OPTIMASI FUNGSI KEANGGOTAAN *FUZZY* MENGGUNAKAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) PADA SISTEM INFERENSI *FUZZY* PENENTUAN JURUSAN SISWA SMA

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

Prisdhika Juningdiyah

0910960012

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

OPTIMASI FUNGSI KEANGGOTAAN *FUZZY* MENGGUNAKAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) PADA SISTEM INFERENSI *FUZZY* PENENTUAN JURUSAN SISWA SMA

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

Prisdhika Juningdiyah

0910960012

Skripsi ini telah disetujui oleh dosen pembimbing

pada tanggal 11 Juni 2014

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Candra Dewi, S.Kom., M.Sc.	**Indriati, ST., M.Kom.**
NIP. 19771114 200312 2 001	**NIK. 831013 06 1 2 0035**

LEMBAR PENGESAHAN

OPTIMASI FUNGSI KEANGGOTAAN *FUZZY* MENGGUNAKAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) PADA SISTEM INFERENSI *FUZZY* PENENTUAN JURUSAN SISWA SMA

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

Prisdhika Juningdiyah

0910960012

Skripsi ini telah diuji dan dinyatakan lulus

pada tanggal 27 Juni 2014

Dosen Penguji I,

Novanto Yudistira, S.Kom., M.Sc.
NIK. 83110 16 1 1 0425

Dosen Penguji II,

Dian Eka Ratnawati, S.Si., M.Kom
NIP. 19730619 200212 2 001

Dosen Penguji III,

Wijaya Kurniawan, S.T., M.T.
NIK. 820125 16 1 1 0418

Mengetahui,
Ketua Program Studi Informatika/Ilmu Komputer,

Drs. Marji, MT.
NIP. 19670801 199203 1 001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	: Prisdhika Juningdiyah
NIM	: 0910960012
Program Studi	: Informatika/Ilmu Komputer
Jurusan	: Informatika/Ilmu Komputer
Fakultas	: Program Teknologi Informasi dan Ilmu Komputer
Penulis skripsi berjudul	: Optimasi Fungsi Keanggotaan *Fuzzy* Menggunakan Algoritma *Particle Swarm Optimization* (Pso) Pada Sistem Inferensi *Fuzzy* Penentuan Jurusan Siswa SMA

Dengan ini menyatakan bahwa:

1. Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran dan penuh tanggung jawab dan digunakan sebagaimana mestinya.

Malang, Juni 2014

Yang menyatakan,

Prisdhika Juningdiyah

NIM. 0910960012

Kata Pengantar

Segala puji syukur penulis panjatkan hanya bagi Allah SWT, yang atas limpahan rahmat, taufik dan hidayah-Nya, penulis mampu menyelesaikan skripsi dengan judul "Optimasi Fungsi Keanggotaan *Fuzzy* Menggunakan Algoritma *Particle Swarm Optimization* (Pso) Pada Sistem Inferensi *Fuzzy* Penentuan Jurusan Siswa SMA".

Tugas akhir ini dikerjakan demi memenuhi salah satu syarat guna memperoleh gelar Sarjana Komputer pada program studi Ilmu Komputer/ Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya-Malang. Penulis menyadari bahwa tugas akhir ini bukanlah tujuan akhir dari belajar karena belajar adalah sesuatu yang tidak terbatas.

Terselesaikannya skripsi ini tentunya tak lepas dari dorongan dan bantuan dari berbagai pihak. Oleh karena itu, penulis ingin mengungkapkan rasa terima kasih yang sebesar-besarnya kepada semua pihak atas bantuan dan dukungan selama penulisan proposal skripsi, antara lain:

1. Ibu, Bapak dan seluruh keluarga yang memberikan motivasi dan doa
2. Candra Dewi, S.Kom, MSc yang telah memberikan bimbingan, arahan, dan nasehat untuk terselesaikannya tugas akhir ini
3. Indriati, ST, M.Kom yang telah memberikan bimbingan, arahan, dan nasehat untuk terselesaikannya tugas akhir ini.
4. Seluruh Bapak-Ibu Dosen yang telah memberikan ilmunya selama ini.
5. Ratna Putri P.S. yang selalu bersama-sama memecahkan permasalahan dalam menyelesaikan tugas akhir ini.
6. Seluruh teman-teman Ilmu Komputer, terutama teman-teman Ilkom B 2009.
7. Pihak SMA Negeri 3 Malang yang telah membantu memberikan data untuk tugas akhir ini.
8. Dan seluruh pihak yang telah membantu baik secara langsung ataupun tidak yang tidak dapat disebutkan satu-persatu

Semoga Allah SWT membalas kebaikan dan ketulusan semua pihak yang telah membantu menyelesaikan skripsi ini dengan melimpahkan rahmat dan karunia-Nya.

Penulis sadar bahwa skripsi ini masih memiliki banyak kekurangan, kritik dan saran yang bersifat membangun sangat diharapkan untuk memperbaiki skripsi ini agar menjadi lebih baik lagi.

Malang, Juni 2014

Penulis

OPTIMASI FUNGSI KEANGGOTAAN *FUZZY* MENGGUNAKAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) PADA SISTEM INFERENSI *FUZZY* PENENTUAN JURUSAN SISWA SMA

ABSTRAK

Penjurusan merupakan bagian dari program pelaksanaan pendidikan yang berfungsi untuk mengarahkan siswa sesuai dengan bakat dan minatnya. Dengan penjurusan yang tepat dan sesuai dengan bakat serta minat siswa diharapkan prestasi siswa akan meningkat pula. Salah satu metode yang dapat digunakan sebagai metode untuk membuat sebuah sistem pendukung keputusan penentuan jurusan siswa adalah dengan menggunakan sistem inferensi *fuzzy*. Dalam beberapa penelitian dikatakan bahwa optimasi fungsi keanggotaan *fuzzy* dilakukan sebagai usaha untuk meningkatkan hasil dari sistem inferensi *fuzzy*. Pada penelitian ini dilakukan pencarian rekomendasi jurusan siswa SMA dengan menggunakan sistem inferensi *fuzzy* model Mamdani dengan algoritma *Particle Swarm Optimization* (PSO) sebagai metode yang digunakan untuk melakukan optimasi fungsi keanggotaan *fuzzy*. Data yang digunakan pada penelitian ini merupakan data penentuan jurusan siswa SMA Negeri 3 Malang kelas X pada tahun 2013 yang terdiri atas tiga belas parameter berupa data nilai dan minat siswa. Dari hasil pengujian didapatkan kombinasi parameter PSO terbaik untuk penelitian ini adalah : c_1= 1, c_2=1.5, ω_{min}= 0.5, ω_{max}= 0.7, jumlah partikel=80, dan iterasi maksimum=300, sedangkan akurasi terbaik hasil sistem inferensi *fuzzy* sebesar 96%.

Kata kunci: penentuan jurusan siswa SMA, sistem inferensi *fuzzy* Mamdani, optimasi fungsi keanggotaan *fuzzy*, Particle Swarm Optimization (PSO)

OPTIMIZING FUZZY MEMBERSHIP FUNCTIONS USING PARTICLE SWARM OPTIMIZATION (PSO) ON FUZZY INFERENCE SYSTEM DETERMINATION MAJOR'S OF SENIOR HIGH SCHOOL STUDENTS

ABSTRACT

Defining major is a part of educational process that aim to direct student according to their talent and interest. With the right major according to their talent and interest, students are expected to develop their achievement. One method that can be used as a decision support in defining high school major is fuzzy inference system. Some research prove that optimization of fuzzy membership function can be applied as a method in attempt to improve the results of the fuzzy inference system. This research give a recommendation for high school to determine their student's major by using Mamdani fuzzy inference system for fuzzy inference model's and Particle Swarm Optimization (PSO) algorithm as method for optimizing membership function in its implementation. The data that is used in this research is 10^{th} grade student of SMAN 3 Malang major determination in 2013 consisting of thirteen parameters in the form of data values and interests of students. The test results obtained the best combination of PSO's parameters for this research is: c_1= 1, c_2=1.5, ω_{min}= 0.5, ω_{max}= 0.7, number of particles= 80, and maximum iteration= 300, while the best accuracy of fuzzy inference system is 96%.

Key phrases: *High school major determination, Mamdani fuzzy inference system, fuzzy membership function optimization, Particle Swarm Optimization (PSO)*

DAFTAR ISI

Kata Pengantar i
ABSTRAK iii
ABSTRACT iv
DAFTAR ISI v
DAFTAR GAMBAR ix
DAFTAR TABEL xi
DAFTAR LAMPIRAN xiii
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 3
1.3. Batasan Masalah 3
1.4. Tujuan 3
1.5. Manfaat 4
1.6. Sistematika Penulisan 4
BAB II KAJIAN PUSTAKA DAN DASAR TEORI 5
2.1. Penjurusan SMA 5
2.1.1. KKM(Kriteria Ketuntasan Minimal) 6
2.2. Logika Fuzzy 7
2.2.1. Pengertian Logika *Fuzzy* 7
2.2.2. Himpunan Fuzzy 7
2.2.3. Fungsi Keanggotaan 8
2.2.4. Sistem Inferensi *Fuzzy* 11
2.2.4.1. Metode Mamdani 12
2.3. Algoritma Particle swarm optimization 13
2.3.1. Nilai *cost* 15
2.4. Akurasi 16
BAB III METODE PENELITIAN DAN PERANCANGAN 17

3.1. Studi Literatur 18
3.2. Data Penelitian 18
3.3. Analisa dan Perancangan Sistem 18
3.3.1. Deskripsi Umum Sistem 18
3.3.2. Optimasi fungsi keanggotaan 19
3.3.3. Sistem Inferensi Fuzzy Mamdani 32
3.3.4. Analisa dan Perancangan Antarmuka 42
3.3.4.1. Antarmuka Proses Optimasi Fungsi Keanggotaan 42
3.3.4.2. Antarmuka Pengujian Menggunakan Sistem Inferensi *Fuzzy* . 43
3.3.4.3. Antarmuka Tampilan Fungsi Keanggotaan 45
3.4. Perhitungan Manual 46
3.4.1. Proses Optimasi Fungsi Keanggotaan Menggunakan Algoritma *Particle Swarm Optimization* 46
3.4.2. Sistem Inferensi *Fuzzy* Mamdani 58
3.5. Perancangan Pengujian dan Analisis 65
BAB IV IMPLEMENTASI 69
4.1. Lingkungan Implementasi 69
4.1.1. Lingkungan Implementasi Perangkat Keras 69
4.1.2. Lingkungan Implementasi Perangkat Lunak 69
4.2. Implementasi Program 69
4.2.1. Proses Optimasi Fungsi Keanggotaan Menggunakan algoritma Particle Swarm Optimization 77
4.2.1.1. Proses Optimasi Seluruh Fungsi Keanggotaan Awal 78
4.2.1.2. Proses Optimasi Fungsi Keanggotaan Satu Parameter 78
4.2.1.3. Proses Optimasi dengan Algoritma *Particle Swarm Optimization* 80
4.2.1.4. Proses Penentuan Partikel Acak 82
4.2.1.5. Proses Inisialisasi 83
4.2.1.6. Proses Perhitungan berat inersia (ω) 83

4.2.1.7. Proses Proses Hitung *Cost* Partikel 84
4.2.1.8. Proses Cari *Cost* Pbest 84
4.2.1.9. Proses Cari *Cost* Gbest 85
4.2.1.10. Proses Pencarian Pbest 85
4.2.1.11. Proses Pencarian Gbest 86
4.2.1.12. Proses Perhitungan Kecepatan Baru 86
4.2.1.13. Proses Perhitungan Posisi Baru 87
4.2.2. Sistem Inferensi *Fuzzy* Mamdani 87
4.2.2.1. Proses Fuzzifikasi 88
4.2.2.2. Proses Pemilihan Aturan 89
4.2.2.3. Proses Implikasi 90
4.2.2.4. Proses Komposisi 91
4.2.2.5. Proses Defuzzifikasi 92
4.2.2.6. Proses Perhitungan Tingkat Akurasi 93
4.2.2.7. Proses Perhitungan Persen Akurasi 94
4.2. Implementasi Antarmuka 94
4.2.1. Antarmuka Proses Optimasi Fungsi Keanggotaan 94
4.2.2. Antarmuka Pengujian Menggunakan Sistem Inferensi *Fuzzy* 96
4.2.3. Antarmuka Tampilan Fungsi Keanggotaan 97
BAB V ANALISA HASIL DAN PEMBAHASAN 99
5.1. Implementasi Pengujian 99
5.1.1. Skenario Pengujian 99
5.1.1.1. Skenario Pengujian Parameter Optimasi 99
5.1.1.2. Skenario Pengujian Sistem Inferensi *Fuzzy* Mamdani 100
5.1.2. Hasil Pengujian 100
5.1.2.1. Hasil ujicoba untuk parameter optimasi 100
5.1.2.2. Hasil ujicoba untuk sistem inferensi *fuzzy* Mamdani 103
5.2. Analisa Hasil Pengujian 104
5.2.1. Analisa Hasil Pengujian Parameter PSO 104

5.2.2. Analisa Hasil Pengujian Sistem Inferensi *Fuzzy* 107
BAB VI PENUTUP 109
6.1. Kesimpulan 109
6.2. Saran 109

DAFTAR GAMBAR

Gambar 2.1. Kurva segitiga 9
Gambar 2.2. Kurva trapezium 9
Gambar 2.3. Karakteristik fungsi *gauss* 10
Gambar 2.4. Karakteristik fungsi *generalized bell* 10
Gambar 2.5. Kurva *sigmoid* 11
Gambar 2.6. Struktur Sistem Inferensi *Fuzzy* 11
Gambar 2.7 Fungsi implikasi *MIN* 12
Gambar 2.8. Algoritma dasar PSO 15
Gambar 3.1. Desain Penelitian 17
Gambar 3.2. Gambaran umum sistem 19
Gambar 3.3. Alur Proses Sistem Optimasi 20
Gambar 3.4.a. Proses Optimasi Fungsi Keanggotaan dengan menggunakan *Particle Swarm Optimization* (a) 21
Gambar 3.4.b. Proses Optimasi Fungsi Keanggotaan dengan menggunakan *Particle Swarm Optimization* (b) 22
Gambar 3.5. Proses Inisialisasi Awal 24
Gambar 3.6. Hitung *cost* partikel 25
Gambar 3.7. Proses cari Pbest 26
Gambar 3.8. Proses cari *cost* Pbest 28
Gambar 3.9. Proses mencari *cost* 29
Gambar 3.10. Proses Mencari Gbest 31
Gambar 3.11. Sistem inferensi *fuzzy* 33
Gambar 3.12. Proses fuzzifikasi 34
Gambar 3.13. Proses pemilihan aturan 35
Gambar 3.14. Proses implikasi 37
Gambar 3.15. Proses komposisi 39
Gambar 3.16. Proses Defuzzifikasi 41
Gambar 3.17. Antarmuka Proses Optimasi Fungsi Keanggotaan 42
Gambar 3.18.b. Antarmuka Proses Pengujian Menggunakan Sistem Inferensi *Fuzzy* 44

Gambar 3.19.a. Antarmuka Tampilan Fungsi Keanggotaan Awal 45
Gambar 3.19.b. Antarmuka Tampilan Fungsi Keanggotaan Optimal 45
Gambar 4.1. *Class* Diagram *Package* Kode.Excel 73
Gambar 4.2. *Class* Diagram *Package* Kode.Tipe 74
Gambar 4.3. *Class* Diagram *Package* Kode.PSO 75
Gambar 4.4. Class Diagram Package Kode.Fuzzy 76
Gambar 4.5. Antarmuka Proses Optimasi Fungsi Keanggotaan..... 95
Gambar 4.6.b. Antarmuka Proses Pengujian Menggunakan Sistem Inferensi *Fuzzy* 96
Gambar 4.7.a. Antarmuka Tampilan Fungsi Keanggotaan Awal 98
Gambar 4.7.b. Antarmuka Tampilan Fungsi Keanggotaan Optimal 98
Gambar 5.1. Grafik hubungan c_1 dan nilai *cost* rata-rata..... 104
Gambar 5.2. Grafik hubungan c_2 dan nilai *cost* rata-rata..... 105
Gambar 5.3. Grafik hubungan inersia minimum (ωminimum) dan nilai *cost* rata-rata 105
Gambar 5.4. Grafik hubungan inersia maksimum ($\omega_{maximum}$) dan nilai *cost* rata-rata 106
Gambar 5.5. Grafik hubungan jumlah partikel dan nilai *cost* rata-rata..... 106
Gambar 5.6. Grafik hubungan iterasi maksimum dan nilai *cost* rata-rata 107

DAFTAR TABEL

Tabel 3.1. Contoh data latih yang digunakan 47
Tabel 3.2. Contoh data yang telah ditransformasi 47
Tabel 3.3. Partikel acak untuk proses optimasi 48
Tabel 3.4. Evaluasi nilai fungsi tujuan iterasi ke-0 49
Tabel 3.5. Pbest iterasi ke-0 50
Tabel 3.6. Hasil perhitungan $v_{(k+1)}$ iterasi 0 51
Tabel 3.7. Hasil perhitungan $x_{(k+1)}$ iterasi 0 51
Tabel 3.8. Evaluasi nilai fungsi tujuan iterasi 1 52
Tabel 3.9. Pbest iterasi ke-1 53
Tabel 3.10. Hasil perhitungan $v_{(k+1)}$ iterasi ke-6 54
Tabel 3.11. Hasil perhitungan $x_{(k+1)}$ iterasi ke-6 54
Tabel 3.12. Evaluasi nilai fungsi tujuan iterasi ke-7 55
Tabel 3.13. Pbest iterasi ke-7 55
Tabel 3.14. Daftar hasil perhitungan nilai *cost* dan Gbest seluruh 56
Tabel 3.15. Partikel untuk parameter NIPA 57
Tabel 3.16. Partikel acak untuk parameter NIPS 57
Tabel 3.17. Hasil optimasi fungsi keanggotaan semua parameter 58
Tabel 3.18. Tabel pencarian parameter c_1 dan c_2 66
Tabel 3.19. Tabel pencarian parameter inersia minimum (ω_{min}) dan inersia maksimum (ω_{max}) 66
Tabel 3.20. Tabel pencarian parameter jumlah partikel 67
Tabel 3.21. Tabel pencarian parameter iterasi maksimum 67
Tabel 3.19. Tabel akurasi hasil pengujian 68
Tabel 4.1. Kelas-kelas pembentuk aplikasi dalam *package* Antarmuka 70
Tabel 4.2. Kelas-kelas pembentuk aplikasi dalam *package* Kode.Excel 71
Tabel 4.3. Kelas-kelas pembentuk aplikasi dalam *package* Kode.Tipe 71
Tabel 4.4. Kelas-kelas pembentuk aplikasi dalam *package* Kode.PSO 71
Tabel 4.5. Kelas-kelas pembentuk aplikasi dalam *package* Kode.Fuzzy 72
Tabel 4.6. *Method* penyusun `PSO.java` 77
Tabel 4.7. Method penyusun `ProsesPSO.java` 77

Tabel 4.8. *Method* penyusun `Fuzzy.java` 87
Tabel 4.9. *Method* penyusun `Akurasi.java` 88
Tabel 5.1. Hasil ujicoba pencarian parameter c1 dan c2..... 101
Tabel 5.2. Hasil ujicoba pencarian parameter ωmin dan ωmax..... 102
Tabel 5.3. Hasil ujicoba pencarian parameter jumlah partikel..... 102
Tabel 5.4. Hasil ujicoba pencarian parameter iterasi maksimum 103
Tabel 5.5. Hasil ujicoba sistem inferensi *fuzzy* 103

DAFTAR LAMPIRAN

Lampiran 1: Data Penjurusan SMA Negeri 3 Malang Tahun 2013 113
Lampiran 2: Tabel Hasi l Ujicoba 118

BAB I
PENDAHULUAN

1.1. Latar Belakang

Penjurusan merupakan bagian dari program pelaksanaan pendidikan yang berfungsi untuk mengarahkan siswa sesuai dengan bakat dan minatnya. Penjurusan pada Sekolah Menengah Atas (SMA) dimulai pada semester satu kelas XI. Penjurusan yang tepat dan sesuai dengan kemampuan serta minat siswa sangat diperlukan Dengan adanya penjurusan, diharapkan setiap siswa dapat lebih fokus pada kemampuan yang telah dimiliki [ABS-13].

Implementasi teknologi informasi dalam dunia pendidikan dapat digunakan dalam berbagai bidang, salah satunya adalah sistem yang dapat membantu proses pemilihan jurusan di SMA [SAA-13]. Salah satu metode yang dapat digunakan sebagai metode untuk membuat sebuah sistem pendukung keputusan adalah dengan logika *fuzzy*, yaitu dengan sistem inferensi *fuzzy*. Sistem inferensi *fuzzy* adalah suatu kerangka komputasi yang didasarkan pada teori himpunan *fuzzy*, aturan *fuzzy* dan penalaran *fuzzy*.

Logika fuzzy merupakan salah satu komponen pembentuk *soft computing* yang diperkenalkan oleh Lotfi A. Zadeh pada tahun 1965. Dasar dari logika *fuzzy* adalah teori himpunan *fuzzy*. Sebuah himpunan fuzzy memiliki tiga komponen utama yaitu derajat keanggotaan, nilai domain yang memungkinkan untuk himpunan tersebut, dan fungsi keanggotaan. Fungsi keanggotaan merupakan kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah melalui pendekatan fungsi [KSH-10].

Dalam beberapa tahun terakhir telah diteliti beberapa metode untuk mengoptimasi fungsi keanggotaan. Optimasi fungsi keanggotaan tersebut dilakukan sebagai cara untuk meningkatkan kinerja dari logika fuzzy. Sebagaimana yang telah dilakukan pada [SAH-10] optimasi fungsi keanggotaan menggunakan algoritma genetika terbukti dapat meningkatkan kinerja dari logika

fuzzy dengan menurunkan kesalahan atau error yang terjadi pada hasil klasifikasi sebesar 1,8%. Dengan kata lain optimasi pada fungsi keanggotaan memberikan hasil yang lebih akurat pada hasil klasifikasi menggunakan logika *fuzzy*.

Metode lain yang dapat digunakan untuk melakukan optimasi terhadap fungsi keanggotaan *fuzzy* adalah algoritma *Particle Swarm Optimization* (PSO). Algoritma *Particle Swarm Optimization* (PSO) merupakan algoritma optimasi yang diperkenalkan Kennedy dan Eberhart pada tahun 1995 sebagai pencarian stokastik melalui masalah ruang n-dimensi yang bertujuan untuk meminimalkan (atau memaksimalkan) dari fungsi tujuan dari masalah [PES-10]. Jika dibandingkan dengan algoritmqa genetika, algoritma PSO memiliki beberapa kelebihan antara lain lebih sederhana dan lebih sedikit parameter yang harus diatur [RVA-13]. Selain itu, PSO menghasilkan konvergensi lebih cepat bila dibandingkan dengan algoritma genetika, karena keseimbangan antara eksplorasi dan eksploitasi di ruang pencarian [PES-10].

Penelitian tentang optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization* antara lain dilakukan oleh Elijabah dan Omizegba pada tahun 2010 [OEA-10]. Pada penelitian tersebut nilai parameter yang akan dioptimalkan menggunakan PSO adalah titik pusat, kiri dan kanan masing-masing kaki dari fungsi keanggotaan. Setelah partikel mencapai hasil yang optimal, nilai parameter akan dioptimalkan oleh PSO dan akan digunakan untuk membangun fungsi keanggotaan fuzzy yang baru. Hasil dari penelitian tersebut menunjukkan bahwa untuk optimasi fungsi keanggotaan *fuzzy* menggunakan PSO dengan menggunakan masukan atau *input* dari ahli memberikan kinerja yang lebih baik daripada optimasi menggunakan algoritma genetika, baik secara kecepatan maupun nilai MSE yang dihasilkan.

Berdasarkan paparan informasi tersebut, penulis tertarik untuk melakukan penelitian tentang "Optimasi Fungsi Keanggotaan *Fuzzy* Menggunakan Algoritma *Particle Swarm Optimization* Pada Sistem Inferensi *Fuzzy* Penentuan Jurusan Siswa SMA".

1.2. Rumusan Masalah

1. Bagaimana proses optimasi fungsi keanggotaan *fuzzy* menggunakan algoritma *Particle Swarm Optimization* pada inferensi penentuan jurusan siswa SMA?
2. Bagaimana kombinasi nilai parameter dari algoritma *Particle Swarm Optimization* yang menghasilkan fungsi keanggotaan optimal?
3. Bagaimana tingkat akurasi hasil sistem inferensi *fuzzy* yang menggunakan fungsi keanggotaan yang telah dioptimasi dengan menggunakan algoritma PSO (*Particle Swarm Optimization*) pada sistem inferensi *fuzzy* penentuan jurusan siswa SMA?

1.3. Batasan Masalah

Penelitian ini dibatasi oleh hal-hal sebagai berikut:

1. Metode sistem inferensi *fuzzy* yang digunakan adalah metode Mamdani
2. Data yang digunakan adalah data penentuan jurusan siswa SMA Negeri 3 Malang tahun 2013
3. Data yang digunakan merupakan data yang sudah memenuhi KKM (Kriteria Ketuntasan Minimal)
4. Parameter yang akan dioptimasi adalah beberapa parameter *input* berupa nilai akademik yang berkaitan dengan penentuan jurusan.

1.4. Tujuan

Tujuan dari penelitian ini antara lain :

1. Menerapkan algoritma *Particle Swarm Optimization* untuk melakukan optimasi fungsi keanggotaan *fuzzy* pada inferensi penentuan jurusan siswa SMA.
2. Mengetahui kombinasi parameter algoritma *Particle Swarm Optimization* untuk menghasilkan fungsi keanggotaan yang optimal.
3. Mengetahui tingkat akurasi hasil inferensi *fuzzy* menggunakan kan fungsi keanggotaan yang telah dioptimasi pada inferensi penentuan jurusan siswa SMA menggunakan algoritma *Particle Swarm Optimization*.

1.5. Manfaat

Manfaat yang diharapkan dari penelitian ini adalah didapatkan sebuah sistem pendukung keputusan bagi sekolah menengah atas dalam menentukan jurusan untuk siswanya. Kemudian penelitian ini juga bisa menjadi pertimbangan agar optimasi fungsi keanggotaan dapat diterapkan dalam sistem inferensi *fuzzy* dengan tujuan membuat kinerja sistem inferensi *fuzzy* menjadi lebih baik. Pada sisi akademisi, penelitian dapat bermanfaat sebagai tambahan referensi guna dikembangkan kearah penelitian yang lebih lanjut.

1.6. Sistematika Penulisan

Sistematika penulisan tugas akhir ini adalah sebagai berikut :

Bab I Pendahuluan – membahas latar belakang masalah, tujuan penelitian, rumusan masalah, batasan masalah dan sistematika penulisan yang akan digunakan dalam penelitian ini.

Bab II Kajian Pustaka dan Dasar Teori – membahas kajian pustaka dan landasan teori yang berhubungan dengan penelitian ini.

Bab III Metodologi Penelitian dan Perancangan Sistem– membahas langkah-langkah serta perancangan sistem yang akan dilakukan dalam penelitian.

Bab IV Implementasi – membahas tentang implementasi algoritma *Particle Swarm Optimization* untuk pembangkitan fungsi keanggotaan *fuzzy* pada sistem inferensi *fuzzy* penentuan jurusan siswa SMA.

Bab V Analisa Hasil dan Pembahasan – membahas tentang hasil pengujian dari implementasi sistem

Bab VI Penutup – Bagian ini berisi kesimpulan yang diperoleh dari penelitian dan saran untuk penelitian selanjutnya.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini berisi kajian pustaka dan dasar teori yang berhubungan dengan implementasi algoritma *Particle Swarm Optimization* dalam membangkitan fungsi keanggotaan pada sistem inferensi fuzzy penentuan jurusan siswa SMA.

2.1. Penjurusan SMA

Penjurusan merupakan bagian dari program pelaksanaan pendidikan yang berfungsi untuk mengarahkan siswa sesuai dengan bakat dan minatnya. Sebagaimana disebutkan dalam Panduan Penyusunan Laporan Hasil Peserta Didik (Berdasarkan KTSP) Sekolah Menengah Atas (SMA), penentuan jurusan bagi peserta didik pada dilakukan mulai akhir semester 2(dua) kelas X dan pelaksanaan penjurusan program dimulai pada semester satu kelas XI. Adapun kriteria penjurusan meliputi hal-hal berikut [DEP-06]:

a. Nilai Akademik

Peserta didik yang naik ke kelas XI dan akan mengambil program tertentu yaitu : Ilmu Pengetahuan IPA (IPA) atau Ilmu Pengetahuan Sosial (IPS) atau Bahasa, boleh memiliki nilai yang tidak tuntas paling banyak 3 (tiga) mata pelajaran pada mata pelajaran yang bukan ciri khas program tersebut. Peserta didik yang naik ke kelas XI, dan yang bersangkutan mendapat nilai tuntas 3 (tiga) mata pelajaran, maka nilai tersebut harus dijadikan dasar untuk menentukan program yang dapat diikuti oleh peserta didik, contoh :

- Apabila mata pelajaran yang tidak tuntas adalah Fisika, Kimia dan Geografi (2 mata pelajaran ciri khas program IPA dan 1 ciri khas program IPS), maka siswa tersebut secara akademik dapat masuk ke program Bahasa.
- Apabila mata pelajaran yang tidak tuntas adalah Bahasa Indonesia, Bahasa Inggris dan Fisika (2 mata pelajaran ciri khas program Bahasa dan 1 ciri khas program IPA), maka siswa tersebut secara akademik dapat masuk ke program IPS.
- Apabila mata pelajaran yang tidak tuntas adalah Ekonomi, Sosiologi, Bahasa Inggris (2 mata pelajaran ciri khas program IPS dan 1 ciri khas

program Bahasa), maka siswa tersebut secara akademik dapat masuk ke program IPA.

- Apabila mata pelajaran yang tidak tuntas adalah Fisika, Ekonomi, dan Bahasa Indonesia (mencakup semua mata pelajaran yang menjadi ciri khas ketiga program di SMA) maka peserta didik tersebut :
  - Perlu diperhatikan minat peserta didik.
  - Perlu diperhatikan prestasi Pengetahuan, Praktik dan Sikap pada mata pelajaran yang menjadi ciri khas program IPA seperti Fisika, Kimia dan Biologi dibandingkan dengan mata pelajaran yang menjadi ciri khas program IPS (Ekonomi, Geografi, Sosiologi) dan dibandingkan dengan mata pelajaran yang menjadi ciri khas program Bahasa (Bahasa Indonesia, Bahasa Inggris). Perbandingan nilai prestasi siswa dimaksud dapat dilakukan melalui program remidial dan diakhiri dengan ujian. Apabila nilai dari setiap mata pelajaran yang menjadi ciri khas program tertentu terdapat nilai prestasi yang lebih unggul daripada program lainnya, maka siswa tersebut dapat dijuruskan ke program yang nilai prestasi mata pelajarannnya lebih unggul tersebut. Apabila antara minat dan prestasi ketiga aspek tidak cocok atau sesuai, wali kelas dengan pertimbangan masukan dari guru Bimbingan dan Konseling dapat memutuskan program apa yang dapat dipilih oleh peserta didik.

b. Minat dan Bakat

Untuk mengetahui potensi dan minat peserta didik dapat dilakukan melalui angket/kuesioner dan wawancara, atau cara lain yang dapat digunakan untuk mendeteksi potensi, minat, dan bakat.

2.1.1. KKM(Kriteria Ketuntasan Minimal)

Salah satu prinsip penilaian pada kurikulum berbasis kompetensi adalah menggunakan acuan criteria, yakni menggunakan criteria tertentu dalam menentukan kelulusan peserta didik. Criteria paling rendah untuk menyatakan peserta didik mencapai ketuntasan dinamakan Kriteria Ketuntasan Minimal (KKM) [DEP-08].

Kriteria ketuntasan minimal ditetapkan oleh satuan pendidikan berdasarkan hasil musyawarah guru mata pelajaran di satuan pendidikan atau beberapa satuan pendidikan yang memiliki karakteristik yang hampir sama [DEP-08].

Kriteria ketuntasan menunjukkan persentase tingkat pencapaian kompetensi sehingga dinyatakan dengan angka maksimal 100(seratus). Angka maksimal 100 merupakan criteria ketuntasan ideal. Target ketuntasan secara nasional diharapkan mencapai minimal 75. Satuan pendidikan dapat memulai dari kriteria ketuntasan minimal dibawah target nasional kemudian ditingkatkan secara bertahap [DEP-08].

2.2. Logika Fuzzy

2.2.1. Pengertian Logika *Fuzzy*

Logika fuzzy adalah sebuah bentuk atau sistem dari logika[COE-05]. Logika fuzzy merupakan salah satu komponen pembentuk *soft computing* yang diperkenalkan oleh Lotfi A. Zadeh pada tahun 1965. Dasar dari logika *fuzzy* adalah teori himpunan *fuzzy*[KSH-10].

2.2.2. Himpunan Fuzzy

Gagasan inti dari logika *fuzzy* terletak pada konsep dari himpunan *fuzzy*. Dalam himpunan fuzzy, sebuah elemen bisa memiliki tiga kondisi, yaitu : bukan anggota dari sebuah himpunan, anggota penuh dari sebuah himpunan, atau anggota parsial dari sebuah himpunan[COE-05].

Pada himpunan tegas (*crisp*), nilai keanggotaan suatu x dalam suatu himpunan A, yang sering ditulis dengan $\mu_A(x)$, memiliki dua kemungkinan, yaitu 0 atau 1. Himpunan *fuzzy* juga memiliki nilai keanggotaan pada interval 0 sampai 1, namun interpretasi keduanya sangat berbeda. Keanggotaan *fuzzy* memeberikan suatu ukuran terhadap pendapat atau keputusan, sedangkan probabilitas mengindikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang [KSH-10].

Himpunan *fuzzy* memiliki dua atribut yaitu [KSH-10] :

Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: muda, parobaya, tua.

Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti: 40, 25, 50, dsb.

2.2.3. Fungsi Keanggotaan

Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaan yang memiliki nilai interval antara 0 dan 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang dapat digunakan untuk merepresentasikan fungsi keanggotaan, diantaranya adalah representasi linear, Representasi Kurva Segitiga, Representasi Kurva Trapesium, Representasi Kurva Bentuk Bahu, Representasi Kurva-S, dan Representasi Kurva Bentuk Lonceng (*Bell Curve*) yang terbagi lagi menjadi Kurva PI, Kurva Beta, dan Kurva Gauss [KSH-10].

1. Fungsi keanggotaan segitiga, disifati oleh parameter {*a*,*b*,*c*} yang didefinisikan seperti pada Persamaan (2-1).

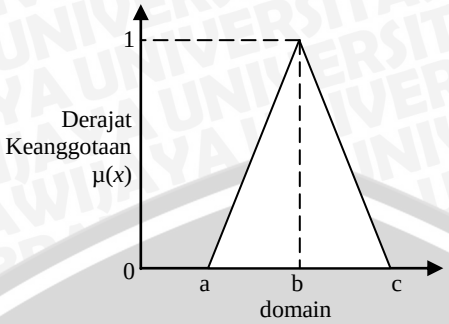
$$segitiga(x;a,b,c)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ \dfrac{c-x}{c-b}, & b\le x\le c\\ 0, & c\le x\end{cases} \qquad (2\text{-}1)$$

dengan : a = batas bawah parameter

b = nilai puncak parameter

c = batas atas parameter

Parameter {*a*,*b*,*c*} (dengan $a < b < c$) yang menentukan koordinat x dari ketiga sudut segitiga tersebut, seperti terlihat pada gambar berikut:



Gambar 2.1. Kurva segitiga

2. Fungsi keanggotaan trapesium, disifati oleh parameter {*a*,*b*,*c*,*d*} yang didefinisikan pada Persmaan (2-2).

$$trapesium(x;a,b,c,d) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases} \qquad (2\text{-}2)$$

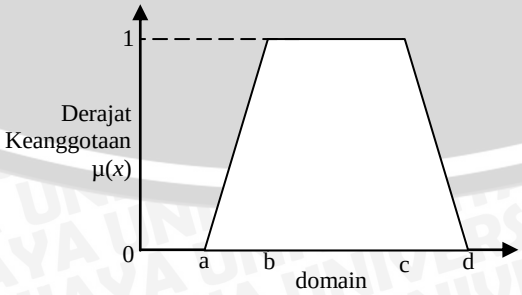
dengan : *a* = batas bawah parameter

b = nilai tertinggi bawah parameter

c = nilai tertinggi atas parameter

d = batas atas parameter

Parameter {*a*,*b*,*c*,*d*} pada kurva trapesium adalah sebagai berikut:



Gambar 2.2. Kurva trapezium

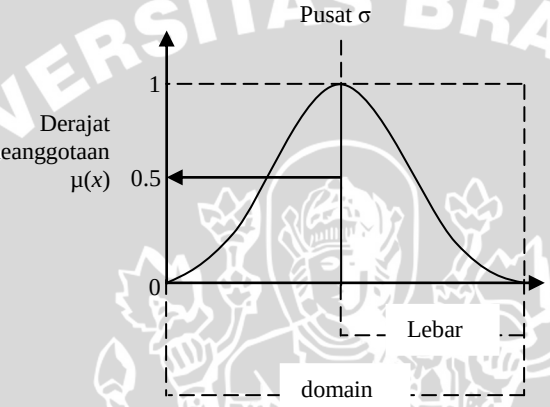
3. Fungsi keanggotaan *gaussian*, disifati oleh parameter $\{c,\sigma\}$ yang didefinisikan pada Persamaan (2-22).

$$gaussian(x;c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \tag{2-3}$$

dengan : c = lebar kurva

σ = nilai domain pusat kurva

Karakteristik fungsi keanggotaan *gauss* ditentukan oleh parameter c dan σ seperti terlihat pada gambar berikut:

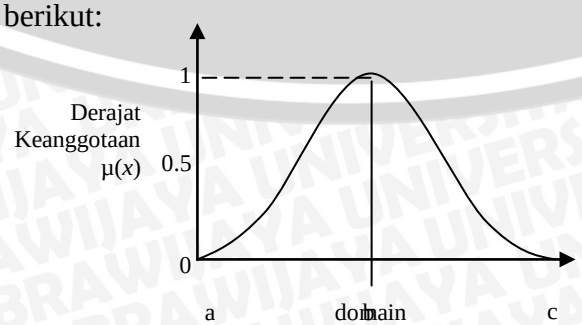


Gambar 2.3. Karakteristik fungsi *gauss*

4. Fungsi keanggotaan *generalized bell*, disifati oleh parameter {*a*,*b*,*c*} yang didefinisikan seperti pada Persamaan (2-4).

$$bell(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{2-4}$$

Parameter *b* selalu positif, supaya kurva menghadap kebawah, seperti terlihat pada gambar berikut:

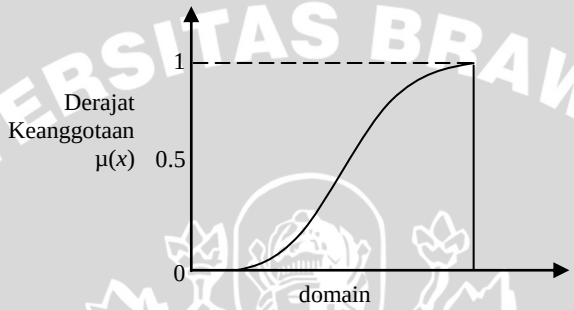


Gambar 2.4. Karakteristik fungsi *generalized bell*

5. Fungsi keanggotaan *sigmoid*, disifati oleh parameter {*a*,*c*} yang didefinisikan seperti pada Persamaan (2-5).

$$sig(x;a,c) = \frac{1}{1+\exp\left[-a\left(x-c\right)\right]} \qquad (2\text{-}5)$$

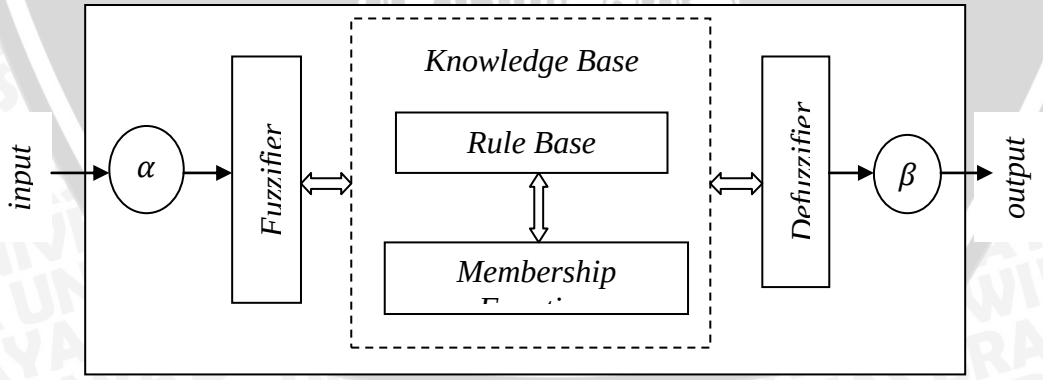
Parameter *a* digunakan untuk menentukan kemiringan kurva pada saat $x = c$. Polaritas dari *a* akan menentukan kurva itu kanan atau kiri terbuka, seperti terlihat pada gambar berikut:



Gambar 2.5. Kurva *sigmoid*

2.2.4. Sistem Inferensi *Fuzzy*

Sistem inferensi *fuzzy* atau biasa disebut *fuzzy inference system* (FIS) merupakan mekanisme yang menghubungkan pengetahuan dasar, penerapan aturan, dan produksi semua hasil [COE-05]. Pada umumnya, susunan lengkap dari sebuah FIS dapat ditunjukkan dengan gambar berikut [OEA-10] :



Gambar 2.6. Struktur FIS (Omizegba, 2010)

Pada gambaran struktur FIS tersebut, α dan β merupakan factor skala dari masing-masing *input* dan *output*, *fuzzifier* atau fuzzifikasi adalah proses konversi

dari *input* yang merupakan bilangan *crisp* menjadi data *fuzzy* untuk diproses melalui sistem inferensi *fuzzy*. *Membership function* (fungsi keanggotaan) bersama dengan *rule base* (aturan dasar) menjadi komponen penting yang disebut sebagai *knowledge base* (pengetahuan dasar) dari sebuah FIS. Setelah dilakukan data diproses melalui sistem inferensi *fuzzy*, *output* yang berupa bilangan *fuzzy* akan dikonversi kembali menjadi bentuk bilangan *crisp* dengan proses *deffuzifier* atau defuzzifikasi [OEA-09].

