2
83	5553781	50000	954500	211150	40000	80750	2
84	5500000	196570	275000	271168	54794	111100	2
85	5313187	26160	2162350	711757	0	31355	1
86	5408575	39460	532000	120730	0	0	2
87	5341028	56080	508500	69816	0	0	2
88	5200000	537810	183500	704401	0	139300	3
89	5161669	33880	271500	30312	0	0	2
90	5024969	113730	468600	200139	71795	0	2
91	5000000	79100	460500	200000	0	0	2
92	4944004	160362	324300	100000	0	97900	2
93	5000000	231679	0	270259	0	63950	2
94	4897521	57692	247000	52516	0	32000	2
95	5000000	0	197000	258638	297395	0	2
96	4887963	91200	0	64300	0	0	2
97	5046926	0	247000	0	0	0	2
98	4799278	63861	192500	73595	0	0	2
99	4803789	25920	237500	119419	0	108925	2
100	4734969	7850	647500	134056	0	68250	2
101	4753545	17000	225500	100000	0	0	2
102	4700000	208205	382000	222780	163878	55800	2
103	4715302	0	463850	92157	0	0	2
104	4577690	46383	325000	76272	0	51950	2
105	4532576	32058	475500	71630	0	0	2
106	4550000	62046	271000	127750	0	0	2
107	4582673	0	230000	0	78603	0	2
108	4500000	0	0	66000	0	0	2
109	4067400	273542	2076500	350000	0	0	1
110	4326000	0	388000	0	0	0	2
111	4159100	30520	231500	100000	0	75800	2
112	4000000	56420	2317600	372953	0	0	1
113	3985000	236039	303500	124311	42160	0	2
114	4147400	0	231500	85500	0	0	2
115	1500000	114963	504000	166597	0	66300	2
116	4000000	181572	205000	200000	0	0	2
117	3980500	30000	155500	100000	0	0	2
118	4000000	66224	0	461000	0	12000	2
119	3914700	138090	0	63021	0	56740	2
120	3677132	67760	2386650	299871	68130	52760	1
121	3715300	62243	869100	100000	0	0	2
122	3640500	6000	174500	39058	0	0	2
123	3600000	65446	173000	91875	0	0	2
124	3500000	169248	211500	51786	0	0	2

125	3582034	0	353500	75000	0	0	2
126	3467600	35702	155000	115905	92700	82900	2
127	3173700	17856	205000	100000	0	0	2
128	3000000	101000	2533000	128546	50000	80300	1
129	3000000	123380	365000	200000	68312	87460	2
130	3000000	72820	242000	86046	79635	72700	2
131	450000	167230	231500	200000	0	0	2
132	3000000	82750	329500	78600	0	0	2
133	3000000	74180	230000	102664	43665	0	2
134	420000	43351	652500	42335	0	0	2
135	0	88404	234800	88640	58160	0	2
136	500000	42862	302500	16200	0	0	2
137	3000000	7106	478500	83000	0	0	2
138	500000	9030	155000	24010	0	32000	2
139	564500	8000	162500	25000	0	0	2
140	454000	6000	134000	30000	0	0	2
141	0	120380	0	203500	58533	0	2
142	3000000	141554	0	103175	0	8000	2
143	0	21986	0	120735	0	79900	2
144	2850000	145360	390250	324737	0	55000	2
145	0	0	180500	0	0	0	2
146	2951900	0	0	0	0	0	2
147	2744400	97376	484000	158623	0	130900	2
148	2868739	0	278500	25000	0	0	2
149	2590000	1318312	227700	127113	76641	195484	3
150	2600000	372255	162500	536117	58440	51640	2
151	2600000	103750	205100	272250	0	0	2
152	2500000	50907	254500	115367	86728	81540	2
153	2500000	54917	147500	373236	152472	80875	2
154	2500000	52780	225500	246170	0	0	2
155	2434200	41971	231500	26336	0	32260	2
156	2500000	0	1048800	27833	0	0	2
157	2500000	5000	0	0	0	0	2
158	2450000	0	0	49143	0	0	2
159	2361510	0	0	0	0	0	2
160	2056160	24992	164000	65253	0	104000	2
161	2035000	69009	189500	31574	0	0	2
162	2000000	53737	268000	100000	0	59450	2
163	2000000	43902	233000	47457	0	15000	2
164	2000000	27200	212500	100000	0	0	2
165	2000000	10089	200000	67658	46970	0	2
166	2000000	0	173000	84900	0	0	2
167	1900000	8010	418000	50000	0	0	2