Terdapat tiga metode dalam sistem inferensi fuzzy yang sering digunakan, yaitu metode tsukamoto, mamdani dan sugeno [KSH-10]. Perbedaan antara ketiga sistem inferensi fuzzy terletak pada konsekuensi dari aturan *fuzzy*, dan dengan demikian metode tersebut memiliki agregasi dan prosedur defuzzifikasi yang berbeda [JSM-97].

2.2.4.1. Metode Mamdani

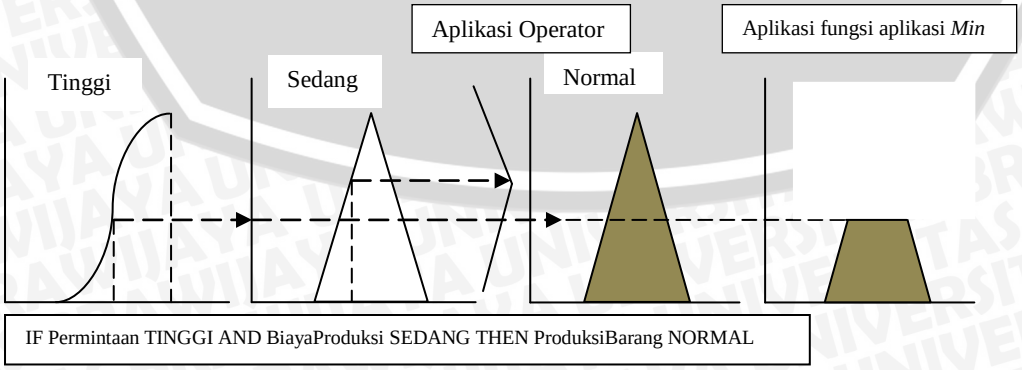
Metode Mamdani sering dikenal sebagai metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan output, diperlukan 4 tahapan yaitu [KSH-10]:

1. Pembentukan himpunan fuzzy

   Pada metode Mamdani, baik variabel *input* maupun *output* dibagi menjadi satu atau lebih himpunan fuzzy.

2. Aplikasi fungsi implikasi

   Pada metode Mamdani, fungsi implikasi yang digunakan adalah min. Fungsi ini akan memotong *output* himpunan *fuzzy*. Gambar 2.7 menunjukkan salah satu contoh penggunaan fungsi *min*.



Gambar 2.7 Fungsi implikasi *MIN* (Kusumadewi,2010)

3. Komposisi aturan

   Tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar aturan. Terdapat tiga metode yang digunakan yaitu max, additive dan probabilistik OR (probor).

   a. Pada metode max, solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah *fuzzy*, dan mengaplikasikannya ke *output* dengan operator OR (union).
   b. Pada metode *additive*, solusi himpunan *fuzzy* diperoleh dengan cara melakukan *bounded-sum* terhadap semua *output* daerah *fuzzy*.
   c. Pada metode probor, solusi himpunan *fuzzy* diperoleh dengan melakukan *product* terhadap semua *output* daerah *fuzzy*.

4. Penegasan (defuzzy)

   Input dari proses defuzzifikasi adalah himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut.

2.3. Algoritma Particle swarm optimization

Algoritma *Particle Swarm Optimization* (PSO) diperkenalkan diperkenalkan Kennedy dan Eberhart pada tahun 1995 sebagai pencarian stokastik melalui masalah ruang n-dimensi yang bertujuan untuk meminimalkan (atau memaksimalkan) dari fungsi tujuan dari masalah. PSO menghasilkan konvergensi lebih cepat bila dibandingkan dengan algoritma genetika, karena keseimbangan antara eksplorasi dan eksploitasi di ruang pencarian [PES--10].

Algoritma ini terinspirasi dari formasi dari sekumpulan hewan seperti burung dan ikan. Prinsip dibalik PSO adalah setiap individu pada sekumpulannya, disebut sebagai partikel, akan bergerak menuju partikel dengan kinerja terbaik (pemimpin) dalam sekumpulan saat mencari posisi terbaik dari masing-massing partikel [FKH-08].

Algoritma PSO adalah metode optimasi berbasis populasi yang dapat melakukan pencarian solusi optimal menggunakan populasi partikel. Setiap

sekawanan dalam PSO merupakan solusi di dalam ruang solusi Definisi PSO disajikan sebagai berikut [PES-10]:

Setiap partikel individu *i* memiliki karakteristik sebagai berikut:

1. Posisi saat ini di ruang pencarian, x_{id}
2. Sebuah kecepatan arus, v_{id}
3. Posisi personal terbaik dalam ruang pencarian, p_{id}.

- Posisi personal terbaik, p_{id}, berhubungan dengan posisi dalam ruang pencarian di mana partikel *i* memiliki nilai kesalahan terkecil yang ditentukan oleh fungsi tujuan *f*, dengan asumsi meminimalisasi tugas.
- Posisi terbaik global, p_{gd} mewakili posisi yang memiliki nilai kesalahan terendah di antara semua anggota.

Selama iterasi setiap partikel dalam sekawanan selalu memperbarui kecepatan partikel dan posisinya. Untuk memperbarui kecepatan partikel digunakan persamaan (2-6), sedangkan untuk posisi partikel diperbarui dengan persamaan (2-7).

$$v_{id} = \omega * v_{id} + c1 * rand() * (p_{id} - x_{id}) + c2 * rand() * (p_{gd} - x_{id}) \quad (2\text{-}6)$$

Dengan: v_{id} = kecepatan partikel
ω = berat inersia
c1 = konstanta
rand = bilangan acak
p_{id} = posisi terbaik masing-masing partikel
x_{id} = posisi partikel
c2 = konstanta
p_{gd} = posisi terbaik seluruh partikel

$$x(t+1) = x(t) + v(t+1) \quad (2\text{-}7)$$

Dengan: $x(t+1)$ = posisi baru partikel
x(t) = posisi partikel
v(t+1) = kecepatan baru partikel

Berat inersia digunakan sebagai control parameter pada algoritma PSO untuk mengontrol efek dari kecepatan sebelumnya pada kecepatan yang baru. Berat inersia yang berkurang secara linier dipercaya dapat meningkatkan performa dari PSO. berat inersia linier dapat direpresentasikan sebagai berikut [DSM-13]:

$$\omega = \omega_{max} - iterasi \times \frac{\omega_{max} - \omega_{min}}{iterasi\ maksimum} \quad (2\text{-}8)$$

Algoritma dasar PSO dapat dijelaskan dengan *pseudo code* sebagai berikut:

The basic PSO algorithm

for all particles{
initialize velocities and positions
}//end for
while stopping criteria is unsatisfied{
for each particle{
1. compute velocities by equation

$$v_i = \omega v_i + c_1 r_1 (\hat{x}_i - x_j) + c_2 r_2 (\hat{g} - x_j)$$

2. increment positions by equation

$$x_j = x_j + v_i$$

if present fitness value is better than current local best value
3. update local best positions
if present fitness value is better than current global best value
4. update global best positions
}//end for
}//end while

Gambar 2.8. Algoritma dasar PSO (Fang, 2008)

2.3.1. Nilai *cost*

Evaluasi dari algoritma optimasi PSO dilakukan pada setiap iterasi, salah satu cara yang dapat digunakan untuk melakukan evalusi adalah menggunakan MSE (mean-square-error). Adapun formula untuk menyatakan nilai MSE dapat dilihat pada persamaan(2-8) [OEA-09]:

$$MSE = \frac{\sum_{k=1}^{q} (y_k - \hat{y}_k)^2}{\sum_{k=1}^{q} (\hat{y}_k)^2} \quad (2\text{-}9)$$

Dengan y_k dan $\hat{y}_k$ merepresentasikan nilai sesungguhnya dan nilai perkiraan.

2.4. Akurasi

Salah satu cara untuk mengetahui hasil penelitian adalah melihat akurasi. Akurasi merupakan kedekatan suatu angka atau hasil pengujian terhadap angka ataupun data sebenarnya (*true value* atau *reference value*) [NUG-06]. Dalam penelitian ini perhitungan akurasi akan melibatkan hasil penelitian dan data nyata yang didapatkan dari sumber, berikut persamaan untuk perhitungan akurasi:

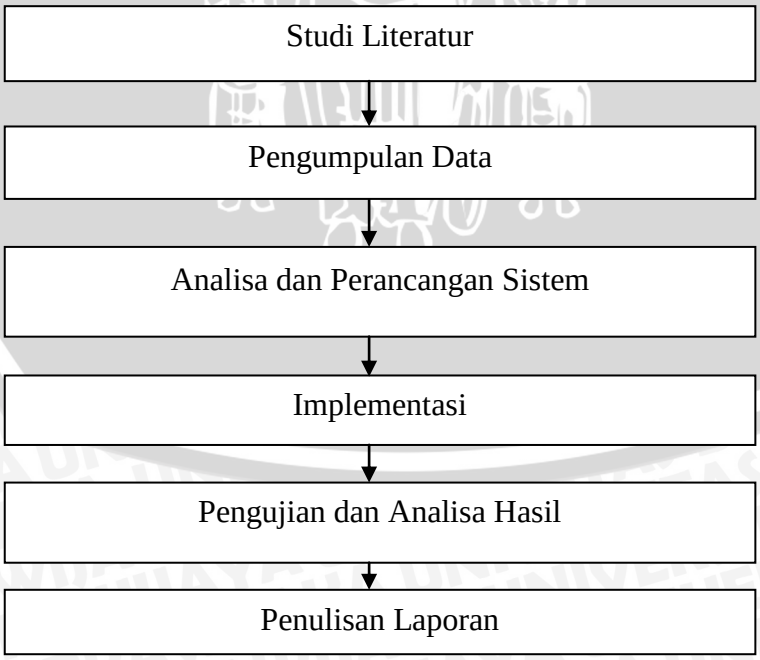
$$TingkatAkurasi = \frac{\sum dataUjiBenar}{\sum totalDataUji} \quad (2\text{-}10)$$

$$Akurasi(\%) = \frac{\sum dataUjiBenar}{\sum totalDataUji} x100\% \quad (2\text{-}11)$$

BAB III
METODE PENELITIAN DAN PERANCANGAN

Pada bab ini dijelaskan langkah-langkah yang dilakukan untuk mengerjakan penelitian tentang optimasi fungsi keanggotaan *fuzzy* menggunakan algoritma *Particle Swarm Optimization* pada sistem inferensi *fuzzy* penentuan jurusan siswa SMA ini. Alur pengerjaan penelitian ini secara umum dapat dilihat pada Gambar 3.1. dengan tahapan-tahapan sebagai berikut:

1. Mempelajari literatur yang berkaitan dengan pengerjaan penelitian, diantaranya mengenai penentuan jurusan siswa SMA, sistem inferensi *fuzzy*, dan algoritma *Particle Swarm Optimization*.
2. Mengumpulkan data penentuan jurusan siswa SMA Negeri 3 Malang tahun 2013.
3. Melakukan analisa dan perancangan sistem mengenai algoritma *Particle Swarm Optimization* untuk optimasi fungsi keanggotaan dan sistem inferensi *fuzzy* Mamdani untuk penentuan jurusan siswa SMA.
4. Mengimplementasikan sistem yang telah dibangun.
5. Melakukan uji coba sistem serta pembahasan terhadap hasil penelitian.
6. Membuat laporan tertulis dengan format sesuai ketentuan.



Gambar 3.1. Desain Penelitian

3.1. Studi Literatur

Studi literatur merupakan proses mempelajari tentang penunjang dasar teori yang akan dilakukan, serta penelitian yang pernah dilakukan sebelumnya. Sumber yang digunakan untuk studi literatur antara lain: jurnal, buku, halaman web, dan laporan penelitian sebelumnya. Teori yang dipelajari diantaranya adalah tentang penjurusan siswa SMA, sistem inferensi *fuzzy*, dan algoritma *Particle Swarm Optimization*.

3.2. Data Penelitian

Data yang akan digunakan pada penelitian ini merupakan data penentuan jurusan siswa SMA Negeri 3 Malang kelas X pada tahun 2013. Atribut data yang digunakan dalam penelitian ini terdiri dari nilai beberapa mata pelajaran yang berkaitan dengan penentuan jurusan berupa tigabelas parameter yang terdiri atas matematika, PPK fisika, praktikum fisika, PPK biologi, praktikum biologi, PPK kimia, praktikum kimia, sejarah, ekonomi, geografi, sosiologi, minat IPA dan minat IPS. Atribut yang digunakan sebagai input dari data tersebut semua bertipe numerik. Seluruh data akan dikelompokkan menjadi dua kelas, yaitu kelas IPA dan kelas IPS. Pada penelitian ini data yang akan digunakan adalah data dari tiga kelas, yang berjumlah 105 data. Untuk data selengkapnya dapat dilihat pada Lampiran1.

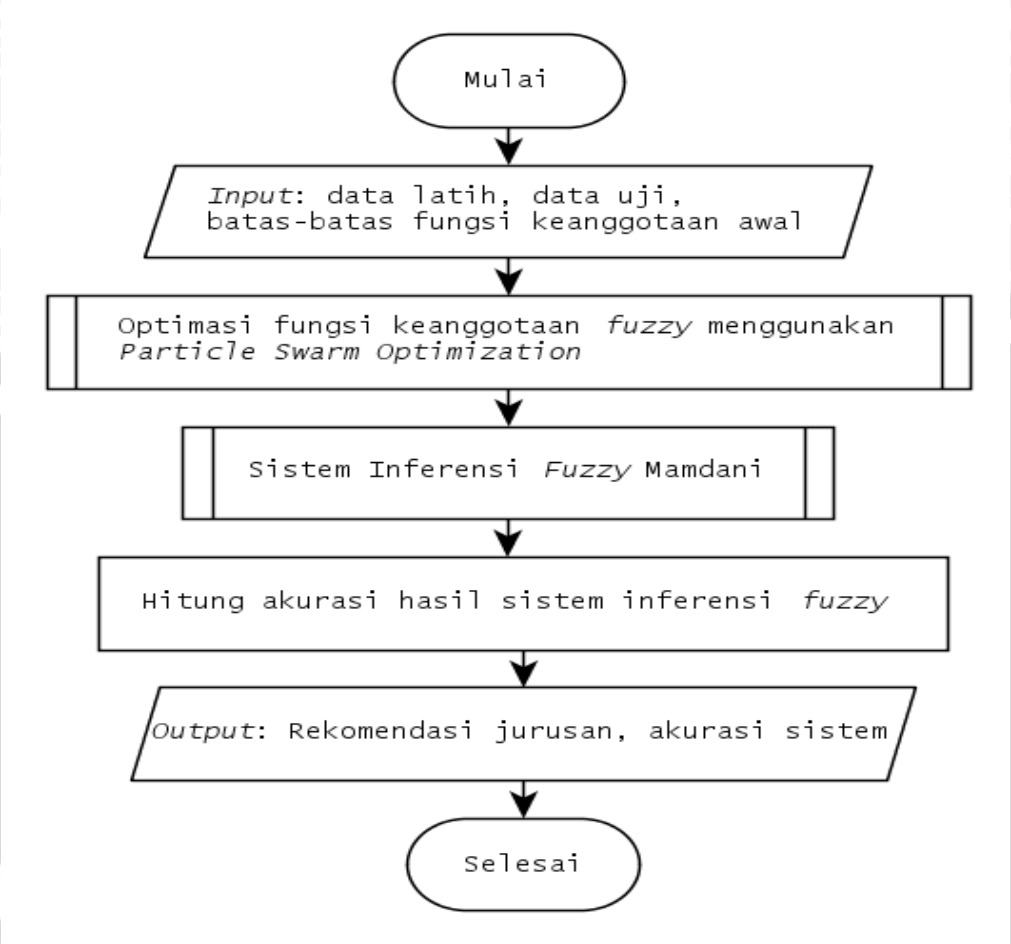
3.3. Analisa dan Perancangan Sistem

3.3.1. Deskripsi Umum Sistem

Secara garis besar, alur sistem yang akan dijalankan dapat dilihat pada Gambar 3.2. dengan tahap-tahap sebagai berikut:

1. *Input* yang dibutuhkan adalah data latih, data uji, dan batas-batas fungsi keanggotaan awal.
2. Proses optimasi fungsi keanggotaan awal dengan menggunakan algoritma *Particle Swarm Optimization* (PSO) .
3. Proses penentuan rekomendasi jurusan menggunakan sistem inferensi *fuzzy* Mamdani menggunakan batas-batas fungsi keanggotaan awal dan batas-batas fungsi keanggotaan yang telah dioptimasi.

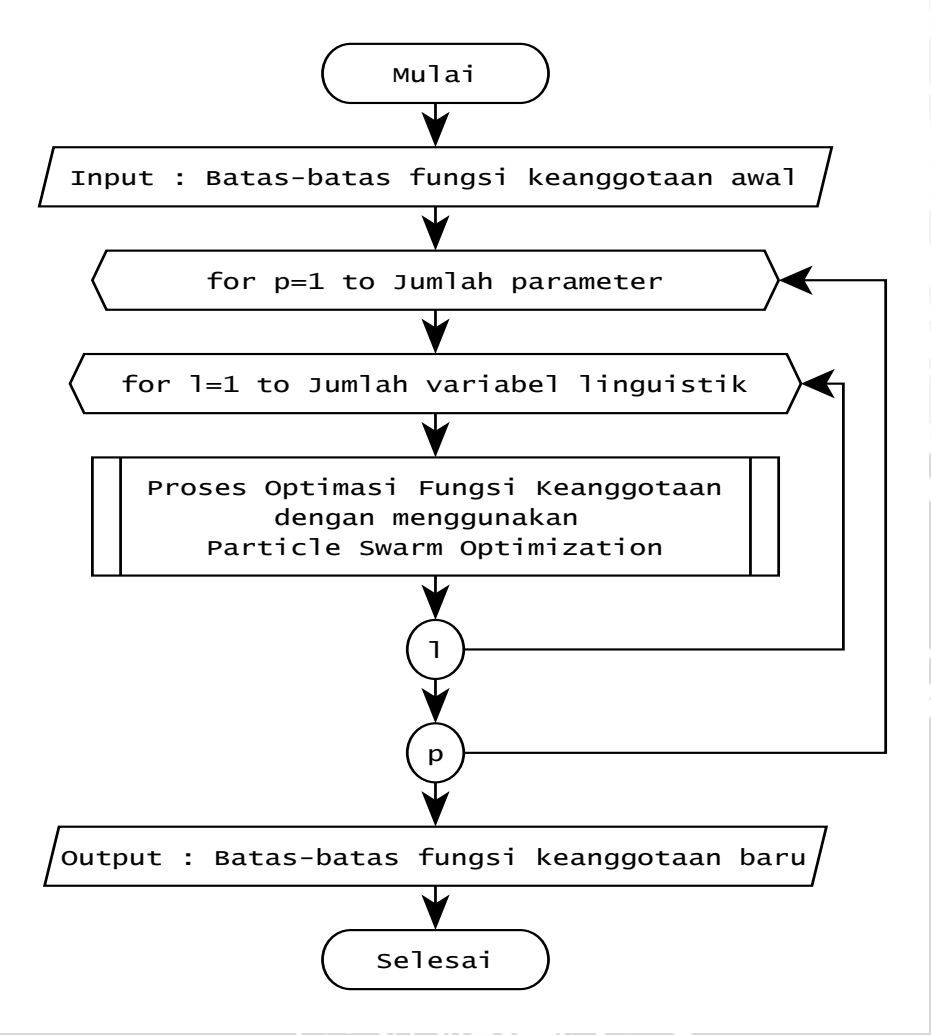
4. Proses menghitung akurasi menggunakan Persamaan (2-10) dan Persamaan (2-11)
5. *Output* yang dihasilkan berupa rekomendasi jurusan dan akurasi sistem.



Gambar 3.2. Gambaran umum sistem

3.3.2. Optimasi fungsi keanggotaan

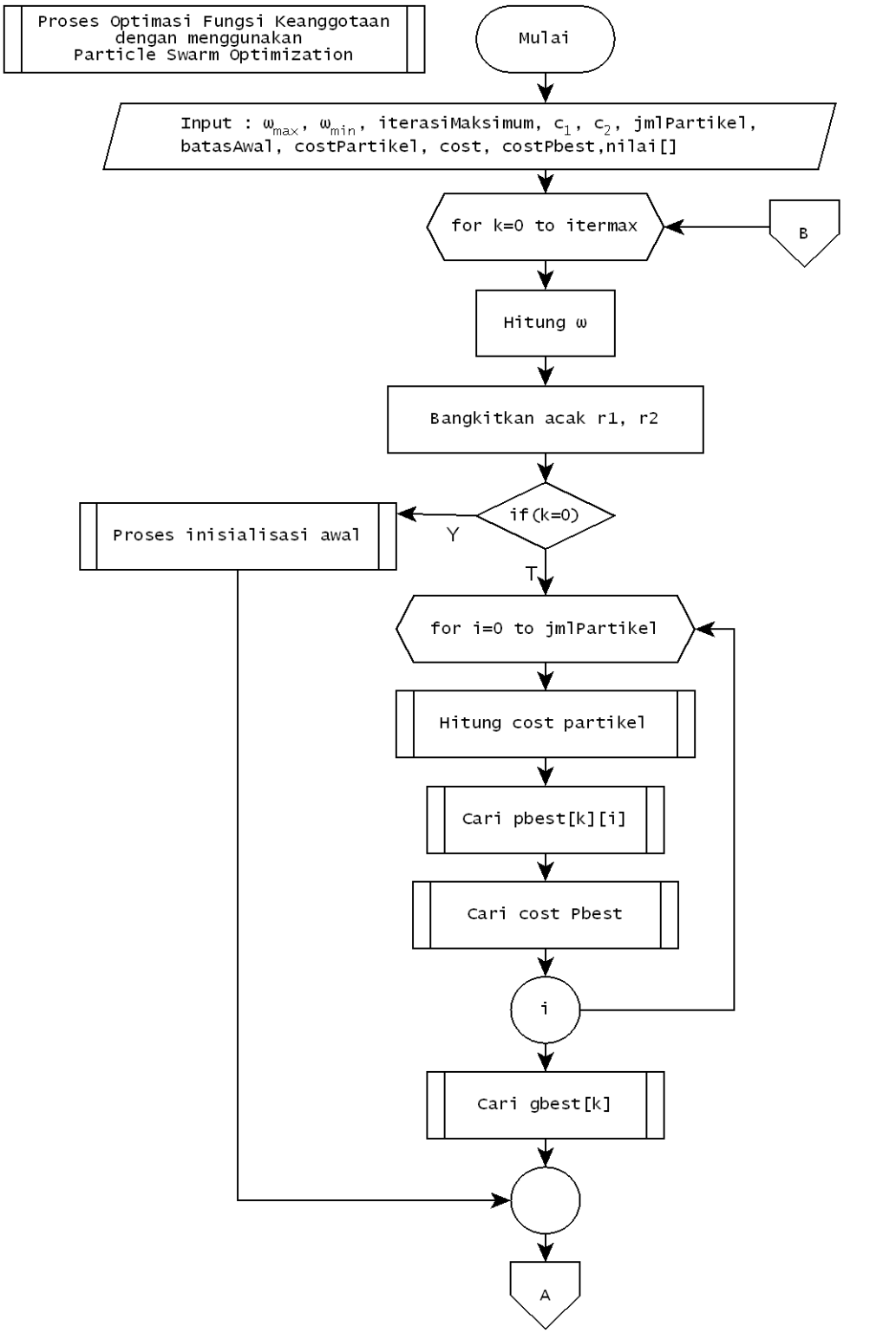
Pada bagian ini, akan dilakukan proses optimasi batas-batas dari setiap variabel linguistik untuk setiap parameter menggunakan algoritma *Particle Swarm Optimization* yang secara umum ditunjukkan oleh Gambar 3.3.



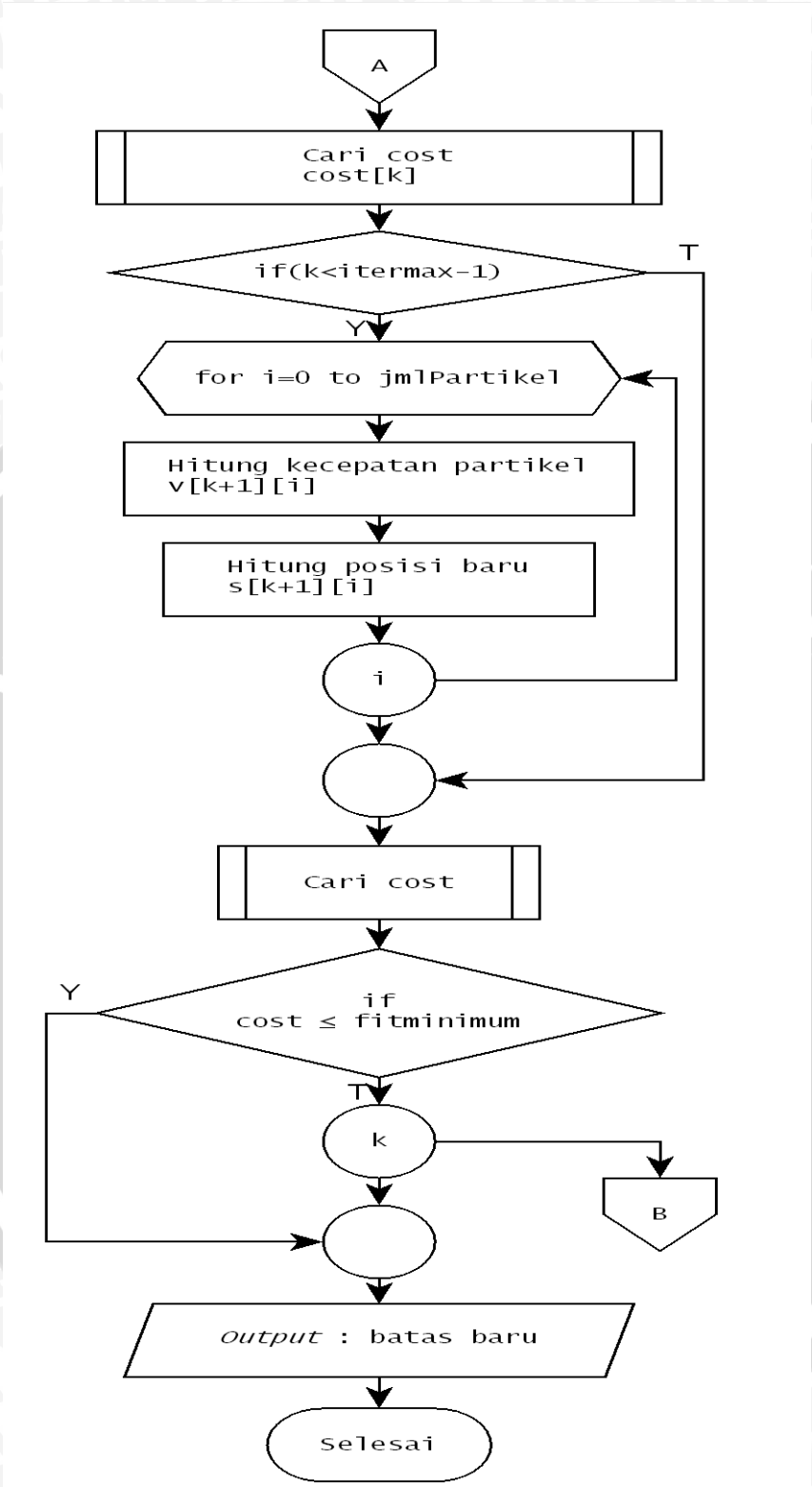
Gambar 3.3. Alur Proses Sistem Optimasi

Proses Optimasi Fungsi Keanggotaan dengan menggunakan *Particle Swarm Optimization*

Tahap ini berisi proses optimasi fungsi keanggotaan untuk satu variabel linguistik dengan menggunakan algoritma *Particle Swarm Optimization* sebagaimana ditunjukkan pada Gambar 3.4.a. dan Gambar 3.4.b.



Gambar 3.4.a. Proses Optimasi Fungsi Keanggotaan dengan menggunakan *Particle Swarm Optimization* (a)



Gambar 3.4.b. Proses Optimasi Fungsi Keanggotaan dengan menggunakan *Particle Swarm Optimization* (b)

Keterangan dari tahapan tersebut adalah sebagai berikut:

1. *Input* nilai untuk variabel iterasi maksimum (*itermax*), inersia maksimum (ω_{max}) dan inersia minimum (ω_{min}), konstanta (*c1*dan *c2*), jumlah partikel yang digunakan (*JmlPartikel*), batas awal variabel linguistik (*batasAwal[]*) dan *cost* minimum (*fitminimum*).
2. Iterasi *k* = 0 hingga *itermax* dilakukan langkah berikut :
   a. Hitung nilai ω untuk masing-masing iterasi dengan menggunakan persamaan (2-8).
   b. Bangkitkan secara acak nilai variabel r_1 dan r_2.
   c. Untuk iterasi ke-0, dilakukan proses inisialisasi awal seperti pada Gambar 3.5.
   d. Untuk iterasi lebih dari 0, dilakukan langkah berikut:
      i. Menghitung nilai *cost* partikel, yang dapat dilihat pada Gambar 3.6.
      ii. Mencari nilai Pbest masing-masing partikel, yang dapat dilihat pada Gambar 3.7.
      iii. Mencari nilai *cost* Pbest, yang dapat dilihat pada Gambar 3.8.
      iv. Proses mencari nilai Gbest yang dapat dilihat pada Gambar 3.10.
   e. Mencari nilai *cost* dari setiap iterasi, yang dapat dilihat pada Gambar 3.9.
   f. Untuk iterasi kurang dari (*itermax*-1):
      i. Menghitung nilai kecepatan terbaru (*v*) dengan menggunakan Persamaan (2-6).
      ii. Menghitung nilai posisi (*s*) terbaru dengan menggunakan Persamaan (2-7).
3. Jika perhitungan telah memenuhi nilai *cost* minimum yang menjadi target (*fitminimum*) sebelum memenuhi iterasi minimum, maka iterasi dihentikan. Jika belum memenuhi nilai *cost* minimum, maka iterasi berlanjut ke iterasi berikutnya.
4. Hasil akhir dari proses ini adalah nilai batas baru dari variabel linguistik.

1) Proses inisialisasi awal

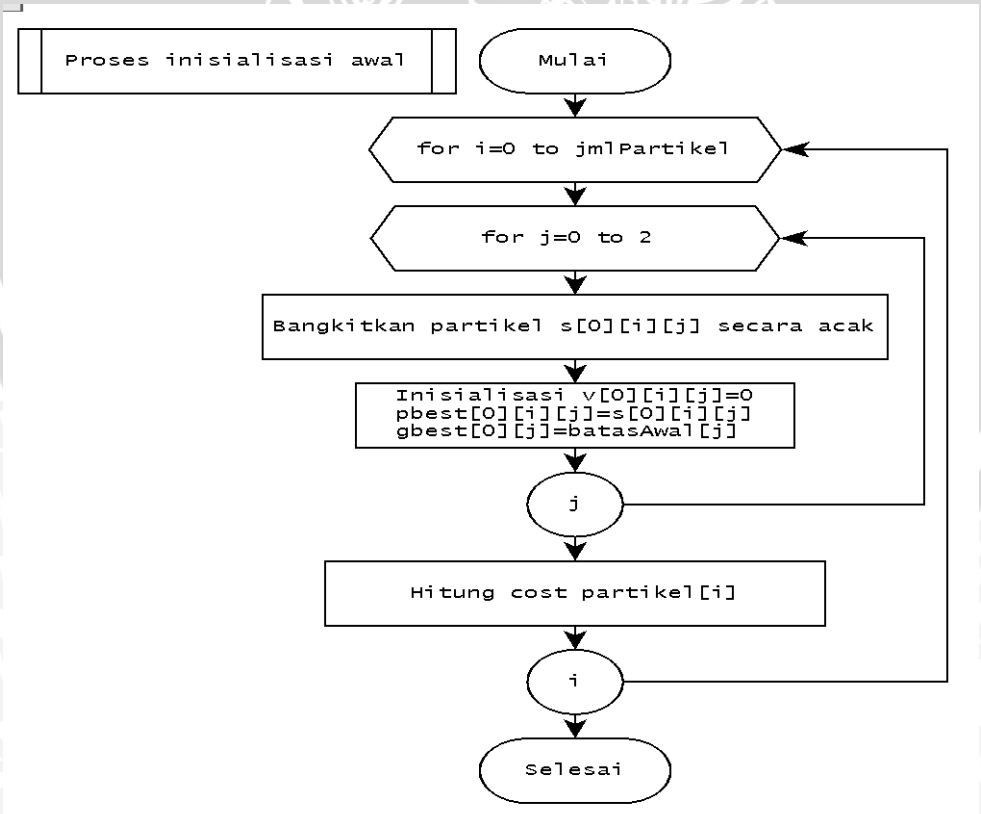
Proses inisialisasi awal ini merupakan proses untuk menentukan nilai awal dari beberapa variabel yang ada pada proses optimasi menggunakan PSO, yaitu:

Untuk setiap partikel *i*, dilakukan:

Untuk setiap titik *j*=0 hingga *j*=2 pada setiap partikel *i*, dilakukan:

a. Posisi partikel(*s[0][i][j]*) merupakan posisi awal partikel yang dibangkitkan secara acak.
b. Kecepatan awal untuk semua partikel adalah 0, *v[0][i][j]*=0.
c. Posisi lokal terbaik untuk setiap partikel (Pbest) awal sama dengan posisi saat dilakukan pembangkitan partikel secara acak, *pbest[0][i][j]= s[0][i][j]*.
d. Posisi terbaik untuk setiap iterasi (Gbest) awal sama dengan batas awal dari masing-masing variabel linguistik *gbest[0][j]=batasAwal[j]*.

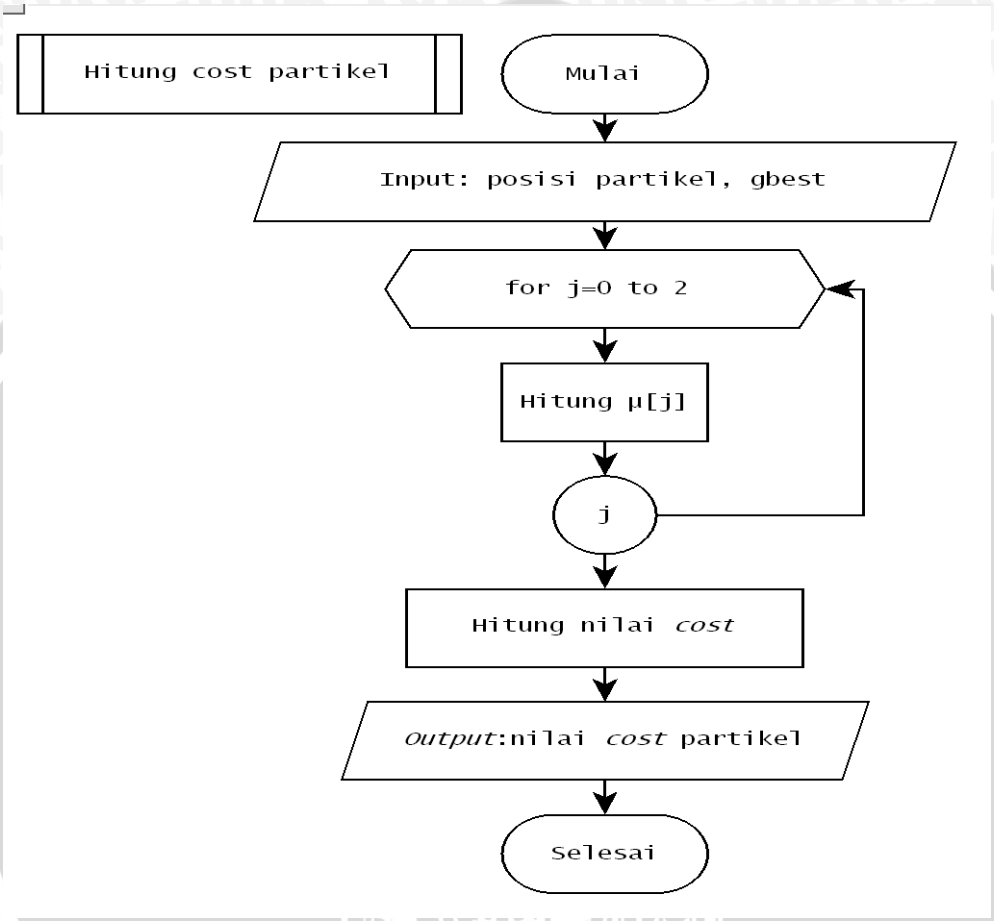
Adapun gambaran dari proses ini dapat dilihat pada Gambar 3.5.



Gambar 3.5. Proses Inisialisasi Awal

2) Proses hitung *cost* partikel

Proses hitung *cost* partikel ini merupakan proses untuk menghitung nilai *cost* dari masing-masing partikel. Gambaran proses ini terdapat pada Gambar 3.6.



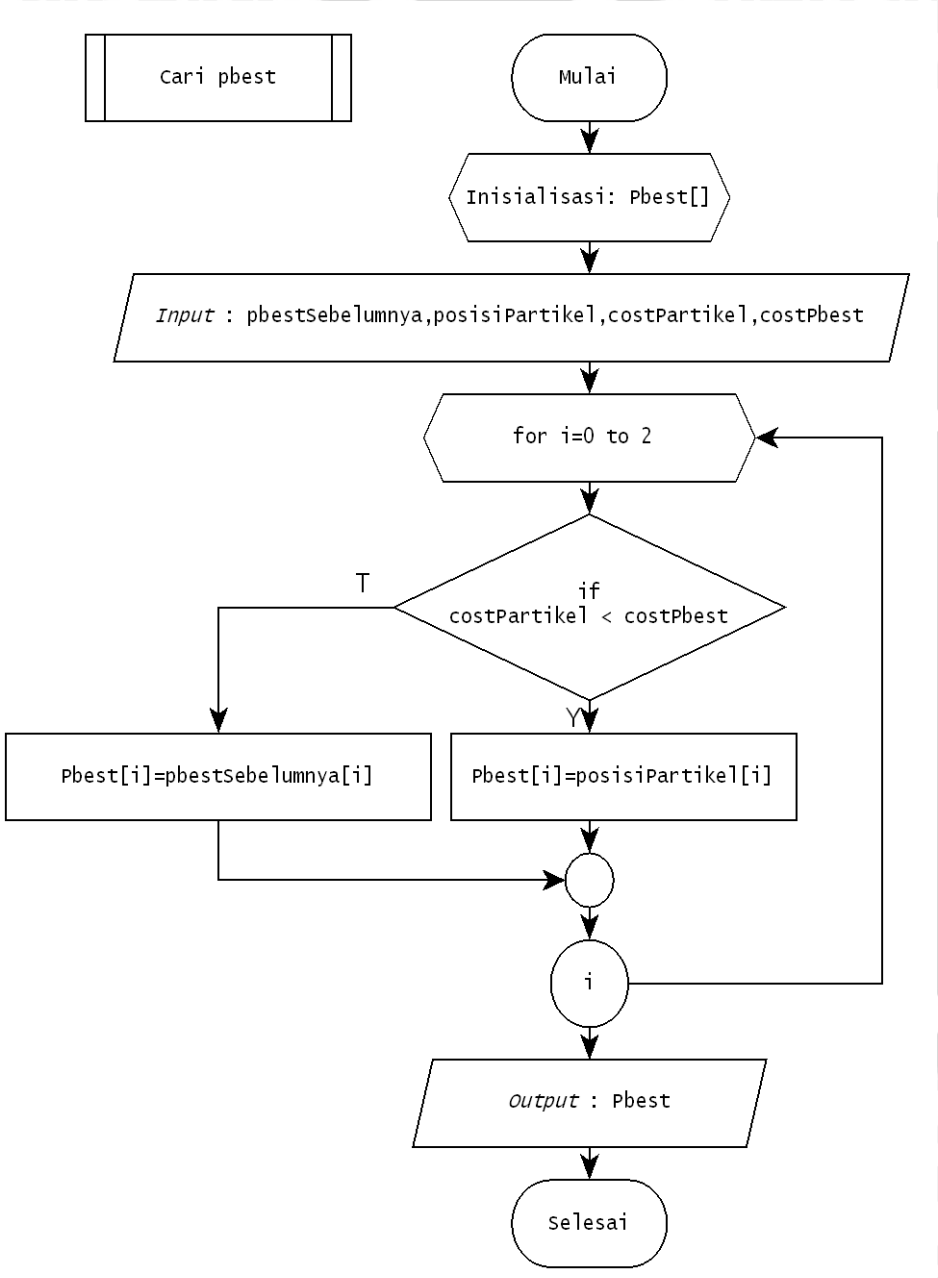
Gambar 3.6. Hitung *cost* partikel

Keterangan:

a. *Input* berupa posisi partikel dan posisi Gbest
b. Untuk titik *j*=0 hingga *j*=2 lakukan:
   Hitung derajat keanggotaan dari masing-masing titik dalam partikel (*μ[j]*)
c. Hitung nilai *cost* dengan menggunakan rumus MSE seperti pada persamaan (2-9)
d. *Output* dari proses ini adalah nilai *cost* dari masing-masing partikel.