168	1700000	80693	186500	177910	0	0	2
169	1800000	0	0	0	0	0	2
170	1500000	16949	231500	90000	0	0	2
171	1500000	35009	179000	79156	0	53780	2
172	1504050	0	254000	0	0	0	2
173	1400000	17376	192500	30000	0	0	2
174	1350000	101924	489000	23118	0	0	2
175	1300000	490358	189500	340940	0	0	2
176	1300000	36220	0	250437	0	0	2
177	1200000	209425	234250	83086	0	0	2
178	1000000	116352	234500	40858	0	0	2
179	1000000	11664	53000	42404	0	51620	2
180	817000	48221	403500	76000	0	0	2
181	1874000	0	152900	0	0	0	2
182	5000000	14124	2202000	393793	0	6616	1
183	10000000	50000	1999500	498019	50000	0	1
184	1330000	0	123500	27000	0	0	2
185	1200000	0	174000	80000	0	0	2
186	13500000	45644	392500	595216	745995	52090	3
187	5000000	57574	218800	400000	172000	0	2
188	2949800	125000	0	134989	0	65784	2
189	10000000	404500	1758500	406000	503315	220000	3
190	4900000	0	288900	0	0	0	2
191	17981706	0	4515000	0	430347	0	1
192	13287760	77620	1808000	86427	267808	125000	3
193	7069800	23068	419500	300261	601898	0	3
194	3000000	140000	288500	85000	261000	22300	2
195	11052142	62500	799500	105836	0	200000	3
196	10000000	0	67000	150000	0	0	2
197	8287027	68265	228500	46038	53790	0	2
198	1000000	12758	287500	105428	0	65500	2
199	8211600	39884	706000	117000	57000	0	2
200	4742000	5000	836500	58498	0	0	2
201	3418690	102440	176000	136551	0	29640	2
202	9200000	59975	449625	323865	252915	66900	2
203	2000000	23000	179000	95779	0	0	2
204	16472633	450000	1300000	200000	170000	15000	2
205	31461167	239400	2762750	1020620	223412	190000	1
206	1500000	78956	137000	31281	0	31600	2
207	9368107	14476	464500	190600	143800	27000	2
208	4959900	633296	1385000	412691	0	132700	3
209	1800000	105340	0	193685	0	0	2
210	11418527	389700	1303500	209800	36320	59310	2

211	1200000	71138	171500	79689	0	39400	2
212	8666532	188150	2475000	189865	44308	57000	1
213	14589300	57725	2728000	669194	34345	0	1
214	1500000	75616	182000	60000	0	50000	2
215	1700000	139345	388000	68299	0	39000	2
216	13850000	713420	1838000	1000000	151220	42545	1
217	2700000	75855	234800	100000	0	60000	2
218	6274966	127891	388500	183263	0	0	2
219	20000000	0	2221000	725000	0	600000	3
220	2906300	19185	174500	98060	0	22000	2
221	3230800	6000	174500	76145	0	0	2
222	1500000	63040	420000	150000	0	42500	2
223	5527095	60603	0	53864	0	45630	2
224	8000000	653701	1918000	500000	100000	100000	1
225	5378034	98318	578000	180301	0	234000	3
226	2750000	16512	517000	135000	0	0	2
227	3091683	85000	186000	250000	0	0	2
228	5058400	24288	215000	85145	0	0	2
229	5000000	915085	2349300	583952	443778	0	1
230	4244892	45341	234500	50000	0	0	2
231	1500000	73568	580000	22000	0	30000	2
232	6600000	0	205000	65000	0	0	2
233	8089296	17570	3479500	400000	55000	60000	1
234	4000000	92000	404500	250000	0	52000	2
235	11730881	210000	1643000	300000	265265	110484	3
236	2500000	214000	825000	200000	45000	0	2
237	2358000	0	180300	156891	0	0	2
238	8400000	117320	1748000	443000	266000	139300	3
239	12349374	59878	3288000	294003	38000	0	1
240	2500000	0	368000	73000	0	0	2
241	8671728	132520	2748000	125983	64758	0	1
242	8408311	371280	2186500	324000	0	73250	1
243	11476000	30840	461500	221965	205007	0	2
244	10000000	402210	2565000	696343	91075	0	1
245	3000000	415220	120500	213351	50000	0	2
246	27137910	443200	1511400	659080	357926	233500	3
247	14380000	312600	1773000	450000	60000	0	1
248	13500000	164100	327000	149926	0	0	2
249	4589077	45626	195500	221965	35553	31100	2
250	8180000	87154	2351500	159643	40448	50800	1
251	22362768	318217	1274500	626984	433611	214200	3
252	22742831	344400	2733000	1611492	162172	0	1
253	9000000	11204	2573000	190546	197700	51050	1