3) Proses cari Pbest

Proses mencari Pbest ini dilakukan untuk mencari posisi lokal terbaik setiap partikel. Proses ini dilakukan dengan membandingkan posisi lokal terbaik setiap partikel (Pbest) pada iterasi sebelumnya dengan posisi yang baru sebagaimana dapat dilihat pada Gambar 3.7.



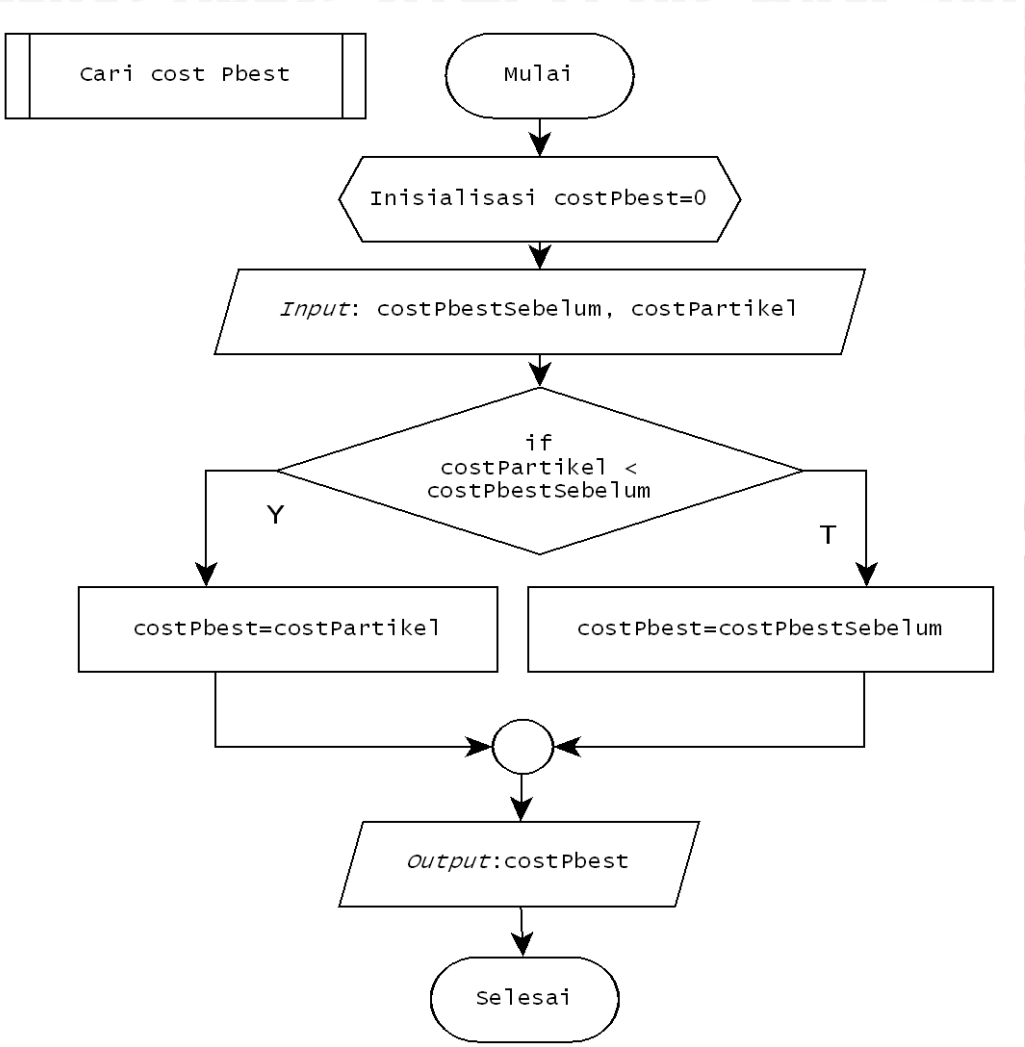
Gambar 3.7. Proses cari Pbest

Keterangan:

a. Inisialisasi array untuk menyimpan posisi Pbest (*Pbest[]*)
b. *Input* berupa posisi Pbest sebelumnya (*pbestSebelumnya*), posisi partikel saat ini (*posisiPartikel*), nilai *cost* partikel saat ini (*costPartikel*), dan nilai *cost* Pbest pada iterasi sebelumnya(*costPbest*)
c. Untuk titik *i*=0 hingga *i*=2, lakukan:
   Cek apakah nilai *cost* partikel saat ini lebih kecil daripada nilai *cost* Pbest iterasi sebelumnya (*costPartikel*<*costPbest*). Jika ya maka posisi Pbest diperbarui dengan posisi partikel saat ini (*Pbest[i]=posisiPartikel[i]*), namun jika tidak maka posisi Pbest tetap sama dengan iterasi sebelumnya (*Pbest[i]=pbestsebelumnya[i]*).
d. *Output* berupa posisi lokal terbaik partikel (*Pbest*).

4) Proses cari *cost* Pbest

Proses mencari *cost* Pbest dilakukan dengan membandingkan nilai *cost* Pbest iterasi sebelumnya dengan *cost* partikel saat ini. Gambaran untuk proses ini dapat dilihat pada Gambar 3.8.



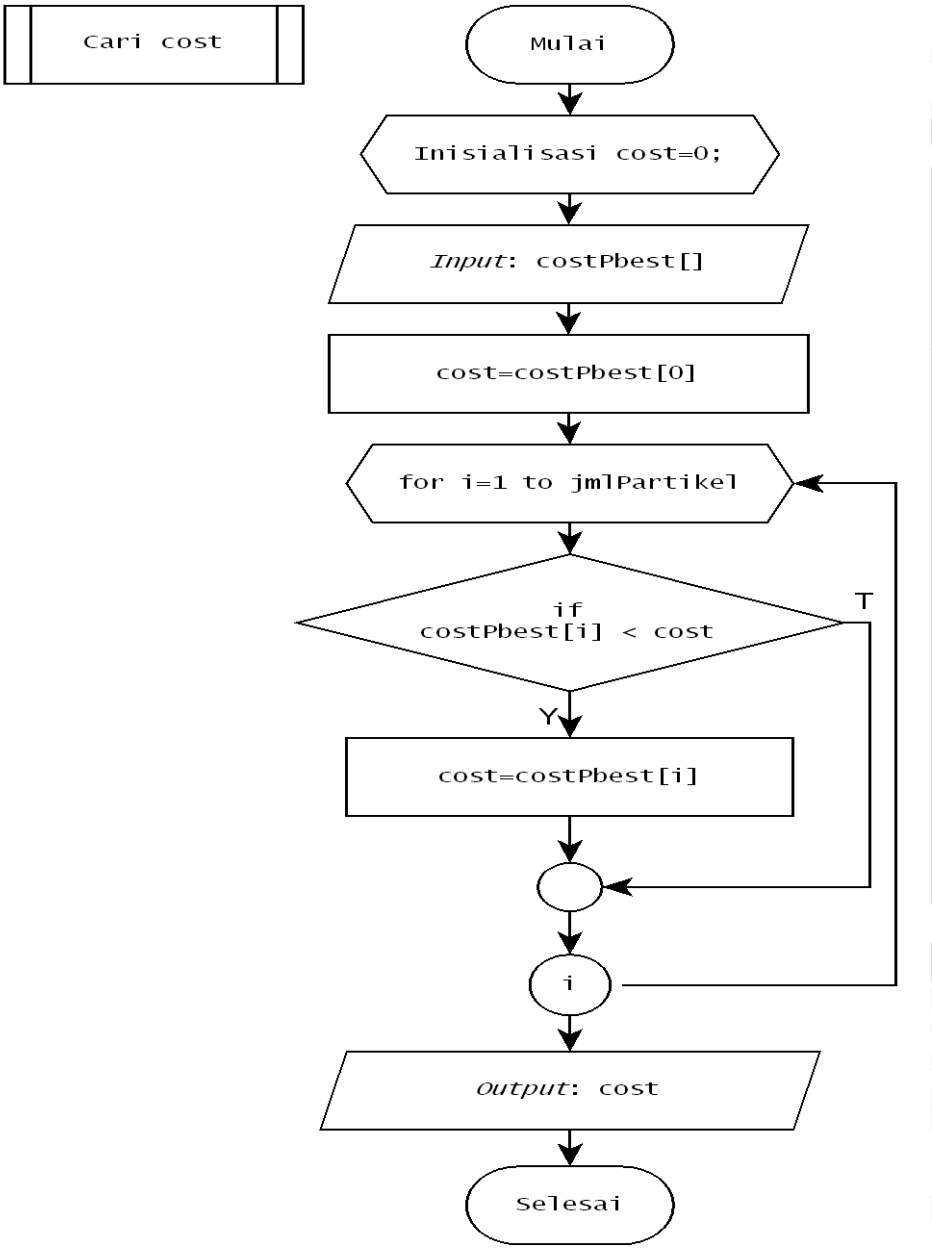
Gambar 3.8. Proses cari *cost* Pbest

Keterangan:

a. Inisialisasi *costPbest*=0
b. *Input* berupa nilai *cost* Pbest iterasi sebelumnya (*costPbestSebelum*) dan *cost* partikel (*costPartikel*)
c. Lakukan pengecekan, jika nilai *cost* partikel lebih kecil dari *cost* Pbest sebelumnya (*costPartikel*<*costPbestSebelum*) maka nilai *cost* Pbest diperbarui dengan nilai *cost* partikel saat ini (*costPbest*=*costPartikel*), jika tidak maka nilai *cost* Pbest sama dengan iterasi sebelumnya (*costPbest*=*costPbestSebelum*).
d. *Output* berupa nilai *cost* Pbest partikel (*costPbest*).

5) Proses mencari *cost*

Proses mencari *cost* dilakukan dengan membandingkan nilai *cost* Pbest dari setiap partikel pada iterasi yang berlangsung. Nilai *cost* Pbest paling minimum dari iterasi yang berlangsung akan dijadikan sebagai nilai *cost* dari iterasi tersebut. Adapun gambaran proses ini dapat dilihat pada Gambar 3.9.



Gambar 3.9. Proses mencari *cost*

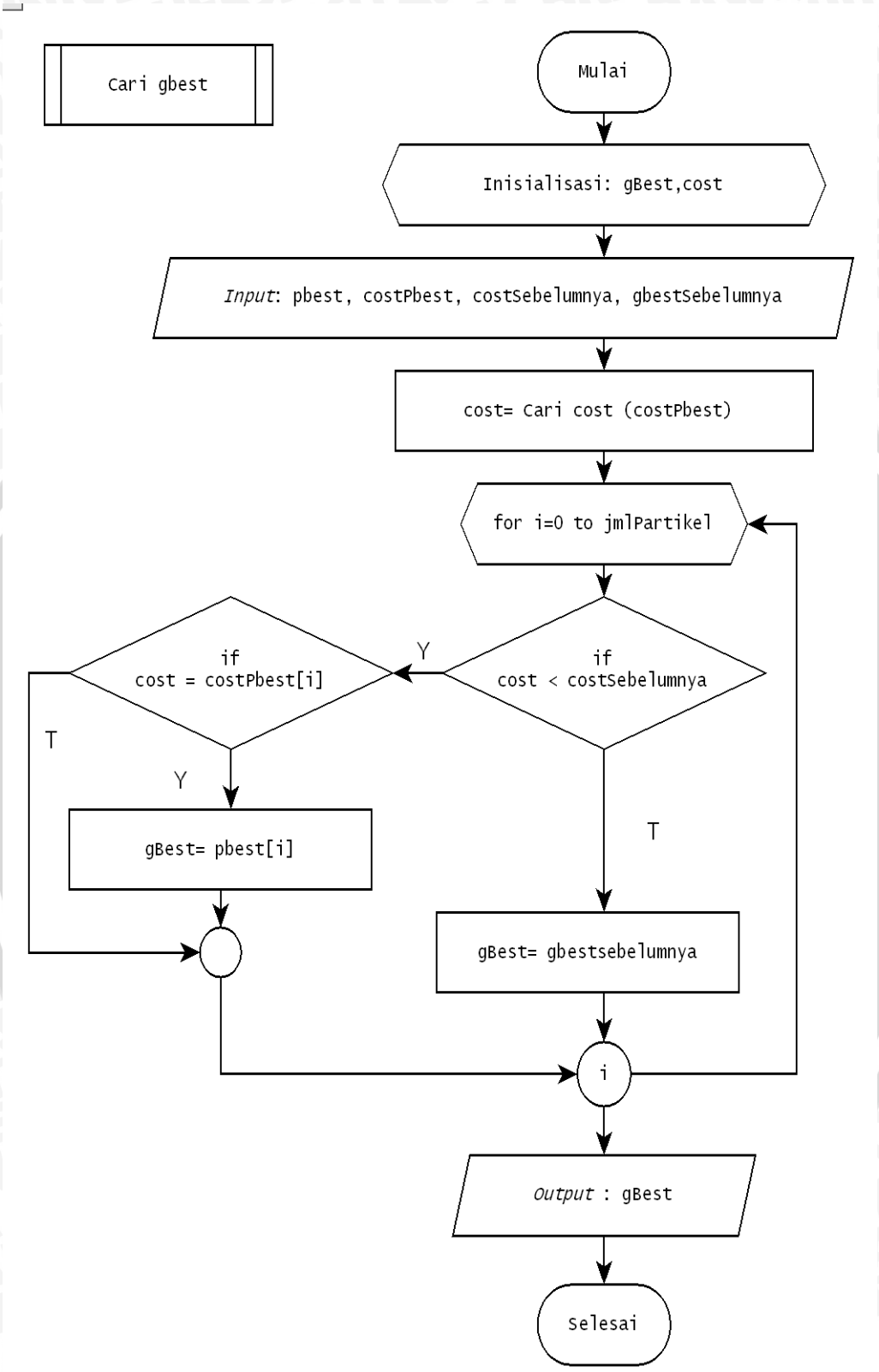
Keterangan:

a. Inisialisasi variabel local *cost*=0
b. *Input costPbest* partikel
c. Set nilai *cost* dengan *costPbest* partikel pertama
d. Untuk i=1 hingga jumlah partikel lakukan:
   Jika *costPbest* partikel *costPbest[i]<cost* maka perbarui nilai *cost* dengan *costPbest[i]*
e. *Output* berupa nilai *cost*.

6) Proses mencari Gbest

Tujuan dari proses mencari Gbest ini adalah untuk mencari posisi global terbaik dari setiap iterasi. Proses ini dilakukan dengan membandingkan antara nilai *cost* dari iterasi yang tengah berjalan dengan *cost* iterasi sebelumnya. Gambaran proses ini terdapat pada Gambar 3.10. dengan keterangan sebagai berikut :

a. Inisialisasi variabel local *gBest*, dan *cost*
b. *Input* posisi local terbaik setiap partikel (*pbest*), *cost* Pbest setiap partikel (*costPbest*), *cost* iterasi sebelumnya (*costSebelumnya*), dan posisi global terbaik iterasi sebelumnya (*gbestSebelumnya*)
c. Hitung *cost*, dimana *cost* merupakan hasil perhitungan nilai *cost* yang dihitung menggunakan proses mencari *cost* seperti yang telah dijelaskan pada proses 5.
d. Untuk i=0 hingga jumlah partikel, lakukan:
   Jika nilai *cost<costSebelumnya*, maka lakukan pengecekan apakah *cost=costPbest[i]*, jika yam aka perbarui posisi Gbest dengan posisi Pbest partikel ke-i (*gBest=pBest[i]*), jika tidak maka posisi Gbest sama dengan posisi Gbest iterasi sebelumnya (*gbest=gbestSebelumnya*)
e. *Output* berupa posisi Gbest.

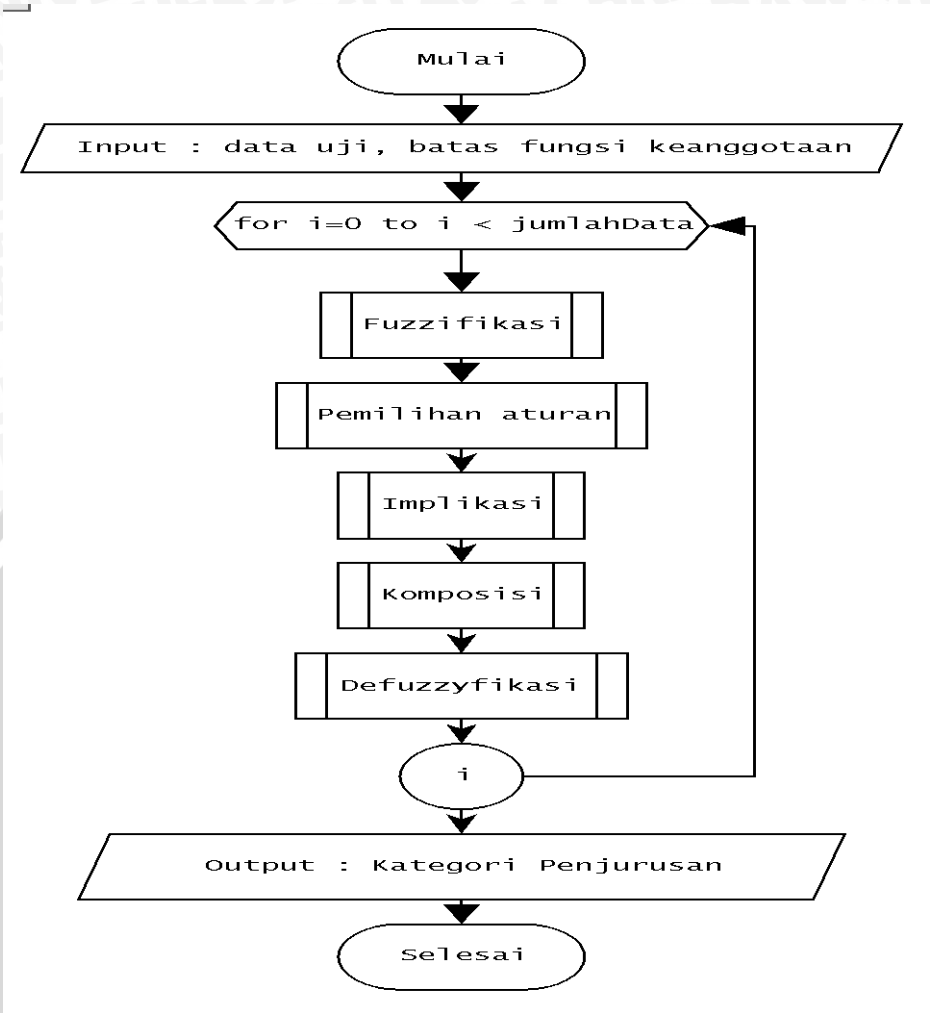


Gambar 3.10. Proses Mencari Gbest

3.3.3. Sistem Inferensi Fuzzy Mamdani

Alur sistem inferensi *fuzzy* Mamdani yang akan dibangun ditunjukkan pada Gambar 3.11 dengan tahapan sebagai berikut:

1. *Input* yang dibutuhkan adalah data uji dan batas-batas fungsi keanggotaan.
2. Untuk i=0 hingga jumlah data, dilakukan:
   a. Proses fuzzifikasi, yaitu proses pembentukan bilangan *fuzzy* yang ditunjukkan oleh Gambar 3.12.
   b. Proses pemilihan aturan, yaitu memilih aturan yang akan digunakan. Alur proses ini dapat dilihat pada Gambar 3.13.
   c. Proses implikasi, pada sistem inferensi *fuzzy* ini digunakan metode Min dengan proses seperti pada Gambar 3.14.
   d. Proses komposisi dengan metode Max, yang ditunjukkan oleh Gambar 3.15.
   e. Proses defuzzifikasi, yang ditunjukkan oleh Gambar 3.16.
3. *Output* yang dihasilkan dari proses sistem inferensi *fuzzy* ini berupa rekomendasi kategori penjurusan.



Gambar 3.11. Sistem inferensi *fuzzy*

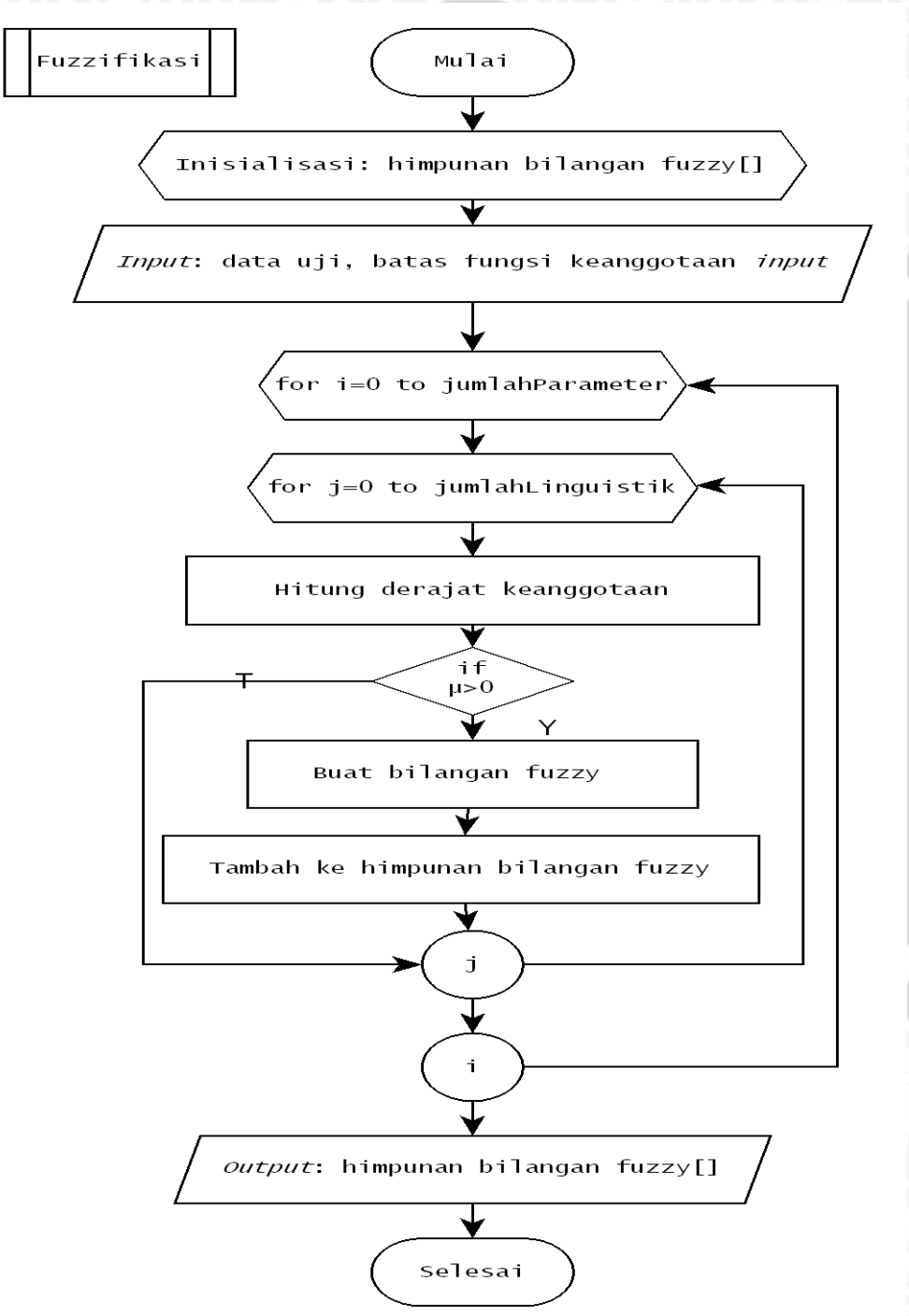
1) Pembentukan himpunan *fuzzy* (fuzzifikasi)

Masing-masing parameter ditransformasikan ke dalam himpunan *fuzzy* dengan fungsi keanggotaan yang sesuai. Proses ini ditunjukkan oleh Gambar 3.12 dengan tahapan sebagai berikut:

1. Inisialisasi himpunan bilangan *fuzzy* yang nantinya digunakan untuk menyimpan hasil yang berupa bilangan *fuzzy*.
2. *Input* yang dibutuhkan berupa dat auji dan batas-batas fungsi keanggotaan variabel *input.*
3. Untuk iterasi *i* = 0 hingga jumlah parameter lakukan langkah berikut :
   Iterasi *j*= 0 hingga jumlah linguistik lakukan langkah berikut :
   a. Hitung nilai derajat keanggotaan (μ) data dengan menggunakan fungsi keanggotaan segitiga

b. Jika derajat keanggotaan lebih dari 0 ($\mu>0$) dilakukan:
   Pembuatan bilangan *fuzzy* dengan parameter *i*, variabel linguistik *j*, dan derajat keanggotaan μ.

4. Output dari proses ini adalah himpunan bilangan *fuzzy*.



Gambar 3.12. Proses fuzzifikasi

2) Proses pemilihan aturan

Proses pemilihan aturan merupakan proses pemilihan aturan yang akan digunakan pada sistem inferensi *fuzzy*. Alur proses pemilihan aturan adalah:

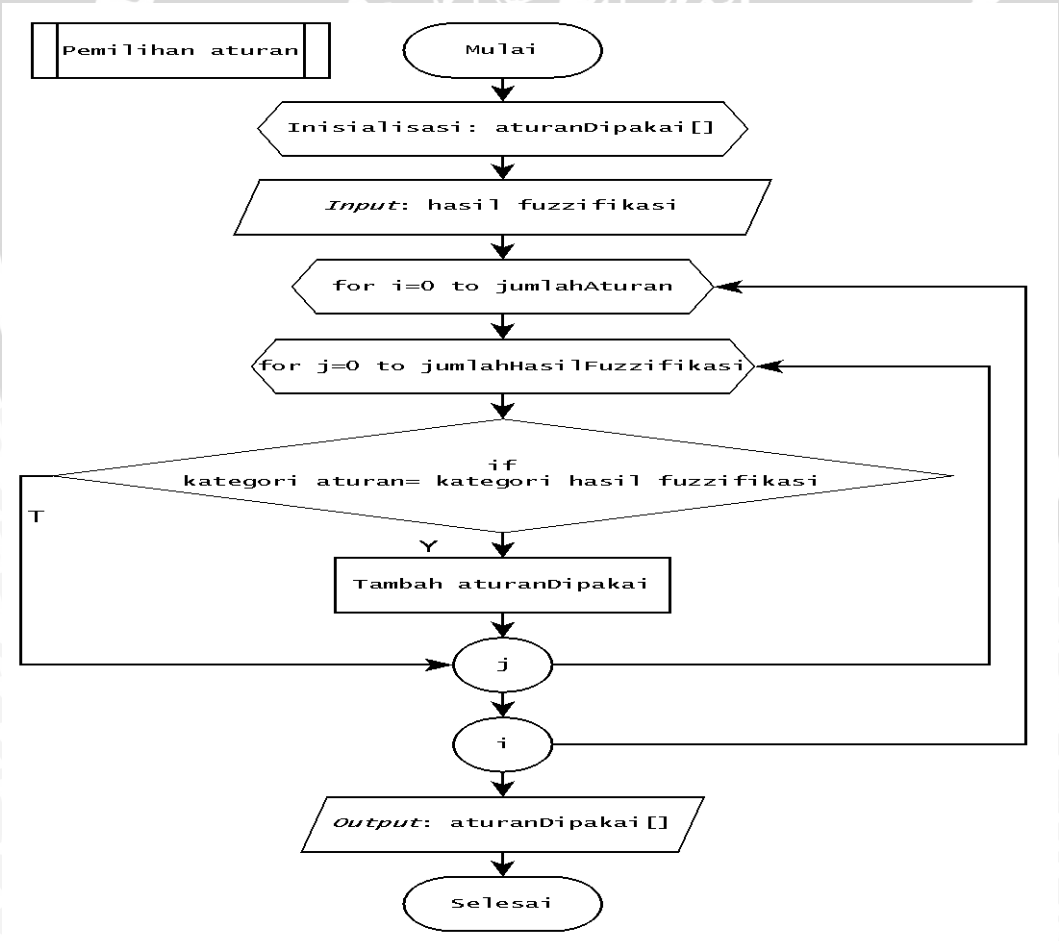
1. Inisialisasi himpunan aturan yang akan digunakan.
2. *Input* yang dibutuhkan adalah bilangan *fuzzy* hasil proses fuzzifikasi.
3. Iterasi *i*=0 hingga jumlah aturan, dilakukan:

Untuk iterasi *j*=0 hingga jumlah hasil fuzzifikasi, lakukan:

Jika kategori aturan=kategori hasil fuzzifikasi, lakukan:

Tambahkan aturan ke-*i* ke dalam himpunan aturan yang akan dipakai.

4. *Output* yang dihasilkan berupa himpunan aturan yang akan digunakan. Proses pemilihan aturan ini ditunjukkan oleh Gambar 3.13.

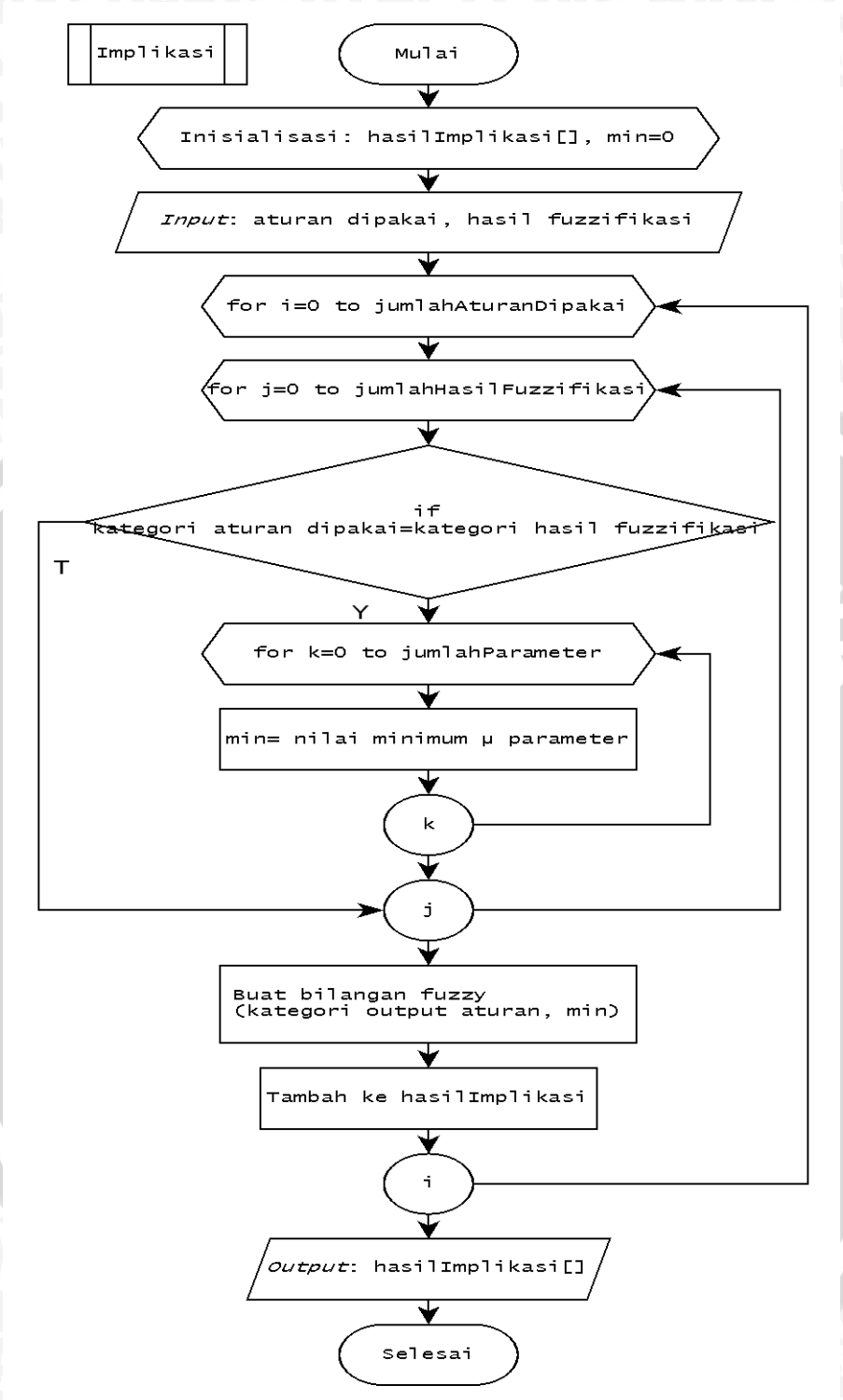


Gambar 3.13. Proses pemilihan aturan

3) Proses implikasi

Proses implikasi untuk penelitian ini, digunakan metode Min yang dapat diihat pada Gambar 3.14. dengan alur sebagai berikut:

1. Inisialisasi himpunan hasil implikasi, dan variabel min (*min*=0).
2. *Input* yang dibutuhkan adalah aturan yang dipakai, dan hasil fuzzifikasi berupa bilangan *fuzzy*.
3. Iterasi *i*=0 hingga jumlah aturan yang dipakai, lakukan:
   Untuk iterasi *j*=0 hingga jumlah fuzzifikasi:
   Dilakukan pengecekan apakah kategori parameter aturan= kategori hasil fuzzifikasi, jika ya lakukan:
   Untuk iterasi *k*=0 hingga jumlah parameter:
   Cari nilai minimum derajat keanggotaan (μ) dari semua parameter penyusun aturan *i*, kemudian set nilai variabel *min* dengan nilai tersebut.
4. Buat bilangan *fuzzy* dengan variabel linguistik sesuai *output* aturan, dan nilai derajat keanggotaan *min*.
5. Tambahkan bilangan *fuzzy* tersebut ke dalam himpunan hasil implikasi.
6. *Output* yang dihasilkan berupa himpunan hasil implikasi berbentuk bilangan *fuzzy*.



Gambar 3.14. Proses implikasi

4) Proses komposisi

Proses komposisi untuk penelitian ini menggunakan metode Max sebagaimana ditunjukkan Gambar 3.15 dengan alur sebagai berikut:

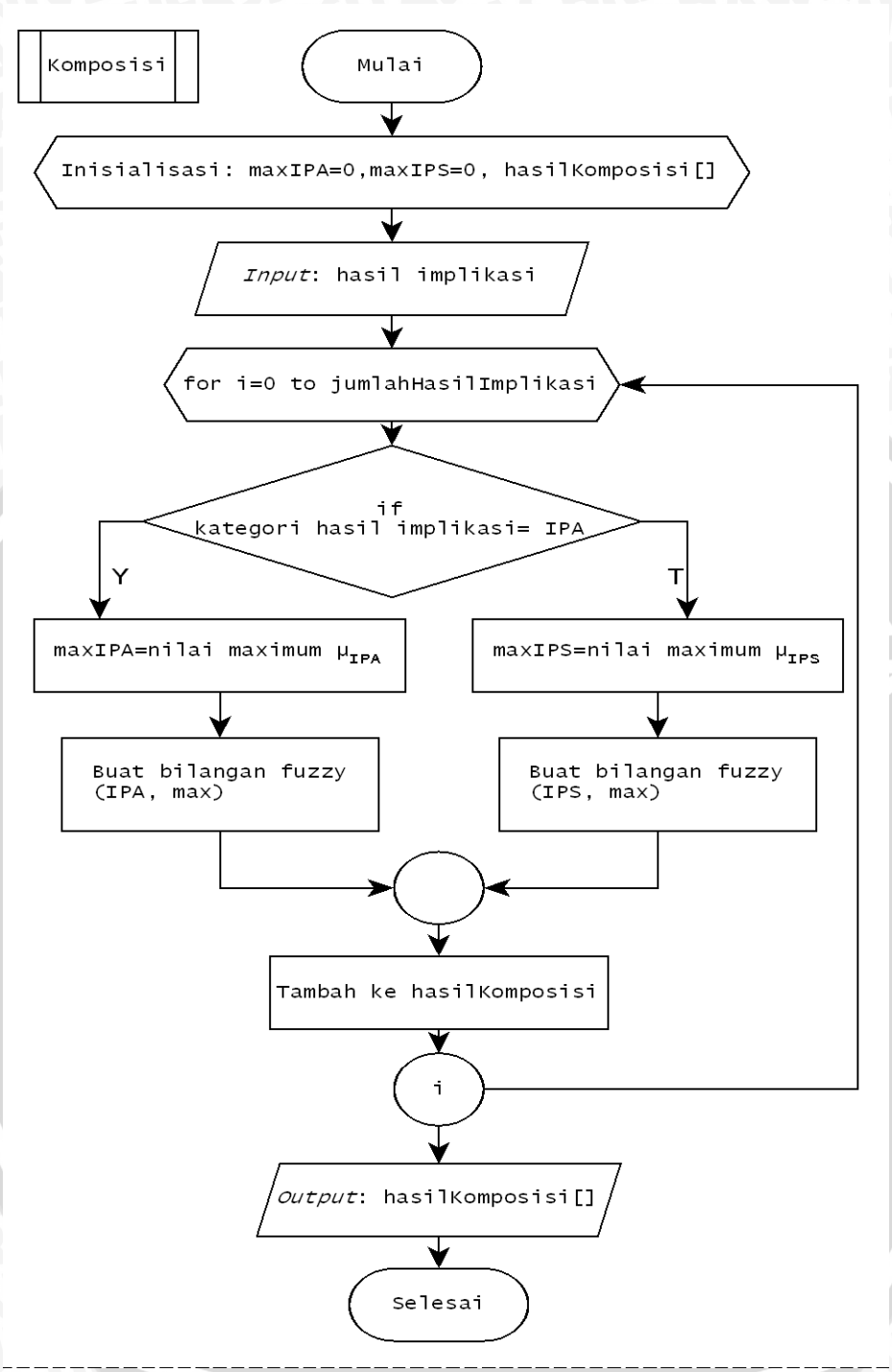
1. Inisialisasi variabel untuk menyimpan nilai maksimum dari masing-masing variabel output (max_{IPA}, max_{IPS}) dan himpunan hasil komposisi.
2. *Input* yang dibutuhkan berupa himpunan hasil implikasi.
3. Untuk iterasi *i*=0 hingga jumlah hasil implikasi:

   Lakukan pengecekan apakah kategori hasil implikasi *i* adalah IPA, jika ya maka:

   a. max_{IPA} diatur sama dengan nilai maksimum dari derajat keanggotaan (μ) yang berkategori IPA,
   b. Buat bilangan *fuzzy* dengan kategori IPA dan nilai derajat keanggotaan max_{IPA}.

   Jika hasil implikasi *i* bukan IPA, maka:

   a. max_{IPS} diatur sama dengan nilai maksimum dari derajat keanggotaan (μ) yang berkategori IPS,
   b. Buat bilangan *fuzzy* dengan kategori IPS dan nilai derajat keanggotaan max_{IPS}.
4. Tambahkan bilangan *fuzzy* yang dibuat ke dalam himpunan hasil komposisi.
5. *Output* yang dihasilkan berupa himpunan hasil komposisi yang terdiri atas bilangan *fuzzy*.

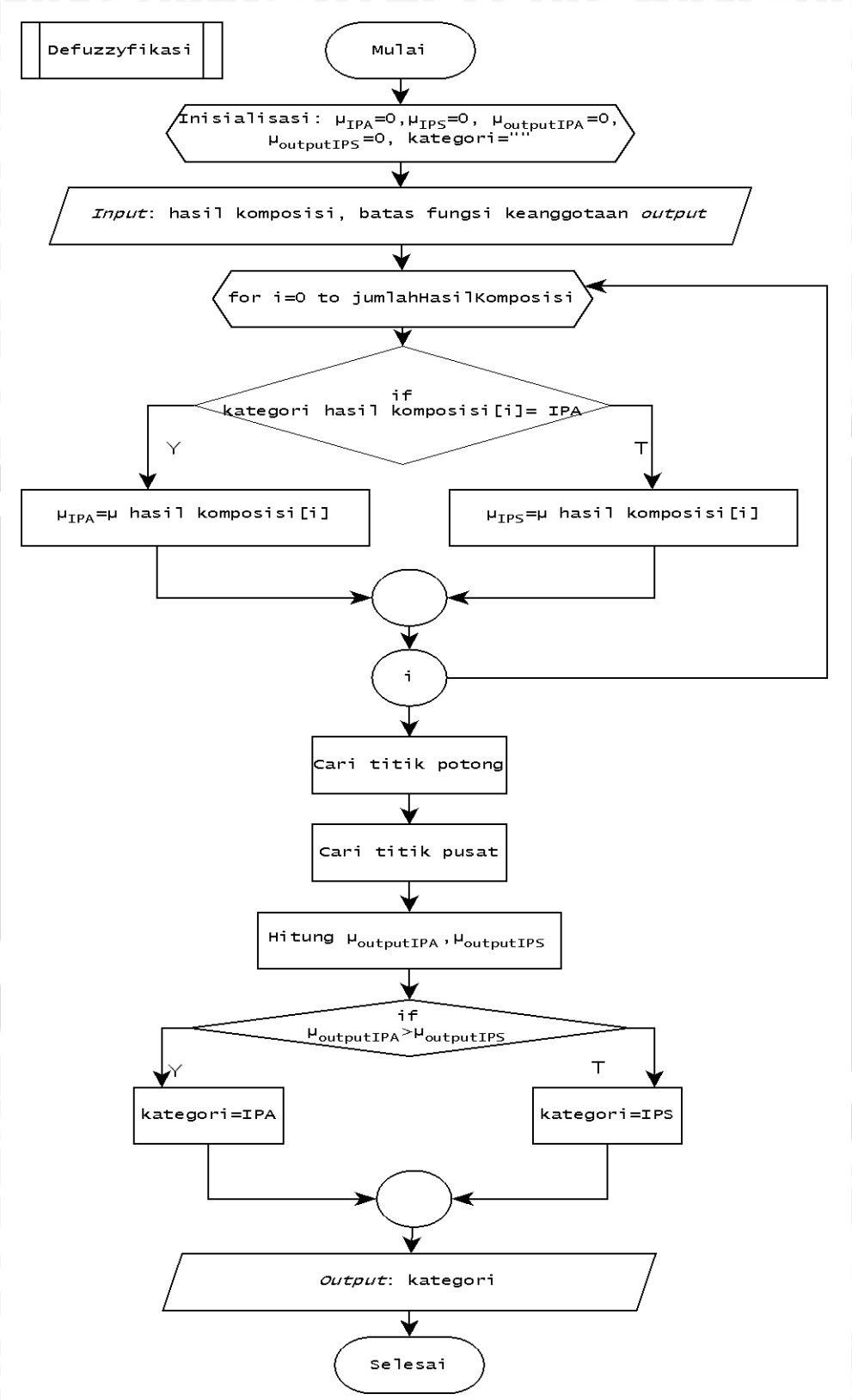


Gambar 3.15. Proses komposisi

5) Proses defuzzikasi

Proses ini merupakan proses untuk mengubah bilangan *fuzzy* menjadi bilangan dalam bentuk bilangan *crisps*. Metode yang digunakan adalah metode centroid dimana solusi diperoleh dengan cara mencari titik pusat (z*) dari masing-masing daerah fuzzy. Adapun gambaran dari proses defuzzifikasi dapat dilihat pada Gambar 3.16 dengan alur sebagai berikut:

1. Inisialisasi beberapa variabel yang digunakan dalam proses, yaitu μ_{IPA} μ_{IPS} $\mu_{outputIPA}$ $\mu_{outputIPS,}$ dan *kategori*.
2. *Input* yang dibutuhkan berupa himpunan bilangan *fuzzy* hasil komposisi dan batas-batas fungsi keanggotaan variabel *output*.
3. Untuk iterasi i=0 hingga jumlah hasil komposisi, lakukan:
   Cek apakah kategori hasil komposisi=IPA, jika ya atur nilai μ_{IPA} dengan nilai derajat keanggotaan hasil komposisi ke *i*, jika tidak atur nilai μ_{IPS} dengan nilai derajat keanggotaan hasil komposisi ke *i*
4. Cari titik potong, yaitu nilai *x* yang memiliki derajat keanggotaan μ_{IPA} dan μ_{IPS}.
5. Cari titik pusat(z*) menggunakan rumus dari pencarian titik pusat menggunakan metode centroid.
6. Hitung masing-masing derajat keanggotaan untuk output IPA dan IPS ($\mu_{outputIPA}$ dan $\mu_{outputIPS}$).
7. Lakukan pengecekan, apakah derajat keanggotaan untuk output IPA lebih besar daripada derajat keanggotaan untuk output IPS ($\mu_{outputIPA} > \mu_{outputIPS}$), jika ya maka kategori= IPA, jika tidak maka kategori=IPS.
8. *Output* dari proses ini berupa kategori jurusan.



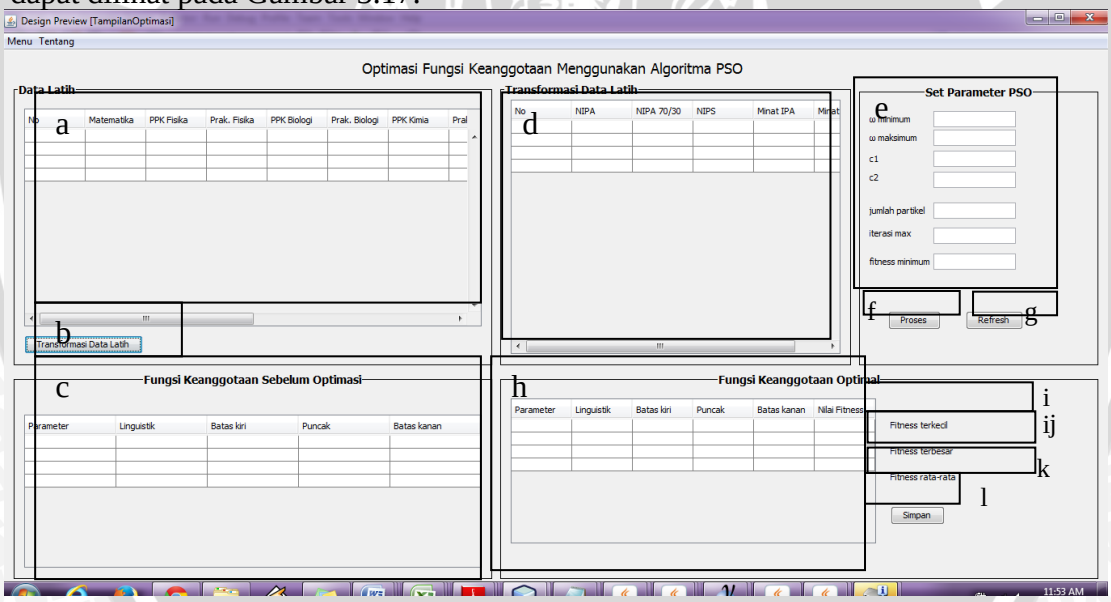
Gambar 3.16. Proses Defuzzifikasi

3.3.4. Analisa dan Perancangan Antarmuka

Pada subbab sebelumnya telah dijelaskan bagaimana perancangan jalannya sistem yang dimulai dari proses optimasi terhadap batas-batas fungsi keanggotaan hingga proses pengujian dengan menggunakan sistem inferensi *fuzzy* Mamdani. Berdasarkan alur proses dari sistem yang akan dibangun, maka perancangan antarmuka juga disesuaikan dengan kebutuhan sistem. Pada penelitian ini akan dirancang 2 tampilan utama antarmuka, yaitu antarmuka untuk proses optimasi fungsi keanggotaan menggunakan data latih dan proses pengujian data uji menggunakan aturan yang dipilih.

3.3.4.1. Antarmuka Proses Optimasi Fungsi Keanggotaan

Antarmuka ini digunakan untuk menampilkan antarmuka proses optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization*, yang dapat dilihat pada Gambar 3.17.



Gambar 3.17. Antarmuka Proses Optimasi Fungsi Keanggotaan

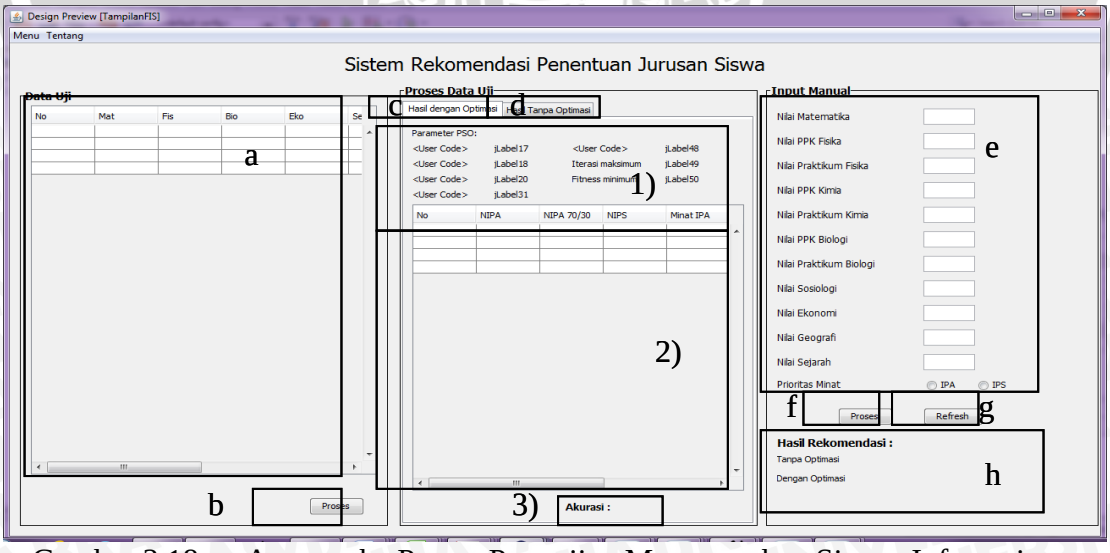
Bagian dari antarmuka tersebut antara lain :

a. Tabel untuk menampilkan data latih
b. Tombol yang digunakan untuk melakukan transformasi data
c. Tabel yang berisi fungsi keanggotaan awal sebelum dioptimasi
d. Tabel yang menampilkan data latih yang telah ditransformasi

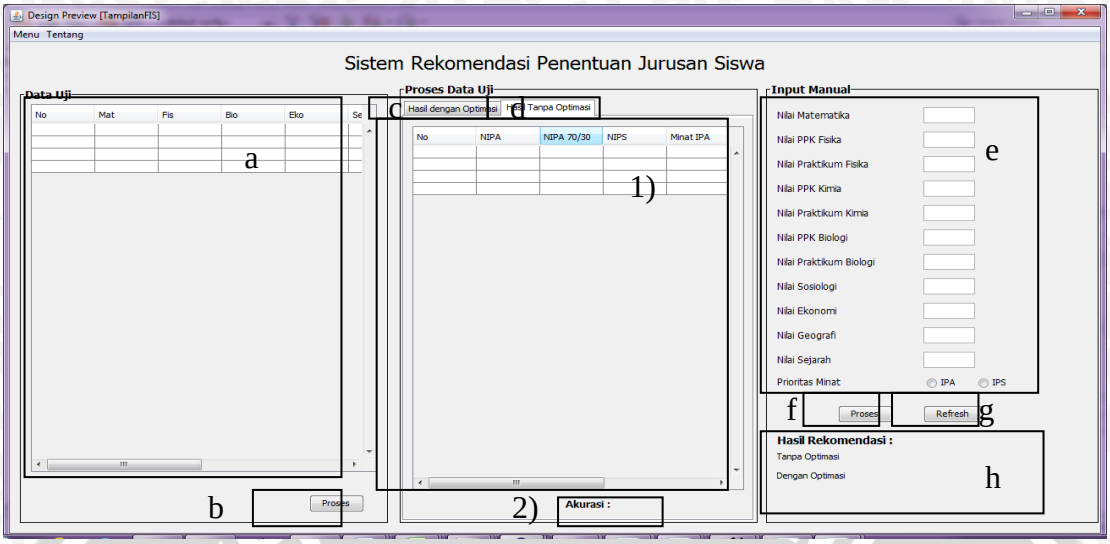
e. *Field* untuk menentukan parameter PSO, yang terdiri atas ω (inersia) maksimum, ω (inersia) minimum, iterasi maksimum, konstanta c_1 , c_2, jumlah partikel, dan *cost* minimum
f. Tombol untuk melakukan proses optimasi
g. Tombol *refresh*, untuk mengosongkan kembali *field* parameter yang dijelaskan pada poin e
h. Tabel yang menunjukkan batas-batas fungsi keanggotaan baru yang merupakan hasil optimasi fungsi keanggotaan awal
i. *Text* yang menampilkan nilai *fitness/cost* terkecil dari satu kali proses optimasi seluruh fungsi keanggotaan
j. *Text* yang menampilkan nilai *fitness/cost* terbesar dari satu kali proses optimasi seluruh fungsi keanggotaan
k. *Text* yang menampilkan nilai *fitness/cost* rata-rata dari satu kali proses optimasi seluruh fungsi keanggotaan
l. Tombol untuk menyimpan batas-batas fungsi keanggotaan baru.

3.3.4.2.Antarmuka Pengujian Menggunakan Sistem Inferensi *Fuzzy*

Antarmuka ini digunakan untuk melakukan pengujian menggunakan sistem inferensi *fuzzy* Mamdani. Adapun gambaran dari antarmuka ini dapat dilihat pada Gambar 3.18.a., dan Gambar 3.18.b.



Gambar 3.18.a. Antarmuka Proses Pengujian Menggunakan Sistem Inferensi *Fuzzy*



Gambar 3.18.b. Antarmuka Proses Pengujian Menggunakan Sistem Inferensi *Fuzzy*

Antarmuka ini terdiri dari tiga bagian utama, yaitu bagian untuk menampilkan data uji, hasil sistem inferensi *fuzzy* untuk data uji, dan *field* pengujian manual. Adapun keterangan untuk Gambar 3.18.a., dan Gambar 3.18.b. adalah:

a. Tabel yang menampilkan data uji
b. Tombol untuk melakukan proses perhitungan data uji menggunakan sistem inferensi *fuzzy*
c. *Tab* untuk menampilkan hasil perhitungan sistem inferensi *fuzzy* untuk data uji menggunakan batas yang telah optimal, terdiri atas:
    1) *Text* yang menampilkan parameter PSO yang digunakan untuk mengoptimasi fungsi keanggotaan
    2) Tabel yang menampilkan parameter berupa nilai NIPA, NIPA70/30, NIPS, minat siswa, hasil sistem, dan hasil sebenarnya
    3) *Text* yang menampilkan persen akurasi perhitungan.
d. *Tab* untuk menampilkan hasil perhitungan sistem inferensi *fuzzy* untuk data uji menggunakan batas awal sebelum dioptimasi, terdiri atas:
    1) Tabel yang menampilkan parameter berupa nilai NIPA, NIPA70/30, NIPS, minat siswa, hasil sistem menggunakan fungsi keanggotaan awal, dan hasil sebenarnya

2) *Text* yang menampilkan persen akurasi perhitungan.

e. *Field* untuk memasukkan data nilai dan minat siswa

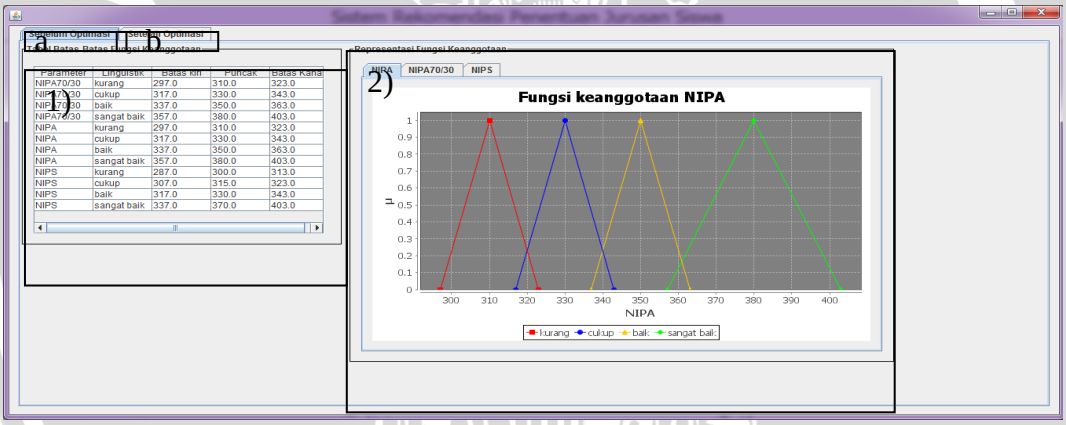
f. Tombol untuk melakukan proses penghitungan data input

g. Tombol untuk mengosongkan kembali *field* yang dijelaskan pada poin e

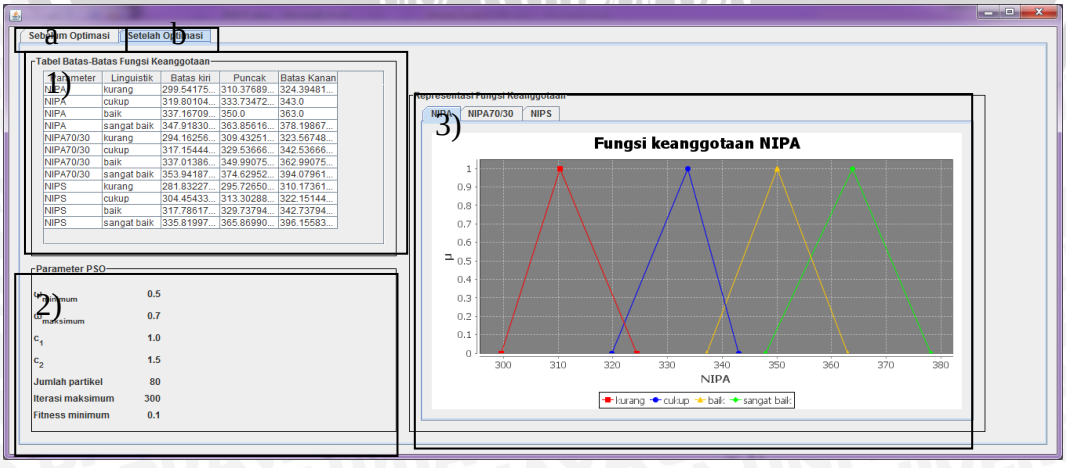
h. *Text* yang digunakan untuk menampilkan hasil perhitungan, baik yang menggunakan batas yang telah optimal, maupun yang menggunakan batas awal.

3.3.4.3.Antarmuka Tampilan Fungsi Keanggotaan

Antarmuka ini digunakan untuk menampilkan batas-batas fungsi keanggotaan baik sebelum maupun sesudah optimasi dalam bentuk tabel dan representasi fungsi keanggotaan tersebut dalam bentuk grafik. Adapun gambaran dari antarmuka ini dapat dilihat pada Gambar 3.19.a. dan 3.19.b.



Gambar 3.19.a. Antarmuka Tampilan Fungsi Keanggotaan Awal



Gambar 3.19.b. Antarmuka Tampilan Fungsi Keanggotaan Optimal

Seperti yang dapat dilihat pada Gambar 3.19.a dan 3.19.b. antarmuka ini memiliki dua *tab* yaitu:

a. Sebelum optimasi, terdiri atas:
   1) Tabel yang menampilkan daftar fungsi keanggotaan awal
   2) Bagian berupa *tab* yang berisi grafik dari fungsi keanggotaan untuk masing-masing parameter.
b. Setelah optimasi, terdiri atas:
   1) Tabel yang menampilkan daftar batas fungsi keanggotaan optimal
   2) *Text* yang menampilkan parameter PSO yang digunakan dalam proses optimasi
   3) Bagian berupa *tab* yang berisi grafik dari fungsi keanggotaan untuk masing-masing parameter.

3.4. Perhitungan Manual

Perhitungan manual dilakukan untuk mengimplementasikan sistem secara matematis lewat perhitungan langkah demi langkah pada data latih dan data uji. Proses perhitungan manual dibedakan menjadi dua bagian sesuai dengan analisa perancangan sistem, yaitu proses perhitungan manual proses optimasi fungsi keanggotaan yang menerapkan algoritma *Particle Swarm Optimization* dan sistem inferensi *fuzzy* Mamdani.

3.4.1. Proses Optimasi Fungsi Keanggotaan Menggunakan Algoritma *Particle Swarm Optimization*

Sebelum melakukan optimasi dengan algoritma *Particle Swarm Optimization*, dipilih beberapa data sebagai data latih untuk proses ini. Jumlah atribut yang digunakan sebanyak tigabelas parameter yang terdiri dari nilai matematika, PPK fisika, praktikum fisika, PPK biologi, praktikum biologi, PPK kimia, praktikum kimia, sejarah, ekonomi, geografi, sosiologi, minat IPA, dan minat IPS. Contoh data latih yang digunakan dapat dilihat pada tabel 3.1.

Tabel 3.1. Contoh data latih yang digunakan

mat	fis		bio		kim		sej	eko	geo	sos	minat		jurusan
	ppk	prak	ppk	prak	ppk	prak					ipa	ips	
84	92	91	84	89	90	92	87	89	82	77	1	2	ipa
78	94	86	82	89	87	92	75	80	75	75	1	2	ipa
76	82	87	80	88	82	90	82	80	82	76	1	2	ips
83	84	91	87	86	90	78	83	79	78	82	1	2	ipa
79	75	83	80	80	84	92	76	83	82	85	1	2	ips
84	85	95	80	87	85	92	81	83	79	84	1	2	ipa
75	75	88	80	80	78	92	78	80	77	85	1	2	ips
75	78	86	85	89	86	90	83	79	78	80	2	1	ips
97	90	91	87	89	95	93	82	89	84	83	1	2	ipa
88	78	92	80	93	87	83	78	85	75	82	1	2	ipa

Nilai-nilai yang ada pada data latih ditransformasi terlebih dahulu menjadi tiga parameter, yaitu NIPA, NIPA70/30, dan NIPS. Rumus untuk melakukan transformasi nilai tersebut ditunjukkan pada persamaan (3-1), (3-2), dan (3-3). Sedangkan contoh data yang telah ditransformasi dapat dilihat pada Tabel 3.2.

$$NIPA = nilai\ matematika + nilai\ ppk\ fisika + nilai\ ppk\ biologi + nilai\ ppk\ kimia \quad (3\text{-}1)$$

$$NIPA70/30 = nilai\ matematika + (0.7 \times nilai\ ppk\ fisika + 0.3 \times nilai\ praktikum\ fisika) + (0.7 \times nilai\ ppk\ biologi + 0.3 \times nilai\ praktikum\ biologi) + (0.7 \times nilai\ ppk\ kimia + 0.3 \times nilai\ praktikum\ kimia) \quad (3\text{-}2)$$

$$NIPS = nilai\ sejarah + nilai\ ekonomi + nilai\ geografi + nilai\ sosiologi \quad (3\text{-}3)$$

Tabel 3.2. Contoh data yang telah ditransformasi

NIPA 70/30	NIPA	NIPS
352	350	335
342	341	305
326	320	320
342	344	322
323	318	326
341	334	327
316	308	320
329	324	320

369	369	338
340	333	320

Pada perhitungan manual optimasi fungsi keanggotaan ini, digunakan parameter NIPA70/30 dengan variabel linguistik kurang sebagai contoh perhitungan manual. Langkah pertama yang dilakukan untuk melakukan optimasi adalah menentukan beberapa parameter PSO dan batasan awal fungsi keanggotaan, sebagai berikut:

*Iterasi ke-0
---

a. Parameter PSO:

jumlah partikel = 10
c_1 = 1
c_2 = 1
ω_{max} = 0.9
ω_{min} = 0.4
iterasi maksimum = 10
cost minimum = 0.010

b. Batasan awal:

xa = 297 ya = 0
xb = 310 yb = 1
xc = 323 yc = 0

Langkah kedua membentuk beberapa partikel sesuai dengan jumlah partikel yang ditentukan, dimana setiap partikel tersusun atas tiga data dari parameter NIPA70/30 secara acak:

Tabel 3.3. Partikel acak untuk proses optimasi

	a	b	c
x_1	318	325	326
x_2	317	323	352
x_3	313	327	351
x_4	321	340	340
x_5	317	342	370
x_6	340	340	340
x_7	316	340	369
x_8	326	340	342
x_9	313	316	340
x_{10}	340	318	342

Langkah ketiga adalah melakukan evaluasi nilai fungsi tujuan, dimana tujuan optimasi ini adalah meminimalkan nilai *cost* yang menggunakan rumus MSE(*mean square error*) sesuai persamaan (2-8) dengan y merupakan nilai derajat keanggotaan dari partikel-partikel PSO, dan ŷ merupakan nilai derajat keanggotaan dari batas awal sebagaimana tercantum pada tabel 3.4. Rumus perhitungan MSE dalam perhitungan ini dapat dituliskan sebagai berikut:

$$MSE = \frac{(y_{1a}-\hat{y}_a)^2+(y_{1b}-\hat{y}_b)^2+(y_{1c}-\hat{y}_c)^2}{\hat{y}_a{}^2+\hat{y}_b{}^2+\hat{y}_c{}^2} \quad (3\text{-}4)$$

Tabel 3.4. Evaluasi nilai fungsi tujuan iterasi ke-0

	y			$\sum(y-\hat{y})^2$	*cost*
	a	b	c		
x_1	1.6154	2.1538	-0.2308	3.9941	3.9941
x_2	1.5385	2.0000	-2.2308	8.3432	8.3432
x_3	1.2308	2.3077	-2.1538	7.8639	7.8639
x_4	1.8462	3.3077	-1.3077	10.4438	10.4438
x_5	1.5385	3.4615	-3.6154	21.4970	21.4970
x_6	3.3077	3.3077	-1.3077	17.9763	17.9763
x_7	1.4615	3.3077	-3.5385	19.9822	19.9822
x_8	2.2308	3.3077	-1.4615	12.4379	12.4379
x_9	1.2308	1.4615	-1.3077	3.4379	3.4379
x_{10}	3.3077	1.6154	-1.4615	13.4556	13.4556

Langkah keempat adalah menentukan kecepatan awal dari partikel, semua partikel diasumsikan memiliki kecepatan awal 0 ($v_{1a} = v_{2a=} \ldots = v_{1b} = v_{2b} = \ldots = v_{1c} = v_{2c} = \ldots = 0$).

Langkah kelima, menentukan Pbest dan Gbest awal. Pbest pada iterasi ke-0 memiliki posisi yang sama dengan partikel acak yang telah ditentukan sebelumnya seperti pada Tabel 3.3. Pbest juga memiliki nilai *cost* yang bernilai sama dengan *cost* partikel seperti yang tercantum pada Tabel 3.4. Posisi Pbest dan nilai *cost* Pbest ditunjukkan pada Tabel 3.5. Sedangkan posisi Gbest iterasi ke-0, diinisialisasi sama dengan batas awal sebelum dilakukan optimasi.

Tabel 3.5. Pbest iterasi ke-0

	y			*cost*
	a	b	c	Pbest
$pbest_1$	318	325	326	3.9941
$pbest_2$	317	323	352	8.3432
$pbest_3$	313	327	351	7.8639
$pbest_4$	321	340	340	10.4438
$pbest_5$	317	342	370	21.4970
$pbest_6$	340	340	340	17.9763
$pbest_7$	316	340	369	19.9822
$pbest_8$	326	340	342	12.4379
$pbest_9$	313	316	340	3.4379
$pbest_{10}$	340	318	342	13.4556

Posisi Gbest iterasi ke-0:

$Gbest_a$ = 297

$Gbest_b$ = 310

$Gbest_c$ = 323

Langkah keenam adalah menghitung kecepatan partikel untuk iterasi berikutnya, sehingga pada iterasi ke-0 ini dihitung kecepatan partikel untuk iterasi ke-1 sesuai dengan persamaan (2-6). Sebelum menghitung kecepatan, ditentukan nilai konstanta r_1, r_2 dan ω terlebih dahulu. Konstanta r_1 dan r_2 merupakan bilangan acak yang dibangkitkan pada setiap iterasi, sedangkan ω merupakan hasil perhitungan berat inersia sesuai rumus (2-7). Konstanta r_1, r_2, dan ω yang digunakan pada iterasi ini adalah : r_1= 0.9044, r_2= 0.7177, ω=0.9000. Hasil perhitungan kecepatan dari langkah keenam ini ditunjukkan pada Tabel 3.6.

Tabel 3.6. Hasil perhitungan $v_{(k+1)}$ iterasi 0

		v	
	a	b	c
v_1	-15.0712	-10.7651	-2.1530
v_2	-14.3535	-9.3298	-20.8126
v_3	-11.4828	-12.2005	-20.0949
v_4	-17.2242	-21.5302	-12.2005
v_5	-14.3535	-22.9656	-33.7307
v_6	-30.8600	-21.5302	-12.2005
v_7	-13.6358	-21.5302	-33.0130
v_8	-20.8126	-21.5302	-13.6358
v_9	-11.4828	-4.3060	-12.2005
v_{10}	-30.8600	-5.7414	-13.6358

Langkah ketujuh menghitung posisi baru dari masing masing partikel sesuai persamaan (2-7). Hasil perhitungan posisi baru dari setiap partikel ditunjukkan pada Tabel 3.7.

Tabel 3.7. Hasil perhitungan $x_{(k+1)}$ iterasi 0

	a	b	c
x_1	302.93	314.23	323.85
x_2	302.65	313.67	331.19
x_3	301.52	314.80	330.91
x_4	303.78	318.47	327.80
x_5	302.65	319.03	336.27
x_6	309.14	318.47	327.80
x_7	302.36	318.47	335.99
x_8	305.19	318.47	328.36
x_9	301.52	311.69	327.80
x_{10}	309.14	312.26	328.36

Setelah menghitung posisi baru dari masing-masing partikel, dilakukan langkah kedelapan, yaitu melakukan evaluasi kembali untuk nilai *cost* dari setiap posisi partikel untuk iterasi ke-1 dengan cara yang sama pada langkah ketiga, namun yang digunakan sebagai batas awal adalah Gbest pada iterasi sebelumnya, dan didapatkan hasil yang tercantum pada Tabel 3.8.

Tabel 3.8. Evaluasi nilai fungsi tujuan iterasi 1

	y			$\sum(y-\hat{y})^2$	*cost*
	a	b	c		
x_1	0.4562	1.3254	-0.0654	0.3182	0.3182
x_2	0.4346	1.2823	-0.6300	0.6655	0.6655
x_3	0.3477	1.3692	-0.6085	0.6274	0.6274
x_4	0.5215	1.6515	-0.3692	0.8328	0.8328
x_5	0.4346	1.6946	-1.0208	1.7134	1.7134
x_6	0.9338	1.6515	-0.3692	1.4329	1.4329
x_7	0.4123	1.6515	-0.9992	1.5930	1.5930
x_8	0.6300	1.6515	-0.4123	0.9914	0.9914
x_9	0.3477	1.1300	-0.3692	0.2741	0.2741
x_{10}	0.9338	1.1738	-0.4123	1.0723	1.0723

Pada langkah kesembilan ini dilakukan pencarian posisi Pbest dari setiap partikel dan posisi Gbest seperti pada langkah kelima. Jika nilai *cost* Pbest pada iterasi ini lebih kecil daripada nilai *cost* Pbest pada iterasi sebelumnya (*cost* $Pbest_1$< *cost* $Pbest_0$) maka posisi Pbest yang dipilih adalah posisi baru partikel ($Pbest_1$), jika tidak maka nilai Pbest sama dengan iterasi sebelumnya ($Pbest_1$). Cara yang sama digunakan untuk mencari posisi Gbest, dan yang menjadi pembanding adalah nilai *cost* minimum dari *cost* Pbest. Jika nilai minimum *cost* Pbest iterasi ini lebih kecil dari nilai minimum *cost* Pbest iterasi sebelumnya (minimum *cost* $Pbest_1$< *cost* $Pbest_0$) maka posisi Gbest yang dipilih adalah posisi Pbest yang memiliki nilai *cost* minimum dari iterasi ini, jika tidak maka posisi Gbest sama dengan iterasi sebelumnya. Hasil posisi Pbest pada iterasi ke-1 dapat dilihat pada Tabel 3.9.

Tabel 3.9. Pbest iterasi ke-1

	y			*cost*
	a	b	c	Pbest
$pbest_1$	302.93	314.23	323.85	0.3182
$pbest_2$	302.65	313.67	331.19	0.6655
$pbest_3$	301.52	314.8	330.91	0.6274
$pbest_4$	303.78	318.47	327.8	0.8328
$pbest_5$	302.65	319.03	336.27	1.7134
$pbest_6$	309.14	318.47	327.8	1.4329
$pbest_7$	302.36	318.47	335.99	1.5930
$pbest_8$	305.19	318.47	328.36	0.9914
$pbest_9$	301.52	311.69	327.8	0.2741
$pbest_{10}$	309.14	312.26	328.36	1.0723

Berdasarkan nilai yang tercantum pada Tabel 3.9 nilai *cost* minimum yang didapatkan adalah 0.2741 yaitu pada posisi Pbest partikel ke-9, sehingga posisi partikel tersebut digunakan sebagai posisi Gbest sebagai berikut:

$Gbest_a = 301.52$
$Gbest_b = 311.69$
$Gbest_c = 327.8$

Setelah Pbest dan Gbest baru dari iterasi ke-1 ditemukan, perhitungan akan dilanjutkan dengan mengulang kembali langkah keenam hingga langkah kesembilan sampai didapatkan nilai *cost* yang diharapkan (kurang atau sama dengan *cost* minimum) atau hingga mencapai jumlah iterasi maksimum. Jika perhitungan telah mencapai iterasi maksimum namun belum dapat mencapai target nilai *cost* yang diharapkan, maka solusi yang dipilih adalah posisi Gbest dari iterasi yang memiliki nilai *cost* terkecil. Pada perhitungan manual ini, perhitungan dihentikan saat iterasi ke-7, karena nilai *cost* yang diharapkan telah terpenuhi. Berikut ini merupakan penjelasan secara singkat untuk perhitungan iterasi ke-7:

Seperti yang telah dijelaskan sebelumnya, akan dilakukan kembali langkah keenam yaitu menghitung kecepatan setiap partikel $v_{(k+1)}$ pada iterasi ke-6 dengan

r_1= 0.9342, r_2= 0.1629, dan ω= 0.6000. Kecepatan partikel $v_{(k+1)}$ pada iterasi ke-6 dapat dilihat pada tabel 3.10.

Tabel 3.10. Hasil perhitungan $v_{(k+1)}$ iterasi ke-6

	v		
	a	b	c
v_1	-13.6530	-10.5576	0.2276
v_2	-12.8939	-9.0359	-19.6153
v_3	-9.8386	-12.0937	-18.8418
v_4	-15.9328	-22.0112	-10.4415
v_5	-12.8939	-23.5305	-33.3593
v_6	-30.4561	-22.0112	-10.4415
v_7	-12.1206	-22.0112	-32.5881
v_8	-19.7606	-22.0112	-11.9753
v_9	-9.8386	-3.6910	-10.4415
v_{10}	-30.4561	-5.2235	-11.9753

Langkah selanjutnya adalah mengulang langkah ketujuh, yaitu menghitung posisi baru $x_{(k+1)}$ untuk iterasi ke-6. Hasil posisi baru tersebut ditunjukkan oleh Tabel 3.11.

Tabel 3.11. Hasil perhitungan $x_{(k+1)}$ iterasi ke-6

	a	b	c
x_1	301.80	312.25	326.85
x_2	301.74	312.12	328.53
x_3	301.48	312.39	328.47
x_4	302.00	313.23	327.76
x_5	301.74	313.35	329.69
x_6	303.22	313.23	327.76
x_7	301.67	313.23	329.63
x_8	302.32	313.23	327.88
x_9	301.48	311.67	327.76
x_{10}	303.22	311.80	327.88

Langkah berikutnya adalah melakukan evaluasi nilai fungsi tujuan (nilai *cost*) untuk iterasi ke-7 seperti langkah kedelapan. Hasil dari proses ini dapat dilihat pada Tabel 3.12.

Tabel 3.12. Evaluasi nilai fungsi tujuan iterasi ke-7

	y			$\sum(y-\hat{y})^2$	*cost*
	a	b	c		
x_1	-0.1113	1.0075	0.1381	0.0315	0.0315
x_2	-0.1178	0.9936	0.0394	0.0155	0.0155
x_3	-0.1456	1.0225	0.0429	0.0235	0.0235
x_4	-0.0899	1.1124	0.0846	0.0279	0.0279
x_5	-0.1178	1.1253	-0.0288	0.0304	0.0304
x_6	0.0407	1.1124	0.0846	0.0215	0.0215
x_7	-0.1253	1.1124	-0.0253	0.0290	0.0290
x_8	-0.0557	1.1124	0.0776	0.0218	0.0218
x_9	-0.1456	0.9454	0.0846	0.0313	0.0313
x_{10}	0.0407	0.9593	0.0776	0.0093	0.0093

Langkah berikutnya adalah mengulangi langkah kesembilan, yaitu mencari posisi Pbest dan Gbest terbaru. Daftar Pbest untuk iterasi ke-7 ditunjukkan pada Tabel 3.13.

Tabel 3.13. Pbest iterasi ke-7

	y			*cost*
	a	b	c	Pbest
$pbest_1$	301.8	312.25	326.85	0.0315
$pbest_2$	301.74	312.12	328.53	0.0155
$pbest_3$	301.48	312.39	328.47	0.0235
$pbest_4$	302	313.23	327.76	0.0279
$pbest_5$	301.74	313.35	329.69	0.0304
$pbest_6$	303.22	313.23	327.76	0.0215
$pbest_7$	301.67	313.23	329.63	0.0290
$pbest_8$	302.32	313.23	327.88	0.0218
$pbest_9$	302.84	312.18	329.2	0.0267
$pbest_{10}$	303.22	311.8	327.88	0.0093

Nilai minimum untuk *cost* Pbest pada iterasi ke-7 ini adalah 0.093, sehingga posisi Gbest yang dipilih adalah:

$Gbest_a$ = 310.49

$Gbest_b$ = 315.05

$Gbest_c$ = 337.32

Dari perhitungan tersebut hasil nilai *cost* dan Gbest secara keseluruhan ditunjukkan pada Tabel 3.14.

Tabel 3.14. Daftar hasil perhitungan nilai *cost* dan Gbest seluruh iterasi

Iterasi	*cost*	gbest a	gbest b	gbest c
1	0.2741	301.52	311.69	327.80
2	0.2741	301.52	311.69	327.80
3	0.2741	301.52	311.69	327.80
4	0.0267	302.84	312.18	329.20
5	0.0267	302.84	312.18	329.20
6	0.0267	302.84	312.18	329.20
7	0.0093	303.22	311.80	327.88

Berdasarkan hasil perhitungan tersebut, dapat dilihat bahwa sebelum iterasi mencapai batas iterasi maksimum, telah didapatkan nilai *cost* yang memenuhi target, yaitu 0.093 pada iterasi ke-7. Oleh karena itu posisi Gbest dari iterasi ke-7 digunakan sebagai batas baru untuk parameter NIPA70/30 dengan variabel linguistik rendah, yaitu:

Batas kiri = $Gbest_a$ = 310.49

Nilai puncak = $Gbest_b$ = 315.05

Batas kanan = $Gbest_c$ = 337.32

Proses optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization* juga dilakukan pada parameter NIPA dan NIPS. Sebagai contoh perhitungan manual, parameter PSO yang digunakan untuk parameter NIPA dan NIPS sama dengan parameter PSO yang digunakan untuk melakukan optimasi parameter NIPA70/30. Partikel acak yang digunakan untuk parameter NIPA tercantum pada Tabel 3.15 dan untuk parameter NIPS tercantum pada Tabel 3.16.

Tabel 3.15. Partikel untuk parameter NIPA

	a	b	c
x_1	335	360	372
x_2	308	326	344
x_3	308	336	350
x_4	308	338	344
x_5	308	324	340
x_6	314	334	370
x_7	318	324	335
x_8	309	318	335
x_9	313	337	341
x_{10}	310	347	348

Tabel 3.16. Partikel acak untuk parameter NIPS

	a	b	c
x_1	311	318	334
x_2	320	333	334
x_3	320	322	325
x_4	305	320	322
x_5	320	326	333
x_6	320	321	325
x_7	312	323	332
x_8	313	320	326
x_9	320	331	336
x_{10}	322	324	337

Hasil optimasi secara keseluruhan untuk parameter NIPA70/30, NIPA, dan NIPS dapat dilihat pada Tabel 3.17.