254	8233492	0	384300	0	0	0	2
255	5702822	41200	210570	250000	45790	23000	2
256	16340875	32810	1746800	128705	41360	25500	2
257	4274700	149021	0	231437	75334	0	2
258	6068392	109400	572000	287946	34070	0	2
259	3507852	50348	922000	150000	0	0	2
260	3148370	135800	717200	81000	75000	43800	2
261	11542206	0	481500	200000	31570	50700	2
262	5000000	0	299000	325000	0	109000	2
263	5000000	577194	1705000	450000	50000	65000	1
264	5209900	45800	283000	99804	40070	65200	2
265	1500000	34125	306000	94321	20000	26000	2
266	3500000	113139	474000	395525	59585	0	2
267	8807653	380021	2184500	233390	57746	24600	1
268	2000000	59178	377500	102025	0	16700	2
269	8730531	42000	545000	96417	38360	68700	2
270	3040000	63182	2010000	91713	0	7500	2
271	3736600	23000	500000	30700	0	45000	2
272	50733300	359000	3706000	154461	150000	208880	1
273	8529162	22835	203000	130000	140000	0	2
274	13868394	210000	200000	167555	84140	100000	2
275	7854384	85300	400000	184000	73500	0	2
276	10500000	100000	192500	400000	0	0	2
277	13800000	466140	3321875	672743	150000	49500	1
278	19254265	42102	2199000	467140	35600	106325	1
279	5462206	47280	361300	112055	0	0	2
280	1700000	43000	300000	130000	0	0	2
281	7500000	54925	2914600	209835	0	0	1
282	9461253	516600	2294500	271207	82111	191140	3
283	5262898	0	171800	0	0	0	2
284	35000000	697315	2869200	1637348	304084	0	1
285	3000000	117848	162500	269871	0	0	2
286	13414800	109704	2833200	120495	0	150000	1
287	4735574	55236	1155500	251667	0	0	2
288	7552700	15600	298000	100147	0	0	2
289	4500000	68162	1140800	217289	40310	44000	2
290	11387284	162450	2304410	350000	60000	0	1
291	10000000	282606	600000	400000	0	0	2
292	19740000	180000	3118500	513190	74000	26850	1
293	6794560	35752	1443000	165187	77781	0	2
294	9306800	43340	2045500	377057	35709	317660	3
295	2500000	6000	317500	125000	0	0	2
296	5908502	39630	0	50000	0	15000	2

297	3959633	66800	352000	125000	0	0	2
298	6000000	0	625500	78952	0	121800	2
299	4747528	42238	601500	107503	0	0	2
300	1200000	41600	111500	78500	0	0	2
301	7417478	20281	492000	124000	114000	0	2
302	4000000	63180	593500	143297	171570	0	2
303	5200000	269563	210500	320134	61114	166845	3
304	18544450	149624	312600	500000	60000	455750	3
305	12140748	37150	350000	424054	35000	100000	2
306	1500000	67000	221000	40000	0	0	2
307	10621152	58804	355000	142190	0	0	2
308	5000000	334752	156500	495578	146000	15645	2
309	16426340	49184	2324500	99129	43335	0	1
310	7000000	49270	3187000	290000	0	83000	1
311	4431034	0	669700	77000	0	0	2
312	5000000	233369	1800000	867016	193472	197000	3
313	8427182	74385	1106500	77570	0	165000	2
314	8985430	479910	3708000	750000	40000	60800	1
315	2250000	0	207000	50000	0	0	2
316	7989977	21450	1235000	99600	0	0	2
317	3500000	24427	190500	115000	33000	0	2
318	8975139	40900	750000	44000	0	75000	2
319	5358000	0	195500	242240	0	68600	2
320	3000000	455370	938000	571119	236873	18584	1
321	3587840	15035	408000	72077	0	0	2
322	2862300	65199	195500	71000	85220	0	2
323	4249962	40575	391000	95716	0	0	2
324	8289900	24336	434175	70000	0	0	2
325	4605646	74680	190000	101951	0	0	2
326	5000000	12456	303000	150000	0	0	2
327	19020241	349030	3846500	800000	212420	133500	1
328	8200000	57254	227000	200000	0	0	2
329	6926010	101860	252500	339322	0	0	2
330	7465668	61250	1375500	109287	42985	0	2
331	7678033	56179	428500	362933	0	0	2
332	9200802	10840	455500	68499	0	44600	2
333	5671100	354660	2937750	119182	0	48827	1
334	2250000	79000	205000	97335	0	0	2
335	4929500	232937	1446000	352836	304821	150700	3
336	2400000	34239	204500	111429	41170	45500	2
337	38000000	284089	2932000	500000	0	0	1
338	1825000	10736	143500	0	0	0	2
339	8203480	32776	419750	275793	0	32000	2

340	4500000	40000	0	150000	261800	0	2
341	18101880	57725	162500	568944	776050	0	3
342	8409152	521400	1507850	21040	42160	37140	2
343	12257898	101000	2655400	270000	0	50000	1
344	4000000	39170	432250	51851	0	49505	2
345	2000000	170000	203000	100000	0	0	2
346	1810100	32375	394000	126000	0	0	2
347	15995000	55736	746500	210000	32160	0	2

Jumlah anggota *cluster* 1 = 68
Jumlah anggota *cluster* 2 = 246
Jumlah anggota *cluster* 3 = 33