Tabel 3.17. Hasil optimasi fungsi keanggotaan semua parameter

Parameter	Variabel linguistik	Sebelum PSO			Setelah PSO		
		batas kiri	nilai puncak	batas kanan	batas kiri	nilai puncak	batas kanan
NIPA70/30	kurang	297	310	323	303.22	311.80	327.88
	cukup	317	330	343	315.42	328.82	346.16
	baik	337	350	363	325.68	345.46	366.96
	sangat baik	357	380	403	328.00	352.44	379.07
NIPA	kurang	297	310	323	297.03	310.01	323.03
	cukup	317	330	343	314.45	334.47	341.72
	baik	337	350	363	336.25	353.71	366.34
	sangat baik	357	380	403	348.24	372.04	390.66
NIPS	kurang	287	300	313	299.55	312.28	319.27
	cukup	307	315	323	306.21	316.98	322.61
	baik	317	330	343	318.36	330.46	339.81
	sangat baik	337	370	403	321.00	334.78	338.03

3.4.2. Sistem Inferensi *Fuzzy* Mamdani

Contoh data yang digunakan sebagai data uji dalam perhitungan manual ini adalah:

Nilai matematika : 79
Nilai PPK fisika : 75
Nilai praktikum fisika : 83
Nilai PPK biologi : 80
Nilai praktikum biologi : 80
Nilai PPK kimia : 84
Nilai praktikum kimia : 92
Nilai sejarah : 76
Nilai geografi : 83
Nilai ekonomi : 82
Nilai sosiologi : 85
Minat IPA : 1
Minat IPS : 2

Langkah pertama yang dilakukan untuk melakukan perhitungan dengan sistem inferensi *fuzzy* adalah melakukan transformasi data, seperti pada persamaan (3-1), (3-2), dan (3-3). Hasil transformasi nilai dari data uji diatas:

NIPA70/30: 323
NIPA: 318
NIPS: 326

Langkah kedua adalah melakukan fuzzifikasi pada setiap parameter *input*. Hasil optimasi pada Tabel 3.17. digunakan sebagai batas-batas fungsi keanggotaan masing-masing variabel linguistik pada setiap parameter.

a. Untuk parameter NIPA70/30

Representasi fungsi keanggotaan untuk parameter NIPA70/30 dengan menggunakan batas yang optimal adalah:

$$\mu_{kurang}[x] = \begin{cases} 0, x < 303.22 \; dan \; x > 327.88 \\ \frac{(x-303.22)}{(311.80-303.22)}, 303.22 \leq x \leq 311.80 \\ \frac{(327.88-x)}{(327.88-311.80)}, 311.80 < x \leq 327.88 \end{cases} \quad (3\text{-}5)$$

$$\mu_{cukup}[x] = \begin{cases} 0, \; x < 315.42 \; dan \; x > 346.16 \\ \frac{(x-315.42)}{(328.82-315.42)}, 315.42 \leq x \leq 328.82 \\ \frac{(346.16-x)}{(346.16-328.82)}, 328.82 < x \leq 346.16 \end{cases} \quad (3\text{-}6)$$

$$\mu_{baik}[x] = \begin{cases} 0, x < 325.68 \; dan \; x > 366.96 \\ \frac{(x-325.68)}{(345.46-325.68)}, 325.68 \leq x \leq 345.46 \\ \frac{(366.96-x)}{(366.96-345.46)}, 345.46 < x \leq 366.96 \end{cases} \quad (3\text{-}7)$$

$$\mu_{sangat\ baik}[x] = \begin{cases} 0, x < 328 \; dan \; x > 379.07 \\ \frac{(x-328)}{(352.44-328)}, 328 \leq x \leq 352.44 \\ \frac{(379.07-x)}{(379.07-352.44)}, 352.44 < x \leq 379.07 \end{cases} \quad (3\text{-}8)$$

Dengan menggunakan representasi fungsi keanggotaan tersebut, maka nilai derajat keanggotaan (μ) untuk NIPA70/30 323 adalah:

$\mu_{kurang}[323]$: 0.303483
$\mu_{cukup}[323]$: 0.565672
$\mu_{baik}[323]$: 0
$\mu_{sangat\ baik}[323]$: 0

b. Untuk parameter NIPA

Representasi fungsi keanggotaan untuk parameter NIPA dengan menggunakan batas yang optimal adalah:

$$\mu_{kurang}[x] = \begin{cases} 0, x < 297.03 \; dan \; x > 323.03 \\ \frac{(x-297.03)}{(310.01-297.03)}, 297.03 \leq x \leq 310.01 \\ \frac{(323.03-x)}{(323.03-310.01)}, 311.80 < x \leq 327.88 \end{cases} \quad (3\text{-}9)$$

$$\mu_{cukup}[x] = \begin{cases} 0, \; x < 314.45 \; dan \; x > 341.72 \\ \frac{(x-314.45)}{(334.47-314.45)}, 314.45 \leq x \leq 334.47 \\ \frac{(341.72-x)}{(341.72-334.47)}, 334.47 < x \leq 341.72 \end{cases} \quad (3\text{-}10)$$

$$\mu_{baik}[x] = \begin{cases} 0, x < 336.25 \; dan \; x > 366.34 \\ \frac{(x-336.25)}{(353.71-336.25)}, 336.25 \leq x \leq 353.71 \\ \frac{(366.34-x)}{(366.34-345.46)}, 353.71 < x \leq 366.34 \end{cases} \quad (3\text{-}11)$$

$$\mu_{sangat\ baik}[x] = \begin{cases} 0, x < 348.24 \; dan \; x > 390.66 \\ \frac{(x-348.24)}{(372.04-348.24)}, 348.24 \leq x \leq 372.04 \\ \frac{(390.66-x)}{(390.66-372.04)}, 372.04 < x \leq 390.66 \end{cases} \quad (3\text{-}12)$$

Dengan menggunakan representasi fungsi keanggotaan tersebut, maka nilai derajat keanggotaan (μ) untuk NIPA 318 adalah:

$\mu_{kurang}[318]$: 0.386329
$\mu_{cukup}[318]$: 0.177323
$\mu_{baik}[318]$: 0
$\mu_{sangat\ baik}[318]$: 0

c. Untuk parameter NIPS

Representasi fungsi keanggotaan untuk parameter NIPA dengan menggunakan batas yang optimal adalah:

$$\mu_{kurang}[x] = \begin{cases} 0, x < 299.55 \; dan \; x > 319.27 \\ \frac{(x-299.55)}{(312.28-303.22)}, 299.55 \leq x \leq 312.28 \\ \frac{(319.27-x)}{(319.27-312.28)}, 312.28 < x \leq 319.27 \end{cases} \quad (3\text{-}13)$$

$$\mu_{cukup}[x] = \begin{cases} 0, & x < 306.21 \; dan \; x > 322.61 \\ \frac{(x-306.21)}{(316.98-306.21)}, & 306.21 \leq x \leq 316.98 \\ \frac{(322.61-x)}{(322.61-316.98)}, & 316.98 < x \leq 322.61 \end{cases} \quad (3\text{-}14)$$

$$\mu_{baik}[x] = \begin{cases} 0, & x < 318.36 \; dan \; x > 339.81 \\ \frac{(x-318.36)}{(330.46-318.36)}, & 318.86 \leq x \leq 330.46 \\ \frac{(339.81-x)}{(339.81-330.46)}, & 330.46 < x \leq 339.81 \end{cases} \quad (3\text{-}15)$$

$$\mu_{sangat\ baik}[x] = \begin{cases} 0, & x < 321 \; dan \; x > 338.03 \\ \frac{(x-321)}{(334.78-321)}, & 321 \leq x \leq 334.78 \\ \frac{(338.03-x)}{(338.03-334.78)}, & 334.78 < x \leq 338.03 \end{cases} \quad (3\text{-}16)$$

Dengan menggunakan representasi fungsi keanggotaan tersebut, maka nilai derajat keanggotaan (μ) untuk NIPS 326 adalah:

$\mu_{kurang}[326]$: 0
$\mu_{cukup}[326]$: 0
$\mu_{baik}[326]$: 0.631405
$\mu_{sangat\ baik}[326]$: 0.362845

d. Untuk parameter minatIPA

$$\mu_{tinggi}[x] = \begin{cases} 1, x = 1 \\ 0, x \neq 1 \end{cases} \quad (3\text{-}17)$$

$$\mu_{rendah}[x] = \begin{cases} 1, x = 2 \\ 0, x \neq 2 \end{cases} \quad (3\text{-}18)$$

Hasil perhitungan nilai derajat keanggotaan (μ) untuk minatIPA 1:

$\mu_{tinggi}[1]$: 1
$\mu_{rendah}[1]$: 0

e. Untuk parameter minatIPS

$$\mu_{tinggi}[x] = \begin{cases} 1, x = 1 \\ 0, x \neq 1 \end{cases} \quad (3\text{-}19)$$

$$\mu_{rendah}[x] = \begin{cases} 1, x = 2 \\ 0, x \neq 2 \end{cases} \quad (3\text{-}20)$$

Hasil perhitungan nilai derajat keanggotaan (μ) untuk minatIPA 2:

$\mu_{tinggi}[1]$: 0

$\mu_{rendah}[1]$: 1

Langkah ketiga adalah memilih *rule* atau aturan yang ada sesuai dengan perhitungan fuzzifikasi, dimana aturan yang dipilih adalah aturan yang memiliki nilai derajat keanggotaan penyusunnya tidak sama dengan 0 ($\mu \neq 0$). Aturan-aturan yang terpilih adalah:

[R1] IF NIPA70/30 kurang AND NIPA kurang AND NIPS baik AND minatIPA tinggi AND minatIPS rendah THEN jurusan IPS

[R2] IF NIPA70/30 kurang AND NIPA kurang AND NIPS sangat baik AND minatIPA tinggi AND minatIPS rendah THEN jurusanIPS

[R3] IF NIPA70/30 kurang AND NIPA cukup AND NIPS baik AND minatIPA tinggi AND minatIPS rendah THEN jurusan IPA

[R4] IF NIPA70/30 kurang AND NIPA cukup AND NIPS sangat baik AND minatIPA tinggi AND minatIPS rendah THEN jurusanIPA

[R5] IF NIPA70/30 cukup AND NIPA kurang AND NIPS baik AND minatIPA tinggi AND minatIPS rendah THEN jurusan IPA

[R6] IF NIPA70/30 cukup AND NIPA kurang AND NIPS sangat baik AND minatIPA tinggi AND minatIPS rendah THEN jurusanIPA

[R7] IF NIPA70/30 cukup AND NIPA cukup AND NIPS baik AND minatIPA tinggi AND minatIPS rendah THEN jurusan IPA

[R8] IF NIPA70/30 cukup AND NIPA cukup AND NIPS sangat baik AND minatIPA tinggi AND minatIPS rendah THEN jurusanIPA.

Setelah memilih *rule* yang sesuai, dilakukan aplikasi fungsi implikasi dengan menggunakan fungsi MIN, yaitu memilih nilai derajat keanggotaan (μ) minimum dari masing-masing parameter input setiap *rule* yang digunakan sebagai nilai derajat keanggotaan dari parameter output.

[R1] IF NIPA70/30 kurang (0.303482587) AND NIPA kurang (0.386328725) AND NIPS baik (0.631404959) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPS (0.303482587)

[R2] IF NIPA70/30 kurang (0.303482587) AND NIPA kurang (0.386328725) AND NIPS sangat baik (0.362844702) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPS (0.303482587)

[R3] IF NIPA70/30 kurang (0.303482587) AND NIPA cukup (0.177322677) AND NIPS baik (0.631404959) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPA (0.177322677)

[R4] IF NIPA70/30 kurang (0.303482587) AND NIPA cukup (0.177322677) AND NIPS sangat baik (0.362844702) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPA (0.177322677)

[R5] IF NIPA70/30 cukup (0.565671642) AND NIPA kurang (0.386328725) AND NIPS baik (0.631404959) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPA (0.386328725)

[R6] IF NIPA70/30 cukup (0.565671642) AND NIPA kurang (0.386328725) AND NIPS sangat baik (0.362844702) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPA (0.362844702)

[R7] IF NIPA70/30 cukup (0.565671642) AND NIPA cukup (0.177322677) AND NIPS baik (0.631404959) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPA (0.177322677)

[R8] IF NIPA70/30 cukup (0.565671642) AND NIPA cukup (0.177322677) AND NIPS sangat baik (0.362844702) AND minatIPA tinggi (1) AND minatIPS rendah (1) THEN jurusan IPA (0.177322677)

Langkah selanjutnya adalah melakukan komposisi aturan menggunakan metode MAX, yaitu memilih nilai α_{cut} maksimum dari masing-masing parameter output yang dihasilkan dari masing-masing *rule* sebagai berikut:

α_{cut} ipa: 0.177322677 vs 0.177322677 vs 0.386328725 vs
0.362844702 vs 0.177322677 vs 0.177322677
α_{cut} ips: 0.303482587 vs 0.303482587

Hasil yang dipilih adalah:

α_{cut} ipa = 0.386328725 dan α_{cut} ips = 0.303482587

Setelah didapatkan α_{cut} dari masing-masing jurusan, dilakukan proses defuzzifikasi, yaitu mengubah bilangan *fuzzy* yang didapatkan menjadi bilangan *crisps* dengan menggunakan metode centroid.

Adapun representasi fungsi keanggotaan untuk parameter output adalah:

$$\mu_{IPA}[x] = \begin{cases} x, 0 \leq x \leq 1 \\ (2-x), 1 < x \leq 2 \\ 0, x > 2 \end{cases} \quad (3\text{-}21)$$

$$\mu_{IPS}[x] = \begin{cases} (x-1), 1 \leq x \leq 2 \\ (3-x), 2 < x \leq 3 \\ 0, x > 3 \end{cases} \quad (3\text{-}22)$$

Nilai α_{cut} ipa dan α_{cut} ips yang dihasilkan pada proses komposisi aturan tersebut kemudian dicari titik *x* yang sesuai dengan persamaan (3-21) dan (3-22), dan hasil titik *x* yang ditemukan adalah:

titik A: 0.386329
titik B: 1.613671
titik C: 1.696517
titik D: 2.696517

Dari titik-titik tersebut dicari nilai titik pusat z* dengan persamaan:

$$z^* = \frac{\sum_{j=1}^{n} z_j \mu(z_j)}{\sum_{j=1}^{n} \mu(z_j)} \quad (3\text{-}23)$$

dengan z_j= titik yang telah ditemukan dan $\mu_{(zj)}$ adalah nilai α_{cut} dari masing-masing parameter output. Hasil nilai z* yang didapatkan adalah 1.52640801.

Berdasarkan fungsi keanggotaan output pada persamaan (3-21) dan (3-22), dihitung nilai derajat keanggotaan (μ) untuk setiap jurusan IPA dan IPS. Dari proses ini didapatkan nilai derajat keanggotaan untuk IPA (μ_{ipa}) = 0.47359199 dan nilai derajat keanggotaan untuk IPS (μ_{ips}) = 0.52640801.

Karena $\mu_{ips} > \mu_{ipa}$, maka siswa dengan data tersebut direkomendasikan untuk masuk ke jurusan IPS dengan nilai kelayakan 0.52640801. Hasil ini sesuai dengan hasil sebenarnya yang ada pada data, dimana siswa tersebut masuk kedalam jurusan IPS.

Dengan proses yang sama dilakukan pula perhitungan menggunakan batas awal yang belum dioptimasi, pada perhitungan tersebut hasil dari sistem merekomendasikan siswa tersebut untuk masuk ke dalam jurusan IPA, hasil ini tidak sesuai dengan hasil sebenarnya.

3.5. Perancangan Pengujian dan Analisis

Pengujian dan analisis dilakukan untuk mengetahui parameter terbaik untuk proses optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization* dan bagaimana pengaruh fungsi keanggotaan yang telah dianggap optimal untuk proses sistem inferensi *fuzzy* pada rekomendasi penentuan jurusan siswa SMA. Skenario pengujian yang akan dilakukan dibagi menjadi dua tahap, yaitu pencarian parameter optimasi terbaik dan evaluasi hasil akurasi pengujian.

1. Pencarian parameter optimasi terbaik

   Pada skenario pengujian ini akan dilakukan percobaan dengan menggunakan beberapa jumlah data latih. Pengujian ini digunakan untuk mencari kombinasi parameter algoritma *Particle Swarm Optimization* yang menghasilkan fungsi keanggotaan optimal dengan nilai *cost* terkecil. Parameter optimasi yang akan dicari adalah c_1, c_2, inersia minimum (ω_{min}), inersia maksimum (ω_{max}), jumlah partikel, dan iterasi maksimum. Perancangan tabel pengujian untuk proses pencarian parameter terbaik dapat dilihat pada Tabel 3.18. untuk parameter c_1 dan c_2, Tabel 3.19. untuk parameter inersia minimum dan ineria maksimum, Tabel 3.20. untuk parameter jumlah partikel, dan Tabel 3.21. untuk parameter iterasi maksimum.

Tabel 3.18. Tabel pencarian parameter c_1 dan c_2

Jumlah partikel = *x*, ω_{min} =*wmin*, ω_{max}=*wmax*, iterasi maksimum= *maxIter*, *cost* minimum=0,01						
c_1	c_2	Ujicoba ke-	*Cost* minimum	*Cost* maksimum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
Konstanta 1	Konstanta 1	1				
		…				
		n				
	….	1				
		…				
		n				
	Konstanta m	1				
		…				
		n				
….	…	…				
Konstanta k	Konstanta 1	1				
		…				
		n				
		…				
	Konstanta m	1				
		…				
		n				

Hasil terbaik: Rata-rata *cost* minimum=… c_1=… c_2=…

Tabel 3.19. Tabel pencarian parameter inersia minimum (ω_{min}) dan inersia maksimum (ω_{max})

Jumlah partikel = *x*, c_1 =*c1*, c_2=*c2*, iterasi maksimum= *maxIter*, *cost* minimum=0,01						
ω_{min}	ω_{max}	Ujicoba ke-	*Cost* minimum	*Cost* maksimum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
Konstanta 1	Konstanta 1	1				
		…				
		n				
	….	1				
		…				
		n				
	Konstanta m	1				
		…				
		n				
….	…	…				
Konstanta k	Konstanta 1	1				
		…				
		n				
	….	…				

	Konstanta m	1				

Hasil terbaik: Rata-rata *cost* minimum=… ω_{min} =... ω_{max}=...

Tabel 3.20. Tabel pencarian parameter jumlah partikel

Jumlah partikel = *x*, c_1 =*c1*, c_2=*c2*, ω_{min} =*wmin*, ω_{max}=*wmax*, iterasi maksimum= *maxIter*, *cost* minimum=0,01					
Jumlah partikel	Ujicoba ke-	*Cost* minimum	*Cost* maksimum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
Konstanta 1	1				
	…				
	n				
….	1				
	…				
	n				
Konstanta m	1				
	…				
	n				

Hasil terbaik: Rata-rata *cost* minimum=… jumlah partikel=...

Tabel 3.21. Tabel pencarian parameter iterasi maksimum

Jumlah partikel = *x*, c_1 =*c1*, c_2=*c2*, ω_{min} =*wmin*, ω_{max}=*wmax*, jumlah partikel= *x*, *cost* minimum=0,01					
Iterasi maksimum	Ujicoba ke-	*Cost* minimum	*Cost* maksimum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
Konstanta 1	1				
	…				
	n				
….	1				
	…				
	n				
Konstanta m	1				

Hasil terbaik: Rata-rata *cost* minimum=… iterasi maksimum=...

2. Evaluasi akurasi hasil pengujian

Pada skenario pengujian ini akan dicari tingkat akurasi sistem inferensi *fuzzy* menggunakan batas-batas yang telah dianggap optimal terhadap data uji. Untuk menentukan batas-batas yang dianggap optimal digunakan

parameter yang telah didapatkan dari percobaan 1. Perancangan tabel pengujian untuk skenario ini ditunjukkan pada Tabel 3.19.

Tabel 3.19. Tabel akurasi hasil pengujian

Ujicoba ke-	Akurasi (%)
1.	
….	
n	

BAB IV
IMPLEMENTASI

4.1. Lingkungan Implementasi

Dalam menerapkan metode penelitian ke dalam sistem serta dalam pengembangan sistem, perlu adanya beberapa faktor yang perlu diperhatikan dalam menerapkan metode dan pengembangan sistem. Tujuannya yaitu untuk memenuhi kebutuhan dari sistem yang nantinya akan dikembangkan dan metode yang diimplementasikan. Beberapa aspek yang perlu diperhatikan yaitu dari segi perangkat keras (*hardware*) maupun perangkat lunak (*software*).

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem ini adalah sebuah *notebook* dengan spesifikasi sebagai berikut:

Processor Intel® Celeron® CPU B820 @ 1.70GHz

Memory 4.00GB RAM

Monitor

Keyboard

Mouse

4.1.2. Lingkungan Implementasi Perangkat Lunak

Pengembangan sistem dari penelitian ini juga membutuhkan beberapa perangkat lunak. Beberapa perangkat lunak yang digunakan sebagai berikut:

1. Sistem operasi yang digunakan Windows 7 Home Premium 64-bit
2. Aplikasi pembangun GUI dan kode menggunakan NetBeans IDE 6.9.1
3. Bahasa pemrograman yang dipakai yaitu bahasa pemrograman Java
4. Komponen Java yang digunakan yaitu JDK 5
5. Apache POI 3.9
6. Microsoft Excel 2007

4.2. Implementasi Program

Berdasarkan analisa dan perancangan sistem yang telah dijelaskan pada Bab 3, maka pada subbab ini akan dijelaskan implementasi kode untuk membangun aplikasi pada penelitian ini. Kelas-kelas yang digunakan terbagi dalam dua

package yaitu *package* Antarmuka yang berisi kelas untuk membuat antarmuka program, dan *package* Kode yang berisi kelas yang mengimplementasikan algoritma dan fungsi-fungsi lain untuk membangun program.

Package Kode terbagi atas 4 *package*, yaitu *package* Excel yang digunakan untuk mengakses data yang terdapat pada *file* Excel, *package* Tipe yang berisi kelas-kelas yang digunakan sebagai objek tipe data untuk membantu jalannya program, *package* PSO yang berisi kelas yang berhubungan dengan proses optimasi menggunakan algoritma *Particle Swarm Optimization* dan *package* Fuzzy yang berisi kelas-kelas yang berhubungan dengan proses sistem inferensi *fuzzy* Mamdani.

Kelas penyusun setiap *package* dapat dilihat pada Tabel 4.1., Tabel 4.2., Tabel 4.3., Tabel 4.4., dan Tabel 4.5. Tabel 4.1. menunjukkan kelas penyusun *package* Antarmuka, Tabel 4.2. menunjukkan kelas penyusun *package* Kode.Excel, Tabel 4.3. berisi kelas penyusun *package* Kode.Tipe, Tabel 4.4. menunjukkan kelas penyusun *package* Kode.PSO, dan Tabel 4.5. menunjukkan kelas penyusun *package* Kode.Fuzzy.

Tabel 4.1. Kelas-kelas pembentuk aplikasi dalam *package* Antarmuka

No	Nama Kelas	Deskripsi
1.	`TampilanOptimasi.class`	Merupakan kelas yang digunakan untuk membuat tampilan antarmuka untuk proses optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization.*
2.	`TampilanFIS.class`	Merupakan kelas yang digunakan untuk membuat tampilan untuk proses penentuan rekomendasi jurusan menggunakan sistem inferensi *fuzzy* Mamdani.
3.	`TampilanFK.class`	Kelas yang digunakan untuk menampilkan fungsi keanggotaan baik sebelum maupun setelah optimasi dalam bentuk tabel dan bentuk grafik.

Tabel 4.2. Kelas-kelas pembentuk aplikasi dalam *package* Kode.Excel

No	Nama Kelas	Deskripsi
1.	`Excel.class`	Merupakan kelas yang menjadi *interface* untuk kelas-kelas yang digunakan untuk mengakses *file* Excel.
2.	`BacaExcel.class`	Kelas yang menjadi *super class* dari kelas-kelas yang membaca *file* Excel.
3.	`BacaData.class`	Kelas yang digunakan untuk membaca data yang digunakan sebagai *input* yang tersimpan dalam *file* Excel.
4.	`BacaBatas.class`	Kelas yang digunakan untuk membaca batas-batas variabel linguistik yang tersimpan dalam *file* Excel.
5.	`BacaAturan.class`	Kelas yang digunakan untuk membaca aturan yang tersimpan dalam *file* Excel.
6.	`BacaParameter.class`	Kelas yang digunakan untuk membaca parameter PSO yang digunakan untuk optimasi dan tersimpan dalam *file* Excel.
7.	`UpdateExcel.class`	Kelas yang menjadi *super class* dari kelas-kelas yang memperbarui isi dari *file* Excel.
8.	`UpdateBatas.class`	Merupakan kelas yang digunakan untuk memperbarui batas yang dioptimasi yang disimpan dalam *file* Excel.
9.	`UpdateParameter.class`	Merupakan kelas yang digunakan untuk memperbarui parameter PSO yang disimpan dalam *file* Excel.

Tabel 4.3. Kelas-kelas pembentuk aplikasi dalam *package* Kode.Tipe

No	Nama Kelas	Deskripsi
1.	`ParameterInput.class`	Merupakan kelas untuk membentuk tipe data yang berisi parameter dari data yang diinputkan.
2.	`ParameterTrasformasi.class`	Merupakan kelas yang digunakan untuk memebentuk tipe data yang berisi parameter data yang telah ditransformasi.
3.	`BatasLinguistik.class`	Merupakan kelas yang digunakan untuk membentuk batas-batas linguistik.

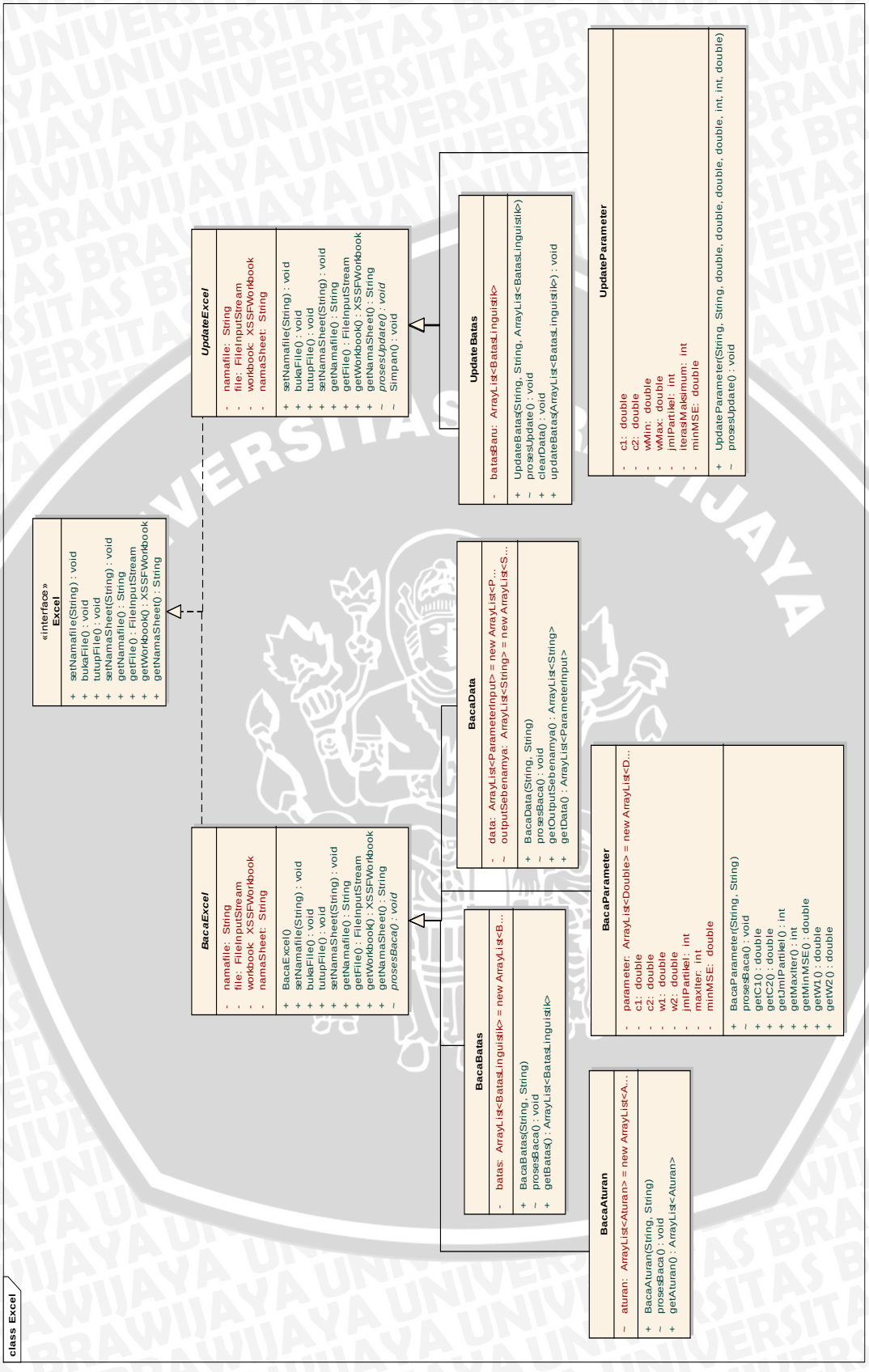
Tabel 4.4. Kelas-kelas pembentuk aplikasi dalam *package* Kode.PSO

No	Nama Kelas	Deskripsi
1.	`ProsesPSO.class`	Kelas yang digunakan untuk melakukan optimasi satu variabel linguistik menggunakan algoritma *Particle Swarm Optimization*
2.	`PSO.class`	Kelas yang digunakan untuk melakukan optimasi seluruh variabel linguistik dari semua fungsi keanggotaan.
3.	`Partikel.class`	Kelas yang digunakan untuk membentuk partikel yang digunakan dalam optimasi

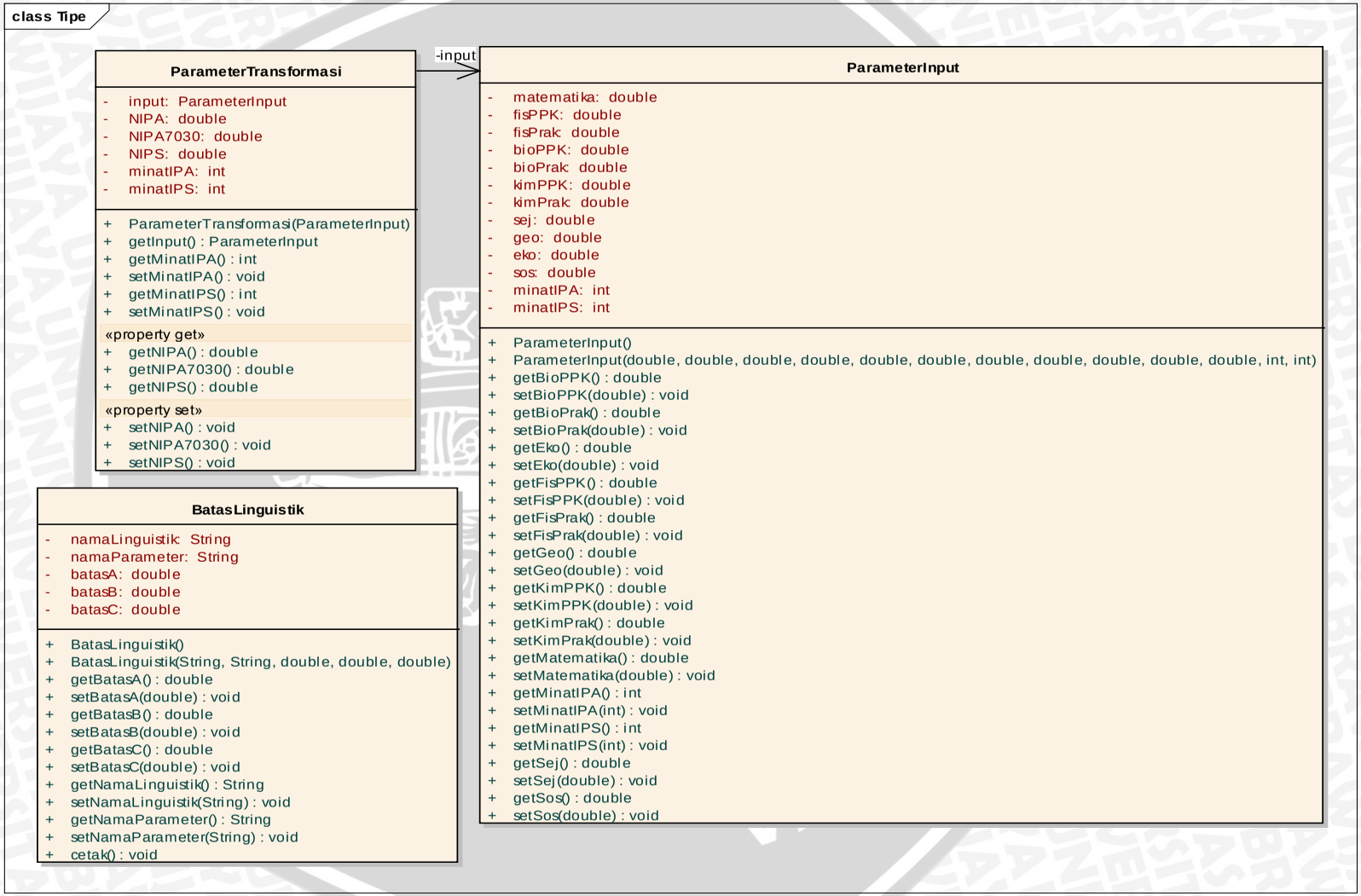
Tabel 4.5. Kelas-kelas pembentuk aplikasi dalam *package* Kode.Fuzzy

No	Nama Kelas	Deskripsi
1.	`Fuzzy.class`	Kelas yang digunakan untuk melakukan sistem inferensi *fuzzy* Mamdani.
2.	`Akurasi.class`	Kelas yang digunakan untuk menghitung tingkat akurasi rekomendasi penjurusan.
3.	`Aturan.class`	Kelas yang digunakan sebagai tipe data untuk aturan-aturan yang akan digunakan dalam sistem inferensi *fuzzy*.
4.	`BilanganFuzzy.class`	Kelas yang digunakan untuk membentuk tipe data bilangan *fuzzy*.

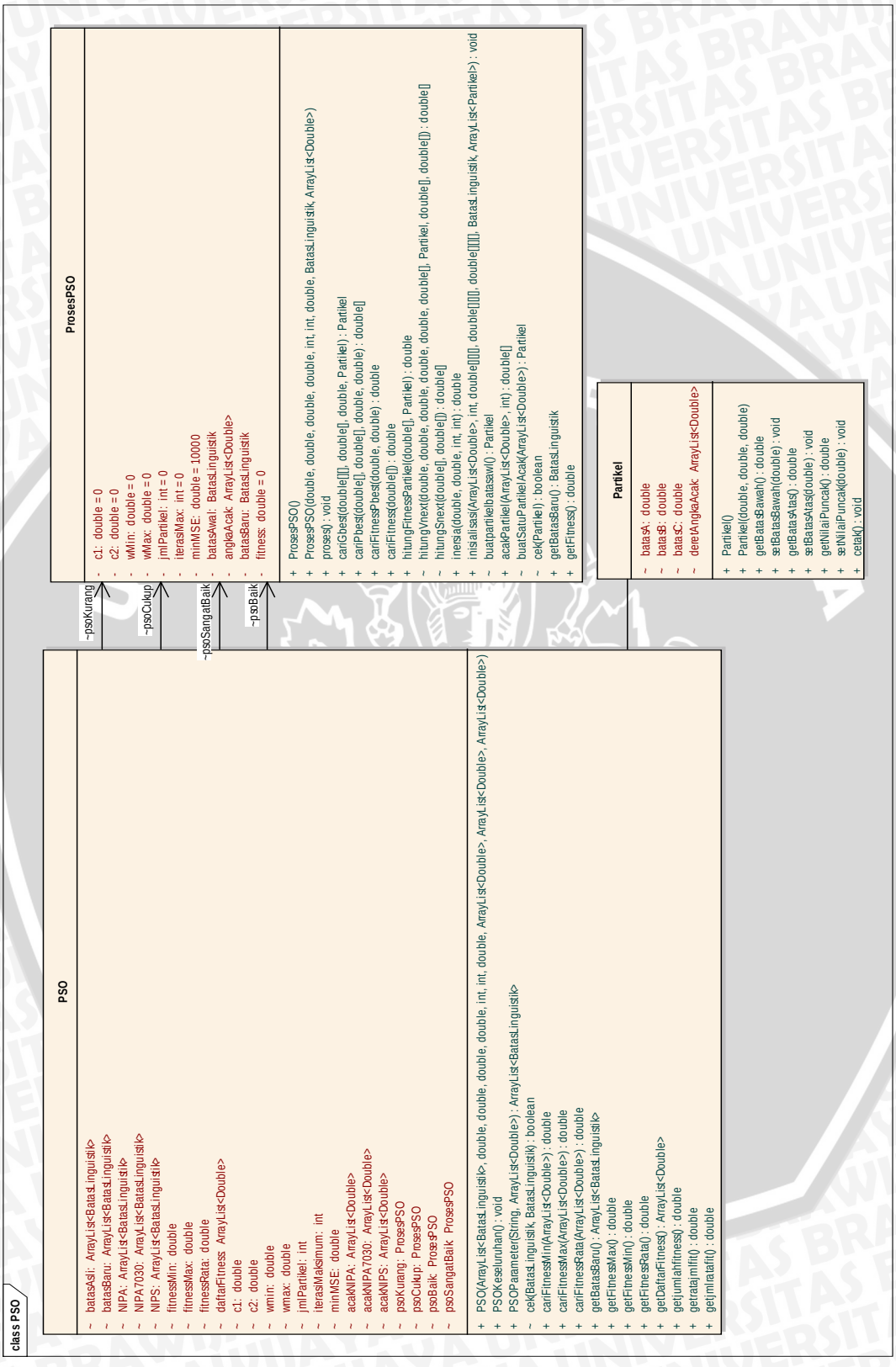
Dari kelas-kelas pembentuk aplikasi yang telah disebutkan pada Tabel 4.1. hingga Tabel 4.5. tidak semua kelas pembangun aplikasi tersebut akan dijelaskan pada subbab ini, hanya kelas-kelas yang berisi proses utama yang berkaitan dengan proses optimasi fungsi keanggotaan dan sistem inferensi *fuzzy* seperti yang telah dirancang pada Bab 3. Kelas-kelas yang akan dijelaskan yaitu: `ProsesPSO.java`, `PSO.java`, `Fuzzy.java`, dan `Akurasi.java`. Secara garis besar hubungan antar kelas yang terdapat pada *package* Kode dapat dilihat pada Gambar 4.1., Gambar 4.2., Gambar 4.3., dan Gambar 4.4.



Gambar 4.1. *Class* Diagram *Package* Kode.Excel



Gambar 4.2. *Class* Diagram *Package* Kode.Tipe



Gambar 4.3. *Class* Diagram *Package* Kode.PSO

Gambar 4.4. Class Diagram Package Kode.Fuzzy

4.2.1. Proses Optimasi Fungsi Keanggotaan Menggunakan algoritma Particle Swarm Optimization

Proses optimasi fungsi keanggotaan menggunakan dua kelas utama yaitu kelas `PSO.java` dan `ProsesPSO.java`. Kelas `PSO.java` merupakan kelas yang akan melakukan optimasi terhadap semua fungsi keanggotaan awal, sedangkan kelas `ProsesPSO.java` merupakan kelas yang berisi proses optimasi terhadap satu fungsi keanggotaan saja. *Method* yang menyusun kelas `PSO.java` dan `ProsesPSO.java` dapat dilihat pada Tabel 4.6. dan Tabel 4.7.

Tabel 4.6. *Method* penyusun `PSO.java`

No	Nama *Method*	Deskripsi
1.	`PSOkeseluruhan()`	Merupakan *method* untuk menjalankan proses optimasi seluruh fungsi keanggotaan awal.
2.	`PSOParameter(String namaParameter, ArrayList<Double> angkaAcak)`	*Method* untuk melakukan optimasi pada setiap fungsi keanggotaan parameter input.
3.	`cariFitnessMin(ArrayList<Double> fitness)`	*Method* untuk mencari nilai *cost* minimum dari semua proses yang telah dijalankan.
4.	`cariFitnessMax(ArrayList<Double> fitness)`	*Method* untuk mencari nilai *cost* maksimum dari semua proses yang telah dijalankan.
5.	`cariFitnessRata(ArrayList<Double> fitness)`	*Method* untuk mencari nilai rata-rata *cost* dari semua proses yang telah dijalankan.

Tabel 4.7. Method penyusun `ProsesPSO.java`

No	Nama *Method*	Deskripsi
1.	`proses ()`	Merupakan *method* untuk menjalankan proses optimasi menggunakan algoritma PSO.
2.	`cariGbest(double pbest[][],double fitnessPbest[], double fitness,Partikel gbest)`	*Method* untuk mencari nilai Gbest.
3.	`cariPbest(double pbestSebelumnya[],double posisi[],double fitnessPartikel, double fitnessPbest)`	*Method* untuk mencari nilai Pbest.
4.	`cariFitnessPbest(double fitnessSebelum, double fitnessSekarang)`	*Method* untuk menghitung nilai *cost* Pbest.
5.	`cariFitness(double fitnessPbest[])`	*Method* untuk menghitung nilai *cost* Gbest.
6.	`cariNilaiMinimum(double angka1, double angka2)`	*Method* untuk mencari nilai minimum dari dua angka yang digunakan dalam pencarian nilai *cost*.
7.	`hitungFitnessPartikel`	*Method* untuk menghitung nilai *cost*

	`(double posisiPartikel[],Partikel gbest)`	masing-masing partikel.
8.	`hitungVnext(double w, double c1, double c2, double r1, double r2,double pbest[],Partikel gbest, double posisi[],double vsebelum[])`	*Method* untuk menghitung nilai kecepatan baru.
9.	`hitungSnext(double v[],double posisisebelum[])`	*Method* untuk menghitung posisi baru.
10.	`inersia(double wMax, double wMin, int iterasiMax, int iterasi)`	*Method* untuk menghitung nilai inersia (ω).
11.	`inisialisasi(ArrayList<Double> angkaAcak, int jmlPartikel, double s[][][], double v[][][], double pbest[][][], BatasLinguistik batasAwal, ArrayList<Partikel> gbest)`	*Method* untuk melakukan inisialisasi awal.
12.	`acakPartikel(ArrayList<Double> angkaAcak, int jmlPartikel)`	*Method* untuk membuat partikel acak.

4.2.1.1. Proses Optimasi Seluruh Fungsi Keanggotaan Awal

Proses optimasi seluruh fungsi keanggotaan awal terdapat pada kelas `PSO.java`, dalam *method* `PSOkeseluruhan()` yang dapat dilihat pada Kode Program 4.1.

```
public void PSOKeseluruhan(){
    NIPA=PSOParameter("NIPA", acakNIPA);
    NIPA7030=PSOParameter("NIPA70/30", acakNIPA7030);
    NIPS=PSOParameter("NIPS", acakNIPS);
    batasBaru.addAll(NIPA);
    batasBaru.addAll(NIPA7030);
    batasBaru.addAll(NIPS);
    fitnessMin=cariFitnessMin(daftarFitness);
    fitnessMax=cariFitnessMax(daftarFitness);
    fitnessRata=cariFitnessRata(daftarFitness);
}
```

Kode Program 4.1. Proses Optimasi Seluruh Fungsi Keanggotaan Awal

4.2.1.2. Proses Optimasi Fungsi Keanggotaan Satu Parameter

Proses optimasi fungsi keanggotaan satu parameter ini terdapat pada kelas `PSO.java`, dalam *method* `PSOParameter()`. Proses ini merupakan proses yang

digunakan untuk mengoptimasi masing-masing parameter input yaitu NIPA, NIPA70/30, dan NIPS. Proses ini menggunakan kelas `ProsesPSO.java` untuk setiap variabel linguistik, dan mencari nilai *fitness* setiap proses. *Method* ini akan mengembalikan nilai berupa Arraylist batas-batas baru dari setiap linguistik yang ada pada parameter yang dioptimasi. Kode untuk proses ini terdapat pada Kode Program 4.2.

```
public       ArrayList<BatasLinguistik>       PSOParameter(String
namaParameter, ArrayList<Double> angkaAcak){

    ArrayList<BatasLinguistik>                    batasParameter=new
        ArrayList<BatasLinguistik>();
    BatasLinguistik kurang=new BatasLinguistik();
    BatasLinguistik cukup=new BatasLinguistik();
    BatasLinguistik baik=new BatasLinguistik();
    BatasLinguistik sangatBaik= new BatasLinguistik();
    BatasLinguistik kurangBaru;
    BatasLinguistik cukupBaru;
    BatasLinguistik baikBaru;
    BatasLinguistik sangatBaikBaru;
    double fitnessKurang;
    double fitnessCukup;
    double fitnessBaik;
    double fitnessSangatBaik;
        for (int i = 0; i < batasAsli.size(); i++) {

           if(batasAsli.get(i).getNamaParameter().equalsIgnoreCas
           e(namaParameter)&&
           batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase(
           "kurang")){
                kurang=batasAsli.get(i);
            }

           if(batasAsli.get(i).getNamaParameter().equalsIgnoreCas
           e(namaParameter)&&
           batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase(
           "cukup")){
                cukup=batasAsli.get(i);
            }

           if(batasAsli.get(i).getNamaParameter().equalsIgnoreCas
           e(namaParameter)&&
           batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase(
           "baik")){
                baik=batasAsli.get(i);
            }

           if(batasAsli.get(i).getNamaParameter().equalsIgnoreCas
           e(namaParameter)&&
           batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase(
           "sangat baik")){
                sangatBaik=batasAsli.get(i);
            }
```

```
        }
    ProsesPSO3 psoKurang=new ProsesPSO3(c1, c2, wmin, wmax,
        jmlPartikel, iterasiMaksimum, minMSE, kurang,
        angkaAcak);
    ProsesPSO3 psoCukup= new ProsesPSO3(c1, c2, wmin, wmax,
        jmlPartikel, iterasiMaksimum, minMSE, cukup, angkaAcak);
    ProsesPSO3 psoBaik= new ProsesPSO3(c1, c2, wmin, wmax,
        jmlPartikel, iterasiMaksimum, minMSE, baik, angkaAcak);
    ProsesPSO3 psoSangatBaik= new ProsesPSO3(c1, c2, wmin, wmax,
        jmlPartikel, iterasiMaksimum, minMSE, sangatBaik,
        angkaAcak);

    psoKurang.proses();
    psoCukup.proses();
    psoBaik.proses();
    psoSangatBaik.proses();
    kurangBaru=psoKurang.getBatasBaru();
    cukupBaru=psoCukup.getBatasBaru();
    baikBaru=psoBaik.getBatasBaru();
    sangatBaikBaru=psoSangatBaik.getBatasBaru();

    fitnessKurang=psoKurang.getFitness();
    fitnessCukup=psoCukup.getFitness();
    fitnessBaik=psoBaik.getFitness();
    fitnessSangatBaik=psoSangatBaik.getFitness();

    batasParameter.add(kurangBaru);
    batasParameter.add(cukupBaru);
    batasParameter.add(baikBaru);
    batasParameter.add(sangatBaikBaru);

    daftarFitness.add(fitnessKurang);
    daftarFitness.add(fitnessCukup);
    daftarFitness.add(fitnessBaik);
    daftarFitness.add(fitnessSangatBaik);

    boolean cek1,cek2,cek3;
    cek1=cek(kurangBaru, cukupBaru);
    cek2=cek(cukupBaru,baikBaru);
    cek3=cek(baikBaru,sangatBaikBaru);


    if(cek1==false&&cek2==false&&cek3==false)
        PSOParameter(namaParameter,angkaAcak);
        return batasParameter;

}
```

Kode Program 4.2. Proses Optimasi Fungsi Keanggotaan Satu Parameter

4.2.1.3. Proses Optimasi dengan Algoritma *Particle Swarm Optimization*

Proses yang terdapat pada method `prosespso()` ini merupakan proses utama optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm*

Optimization yang dijalankan untuk mengoptimasi satu variabel linguistik, sebagaimana yang tercantum pada Kode Program 4.3.

```
public void proses(){
    double s[][][]=new double[iterasiMax][jmlPartikel][3];
    double v[][][]=new double[iterasiMax][jmlPartikel][3];
    double pbest[][][]=new double[iterasiMax][jmlPartikel][3];
    ArrayList<Partikel> gbest=new ArrayList<Partikel>();
    double fitnesPartikel[][] = new double [iterasiMax]
    [jmlPartikel];
    double fitnessPbest[][]=new double[iterasiMax][jmlPartikel];
    ArrayList<Double> fitnessGbest=new ArrayList<Double>();
    double r1=0;
    double r2=0;
    double w=0;
    BatasLinguistik hasil=new BatasLinguistik();

    for (int k = 0; k < iterasiMax; k++) {
        w=inersia(wMax, wMin, iterasiMax, k);
        r1=Math.random();
        r2=Math.random();

        if(k==0){
            inisialisasi(angkaAcak, jmlPartikel, s, v, pbest,
            batasAwal, gbest);
            for (int i = 0; i < jmlPartikel; i++) {
              fitnesPartikel[k][i] = hitungFitnessPartikel
                  (s[k][i], gbest.get(k));
              fitnessPbest[k][i]=fitnesPartikel[k][i];
            }
        }
        if(k>0){
            for (int i = 0; i < jmlPartikel; i++) {
                fitnesPartikel[k][i] = hitungFitnessPartikel
                (s[k][i], gbest.get(k-1));
                pbest[k][i]=cariPbest(pbest[k-1][i], s[k][i],
                fitnesPartikel[k][i], fitnessPbest[k-1][i]);
                fitnessPbest[k][i] = cariFitnessPbest
                (fitnessPbest[k-1][i], fitnesPartikel[k][i]);
            }
            gbest.add(cariGbest(pbest[k], fitnessPbest[k],
            fitnessGbest.get(k-1), gbest.get(k-1)));
        }
        if(k<iterasiMax-1){
            for (int i = 0; i < jmlPartikel; i++) {
                v[k+1][i]=hitungVnext(w, c1, c2, r1, r2,
                  pbest[k][i], gbest.get(k), s[k][i], v[k][i]);
                s[k+1][i]=hitungSnext(v[k+1][i], s[k][i]);
            }
        }
        fitnessGbest.add(cariFitness(fitnessPbest[k]));
        if(fitnessGbest.get(k)<minMSE)
            break;
    }
    fitness=fitnessGbest.get(0);
    batasBaru=new BatasLinguistik(batasAwal.getNamaParameter(),
       batasAwal.getNamaLinguistik(),
```

```
            gbest.get(0).getBatasBawah(),
            gbest.get(0).getNilaiPuncak(),
            gbest.get(0).getBatasAtas());
        for (int i = 1; i < gbest.size(); i++) {
            if (fitnessGbest.get(i)<fitness){
                fitness=fitnessGbest.get(i);
                batasBaru=new
                  BatasLinguistik(batasAwal.getNamaParameter(),
                  batasAwal.getNamaLinguistik(),
                  gbest.get(i).getBatasBawah(),
                  gbest.get(i).getNilaiPuncak(),
                  gbest.get(i).getBatasAtas());
            }
        }
}
```

Kode Program 4.3. Proses Optimasi dengan PSO

4.2.1.4. Proses Penentuan Partikel Acak

Proses ini merupakan proses pembangkitan partikel yang akan digunakan dalam melakukan optimasi. Diawali dengan memilih beberapa bilangan acak dari parameter yang akan dioptimasi, lalu bilangan acak tersebut dibangun menjadi partikel dengan masing-masing partikel menggunakan 3 bilangan acak yang dijadikan sebagai batas a (batas bawah), b (nilai puncak), dan c (batas atas) dari masing-masing partikel seperti yang ditunjukkan oleh Kode Program 4.5. Proses tersebut akan diulang sejumlah partikel acak sebagaimana tercantum pada Kode Program 4.4.

```
public double[][] acakPartikel(ArrayList<Double> angkaAcak, int
jmlPartikel){
    double partikelAcak[][]=new double[jmlPartikel][3];
    Partikel partikel;
    for (int i = 0; i < jmlPartikel; i++) {
        partikel=buatSatuPartikelAcak(angkaAcak);
        partikelAcak[i][0]=partikel.getBatasBawah();
        partikelAcak[i][1]=partikel.getNilaiPuncak();
        partikelAcak[i][2]=partikel.getBatasAtas();
    }
    return partikelAcak;
}
```

Kode Program 4.4. Proses Pembangkitan Partikel Acak

```
Partikel buatSatuPartikelAcak(ArrayList<Double> angkaAcak){
    int indexrandom[]=new int[3];
    Partikel partikel;
    for (int i = 0; i < 3; i++) {
        indexrandom[i]=(int) (Math.random()*angkaAcak.size());
    }
    partikel=new
```

```
    Partikel(angkaAcak.get(indexrandom[0]),angkaAcak.get(indexran
    dom[1]), angkaAcak.get(indexrandom[2]));
       if(cek(partikel)==false)
           buatSatuPartikelAcak(angkaAcak);
    return partikel;
}
```

Kode Program 4.5. Proses Pembentukan Satu Partikel Acak

4.2.1.5. Proses Inisialisasi

Proses inisialisasi merupakan proses untuk melakukan inisialisasi awal dari nilai batas awal, partikel acak, posisi awal partikel, kecepatan awal partikel, Pbest awal ,dan Gbest awal. Proses ini terdapat pada *method* `inisialisasi()`. Kode Program yang digunakan dapat dilihat pada Kode Program 4.6.

```
public void inisialisasi(ArrayList<Double> angkaAcak, int
jmlPartikel, double s[][][], double v[][][], double pbest[][][],
BatasLinguistik batasAwal, ArrayList<Partikel> gbest){

    double partikelAcak[][]=new double[jmlPartikel][3];
    partikelAcak=acakPartikel(angkaAcak, jmlPartikel);
    Partikel gb=new Partikel();

    for (int i = 0; i < jmlPartikel; i++) {
        for (int j = 0; j < 3; j++) {
            s[0][i][j]=partikelAcak[i][j];
            v[0][i][j]=0;
            pbest[0][i][j]=s[0][i][j];
        }
    }

    gb.setBatasBawah(batasAwal.getBatasA());
    gb.setNilaiPuncak(batasAwal.getBatasB());
    gb.setBatasAtas(batasAwal.getBatasC());
    gbest.add(0,gb);
}
```

Kode Program 4.6. Proses Inisialisasi

4.2.1.6. Proses Perhitungan berat inersia (ω)

Proses perhitungan berat inersia (ω) merupakan proses yang digunakan untuk menghitung nilai inersia dari masing-masing iterasi, dimana nantinya nilai inersia ini akan digunakan untuk menghitung kecepatan dari masing-masing partikel. Kode program untuk proses perhitungan ω ditunjukkan pada Kode Program 4.7.

```
public double inersia(double wMax, double wMin, int iterasiMax,
int iterasi){
    double w;
    w=wMax-(((wMax-wMin)/iterasiMax)*iterasi);
    return w;
}
```

Kode Program 4.7. Proses Perhitungan ω

4.2.1.7. Proses Proses Hitung *Cost* Partikel

Proses hitung *cost* partikel seperti yang tercantum pada Kode Program 4.8.. digunakan untuk menghitung nilai *cost* dari masing-masing partikel pada setiap iterasi. Nilai yang dihasilkan dari *method* `hitungFitnessPartikel()` dijadikan sebagai nilai evaluasi dari masing-masing partikel.

```
public          double          hitungFitnessPartikel(double
    posisiPartikel[],Partikel gbest){
    double ytarget[]={0,1,0};
    double myu[]=new double[3];
    double pembilang=0;
    double penyebut=0;
    double mse;
        for (int i = 0; i < 3; i++) {
            if(i==0||i==1)
                myu[i]=(posisiPartikel[i]-
    gbest.getBatasBawah())/(gbest.getNilaiPuncak()-
    gbest.getBatasBawah());
            else
                myu[i]=(gbest.getBatasAtas()-
    posisiPartikel[i])/(gbest.getBatasAtas()-
    gbest.getNilaiPuncak());
            pembilang=pembilang+(Math.pow((ytarget[i]-myu[i]),
    2));
            penyebut=penyebut+Math.pow(ytarget[i], 2);
        }
        mse=(double)(pembilang/penyebut);
        return mse;
}
```

Kode Program 4.8. Proses Perhitungan Nilai *Cost* Partikel

4.2.1.8. Proses Cari *Cost* Pbest

Proses ini merupakan proses untuk mencari nilai minimum *cost* dari masing-masing partikel, nilai yang dihasilkan akan dipilih menjadi nilai *cost* dari masing-masing Pbest yang terpilih. Kode program yang digunakan untuk proses ini dapat dilihat pada Kode Program 4.9.

```
public double cariFitnessPbest(double fitnessSebelum, double
fitnessSekarang){
    double fit=0;
    fit=Math.min(fitnessSebelum, fitnessSekarang);
    return fit;
}
```

Kode Program 4.9. Proses Pencarian Nilai *Cost* Pbest

4.2.1.9. Proses Cari *Cost* Gbest

Proses ini merupakan proses untuk mencari nilai minimum *cost* dari masing-masing iterasi, nilai yang dihasilkan akan dipilih menjadi nilai *cost* dari masing-masing Gbest yang terpilih dari setiap iterasi. Kode program yang digunakan untuk proses ini dapat dilihat pada Kode Program 4.10.

```
public double cariFitness(double fitnessPbest[]){
    double fit1iter;
    fit1iter=fitnessPbest[0];
    for(int a=1;a<fitnessPbest.length;a++){
            fit1iter=Math.min(fit1iter, fitnessPbest[a]);
        }
    return fit1iter;
}
```

Kode Program 4.10. Proses Pencarian Nilai *Cost* Gbest

4.2.1.10. Proses Pencarian Pbest

Proses pencarian Pbest ini dilakukan untuk mencari posisi terbaik dari setiap partikel, dimana yang menjadi pembanding adalah nilai *cost* Pbest yang dihasilkan dari proses yang tercantum pada Kode Program 4.9. Kode program untuk proses pencarian Pbest dapat dilihat pada Kode Program 4.11.

```
public    double[]    cariPbest(double    pbestSebelumnya[],double
    posisi[],double fitnessPartikel, double fitnessPbest){
    double Pbest[]=new double [3];
    for (int i = 0; i < 3; i++) {
        if(fitnessPartikel<fitnessPbest)
            Pbest[i]=posisi[i];
        else
            Pbest[i]=pbestSebelumnya[i];
        }
    return Pbest;
}
```

Kode Program 4.11. Proses Pencarian Pbest

4.2.1.11. Proses Pencarian Gbest

Proses ini merupakan proses untuk mencari posisi terbaik dari semua partikel di setiap iterasi dengan nilai *cost* yang diperoleh dari Kode Program 4.10. Nilai Gbest yang dihasilkan masing-masing iterasi akan digunakan sebagai acuan batas awal untuk iterasi berikutnya. Kode program yang digunakan ditunjukkan pada Kode Program 4.12.

```
public Partikel cariGbest(double pbest[][],double fitnessPbest[],
    double fitness,Partikel gbest){
double gbst[]=new double[3];
double fititer=cariFitness(fitnessPbest);
Partikel partikel=new Partikel();
    for(int i=0;i<pbest.length;i++){
            if(fititer<fitness){
                if(fititer==fitnessPbest[i]){
                    partikel.setBatasBawah(pbest[i][0]);
                    partikel.setNilaiPuncak(pbest[i][1]);
                    partikel.setBatasAtas(pbest[i][2]);
                }
            }
            else{
                partikel=gbest;
            }
    }
return partikel;
}
```

Kode Program 4.12. Proses Pencarian Gbest

4.2.1.12. Proses Perhitungan Kecepatan Baru

Proses perhitungan kecepatan baru yang tercantum pada *method* `hitungVnext` ini merupakan proses untuk menghitung nilai kecepatan (*v*) pada iterasi berikutnya. Kode Program untuk proses ini tercantum pada Kode Program 4.13.

```
double[] hitungVnext(double w, double c1, double c2, double r1,
    double    r2,double    pbest[],Partikel    gbest,    double
    posisi[],double vsebelum[]){
    double vPar[]=new double[3];
           vPar[0]=(vsebelum[0]*w)+(c1*r1*(pbest[0]-
    posisi[0]))+(c2*r2*(gbest.getBatasBawah()-posisi[0]));
           vPar[1]=(vsebelum[1]*w)+(c1*r1*(pbest[1]-
    posisi[1]))+(c2*r2*(gbest.getNilaiPuncak()-posisi[1]));
           vPar[2]=(vsebelum[2]*w)+(c1*r1*(pbest[2]-
    posisi[2]))+(c2*r2*(gbest.getBatasAtas()-posisi[2]));
    return vPar;
}
```

Kode Program 4.13. Proses Perhitungan Kecepatan Baru ($v_{(k+1)}$)

4.2.1.13. Proses Perhitungan Posisi Baru

Proses ini adalah proses untuk mencari nilai posisi terbaru masing-masing partikel untuk iterasi berikutnya. Kode program untuk proses ini dapat dilihat pada Kode Program 4.14.

```
double[] hitungSnext(double v[],double posisisebelum[]){
    double posisi[]=new double [3];
    for (int i = 0; i < 3; i++) {
        posisi[i]=v[i]+posisisebelum[i];
    }
    return posisi;
}
```

Kode Program 4.14. Proses Perhitungan Posisi Baru($s_{(k+1)}$)

4.2.2. Sistem Inferensi *Fuzzy* Mamdani

Kelas utama untuk menjalankan sistem inferensi *fuzzy* Mamdani ini adalah `Fuzzy.java` dan `Akurasi.java` dengan *method* penyusun yang tercantum pada Tabel 4.8. dan 4.9.

Tabel 4.8. *Method* penyusun `Fuzzy.java`

No	Nama *Method*	Deskripsi
1.	`prosesFuzzy()`	Merupakan *method* untuk menjalankan proses sistem inferensi *fuzzy* secara keseluruhan.
2.	`fuzzifikasi()`	*Method* untuk proses fuzzifikasi.
3.	`pemilihanAturan()`	*Method* untuk melakukan pemilihan aturan yang akan digunakan.
4.	`implikasiMin()`	*Method* untuk menghitung nilai implikasi dengan fungsi Min.
5.	`komposisiAturanMax()`	*Method* untuk menghitung nilai komposisi dengan metode Max.
6.	`defuzzifikasi()`	*Method* untuk proses defuzzifikasi.

Tabel 4.9. *Method* penyusun `Akurasi.java`

No	Nama *Method*	Deskripsi
1.	`tingkatAkurasi()`	Merupakan *method* untuk mencari tingkat akurasi.
2.	`persenAkurasi()`	*Method* untuk mencari prosentase tingkat akurasi.

4.2.2.1. Proses Fuzzifikasi

Sebagaimana yang telah dijelaskan pada Bab 3, proses fuzzifikasi merupakan proses konversi dari angka dalam bilangan *crisps* menjadi bilangan *fuzzy*. Kode program untuk proses ini dapat dilihat pada Kode Program 4.15 dan 4.16. pada Kode Program dimana nantinya *method* ini akan mengembalikan nilai dalam bentuk bilangan *fuzzy* yang merupakan *instance* dari kelas `BilanganFuzzy` dan terdiri atas kategori linguistic dan nilai derajat keanggotaan.

```
public void fuzzifikasi(){
    BilanganFuzzy bil=new BilanganFuzzy();
    for (int i = 0; i < batas.size(); i++) {

        if(batas.get(i).getNamaParameter().equalsIgnoreCase
        ("NIPA")){
            bil=prosesFuzzifikasi(NIPA, batas.get(i));
            if(bil.getDerajatKeanggotaan()!=0)
                    hasilFuzzifikasiNIPA.add(bil);
        }
            else if (batas.get(i).getNamaParameter().
                equalsIgnoreCase ("NIPA70/30")){
            bil=prosesFuzzifikasi(NIPA7030, batas.get(i));
            if(bil.getDerajatKeanggotaan()!=0)
                hasilFuzzifikasiNIPA7030.add(bil);
        }
        else{
            bil=prosesFuzzifikasi(NIPS, batas.get(i));
            if(bil.getDerajatKeanggotaan()!=0)
                hasilFuzzifikasiNIPS.add(bil);
        }
        if(bil.getDerajatKeanggotaan()!=0)
            hasilFuzzifikasi.add(bil);
    }

    if(minatIPA==1&&minatIPS==2){
        bil=new BilanganFuzzy("MinatIPA", "tinggi", 1);
        hasilFuzzifikasi.add(bil);
        hasilFuzzifikasiMinatIPA=bil;
        bil=new BilanganFuzzy("MinatIPS", "rendah", 1);
        hasilFuzzifikasi.add(bil);
        hasilFuzzifikasiMinatIPS=bil;
    }

    else{
         bil=new BilanganFuzzy("MinatIPA", "rendah", 1);
```

```
            hasilFuzzifikasi.add(bil);
            hasilFuzzifikasiMinatIPA=bil;
            bil=new BilanganFuzzy("MinatIPS", "tinggi", 1);
            hasilFuzzifikasi.add(bil);
            hasilFuzzifikasiMinatIPA=bil;
        }
}
```

Kode Program 4.15. Fuzzifikasi

```
public     BilanganFuzzy     prosesFuzzifikasi(double     angka,
BatasLinguistik linguistik){
    BilanganFuzzy bilFuzzy;
    double a,b,c,myu;
    a=linguistik.getBatasA();
    b=linguistik.getBatasB();
    c=linguistik.getBatasC();
    if (angka>=a&&angka<=b)
        myu=(double)((angka-a)/(b-a));
    else if(angka>b&&angka<=c)
        myu=(double)((c-angka)/(c-b));
    else
        myu=0;
    bilFuzzy=new     BilanganFuzzy(linguistik.getNamaParameter(),
    linguistik.getNamaLinguistik(), myu);

    return bilFuzzy;
}
```

Kode Program 4.16. Proses Fuzzifikasi

4.2.2.2. Proses Pemilihan Aturan

Proses ini merupakan proses untuk memilih aturan-aturan yang akan dipakai dalam sistem inferensi *fuzzy*. Aturan yang dipilih akan disesuaikan dengan kategori masing-masing parameter yang diperoleh dari proses fuzzifikasi. Proses ini ditunjukkan oleh Kode Program 4.17.

```
public void pemilihanAturan(){
    ArrayList<Aturan> semuaAturan=new ArrayList<Aturan>();
    BacaAturan aturan=new BacaAturan("data.xlsx", "aturan");
    semuaAturan=aturan.getAturan();
    ArrayList<Aturan> tempAtur=new ArrayList<Aturan>();

    for (int i = 0; i < semuaAturan.size(); i++) {
        for (int j = 0; j < hasilFuzzifikasiNIPA.size(); j++) {
            if(hasilFuzzifikasiNIPA.get(j).getKategoriLinguistik
            ().equalsIgnoreCase(semuaAturan.get(i).getKategoriNI
            PA()))
                tempAtur.add(semuaAturan.get(i));
        }
    }

    for (int i = 0; i < tempAtur.size(); i++) {
        if(!tempAtur.get(i).getMinatIPA().equalsIgnoreCase(hasi
```

```
            lFuzzifikasiMinatIPA.getKategoriLinguistik()))
                tempAtur.remove(i);
        }

        ArrayList<Aturan> tempAtur2=new ArrayList<Aturan>();

        for (int i = 0; i < tempAtur.size(); i++) {
            for (int j = 0; j < hasilFuzzifikasiNIPA7030.size();
             j++) {
                    if (tempAtur.get(i).getKategoriNIPA7030().
                equalsIgnoreCase(hasilFuzzifikasiNIPA7030.get(j).get
                            KategoriLinguistik())) {
                    tempAtur2.add(tempAtur.get(i));
                }
            }
        }
        for (int i = 0; i < tempAtur2.size(); i++) {
            for (int j = 0; j < hasilFuzzifikasiNIPS.size(); j++) {
                    if (tempAtur2.get(i).getKategoriNIPS().
                equalsIgnoreCase(hasilFuzzifikasiNIPS.get(j).getKate
                            goriLinguistik())) {
                    aturanDipakai.add(tempAtur2.get(i));
                }
            }
        }
}
```

Kode Program 4.17. Proses Pemilihan Aturan

4.2.2.3. Proses Implikasi

Proses implikasi yang digunakan untuk sistem inferensi *fuzzy* yang diterapkan pada penelitian ini menggunakan metode Min. *Method* ini akan mengembalikan nilai dalam bentuk `Arraylist` yang bertipe `BilanganFuzzy`. Kode Program untuk proses implikasi dapat dilihat pada Kode Program 4.18.

```
public void implikasiMin(){
   BilanganFuzzy hasil;
   double min=0;

   for (int i = 0; i < aturanDipakai.size(); i++) {
     for (int j = 0; j < hasilFuzzifikasiNIPA.size(); j++) {

       if(aturanDipakai.get(i).getKategoriNIPA().equalsIgnoreCase
       (hasilFuzzifikasiNIPA.get(j).getKategoriLinguistik())){

          min=hasilFuzzifikasiNIPA.get(j).
          getDerajatKeanggotaan();

          for (int k = 0; k < hasilFuzzifikasiNIPA7030.size();
          k++) {
            if(aturanDipakai.get(i).getKategoriNIPA7030().equals
            IgnoreCase(hasilFuzzifikasiNIPA7030.get(k).getKatego
            riLinguistik())){
```

```
                if(hasilFuzzifikasiNIPA7030.get(k).getDerajatKea
                nggotaan()<min){

                  min=hasilFuzzifikasiNIPA7030.get(k).getDerajat
                  Keanggotaan();

                  for     (int     l     =     0;     l     <
                  hasilFuzzifikasiNIPS.size(); l++) {

                    if(aturanDipakai.get(i).getKategoriNIPS().e
                    qualsIgnoreCase(hasilFuzzifikasiNIPS.get(l)
                    .getKategoriLinguistik())){

                       if(hasilFuzzifikasiNIPS.get(l).getDeraja
                       tKeanggotaan()<min){

                          min=hasilFuzzifikasiNIPS.get(l).getD
                          erajatKeanggotaan();
                                }
                            }
                          }
                       }
                    }
                 }
              }
           }
           hasil=new                          BilanganFuzzy("jurusan",
          aturanDipakai.get(i).getJurusan(), min);
           hasilImplikasi.add(hasil);
       }
}
```

Kode Program 4.18. Proses Implikasi

4.2.2.4. Proses Komposisi

Proses Komposisi yang digunakan pada penelitian ini menggunakan metode Max. Kode program yang digunakan dapat dilihat pada Kode Program 4.19. *Method* `komposisiAturanMax` akan mengembalikan nilai berbentuk array bertipe `BilanganFuzzy`.

```
public void komposisiAturanMax(){
    ArrayList<BilanganFuzzy> ipa=new ArrayList<BilanganFuzzy>();
    ArrayList<BilanganFuzzy> ips=new ArrayList<BilanganFuzzy>();
    BilanganFuzzy komposisiIPA=new BilanganFuzzy();
    BilanganFuzzy komposisiIPS=new BilanganFuzzy();

    for (int i = 0; i < hasilImplikasi.size(); i++) {
        if       (hasilImplikasi.get(i).getKategoriLinguistik().
         equalsIgnoreCase("ipa")) {
            ipa.add(hasilImplikasi.get(i));
        }
         else
```

```
            ips.add(hasilImplikasi.get(i));
        }

        if(ipa.size()<1)
            ipa.add(new BilanganFuzzy("jurusan", "ipa", 0));
        if(ips.size()<1)
            ips.add(new BilanganFuzzy("jurusan", "ips", 0));

        double maxIPA=ipa.get(0).getDerajatKeanggotaan();
        double maxIPS=ips.get(0).getDerajatKeanggotaan();

        for (int i = 1; i < ipa.size(); i++) {
            if(ipa.get(i).getDerajatKeanggotaan()>maxIPA)
                maxIPA=ipa.get(i).getDerajatKeanggotaan();
            }

        for (int i = 1; i < ips.size(); i++) {
            if(ips.get(i).getDerajatKeanggotaan()>maxIPA)
                maxIPS=ips.get(i).getDerajatKeanggotaan();
            }

        komposisiIPA=new BilanganFuzzy("jurusan", "ipa", maxIPA);
        komposisiIPS=new BilanganFuzzy("jurusan", "ips", maxIPS);
        hasilKomposisi[0]=komposisiIPA;
        hasilKomposisi[1]=komposisiIPS;
    }
```

Kode Program 4.19. Proses Komposisi

4.2.2.5. Proses Defuzzifikasi

Proses deffuzzifikasi adalah proses untuk mengetahui hasil akhir yang direkomendasikan oleh sistem. *Method* ini masih tetap mengembalikan nilai dalam bentuk BilanganFuzzy, karena nilai kelayakan masih dibutuhkan untuk diketahui. Rekomendasi jurusan dan nilai kelayakannya akan ditampilkan langsung pada kelas yang menggunakan proses ini. Proses defuzzifikasi dapat dilihat pada Kode Program 4.20.

```
public void defuzzifikasi(){
    BatasLinguistik   outputIPA=new   BatasLinguistik("jurusan",
    "ipa", 0, 1, 2);
    BatasLinguistik   outputIPS=new   BatasLinguistik("jurusan",
    "ips", 1, 2, 3);
    double titikpotong[]=new double[4];
    double z=0;

    titikpotong[0]=(hasilKomposisi[0].getDerajatKeanggotaan()*(o
    utputIPA.getBatasB()-
    outputIPA.getBatasA()))+outputIPA.getBatasA();

    if(hasilKomposisi[0].getDerajatKeanggotaan()>hasilKomposisi[
    1].getDerajatKeanggotaan()){
        titikpotong[1]=outputIPA.getBatasC()-(hasilKomposisi[0].
```

```
    getDerajatKeanggotaan()*(outputIPA.getBatasC()-
    outputIPA.getBatasB()));
     titikpotong[2]=outputIPA.getBatasC()-
    (hasilKomposisi[1].getDerajatKeanggotaan()*(outputIPA.ge
    tBatasC()-outputIPA.getBatasB()));
}
else     if     (hasilKomposisi[1].getDerajatKeanggotaan()>
hasilKomposisi[0].getDerajatKeanggotaan()){
    titikpotong[1]=hasilKomposisi[0].getDerajatKeanggotaan()
    *(outputIPS.getBatasB()-outputIPS.getBatasA())+
    outputIPS.getBatasA();
    titikpotong[2]=hasilKomposisi[1].getDerajatKeanggotaan()
    *(outputIPS.getBatasB()-outputIPS.getBatasA())+
    outputIPS.getBatasA();
}

titikpotong[3]=outputIPS.getBatasC()-(hasilKomposisi[1].
getDerajatKeanggotaan()*(outputIPS.getBatasC()-
outputIPS.getBatasB()));

z=((hasilKomposisi[0].getDerajatKeanggotaan()*titikpotong[0]
)+(hasilKomposisi[0].getDerajatKeanggotaan()*titikpotong[1])
+(hasilKomposisi[1].getDerajatKeanggotaan()*titikpotong[2])
+(hasilKomposisi[1].getDerajatKeanggotaan()*titikpotong[3]))
/((2*hasilKomposisi[0].getDerajatKeanggotaan())
+(2*hasilKomposisi[1].getDerajatKeanggotaan()));

BilanganFuzzy hasilIPA=prosesFuzzifikasi(z, outputIPA);
BilanganFuzzy hasilIPS=prosesFuzzifikasi(z, outputIPS);

if(hasilIPA.getDerajatKeanggotaan()>hasilIPS.getDerajatKeang
gotaan())
    hasilDefuzzifikasi=hasilIPA;
else          if(hasilIPA.getDerajatKeanggotaan()==hasilIPS.
getDerajatKeanggotaan()&&minatIPA==1){
    hasilDefuzzifikasi=hasilIPA;
    }
else
    hasilDefuzzifikasi=hasilIPS;
}
```

Kode Program 4.20. Proses Defuzzifikasi

4.2.2.6. Proses Perhitungan Tingkat Akurasi

Proses perhitungan tingkat akurasi terdapat pada kelas `Akurasi.java`. Proses ini dilakukan dengan mencari nilai yang sama dari output yang dihasilkan sistem dengan nilai sebenarnya. Kode program untuk proses ini dapat dilihat pada Kode Program 4.21.

```
void tingkatAkurasi(){
    double sama=0;
    for (int i = 0; i < hasilAsli.size(); i++) {
      if(hasilSistem.get(i).equalsIgnoreCase(hasilAsli.get(i))){
            sama++;
        }
    }
    tingkatAkurasi=(double)(sama/hasilAsli.size());
}
```

Kode Program 4.21. Proses Perhitungan Tingkat Akurasi

4.2.2.7. Proses Perhitungan Persen Akurasi

Proses perhitungan persen akurasi merupakan perhitungan tingkat akurasi dalam bentuk prosentase. Kode Program yang menunjukkan proses ini dapat dilihat pada Kode Program 4.22.

```
void persenAkurasi(){
    persenAkurasi=getTingkatAkurasi()*100;
}
```

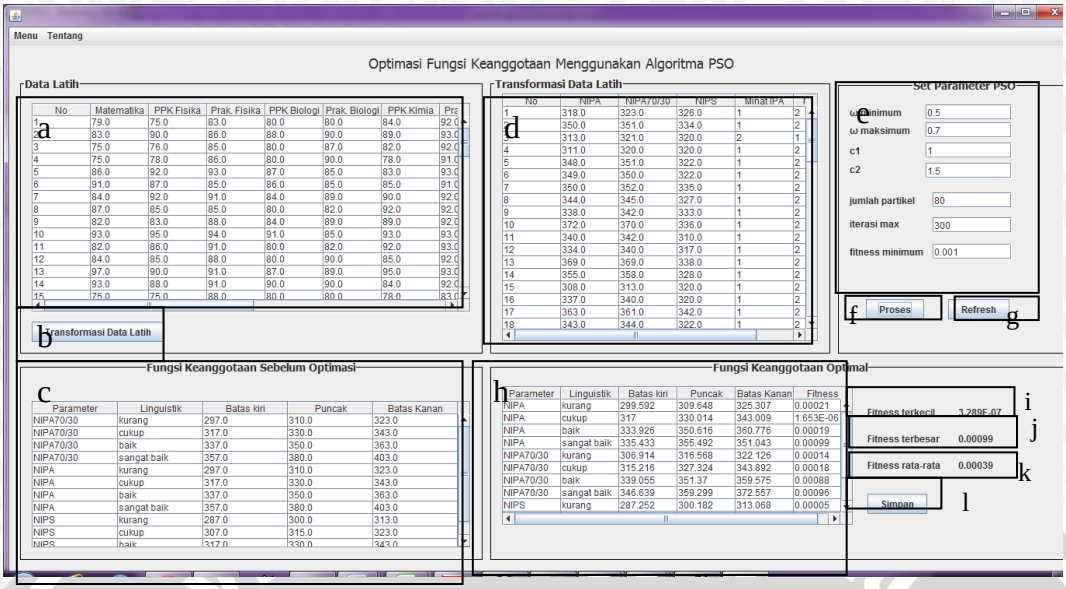
Kode Program 4.22. Proses Perhitungan Akurasi

4.2. Implementasi Antarmuka

Sebagaimana yang telah dirancang sebelumnya pada Bab 3, terdapat dua tampilan utama antarmuka dalam penelitian ini, yaitu antarmuka proses optimasi fungsi keanggotaan dan antarmuka pengujian menggunakan sistem inferensi *fuzzy*.

4.2.1. Antarmuka Proses Optimasi Fungsi Keanggotaan

Antarmuka ini digunakan untuk menampilkan antarmuka proses optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization*, yang dapat dilihat pada Gambar 4.5.



Gambar 4.5. Antarmuka Proses Optimasi Fungsi Keanggotaan

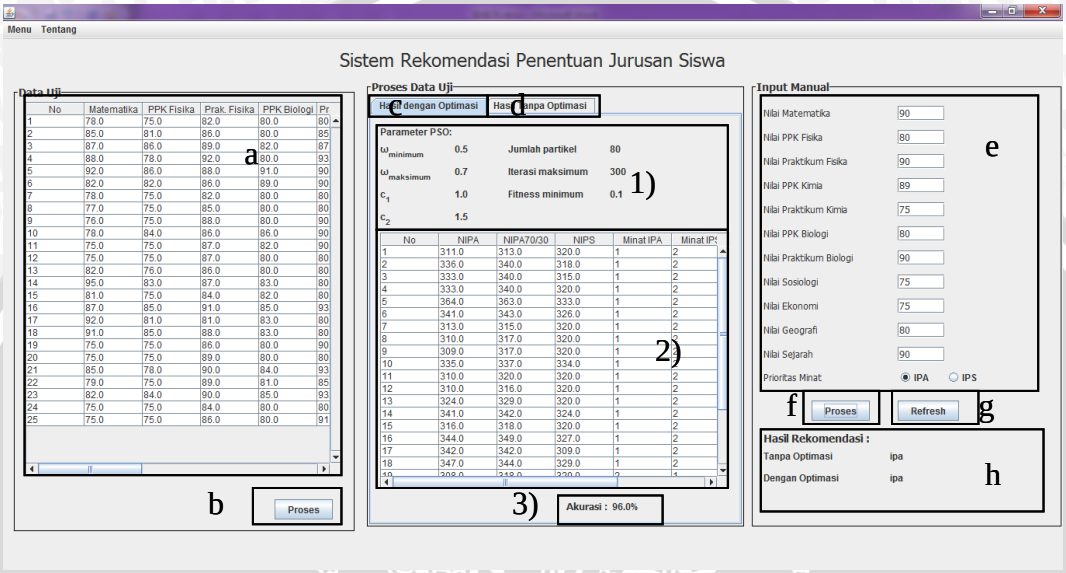
Bagian dari antarmuka tersebut antara lain :

a. Tabel untuk menampilkan data latih
b. Tombol yang digunakan untuk melakukan transformasi data
c. Tabel yang berisi fungsi keanggotaan awal sebelum dioptimasi
d. Tabel yang menampilkan data latih yang telah ditransformasi
e. *Field* untuk menentukan parameter PSO, yang terdiri atas ω (inersia) maksimum, ω (inersia) minimum, iterasi maksimum, konstanta c_1 , c_2, jumlah partikel, dan *cost* minimum
f. Tombol untuk melakukan proses optimasi
g. Tombol *refresh*, untuk mengosongkan kembali *field* parameter yang dijelaskan pada poin e
h. Tabel yang menunjukkan batas-batas fungsi keanggotaan baru yang merupakan hasil optimasi fungsi keanggotaan awal
i. *Text* yang menampilkan nilai *fitness/cost* terkecil dari satu kali proses optimasi seluruh fungsi keanggotaan
j. *Text* yang menampilkan nilai *fitness/cost* terbesar dari satu kali proses optimasi seluruh fungsi keanggotaan
k. *Text* yang menampilkan nilai *fitness/cost* rata-rata dari satu kali proses optimasi seluruh fungsi keanggotaan

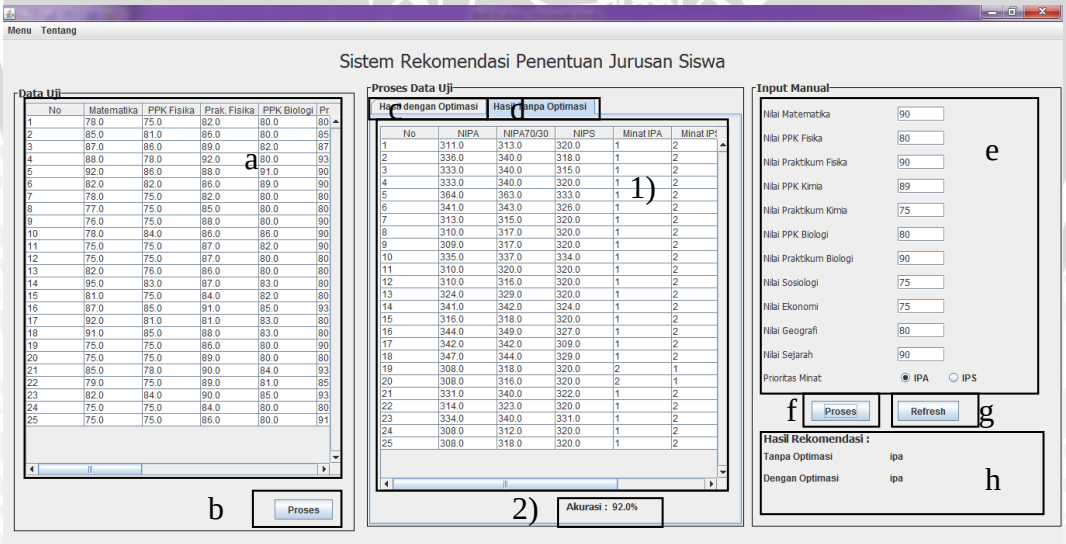
l. Tombol untuk menyimpan batas-batas fungsi keanggotaan baru.

4.2.2. Antarmuka Pengujian Menggunakan Sistem Inferensi *Fuzzy*

Antarmuka ini digunakan untuk melakukan pengujian menggunakan sistem inferensi *fuzzy* Mamdani. Adapun gambaran dari antarmuka ini dapat dilihat pada Gambar 4.6.a., dan Gambar 4.6.b.



Gambar 4.6.a. Antarmuka Proses Pengujian Menggunakan Sistem Inferensi Fuzzy



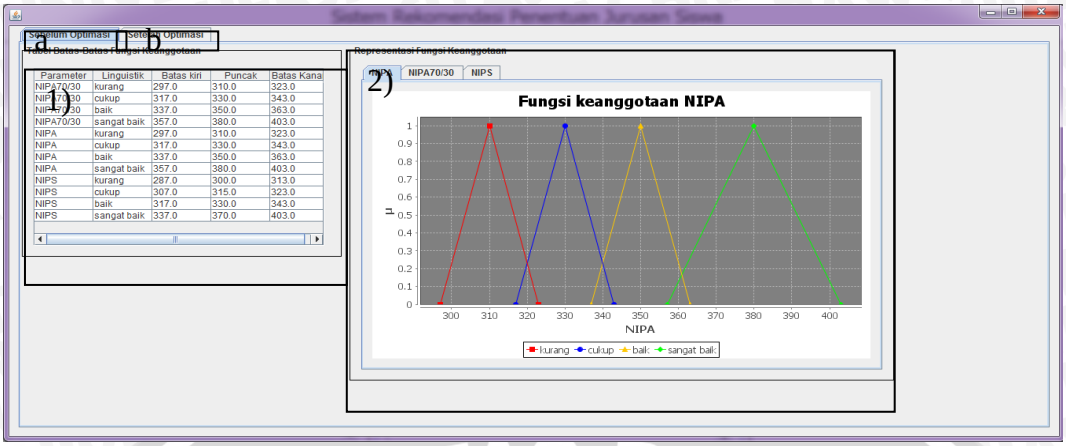
Gambar 4.6.b. Antarmuka Proses Pengujian Menggunakan Sistem Inferensi *Fuzzy*

Antarmuka ini terdiri dari tiga bagian utama, yaitu bagian untuk menampilkan data uji, hasil sistem inferensi *fuzzy* untuk data uji, dan *field* pengujian manual. Adapun keterangan untuk Gambar 4.6.a., dan Gambar 4.6.b. adalah:

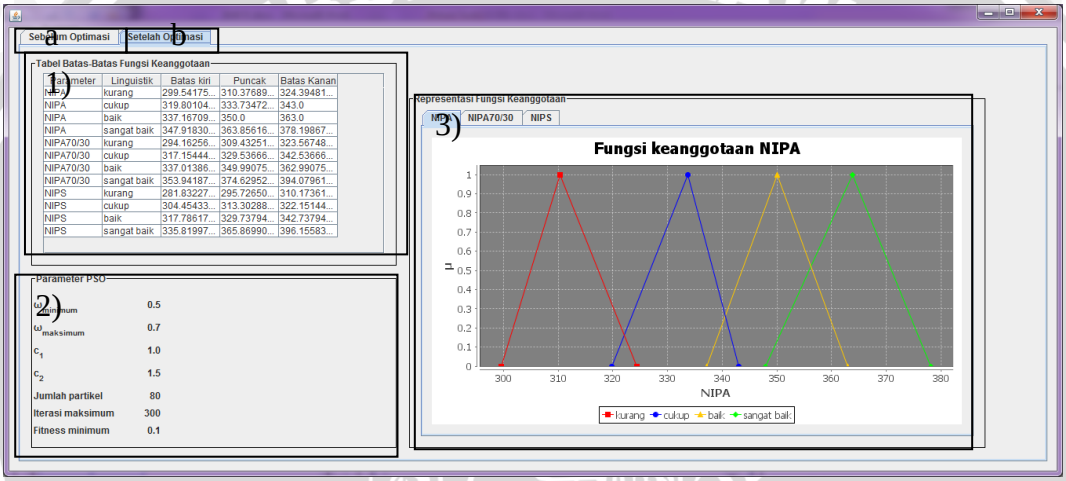
a. Tabel yang menampilkan data uji
b. Tombol untuk melakukan proses perhitungan data uji menggunakan sistem inferensi *fuzzy*
c. *Tab* untuk menampilkan hasil perhitungan sistem inferensi *fuzzy* untuk data uji menggunakan batas yang telah optimal, terdiri atas:
    1) *Text* yang menampilkan parameter PSO yang digunakan untuk mengoptimasi fungsi keanggotaan
    2) Tabel yang menampilkan parameter berupa nilai NIPA, NIPA70/30, NIPS, minat siswa, hasil sistem, dan hasil sebenarnya
    3) *Text* yang menampilkan persen akurasi perhitungan.
d. *Tab* untuk menampilkan hasil perhitungan sistem inferensi *fuzzy* untuk data uji menggunakan batas awal sebelum dioptimasi, terdiri atas:
    3) Tabel yang menampilkan parameter berupa nilai NIPA, NIPA70/30, NIPS, minat siswa, hasil sistem menggunakan fungsi keanggotaan awal, dan hasil sebenarnya
    4) *Text* yang menampilkan persen akurasi perhitungan.
e. *Field* untuk memasukkan data nilai dan minat siswa
f. Tombol untuk melakukan proses penghitungan data input
g. Tombol untuk mengosongkan kembali *field* yang dijelaskan pada poin e
h. *Text* yang digunakan untuk menampilkan hasil perhitungan, baik yang menggunakan batas yang telah optimal, maupun yang menggunakan batas awal.

4.2.3. Antarmuka Tampilan Fungsi Keanggotaan

Antarmuka ini digunakan untuk menampilkan batas-batas fungsi keanggotaan baik sebelum maupun sesudah optimasi dalam bentuk tabel dan representasi fungsi keanggotaan tersebut dalam bentuk grafik. Adapun gambaran dari antarmuka ini dapat dilihat pada Gambar 4.7.a. dan 4.7.b.



Gambar 4.7.a. Antarmuka Tampilan Fungsi Keanggotaan Awal



Gambar 4.7.b. Antarmuka Tampilan Fungsi Keanggotaan Optimal

Seperti yang dapat dilihat pada Gambar 4.7.a dan 4.7.b. antarmuka ini memiliki dua *tab* yaitu:

a. Sebelum optimasi, terdiri atas:
   1) Tabel yang menampilkan daftar fungsi keanggotaan awal
   2) Bagian berupa *tab* yang berisi grafik dari fungsi keanggotaan untuk masing-masing parameter.
b. Setelah optimasi, terdiri atas:
   1) Tabel yang menampilkan daftar batas fungsi keanggotaan optimal
   2) *Text* yang menampilkan parameter PSO yang digunakan dalam proses optimasi
   3) Bagian berupa *tab* yang berisi grafik dari fungsi keanggotaan untuk masing-masing parameter.

BAB V
ANALISA HASIL DAN PEMBAHASAN

5.1. Implementasi Pengujian

5.1.1. Skenario Pengujian

Skenario pengujian yang dilakukan mengacu pada penjabaran dari subbab 3.5, yaitu dibagi menjadi dua tahap, yaitu pengujian untuk optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization* dan pengujian akurasi menggunakan sistem inferensi *fuzzy* Mamdani.

5.1.1.1. Skenario Pengujian Parameter Optimasi

Pada skenario pengujian untuk optimasi, pengujian dilakukan terhadap proses optimasi fungsi keanggotaan menggunakan *Particle Swarm Optimization* dimana bahan pengujian berupa jumlah data latih sejumlah 80 data latih, konstanta c_1, konstanta c_2, inersia maksimumn ($\omega_{maximum}$), inersia minimum ($\omega_{minimum}$), jumlah partikel, dan iterasi maksimum. Tujuan dari pengujian ini adalah untuk menentukan parameter optimasi yang terbaik, yaitu kombinasi dari parameter yang menghasilkan nilai *cost* terendah. Nilai untuk setiap parameter yang diuji merupakan nilai yang diperoleh dari beberapa jurnal yang digunakan sebagai rujukan dibandingkan dengan nilai yang terdapat dalam jurnal yang digunakan sebagi jurnal utama. Skenario pengujian ini dibedakan menjadi empat skenario pengujian dan setiap skenario diulang sebanyak 5 kali.

a. Skenario pengujian untuk pencarian parameter c_1 dan c_2

   Skenario pengujian untuk pencarian parameter c_1 dan c_2 dilakukan dengan kombinasi nilai parameter c_1 dan c_2 yang berbeda-beda, sedangkan parameter lainnya diasumsikan bernilai tetap. Pada penelitian ini, nilai untuk setiap parameter c_1 dan c_2 yang akan dicoba adalah 0,5;1;1,5 dan 2.

b. Skenario pengujian untuk pencarian parameter inersia maksimum dan minimum, ω_{min} dan ω_{max}

   Skenario pengujian untuk pencarian parameter ω_{min} dan ω_{max} dilakukan dengan nilai parameter ω_{min} yang berbeda-beda, sedangkan parameter lainnya diasumsikan bernilai tetap kecuali untuk parameter c_1 dan c_2 menggunakan nilai

yang telah didapatkan dari skenario pengujian a. Dalam penelitian ini nilai yang akan diuji adalah ω_{min} 0,1; 0,3; dan 0,5 sedangkan ω_{max} 0,5; 0,7; dan 0,9.

c. Skenario pengujian untuk pencarian parameter jumlah partikel

Skenario pengujian ini digunakan untuk mencari jumlah partikel yang dapat menghasilkan nilai akurasi tertinggi. Nilai dari jumlah partikel dalam jurnal utama adalah 10, nilai ini akan dibandingkan dengan nilai 20, 60, 80, dan 100 partikel dengan parameter c_1, c_2, ω_{min} dan ω_{max} sesuai dengan hasil percobaan a dan b, dan iterasi maksimum diasumsikan bernilai tetap.

d. Skenario pengujian untuk pencarian parameter iterasi maksimum

Seperti skenario pengujian 1, 2, 3, skenario pengujian ini digunakan untuk mencari jumlah iterasi maksimum yang dapat menghasilkan nilai akurasi tinggi, dengan nilai parameter lain tetap sesuai dengan hasil percobaan a, b, dan c. Sesuai dengan jurnal utama, nilai terasi maksimum adalah 200 hingga 500 iterasi. Nilai iterasi maksimum akan diuji memiliki interval 100.

5.1.1.2. Skenario Pengujian Sistem Inferensi *Fuzzy* Mamdani

Pada skenario pengujian akurasi menggunakan sistem inferensi *fuzzy* Mamdani, digunakan 25 data uji dengan 10 kali percobaan. Skenario ini akan membandingkan rata-rata akurasi dari hasil sistem inferensi *fuzzy* dengan batas-batas yang telah dioptimasi menggunakan algoritma *Particle Swarm Optimization* dan data uji yang diproses menggunakan sistem inferensi *fuzzy* dengan batas-batas yang belum dioptimasi.

5.1.2. Hasil Pengujian

5.1.2.1. Hasil ujicoba untuk parameter optimasi

a. Hasil ujicoba untuk pencarian parameter c_1 dan c_2

Berdasarkan ujicoba yang telah dilakukan, hasil ujicoba untuk parameter c_1 dan c_2 secara umum dapat dilihat pada Tabel 5.1, sedangkan untuk hasil secara keseluruhan dapat dilihat pada lampiran. Parameter lain yang digunakan untuk mencari nilai dari parameter c_1 dan c_2 disesuaikan dengan jurnal utama, yaitu: ω_{min} dan ω_{max} 0.9, jumlah partikel 10, dan iterasi maksimum 200.

Tabel 5.1. Hasil ujicoba pencarian parameter c1 dan c2

c_1	c_2	Rata-rata *cost* keseluruhan
0.5	0.5	0.00391
	1	0.00368
	1.5	0.003398
	2	0.003208
1	0.5	0.004382
	1	0.003424
	1.5	0.003096
	2	0.005938
1.5	0.5	0.00483
	1	0.004094
	1.5	0.00384
	2	0.004212
2	0.5	0.00487
	1	0.003538
	1.5	0.010572
	2	0.013144

Dari hasil percobaan tersebut, c_1= 1 dan c_2=1.5 menjadi nilai yang dipilih karena memiliki nilai *cost* paling kecil diantara nilai lainnya yaitu 0.003096.

b. Hasil ujicoba pencarian parameter inersia minimum dan maksimum (ω_{min} dan ω_{max})

Berdasarkan ujicoba yang telah dilakukan, hasil ujicoba untuk parameter ω_{min} dan ω_{max} secara umum dapat dilihat pada Tabel 5.2, sedangkan untuk hasil yang lebih lengkap dapat dilihat pada lampiran. Parameter lain yang digunakan untuk mencari nilai dari parameter ω_{min} dan ω_{max} disesuaikan dengan jurnal utama, yaitu: jumlah partikel 10, iterasi maksimum 200 dan nilai c_1 dan c_2 disesuaikan berdasarkan pada hasil pengujian a yaitu 1 dan 1.5.

Tabel 5.2. Hasil ujicoba pencarian parameter ωmin dan ωmax

ω_{min}	ω_{max}	Rata-rata *cost* keseluruhan
0.1	0.5	3.69E-03
	0.7	0.003752
	0.9	0.003482
0.3	0.5	0.003546
	0.7	0.004244
	0.9	0.003332
0.5	0.5	0.00324
	0.7	0.003112
	0.9	0.003566

Dari hasil percobaan tersebut, maka nilai yang dipilih adalah ω_{min}= 0.5 dan ω_{max}= 0.7 karena memiliki nilai *cost* rata-rata paling kecil diantara nilai lainnya yaitu 0.003112.

c. Hasil ujicoba pencarian parameter jumlah partikel

Jumlah partikel diujicoba dengan parameter yang sama dengan percobaan sebelumnya, dengan iterasi maksimum 200, nilai c_1= 1, c_2=1.5 (didasarkan pada hasil percobaan a), dan ω_{min}= 0.5, ω_{max}= 0.7 (didasarkan pada hasil percobaan b). Hasil percobaan secara umum dapat dilihat pada Tabel 5.3, sedangkan untuk hasil selengkapnya dapat dilihat pada lampiran.

Tabel 5.3. Hasil ujicoba pencarian parameter jumlah partikel

Jumlah partikel	Rata-rata *cost* keseluruhan
10	0.00383
20	0.003012
60	0.003986
80	0.002618
100	0.004074

Dari hasil percobaan tersebut, nilai *cost* rata-rata terendah diperoleh saat jumlah partikel=80 dengan nilai *cost* 0.002618, oleh karena itu nilai tersebut dipilih sebagai angka untuk jumlah partikel.

d. Hasil ujicoba pencarian parameter iterasi maksimum

Iterasi maksimum diujicoba dengan parameter yang diperoleh dari percobaan sebelumnya, dengan jumlah partikel 80(berdasarkan hasil percobaan c), nilai c_1= 1 dan c_2=1.5 (didasarkan pada hasil percobaan a), dan ω_{min}= 0.5 dan ω_{max}= 0.7 (didasarkan pada hasil percobaan b). Hasil percobaan secara umum dapat dilihat pada Tabel 5.4., sedangkan untuk hasil selengkapnya dapat dilihat pada lampiran.

Tabel 5.4. Hasil ujicoba pencarian parameter iterasi maksimum

Iterasi	Rata-rata *cost* keseluruhan
200	0.00305
300	0.002864
400	0.003512
500	0.003182

Berdasarkan hasil percobaan diatas, untuk iterasi maksimum digunakan nilai 300 karena memiliki *cost* rata-rata minimum yaitu 0.002864.

5.1.2.2. Hasil ujicoba untuk sistem inferensi *fuzzy* Mamdani

Dari hasil percobaan yang telah dilakukan sebelumnya untuk pencarian parameter PSO, didapatkan parameter yang memiliki nilai *cost* minimum adalah : jumlah partikel 80, c_1= 1, c_2=1.5, ω_{min}= 0.5, ω_{max}= 0.7, dan iterasi maksimum=300. Hasil tersebut digunakan sebagai batas optimal untuk menghitung hasil sistem inferensi *fuzzy* dari 25 data uji. Nilai perbandingan akurasi yang dihasilkan dari sistem inferensi *fuzzy* menggunakan batas yang telah optimal dapat dilihat pada Tabel 5.5.

Tabel 5.5. Hasil ujicoba sistem inferensi *fuzzy*

Ujicoba ke-	Akurasi(%)	Rata-rata akurasi(%)
1	96	91.2
2	96	
3	96	
4	92	

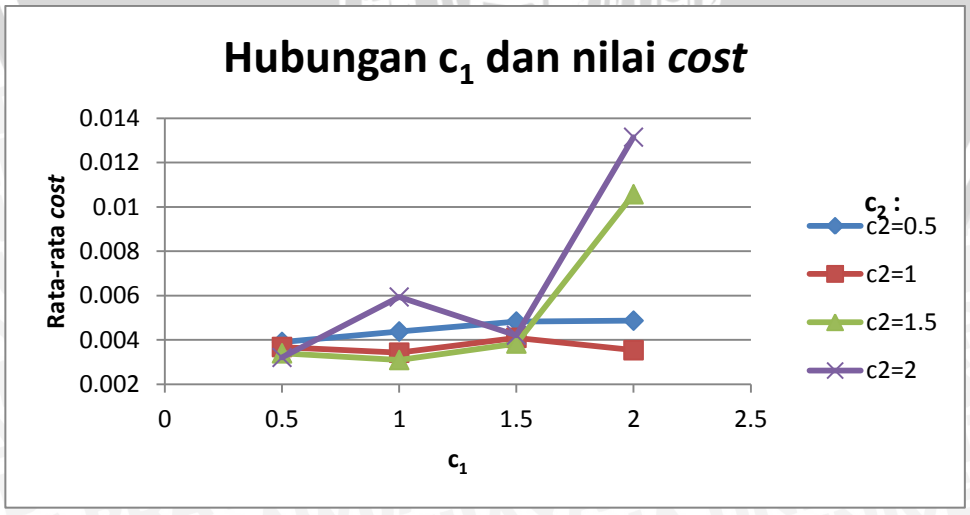
5	96	
6	96	
7	88	
8	96	
9	80	
10	76	

Berdasarkan hasil yang tercantum pada Tabel 5.5. didapatkan nilai rata-rata akurasi untuk dari sistem inferensi *fuzzy* menggunakan batas yang telah optimal adalah 91.2%. Sedangkan nilai persen akurasi yang dihasilkan dari sistem inferensi *fuzzy* menggunakan batas awal yang belum dioptimasi adalah 92%.

5.2. Analisa Hasil Pengujian

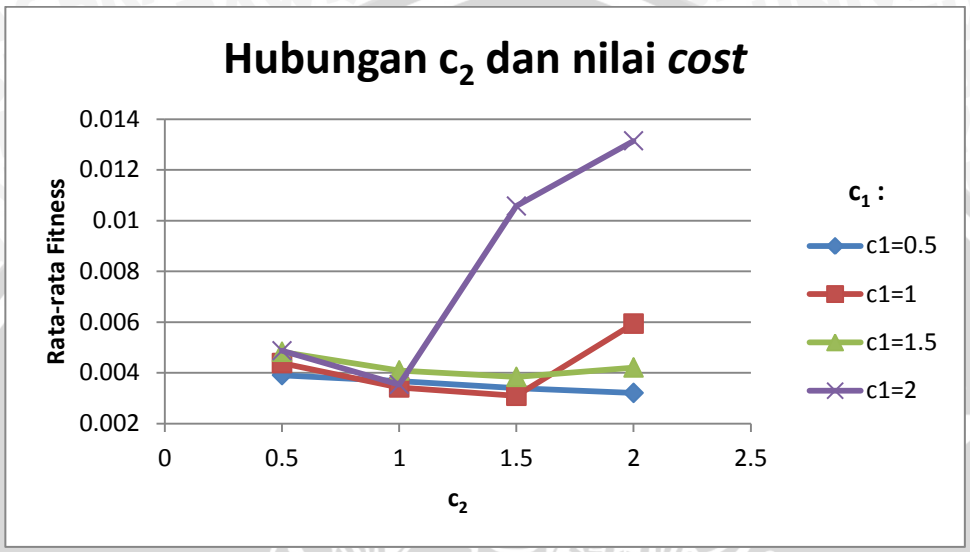
5.2.1. Analisa Hasil Pengujian Parameter PSO

Pada pengujian parameter PSO dicari parameter optimasi yang terbaik, yaitu kombinasi dari parameter yang menghasilkan nilai *cost* terendah. Parameter tersebut adalah konstanta c_1, konstanta c_2, inersia maksimumn ($\omega_{maximum}$), inersia minimum ($\omega_{minimum}$), jumlah partikel, dan iterasi maksimum. Pengaruh hubungan dari parameter PSO tersebut terhadap rata-rata nilai *cost* minimum yang dihasilkan dapat dilihat pada Gambar 5.1 untuk konstanta c_1, Gambar 5.2 untuk konstanta c_2, Gambar 5.3. untuk inersia maksimumn ($\omega_{maximum}$), Gambar 5.4 untuk inersia minimum ($\omega_{minimum}$), Gambar 5.5. untuk pengaruh jumlah partikel, dan. Gambar 5.6. untuk pengaruh iterasi maksimum.



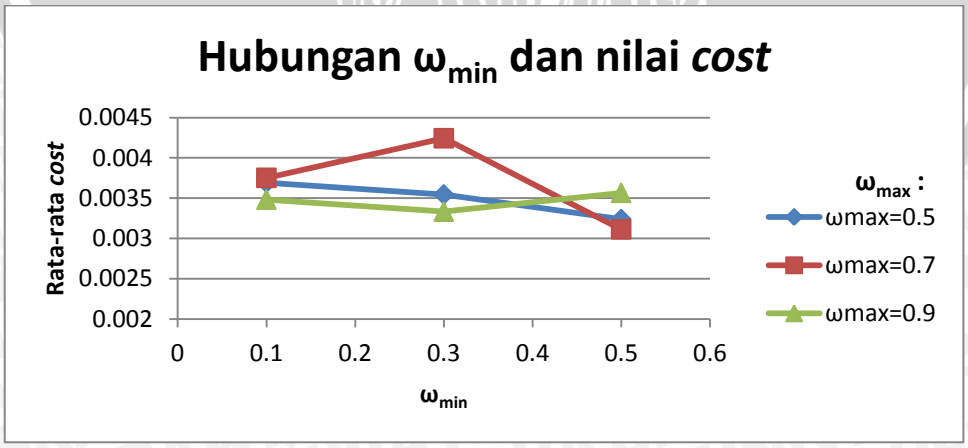
Gambar 5.1. Grafik hubungan c_1 dan nilai *cost* rata-rata

Berdasarkan Gambar 5.1. dapat dilihat bahwa untuk c_1 tidak stabil pada setiap nilainya. Nilai c_1=1 merupakan nilai minimum yang didapatkan pada penelitian ini. Saat c_2 bernilai 0,5 dan 1 nilai c_1 cenderung stabil, sedangkan nilai lainnya tidak. Bahkan untuk c_1=2 mayoritas kombinasi dengan nilai c_2 menunjukkan peningkatan nilai *cost* yang cukup tinggi.



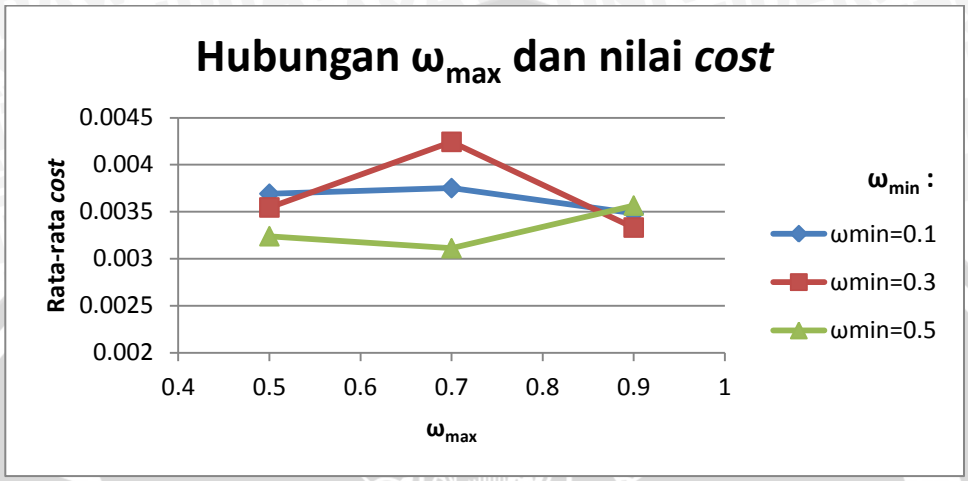
Gambar 5.2. Grafik hubungan c_2 dan nilai *cost* rata-rata

Berdasarkan Gambar 5.2. dapat dilihat bahwa untuk c_2 cenderung berbanding terbalik dengan nilai *cost* saat $c_2 \leq 1{,}5$ kecuali saat dikombinasikan dengan konstanta c_1=2. Rata-rata nilai minimum untuk c_2 didapatkan saat c_2=1.5, kecuali untuk kombinasi dengan c_1=2.



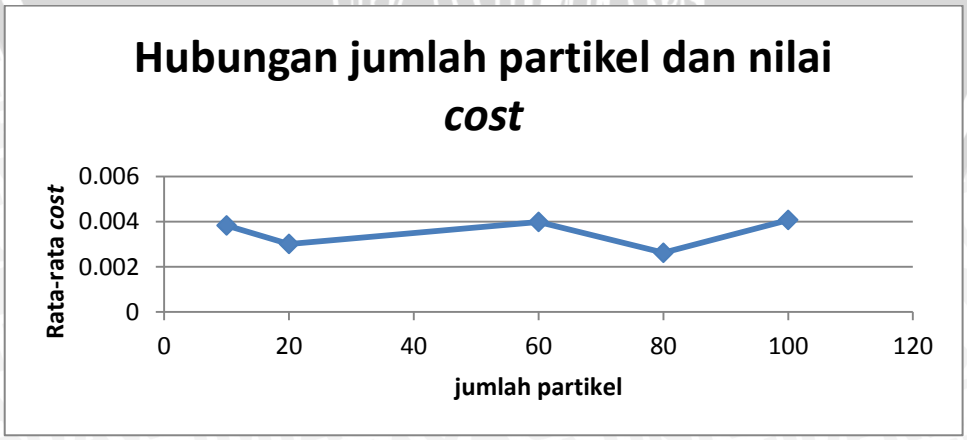
Gambar 5.3. Grafik hubungan inersia minimum (ωminimum) dan nilai *cost* rata-rata

Hubungan antara inersia minimum ($\omega_{minimum}$) dan nilai *cost* rata-rata sebagaimana yang ditunjukkan pada Gambar 5.3. dapat dikatakan tidak stabil. Nilai inersia minimum cenderung berbanding terbalik dengan nilai *cost* rata-rata yang dihasilkan dan mencapai nilai minimum saat bernilai 0,5.



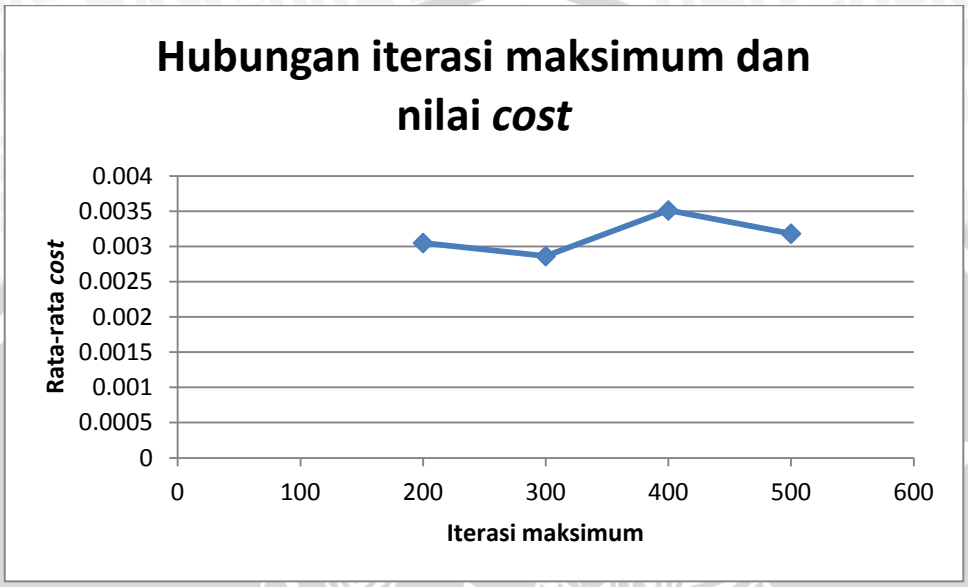
Gambar 5.4. Grafik hubungan inersia maksimum ($\omega_{maximum}$) dan nilai *cost* rata-rata

Hubungan antara inersia maksimum ($\omega_{maximum}$ dan nilai *cost* rata-rata sebagaimana yang ditunjukkan pada Gambar 5.4. dapat dikatakan tidak stabil. Perbandingan nilai inersia maksimum dengan nilai *cost* rata-rata tidak memiliki pola yang pasti, namun nilai *cost* yang dihasilkan dari setiap konstanta tidak berbeda jauh. Nilai inersia minimum yang mencapai *cost* minimum adalah 0,7 saat dikom binasikan dengan inersia minimum 0,5.



Gambar 5.5. Grafik hubungan jumlah partikel dan nilai *cost* rata-rata

Sebagaimana inersia minimum dan inersia maksimum, nilai untuk parameter jumlah partikel dan iterasi maksimum juga tidak stabil. Perbandingan jumlah partikel dan nilai *cost* cenderung naik turun, dan saat jumlah partikel=80 tercapai nilai *cost* yang minimum.



Gambar 5.6. Grafik hubungan iterasi maksimum dan nilai *cost* rata-rata

Untuk iterasi maksimum juga tidak memiliki pola tetap, hal ini dikarenakan untuk proses optimasi iterasi maksimum hanya menjadi pembatas iterasi saat nilai fungsi tujuan (nilai *cost*) yang diharapkan belum tercapai. Sedangkan mayoritas proses optimasi yang dilakukan dapat dihentikan sebelum iterasi maksimum tercapai. Sehingga dapat dikatakan iterasi maaksimum ini tidak terlalu berpengaruh terhadap proses optimasi dalam penelitian ini.

5.2.2. Analisa Hasil Pengujian Sistem Inferensi *Fuzzy*

Berdasarkan hasil yang tercantum pada Tabel 5.5. didapatkan nilai akurasi terbaik dari sistem inferensi *fuzzy* dengan menggunakan batas fungsi keanggotaan yang telah dioptimasi adalah sebesar 96% berbeda 4% dari nilai akurasi yang dihasilkan dari sistem inferensi *fuzzy* menggunakan batas awal yang belum dioptimasi yaitu sebesar 92%. Sedangkan rata-rata akurasi yang didapatkan dari 10 kali percobaan menggunakan batas yang telah dioptimasi adalah 91,2%. Nilai akurasi yang dihasilkan dari sistem inferensi *fuzzy* menggunakan batas yang telah

optimal mayoritas lebih besar daripada menggunakan batas awal yang belum dioptimasi. Namun pada beberapa percobaan, akurasi yang dihasilkan oleh sistem yang menggunakan batas fungsi keanggotaan yang telah dioptimasi lebih rendah daripada akurasi dari sistem yang menggunakan batas fungsi keanggotaan awal.

Hasil akurasi yang berbeda-beda tersebut, dikarenakan hasil fungsi keanggotaan yang dihasilkan dari proses optimasi tidak tetap. Hal ini mungkin terjadi dikarenakan terdapat faktor bilangan acak (bilangan *random*) yang dibangkitkan setiap kali menghitung kecepatan pergerakan partikel. Bilangan tersebut belum tentu sama antara proses optimasi fungsi keanggotaan satu dengan lainnya sehingga batas fungsi keanggotaan yang dihasilkan berbeda pula. Sedangkan dalam sistem inferensi *fuzzy* batas fungsi keanggotaan berpengaruh besar karena akan mempengaruhi pemilihan aturan yang digunakan.

Pada sistem inferensi *fuzzy* menggunakan fungsi keanggotaan yang telah dioptimasi masih belum dapat mencapai nilai akurasi 100%, hal ini dimungkinkan bisa terjadi karena pengaruh dua faktor. Faktor yang pertama, yaitu saat proses optimasi fungsi keanggotaan dimana pada penelitian ini hanya fungsi keanggotaan *input* saja yang dioptimasi sedangkan fungsi keanggotaan *output* tidak mengalami proses optimasi. Faktor kedua, sistem inferensi *fuzzy* yang digunakan adalah metode Mamdani, dimana metode tersebut memiliki kelemahan yaitu parameter yang digunakan dianggap sama atau tidak memberikan prioritas pada parameter tertentu. Sedangkan pada penerapannya, terdapat beberapa kasus pada penenentuan jurusan siswa SMA dimana minat siswa menjadi prioritas yang lebih tinggi dibandingkan dengan nilai akademik, misalnya ketika nilai akademik yang berpengaruh pada masing-masing jurusan dari siswa memiliki nilai yang hampir sama.

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan dapat diambil beberapa kesimpulan antara lain:

1. Proses optimasi fungsi keanggotaan menggunakan algoritma *Particle Swarm Optimization* pada sistem inferensi *fuzzy* penentuan jurusan siswa SMA dilakukan dengan cara menerapkan algoritma PSO untuk proses optimasi setiap fungsi keanggotaan dari variabel linguistik pada setiap parameter *input* berupa nilai yang telah ditransformasi. Fungsi keanggotaan baru yang dihasilkan dari proses optimasi tersebut kemudian digunakan dalam sistem inferensi *fuzzy* Mamdani untuk menentukan rekomendasi jurusan bagi siswa.
2. Kombinasi parameter dari algoritma *Particle Swarm Optimization* untuk optimasi fungsi keanggotaan *fuzzy* terbaik pada penelitian ini adalah : jumlah partikel 80, c_1= 1, c_2=1.5, ω_{min}= 0.5, ω_{max}= 0.7, dan iterasi maksimum=300.
3. Dari beberapa kali percobaan, didapatkan nilai akurasi sistem yang berbeda satu sama lain karena batas linguistik fungsi keanggotaan yang dihasilkan dari proses optimasi fungsi keanggotaan juga berbeda. Hal ini mungkin terjadi dikarenakan terdapat faktor bilangan acak (bilangan *random*) yang dibangkitkan setiap kali menghitung kecepatan pergerakan partikel dalam proses optimasi. Hasil akurasi terbaik yang didapatkan dari penelitian ini adalah sebesar 96%, hasil tersebut berbeda 4% dari hasil akurasi perhitungan tanpa optimasi yaitu sebesar 92%. Namun, secara rata-rata hasil akurasi menggunakan optimasi masih lebih rendah daripada tanpa menggunakan optimasi, yaitu sebesar 91,2%.

6.2. Saran

Saran yang dapat diberikan dari hasil penelitian yang digunakan sebagai bahan pengembangan penelitian yang terkait dengan penelitian ini, antara lain:

1. Dipertimbangkan kembali mengenai penentuan prioritas antara nilai akademik dengan minat siswa dalam penentuan jurusan.

2. Dilakukan percobaan dalam optimasi fungsi keanggotaan *fuzzy* menggunakan metode sistem inferensi *fuzzy* lainnya.
3. Dilakukan percobaan untuk optimasi dengan bentuk fungsi keanggotaan lainnya.

DAFTAR PUSTAKA

[ABS-13] Asmiana, Z., Bu'ulolo, F., Siagian, P., 2013, Penggunaan Sistem Inferensi *Fuzzy* Untuk Penentuan Jurusan di SMA Negeri 1 Bireun, Saintia Matematika vol.1, no.3 pp. 233-247.

[BLJ-09] Blondin, James. 2009. *Particle Swarm Optimization Application in Parameterization of Classifiers*. Armstrong State University.

[COE-05] Cox, Earl. 2005. *Fuzzy Modeling and Genetic Algorithms For Data Mining and Exploration*. Morgan Kauffman. USA.

[DEP-06] Departemen Pendidikan Nasional 2006. Panduan Penyusunan Laporan Hasil Peserta Didik (Berdasarkan KTSP) Sekolah Menengah Atas (SMA). Departemen Pendidikan Nasional Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah Direktorat Pembinaan SMA. Jakarta.

[DEP-08] Departemen Pendidikan Nasional. 2008. Penetapan Kriteria Ketuntasan Minimal. Departemen Pendidikan Nasional Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah Direktorat Pembinaan SMA. Jakarta.

[DSM-13]Debnath, S., Shill,P., Murase, K., 2013. *Particle Swarm Based Adaptive Strategy for Tuning ofFuzzy Logic Controller*. International Journal of Artificial Intelligence &Application. Vol.4, No.1, January 2013.

[FGH-08]Fang, G., Ngai M., Ha, Q., 2008, *Automatic Fuzzy Membership Function Tuning Using the Particle Swarm Optimization,* IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application.

[JSM-97] Jang, Jhy-Shing Roger; Sun, Chuen-Tsal; Mizutani, Eiji.1997. *Neuro-Fuzzy and Soft-Computing A Computational Approach to Learning and Machine Intelligence.* Prentice-Hall, Inc. United States of America.

[KSH-10] Kusumadewi, Sri, Hari Purnomo. 2010. *Aplikasi Logika Fuzzy Untuk Pendukung Keputusan.* Graha Ilmu. Yogyakarta.

[NUG-06]Nugraha, Dany.dkk.2006.Diagnosis Gangguan Sistem Urinari pada Anjing dan Kucing Menggunakan VFI 5.Institut Pertanian Bogor.Bogor.

[OEA-10] Omizegba, Elijah E.; Adebayo, Gbijah E. 2010. *Optimizing Fuzzy Membership Function Using Particle Swarm Algorithm*. Proceeding of IEEE International Conference on System, Man, and Cybernetics. San Antonio.

[PES-10] Permana, E.K., Siti Zaiton Mohd Hashim. 2010. *Fuzzy membership Function Generation using Particle Swarm Optimization*. Int. J. Open Problems Compt. Math., Vol. 3, No. 1, March 2010.

[RVA-13]Rane, Vishal A. 2013. *Partice Swarm Optimization* (PSO) *Algorithm Techniques: Parameters Effect and Analysis*. International Journal of Innovative Research and Development, Vol.2, Issue 7, July, 2013.

[SAA-13] Satriawan, Adityaranda. 2013. Implementasi Metode TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution)* dalam Rekomendasi Penjurusan pada Sekolah Menengah Atas. Skripsi. Universitas Brawijaya. Malang.

[SAH-10] Saleh, Chairul; Avianti, Vira; Hasan, Asmi. *Optimization Of Fuzzy Membership Function Using Genetic Algorithm To Minimize The Mean Square Error Of Credit Status Prediction*. 2010. The 11th Asia Pacific Industrial Engineering and Management Systems Conference, December 2010.

LAMPIRAN

Lampiran 1: Data Penjurusan SMA Negeri 3 Malang Tahun 2013

Kelas X-2														
No	Matematika	Fisika		Biologi		Kimia		Sejarah	Geografi	Ekonomi	Sosiologi	Minat		Jurusan
		PPK	Praktikum	PPK	Praktikum	PPK	Praktikum					IPA	IPS	
1	83	87	90	88	80	86	88	83	80	75	85	1	2	IPA
2	75	78	86	85	87	78	88	76	85	76	83	1	2	IPS
3	89	81	86	82	87	83	88	83	77	76	85	1	2	IPA
4	85	91	89	93	85	91	79	88	86	83	88	1	2	IPA
5	94	79	88	82	80	81	88	82	81	76	86	1	2	IPA
6	87	88	91	89	87	88	92	84	87	80	83	1	2	IPA
7	88	77	79	80	86	83	78	82	81	79	85	2	1	IPS
8	83	84	91	87	86	90	78	83	79	78	82	1	2	IPA
9	75	83	85	82	85	78	92	79	81	77	83	1	2	IPA
10	84	85	95	80	87	85	92	81	83	79	84	1	2	IPA
11	87	85	91	85	93	87	88	83	83	79	82	1	2	IPA
12	92	81	81	83	80	86	88	75	75	77	82	1	2	IPA
13	91	85	88	83	80	88	78	84	79	78	88	1	2	IPA
14	75	76	85	80	87	82	92	80	77	78	85	2	1	IPS
15	75	75	88	80	80	78	83	78	78	79	85	1	2	IPA
16	85	78	90	84	93	84	92	81	82	78	81	1	2	IPA
17	79	75	89	81	85	79	92	76	77	82	85	1	2	IPA
18	82	84	90	85	93	83	88	81	85	81	84	1	2	IPA

19	75	75	84	80	80	78	83	78	80	77	85	1	2	IPA
20	75	77	82	80	85	81	78	77	80	76	87	1	2	IPA
21	78	75	82	80	80	78	78	77	80	78	85	1	2	IPS
22	85	81	86	80	85	90	92	81	78	75	84	1	2	IPA
23	87	86	89	82	87	78	92	81	80	75	79	1	2	IPA
24	88	78	92	80	93	87	83	78	85	75	82	1	2	IPA
25	92	86	88	91	90	95	92	80	86	82	85	1	2	IPA
26	79	75	83	80	80	84	92	76	83	82	85	1	2	IPS
27	78	75	82	80	80	80	78	77	78	85	80	1	2	IPS
28	77	75	85	80	80	78	92	79	77	79	85	1	2	IPS
29	76	75	88	80	90	78	83	76	80	79	85	1	2	IPS
30	78	84	86	86	90	87	88	82	82	82	88	1	2	IPA
31	76	75	77	80	80	78	78	80	76	79	85	1	2	IPA
32	75	75	87	80	80	80	88	79	79	77	85	1	2	IPS
33	82	76	86	80	80	86	92	78	80	77	85	1	2	IPA
34	95	83	87	83	80	80	83	78	84	79	83	1	2	IPA
35	81	75	84	82	80	78	78	79	79	77	85	1	2	IPS

Kelas X-4

No	Matematika	Fisika		Biologi		Kimia		Sejarah	Geografi	Ekonomi	Sosiologi	Minat		Jurusan
		PPK	Praktikum	PPK	Praktikum	PPK	Praktikum					IPA	IPS	
1	82	82	86	89	90	88	91	80	84	80	82	1	2	IPA
2	83	90	86	88	90	89	93	83	85	84	82	1	2	IPA
3	75	75	86	80	90	78	91	76	80	79	85	2	1	IPS
4	75	78	86	80	90	78	91	79	81	75	85	1	2	IPA
5	86	92	93	87	85	83	93	82	83	81	76	1	2	IPA

6	91	87	85	86	85	85	91	84	81	78	79	1	2	IPA
7	84	92	91	84	89	90	92	87	89	82	77	1	2	IPA
8	87	85	85	80	82	92	92	82	86	82	77	1	2	IPA
9	82	83	88	84	89	89	92	83	88	84	78	1	2	IPA
10	93	95	94	91	85	93	93	80	89	84	83	1	2	IPA
11	82	86	91	80	82	92	93	80	75	80	75	1	2	IPA
12	84	85	88	80	90	85	92	81	79	80	77	1	2	IPA
13	97	90	91	87	89	95	93	82	89	84	83	1	2	IPA
14	93	88	91	90	90	84	92	85	86	77	80	1	2	IPA
15	75	75	89	80	80	78	91	78	79	78	85	2	1	IPS
16	83	84	90	83	80	87	93	78	86	75	81	1	2	IPA
17	94	91	85	86	85	92	92	89	87	86	80	1	2	IPA
18	84	87	89	84	80	88	93	78	78	82	84	1	2	IPA
19	88	89	91	83	85	94	93	78	88	82	75	1	2	IPA
20	75	75	85	80	85	78	89	77	80	78	85	1	2	IPA
21	88	79	91	82	80	86	93	84	86	75	76	1	2	IPA
22	94	89	94	90	87	97	95	88	85	85	85	1	2	IPA
23	75	75	88	80	80	78	92	78	80	77	85	1	2	IPS
24	82	83	94	83	82	87	93	79	83	75	76	1	2	IPA
25	91	90	94	81	89	92	95	78	83	79	78	1	2	IPA
26	83	88	90	86	89	87	95	85	86	80	81	1	2	IPA
27	83	75	84	80	80	88	83	76	84	75	85	2	1	IPS
28	77	85	88	87	90	88	92	82	85	81	81	1	2	IPA
29	82	77	81	80	80	86	85	77	81	77	85	1	2	IPA
30	76	76	75	80	80	79	83	76	82	77	85	2	1	IPS

31	84	75	84	80	85	80	80	77	81	77	85	1	2	IPA
32	77	75	87	80	85	78	78	81	82	78	81	1	2	IPA
33	77	75	84	80	90	78	78	79	75	81	85	2	1	IPS
34	88	80	84	81	86	85	85	75	75	75	75	1	2	IPA
35	77	75	87	80	90	88	88	81	80	75	85	1	2	IPA

Kelas X-7

No	Matematika	Fisika		Biologi		Kimia		Sejarah	Geografi	Ekonomi	Sosiologi	Minat		Jurusan
		PPK	Praktikum	PPK	Praktikum	PPK	Praktikum					IPA	IPS	
1	75	77	85	80	91	78	85	78	80	77	85	1	2	IPA
2	78	80	86	84	89	78	88	75	78	82	85	1	2	IPA
3	91	88	87	91	94	90	92	81	80	82	85	1	2	IPA
4	84	85	84	84	91	83	90	75	75	75	80	1	2	IPA
5	88	84	85	84	90	87	90	75	81	75	81	1	2	IPA
6	84	86	87	83	94	80	90	78	75	80	78	1	2	IPA
7	75	75	87	82	90	78	90	83	77	75	85	1	2	IPA
8	85	81	86	81	92	88	90	77	75	76	76	1	2	IPA
9	87	79	85	82	90	87	90	75	79	83	80	1	2	IPA
10	92	82	84	81	88	80	88	77	80	75	81	1	2	IPA
11	84	85	87	80	88	86	91	81	78	81	75	1	2	IPA
12	75	80	86	81	92	78	90	77	79	81	83	2	1	IPS
13	84	83	87	84	93	89	90	80	76	84	80	1	2	IPA
14	75	75	86	80	90	78	90	80	79	76	85	1	2	IPA
15	87	84	86	81	92	81	90	83	81	81	79	1	2	IPA
16	80	84	83	82	91	78	85	77	82	76	85	1	2	IPA
17	78	94	86	82	89	87	92	75	80	75	75	1	2	IPA

18	86	85	85	80	94	83	90	82	78	77	81	1	2	IPA
19	80	81	87	80	88	78	90	82	78	77	83	1	2	IPA
20	85	84	86	87	90	86	92	83	84	86	82	1	2	IPA
21	83	89	86	88	95	90	91	84	85	82	80	1	2	IPA
22	81	80	85	89	91	88	90	81	78	83	83	1	2	IPA
23	83	88	86	83	91	83	88	79	79	76	78	1	2	IPA
24	82	86	85	85	90	84	90	82	78	83	82	1	2	IPA
25	76	82	87	80	88	82	90	82	80	82	76	1	2	IPS
26	97	85	85	80	92	91	90	75	75	75	78	1	2	IPA
27	85	85	86	82	88	83	91	79	80	79	75	1	2	IPA
28	75	75	86	80	91	78	90	80	76	79	85	1	2	IPS
29	86	88	85	82	91	79	90	75	75	78	80	1	2	IPA
30	85	80	86	87	90	83	90	83	88	83	83	1	2	IPA
31	75	77	86	80	92	85	90	81	79	75	85	1	2	IPS
32	75	75	87	80	89	78	90	77	77	81	85	1	2	IPA
33	75	78	86	85	89	86	90	83	79	78	80	2	1	IPS
34	75	75	87	82	91	78	90	77	80	78	85	1	2	IPA
35	88	80	87	80	95	83	90	75	78	75	75	1	2	IPA

Lampiran 2: Tabel Hasi l Ujicoba

Hasil ujicoba pencarian parameter c_1 dan c_2

c1	c2	Ujicoba ke-	*Cost* minimum	*Cost* maximum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
0.5	0.5	1	4.31E-07	0.00912	0.00295	0.00391
		2	0.00038	0.0092	0.00415	
		3	0.00133	0.00944	4.54E-03	
		4	0.00015	0.0095	4.11E-03	
		5	0.00015	0.00891	0.0038	
	1	1	0.00025	0.01	0.00336	0.00368
		2	0.00014	0.00995	0.00421	
		3	0.00005	0.00959	5.40E-03	
		4	8.83E-07	0.0096	2.66E-03	
		5	0.00012	0.00963	0.00277	
	1.5	1	2.62E-06	0.00885	0.00378	0.003398
		2	0.00001	0.00931	0.00438	
		3	0.00011	0.01402	3.54E-03	
		4	9.85E-08	0.00948	2.97E-03	
		5	0.00003	0.00821	0.00232	
	2	1	0.00005	0.0067	0.00238	0.003208
		2	0.00005	0.00979	0.00334	
		3	2.35E-06	0.00992	3.59E-03	
		4	2.76E-06	0.00805	2.55E-03	
		5	0.00004	0.00744	0.00418	
1	0.5	1	0.0001	0.0096	0.00406	0.004382
		2	1.53E-06	0.00959	0.00328	
		3	0.00019	0.00996	6.26E-03	
		4	0.00024	0.00937	4.52E-03	
		5	0.00003	0.00828	0.00379	
	1	1	0.00001	0.00903	0.00296	0.003424
		2	0.00048	0.0088	0.00432	
		3	0.00013	0.00932	3.53E-03	
		4	0.00009	0.00863	2.99E-03	
		5	0.00002	0.0088	0.00332	
	1.5	1	0.00005	0.00913	0.00298	0.003096
		2	9.52E-07	0.00875	0.00275	
		3	9.67E-06	0.00823	3.46E-03	
		4	5.17E-07	0.00785	2.47E-03	
		5	4.09E-06	0.00768	0.00382	
	2	1	0.00013	0.005	0.00155	0.005938
		2	3.30E-06	0.00958	0.00428	

		3	0.00007	0.125	1.62E-02	
		4	0.0005	0.00873	2.97E-03	
		5	0.00004	0.00894	0.00466	
1.5	0.5	1	6.86E-06	0.00929	0.00472	0.00483
		2	7.94E-07	0.00918	0.00452	
		3	0.00061	0.00973	6.05E-03	
		4	2.28E-06	0.00809	4.61E-03	
		5	0.00032	0.00986	0.00425	
	1	1	0.00015	0.00904	0.00435	0.004094
		2	0.00005	0.00946	0.00442	
		3	0.00029	0.00845	4.03E-03	
		4	5.93E-06	0.00886	2.43E-03	
		5	0.00021	0.00987	0.00524	
	1.5	1	0.00011	0.0098	0.00468	0.00384
		2	0.00011	0.00778	0.00198	
		3	0.0004	0.00931	4.56E-03	
		4	0.00026	0.00974	3.24E-03	
		5	0.00002	0.0099	0.00474	
	2	1	5.63E-06	0.0219	0.00358	0.004212
		2	1.15E-06	0.0253	0.00708	
		3	0.00058	0.0095	3.26E-03	
		4	0.00024	0.00854	3.44E-03	
		5	0.00007	0.00941	0.0037	
2	0.5	1	0.00054	0.01484	0.00617	0.00487
		2	0.00146	0.00926	0.00454	
		3	0.0005	0.00937	3.87E-03	
		4	0.00003	0.00985	5.15E-03	
		5	0.00003	0.00981	0.00462	
	1	1	0.00013	0.00669	0.00196	0.003538
		2	0.00004	0.0091	0.00447	
		3	0.00009	0.00807	2.87E-03	
		4	7.97E-09	0.00989	3.88E-03	
		5	0.0001	0.01046	0.00451	
	1.5	1	0.00158	0.32608	0.03434	0.010572
		2	0.00018	0.01559	0.00432	
		3	0.00026	0.01057	4.79E-03	
		4	0.00007	0.00935	3.32E-03	
		5	0.00009	0.00988	0.00609	
	2	1	0.00078	0.07108	0.00874	0.013144
		2	0.00006	0.24314	0.02621	
		3	0.00059	0.17905	0.01891	
		4	0.00037	0.01383	0.00551	
		5	0.00002	0.03636	0.00635	

Hasil ujicoba pencarian parameter ω_{min} dan ω_{max}

ω_{min}	ω_{max}	Ujicoba ke-	*Cost* minimum	*Cost* maximum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
0.1	0.5	1	4.72E-06	0.0098	0.00277	0.003692
		2	0.00023	0.00789	0.00362	
		3	0.00005	0.00873	0.00346	
		4	0.00233	0.00974	0.00542	
		5	0.00001	0.00906	0.00319	
	0.7	1	0.00011	0.00783	0.00399	0.003752
		2	0.00008	0.00806	0.00222	
		3	0.0002	0.00799	0.00375	
		4	0.00017	0.0097	0.00477	
		5	1.84E-07	0.00891	0.00403	
	0.9	1	0.00065	0.00919	0.00429	0.003482
		2	0.00034	0.0097	0.00366	
		3	0.00002	0.0091	0.00381	
		4	9.68E-06	0.00872	0.00207	
		5	0.00009	0.00721	0.00358	
0.3	0.5	1	0.00015	0.00772	0.00287	0.003546
		2	0.00006	0.00957	0.00307	
		3	0.00007	0.00857	0.00405	
		4	9.58E-06	0.00945	0.00422	
		5	3.22E-06	0.00879	0.00352	
	0.7	1	1.74E-07	0.00991	0.00306	0.004244
		2	0.00002	0.00933	0.00417	
		3	0.00003	0.00861	0.00429	
		4	0.00148	0.00962	0.00526	
		5	0.00001	0.0099	0.00444	
	0.9	1	4.00E-05	0.00877	0.0036	0.003332
		2	0.0005	0.0095	0.00282	
		3	9.40E-06	0.00993	0.00552	
		4	4.11E-06	0.00951	0.00278	
		5	2.00E-05	0.00623	0.00194	
0.5	0.5	1	0.00009	0.00946	0.00435	0.00324
		2	0.00006	0.00535	0.00246	
		3	0.00021	0.00845	0.00354	
		4	0.00004	0.00832	0.00238	
		5	1.78E-06	0.0099	0.00347	
	0.7	1	3.00E-05	0.00915	0.00261	0.003112

		2	4.90E-04	0.00785	0.00349	
		3	2.53E-06	0.00938	0.00257	
		4	0.00005	0.00674	0.00256	
		5	0.00001	0.0095	0.00433	
	0.9	1	0.00012	0.00862	0.00254	0.003566
		2	0.00124	0.00859	0.00479	
		3	5.50E-06	0.00611	0.00168	
		4	0.00034	0.00998	0.00538	
		5	0.00002	0.00922	0.00344	

Hasil ujicoba pencarian parameter jumlah partikel

Jumlah partikel	Ujicoba ke-	*Cost* minimum	*Cost* maximum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
10	1	0.00001	0.00941	0.00312	0.00383
	2	5.61E-08	0.00724	0.00265	
	3	0.00023	0.00751	0.00377	
	4	0.00016	0.00941	0.00484	
	5	0.00063	0.00948	0.00477	
20	1	8.24E-06	0.00982	0.00203	0.003012
	2	0.00003	0.00758	0.00315	
	3	0.00009	0.00983	0.00291	
	4	0.00017	0.00877	0.00433	
	5	0.00003	0.00756	0.00264	
60	1	2.26E-06	0.00951	0.00391	0.003986
	2	0.00031	0.00908	0.00498	
	3	0.00013	0.00989	0.00409	
	4	0.00048	0.00813	0.0038	
	5	0.00004	0.00929	0.00315	
80	1	1.05E-06	0.00858	0.00352	0.002618
	2	3.03E-07	0.00436	0.00127	
	3	0.00017	0.0086	0.00328	
	4	2.66E-06	0.00889	0.00208	
	5	0.00007	0.00933	0.00294	
100	1	0.00011	0.00882	0.00331	0.004074
	2	0.00003	0.00963	0.00545	
	3	0.00117	0.00963	0.00444	
	4	0.00074	0.00905	0.00389	
	5	0.00003	0.00875	0.00328	

Hasil ujicoba pencarian parameter iterasi maksimum

Iterasi	Ujicoba ke-	*Cost* minimum	*Cost* maximum	*Cost* rata-rata per ujicoba	Rata-rata *cost* keseluruhan
200	1	2.29E-06	0.00939	0.00406	0.00305
	2	0.00025	0.00794	0.00226	
	3	2.77E-07	0.0084	0.00309	
	4	0.00004	0.00816	0.00248	
	5	0.00006	0.00825	0.00336	
300	1	3.90E-06	0.00986	0.00411	0.002864
	2	0.00002	0.00985	0.00271	
	3	0.00006	0.0099	0.00286	
	4	0.00006	0.00928	0.00272	
	5	9.77E-06	0.00949	0.00192	
400	1	0.00006	0.00888	0.00463	0.003512
	2	0.00033	0.0086	0.003	
	3	4.79E-06	0.00788	0.00362	
	4	0.00004	0.00727	0.00391	
	5	0.00002	0.00592	0.0024	
500	1	2.04E-07	0.00947	0.0043	0.003182
	2	0.00004	0.00934	0.00313	
	3	0.00002	0.00851	0.00283	
	4	0.00007	0.00908	0.0028	
	5	0.00002	0.00992	0.00285	

Hasil batas optimasi menggunakan parameter terpilih yang digunakan untuk ujicoba sistem inferensi *fuzzy*

Ujicoba ke-1

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	299.5418	310.3769	324.3948
	Cukup	319.801	333.7347	343
	Baik	337.1671	350	363
	Sangat Baik	347.9183	363.8562	378.1987
NIPA70/30	Kurang	294.1626	309.4325	323.5675
	Cukup	317.1544	329.5367	342.5367
	Baik	337.0139	349.9908	362.9908
	Sangat Baik	353.9419	374.6295	394.0796
NIPS	Kurang	281.8323	295.7265	310.1736
	Cukup	304.4543	313.3029	322.1514
	Baik	317.7862	329.7379	342.7379
	Sangat Baik	335.82	365.8699	396.1558

Ujicoba ke-2

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	297.402	309.9635	322.8904
	Cukup	317.087	330.2175	342.9565
	Baik	337.4236	350	363.6355
	Sangat Baik	358.0329	381.3146	404.7916
NIPA70/30	Kurang	302.718	312.2188	323
	Cukup	316.7159	330.8522	342.7159
	Baik	339.7382	350.9031	361.201
	Sangat Baik	353.7976	360.8248	421.1261
NIPS	Kurang	298.99	309.9093	319.5352
	Cukup	307.2564	315.2564	322.7686
	Baik	315.3761	331.6239	342.594
	Sangat Baik	337.3213	370.6426	404.0315

Ujicoba ke-3

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	287.44	303.9164	318.6546
	Cukup	316.7222	328.8887	343.5556
	Baik	337	351.8583	363
	Sangat Baik	351.3189	377.9296	398.6707
NIPA70/30	Kurang	298.4939	310.5746	323.3448
	Cukup	317.2251	330.2251	343.6753
	Baik	337.8478	350.8478	361.587
	Sangat Baik	356.4583	379.2082	401.5832
NIPS	Kurang	291.5566	301.4239	312.7152
	Cukup	306.8659	314.8659	323
	Baik	316.5584	329.5584	342.7792
	Sangat Baik	329.7714	350.8398	369.684

Ujicoba ke-4

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	294.2807	309.5803	323.8944
	Cukup	314.7003	333.0662	343
	Baik	335.58	350.9467	363
	Sangat Baik	347.4181	370.9035	389.157
NIPA70/30	Kurang	297.133	310.0818	322.9795
	Cukup	317.4117	328.7649	342.5883
	Baik	336.3	350	363.4667
	Sangat Baik	367.1269	394.2969	422.6582
NIPS	Kurang	288.7209	301.7209	314.059
	Cukup	306.9072	315.1856	323.0928
	Baik	317.0996	329.9751	342.9751
	Sangat Baik	331.2111	354.4957	384.6787

Ujicoba ke-5

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	300.5151	310.9467	323.0488
	Cukup	316.2553	330.7447	343.7447
	Baik	337.3736	350	363.3736
	Sangat Baik	357.8298	381.304	406.912
NIPA70/30	Kurang	292.7016	306.7762	321.0505
	Cukup	316.6715	330.3285	343.9855
	Baik	338.2456	350.9628	359.3356
	Sangat Baik	346.0246	366.4406	380.9925
NIPS	Kurang	283.3345	299.1692	311.8391
	Cukup	305.0489	316.9511	320.0733
	Baik	317.607	330.2023	342.7977
	Sangat Baik	322.3349	342.5798	352.6852

Ujicoba ke-6

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	294.3274	310	322.757
	Cukup	317.6734	329.3215	343.4013
	Baik	337	350	364.1194
	Sangat Baik	355.7918	376.2964	393.1794
NIPA70/30	Kurang	294.6544	309.5309	322.6873
	Cukup	316.7494	329.7494	342.7494
	Baik	338.5987	349.1709	362.5702
	Sangat Baik	354.2789	376.674	396.0927
NIPS	Kurang	284.7332	297.7332	311.605
	Cukup	306.0468	316.7448	320.8775
	Baik	317.7572	330.7572	343
	Sangat Baik	336.0928	368.0031	399.9534

Ujicoba ke-7

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	306.2124	310	323.8375
	Cukup	314.9399	332.2182	343.3846
	Baik	337.0265	349.8674	363.0796
	Sangat Baik	358.5095	383.019	406.5581
NIPA70/30	Kurang	308.116	319.3377	326.2562
	Cukup	318.3365	329.5545	341.6635
	Baik	337.7457	349.9457	364.9255
	Sangat Baik	357.1772	380.8859	404.1694
NIPS	Kurang	290.0562	301.9101	313.7641
	Cukup	307.5914	315	323.2957
	Baik	317.7813	331.172	343
	Sangat Baik	344.7975	383.1289	417.764

Ujicoba ke-8

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	294.3377	309.9941	324.3407
	Cukup	316.9031	330.0323	343
	Baik	335.5516	347.3293	361.153
	Sangat Baik	349.2173	366.3205	385.7295
NIPA70/30	Kurang	298.9019	310.5219	322.5276
	Cukup	316.2088	329.6044	342.2088
	Baik	337.7053	350	362.5298
	Sangat Baik	360.8987	390.5823	421.3798
NIPS	Kurang	284.7037	299.046	311.5886
	Cukup	307.6117	314.259	322.1384
	Baik	316.8864	329.8864	342.9243
	Sangat Baik	334.2246	359.8512	386.1302

Ujicoba ke-9

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	302.0935	312.5679	326.6259
	Cukup	317.2558	328.721	343
	Baik	338.2152	350	362.2709
	Sangat Baik	354.1659	370.8034	392.5732
NIPA70/30	Kurang	303.341	311.9815	320.2258
	Cukup	315.3242	327.4863	344.6758
	Baik	337.6793	350.6793	362.5471
	Sangat Baik	356.6566	380.4349	407.1485
NIPS	Kurang	296.7485	303.4817	312.7922
	Cukup	307.7057	315.7057	323
	Baik	318.7933	329.4022	343
	Sangat Baik	330.736	357.569	382.1813

Ujicoba ke-10

Parameter	Variabel Linguistik	Batas A	Batas B	Batas C
NIPA	Kurang	301.3807	312.8515	324.4514
	Cukup	317.1936	329.6127	342.6127
	Baik	338.0876	350	359.7373
	Sangat Baik	357.3895	380.3895	403.7561
NIPA70/30	Kurang	292.764	305.9816	322.0199
	Cukup	316.8463	329.6926	342.9232
	Baik	336.6424	349.8808	363.5961
	Sangat Baik	358.3055	387.3545	413.4763
NIPS	Kurang	287.0801	300.0308	313.0247
	Cukup	307.6579	313.6842	321.0264
	Baik	318.2211	329.1859	343
	Sangat Baik	335.308	363.9576	394.881

Lampiran 3: Kode Program

1. Kelas PSO.java

```
package Kode.PSO;

import Kode.Excel.BacaBatas;
import Kode.Excel.BacaData;
import Kode.Tipe.BatasLinguistik;
import Kode.Tipe.ParameterInput;
import Kode.Tipe.ParameterTransformasi;
import java.util.ArrayList;

public class PSO {
    ArrayList<BatasLinguistik>                          batasAsli=new
ArrayList<BatasLinguistik>();
    ArrayList<BatasLinguistik>                          batasBaru=new
ArrayList<BatasLinguistik>();
    ArrayList<BatasLinguistik>                                NIPA=new
ArrayList<BatasLinguistik>();
    ArrayList<BatasLinguistik>                           NIPA7030=new
ArrayList<BatasLinguistik>();
    ArrayList<BatasLinguistik>                                NIPS=new
ArrayList<BatasLinguistik>();
    double fitnessMin;
    double fitnessMax;
    double fitnessRata;
    ArrayList<Double> daftarFitness=new ArrayList<Double>();
    double c1;
    double c2;
    double wmin;
    double wmax;
    int jmlPartikel;
    int iterasiMaksimum;
    double minMSE;
    ArrayList<Double> acakNIPA=new ArrayList<Double>();
    ArrayList<Double> acakNIPA7030=new ArrayList<Double>();
    ArrayList<Double> acakNIPS=new ArrayList<Double>();

    ProsesPSO psoKurang;
    ProsesPSO psoCukup;
    ProsesPSO psoBaik;
    ProsesPSO psoSangatBaik;

    public PSO(ArrayList<BatasLinguistik> batasAsli, double c1,
double c2, double wmin, double wmax, int jmlPartikel, int
iterasiMaksimum, double minMSE, ArrayList<Double> acakNIPA,
ArrayList<Double> acakNIPA7030, ArrayList<Double> acakNIPS) {
        this.batasAsli = batasAsli;
        this.c1 = c1;
        this.c2 = c2;
        this.wmin = wmin;
        this.wmax = wmax;
        this.jmlPartikel = jmlPartikel;
        this.iterasiMaksimum = iterasiMaksimum;
        this.minMSE = minMSE;
        this.acakNIPA = acakNIPA;
        this.acakNIPA7030 = acakNIPA7030;
```

```
        this.acakNIPS = acakNIPS;
        PSOKeseluruhan();
    }

public void PSOKeseluruhan(){
    NIPA=PSOParameter("NIPA", acakNIPA);
    NIPA7030=PSOParameter("NIPA70/30", acakNIPA7030);
    NIPS=PSOParameter("NIPS", acakNIPS);
    batasBaru.addAll(NIPA);
    batasBaru.addAll(NIPA7030);
    batasBaru.addAll(NIPS);
    fitnessMin=cariFitnessMin(daftarFitness);
    fitnessMax=cariFitnessMax(daftarFitness);
    fitnessRata=cariFitnessRata(daftarFitness);
}

public       ArrayList<BatasLinguistik>       PSOParameter(String
namaParameter, ArrayList<Double> angkaAcak){

    ArrayList<BatasLinguistik>                  batasParameter=new
ArrayList<BatasLinguistik>();
    BatasLinguistik kurang=new BatasLinguistik();
    BatasLinguistik cukup=new BatasLinguistik();
    BatasLinguistik baik=new BatasLinguistik();
    BatasLinguistik sangatBaik= new BatasLinguistik();
    BatasLinguistik kurangBaru;
    BatasLinguistik cukupBaru;
    BatasLinguistik baikBaru;
    BatasLinguistik sangatBaikBaru;
    double fitnessKurang;
    double fitnessCukup;
    double fitnessBaik;
    double fitnessSangatBaik;
        for (int i = 0; i < batasAsli.size(); i++) {

if(batasAsli.get(i).getNamaParameter().equalsIgnoreCase(namaParame
ter)&&
batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase("kurang")){
                kurang=batasAsli.get(i);
            }

if(batasAsli.get(i).getNamaParameter().equalsIgnoreCase(namaParame
ter)&&
batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase("cukup")){
                cukup=batasAsli.get(i);
            }

if(batasAsli.get(i).getNamaParameter().equalsIgnoreCase(namaParame
ter)&&
batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase("baik")){
                baik=batasAsli.get(i);
            }

if(batasAsli.get(i).getNamaParameter().equalsIgnoreCase(namaParame
ter)&&
batasAsli.get(i).getNamaLinguistik().equalsIgnoreCase("sangat
baik")){
```

```
                sangatBaik=batasAsli.get(i);
            }
        }
    psoKurang=new ProsesPSO(c1, c2, wmin, wmax, jmlPartikel,
iterasiMaksimum, minMSE, kurang, angkaAcak);
    psoCukup= new ProsesPSO(c1, c2, wmin, wmax, jmlPartikel,
iterasiMaksimum, minMSE, cukup, angkaAcak);
    psoBaik= new ProsesPSO(c1, c2, wmin, wmax, jmlPartikel,
iterasiMaksimum, minMSE, baik, angkaAcak);
    psoSangatBaik= new ProsesPSO(c1, c2, wmin, wmax, jmlPartikel,
iterasiMaksimum, minMSE, sangatBaik, angkaAcak);

    kurangBaru=psoKurang.getBatasBaru();
    cukupBaru=psoCukup.getBatasBaru();
    baikBaru=psoBaik.getBatasBaru();
    sangatBaikBaru=psoSangatBaik.getBatasBaru();

    fitnessKurang=psoKurang.getFitness();
    fitnessCukup=psoCukup.getFitness();
    fitnessBaik=psoBaik.getFitness();
    fitnessSangatBaik=psoSangatBaik.getFitness();

    batasParameter.add(kurangBaru);
    batasParameter.add(cukupBaru);
    batasParameter.add(baikBaru);
    batasParameter.add(sangatBaikBaru);

    daftarFitness.add(fitnessKurang);
    daftarFitness.add(fitnessCukup);
    daftarFitness.add(fitnessBaik);
    daftarFitness.add(fitnessSangatBaik);

    boolean cek1,cek2,cek3;
    cek1=cek(kurangBaru, cukupBaru);
    cek2=cek(cukupBaru,baikBaru);
    cek3=cek(baikBaru,sangatBaikBaru);


    if(cek1==false&&cek2==false&&cek3==false)
        PSOParameter(namaParameter,angkaAcak);
        return batasParameter;

}
boolean cek(BatasLinguistik kiri, BatasLinguistik kanan){
    boolean status=false;
    if(kanan.getBatasA()<kiri.getBatasC()){
        status=true;
    }
    return status;
}
public double cariFitnessMin(ArrayList<Double> fitness){
    double fitmin=fitness.get(0);
    for (int i = 1; i < fitness.size(); i++) {
        fitmin=Math.min(fitmin, fitness.get(i));
    }
    return fitmin;
}
```

```
    public double cariFitnessMax(ArrayList<Double> fitness){
        double fitmax=fitness.get(0);
        for (int i = 1; i < fitness.size(); i++) {
            fitmax=Math.max(fitmax, fitness.get(i));
        }
        return fitmax;
    }
    public double cariFitnessRata(ArrayList<Double> fitness){
        double jmlfit=0;
        double fitRata;
        for (int i = 0; i < fitness.size(); i++) {
            jmlfit=jmlfit+fitness.get(i);
        }
        fitRata=(double)jmlfit/fitness.size();
        return fitRata;
    }

    public ArrayList<BatasLinguistik> getBatasBaru() {
        return batasBaru;
    }

    public double getFitnessMax() {
        return fitnessMax;
    }

    public double getFitnessMin() {
        return fitnessMin;
    }

    public double getFitnessRata() {
        return fitnessRata;
    }

    public ArrayList<Double> getDaftarFitness() {
        return daftarFitness;
    }

        public double getjumlahfitness() {
            double jml=0;
            for (int i = 0; i < daftarFitness.size(); i++) {
                jml=jml+daftarFitness.get(i);
            }
            return jml;
    }
        public double getratajmlfit(){
            double jmlnipa=0;
            double jmlnipa7030=0;
            double jmlnips=0;
            double rata;

            for (int i = 0; i < daftarFitness.size(); i++) {

if(batasBaru.get(i).getNamaParameter().equalsIgnoreCase("NIPA")){
                    jmlnipa=jmlnipa+daftarFitness.get(i);
                }

if(batasBaru.get(i).getNamaParameter().equalsIgnoreCase("NIPA70/30
```

```
")){
                    jmlnipa=jmlnipa7030+daftarFitness.get(i);
                }

if(batasBaru.get(i).getNamaParameter().equalsIgnoreCase("NIPS")){
                    jmlnipa=jmlnips+daftarFitness.get(i);
                }

            }
            rata=(jmlnipa+jmlnipa7030+jmlnips)/3;
                return rata;
        }
        public double getjmlratafit(){
            double jmlnipa=0;
            double jmlnipa7030=0;
            double jmlnips=0;
            double ratanipa;
            double ratanipa70;
            double ratanips;
            double jml;

            for (int i = 0; i < daftarFitness.size(); i++) {

if(batasBaru.get(i).getNamaParameter().equalsIgnoreCase("NIPA")){
                    jmlnipa=jmlnipa+daftarFitness.get(i);
                }

if(batasBaru.get(i).getNamaParameter().equalsIgnoreCase("NIPA70/30
")){
                    jmlnipa=jmlnipa7030+daftarFitness.get(i);
                }

if(batasBaru.get(i).getNamaParameter().equalsIgnoreCase("NIPS")){
                    jmlnipa=jmlnips+daftarFitness.get(i);
                }

            }
            ratanipa=jmlnipa/4;
            ratanipa70=jmlnipa7030/4;
            ratanips=jmlnips/4;
            jml=ratanipa+ratanipa70+ratanips;
            return jml;
        }
}
```

2. Kelas Fuzzy.java

```
package Kode.Fuzzy;

import Kode.Excel.BacaAturan;
import Kode.Excel.BacaBatas;
import Kode.Tipe.BatasLinguistik;
import java.util.ArrayList;
```

```
public class Fuzzy {


    ArrayList<Aturan> aturanDipakai=new ArrayList<Aturan>();
    double NIPA=0;
    double NIPA7030=0;
    double NIPS=0;
    int minatIPA=0;
    int minatIPS=0;
    ArrayList<BatasLinguistik>                                  batas=new
ArrayList<BatasLinguistik>();
    ArrayList<BilanganFuzzy>                    hasilFuzzifikasi=new
ArrayList<BilanganFuzzy>();
    ArrayList<BilanganFuzzy>                hasilFuzzifikasiNIPA=new
ArrayList<BilanganFuzzy>();
    ArrayList<BilanganFuzzy>              hasilFuzzifikasiNIPA7030=new
ArrayList<BilanganFuzzy>();
    ArrayList<BilanganFuzzy>                hasilFuzzifikasiNIPS=new
ArrayList<BilanganFuzzy>();
    BilanganFuzzy hasilFuzzifikasiMinatIPA=new BilanganFuzzy();
    BilanganFuzzy hasilFuzzifikasiMinatIPS=new BilanganFuzzy();
    ArrayList<BilanganFuzzy>                      hasilImplikasi=new
ArrayList<BilanganFuzzy>();
    BilanganFuzzy hasilKomposisi[]=new BilanganFuzzy[2];
    BilanganFuzzy hasilDefuzzifikasi=new BilanganFuzzy();
    public Fuzzy() {
    }

    public Fuzzy(double NIPA, double NIPA7030, double NIPS, int
minatIPA, int minatIPS, ArrayList<BatasLinguistik> batas) {
        this.NIPA = NIPA;
        this.NIPA7030 = NIPA7030;
        this.NIPS = NIPS;
        this.minatIPA = minatIPA;
        this.minatIPS = minatIPS;
        this.batas=batas;
        prosesFuzzy();
    }
    public void prosesFuzzy(){
        fuzzifikasi();
        pemilihanAturan();
        implikasiMin();
        komposisiAturanMax();
        defuzzifikasi();
    }

public void fuzzifikasi(){
    BilanganFuzzy bil=new BilanganFuzzy();
    for (int i = 0; i < batas.size(); i++) {

if(batas.get(i).getNamaParameter().equalsIgnoreCase("NIPA")){
            bil=prosesFuzzifikasi(NIPA, batas.get(i));
            if(bil.getDerajatKeanggotaan()!=0)
                    hasilFuzzifikasiNIPA.add(bil);
        }
        else
if(batas.get(i).getNamaParameter().equalsIgnoreCase("NIPA70/30")){
```

```
            bil=prosesFuzzifikasi(NIPA7030, batas.get(i));
            if(bil.getDerajatKeanggotaan()!=0)
                hasilFuzzifikasiNIPA7030.add(bil);
        }
        else{
            bil=prosesFuzzifikasi(NIPS, batas.get(i));
            if(bil.getDerajatKeanggotaan()!=0)
                hasilFuzzifikasiNIPS.add(bil);
        }
        if(bil.getDerajatKeanggotaan()!=0)
            hasilFuzzifikasi.add(bil);
    }
    if(minatIPA==1&&minatIPS==2){
        bil=new BilanganFuzzy("MinatIPA", "tinggi", 1);
        hasilFuzzifikasi.add(bil);
        hasilFuzzifikasiMinatIPA=bil;
        bil=new BilanganFuzzy("MinatIPS", "rendah", 1);
        hasilFuzzifikasi.add(bil);
        hasilFuzzifikasiMinatIPS=bil;
    }
    else{
         bil=new BilanganFuzzy("MinatIPA", "rendah", 1);
        hasilFuzzifikasi.add(bil);
        hasilFuzzifikasiMinatIPA=bil;
        bil=new BilanganFuzzy("MinatIPS", "tinggi", 1);
        hasilFuzzifikasi.add(bil);
        hasilFuzzifikasiMinatIPA=bil;
    }
}

    public ArrayList<BilanganFuzzy> getHasilFuzzifikasi() {
        return hasilFuzzifikasi;
    }

public      BilanganFuzzy      prosesFuzzifikasi(double      angka,
BatasLinguistik linguistik){
   BilanganFuzzy bilFuzzy;
   double a,b,c,myu;
   a=linguistik.getBatasA();
   b=linguistik.getBatasB();
   c=linguistik.getBatasC();
   if (angka>=a&&angka<=b)
       myu=(double)((angka-a)/(b-a));
   else if(angka>b&&angka<=c)
       myu=(double)((c-angka)/(c-b));
   else
       myu=0;
   bilFuzzy=new        BilanganFuzzy(linguistik.getNamaParameter(),
linguistik.getNamaLinguistik(), myu);
   return bilFuzzy;
}

public void pemilihanAturan(){
    ArrayList<Aturan> semuaAturan=new ArrayList<Aturan>();
    BacaAturan aturan=new BacaAturan("data.xlsx", "aturan");
    semuaAturan=aturan.getAturan();
    ArrayList<Aturan> tempAtur=new ArrayList<Aturan>();
```

```
    for (int i = 0; i < semuaAturan.size(); i++) {
        for (int j = 0; j < hasilFuzzifikasiNIPA.size(); j++) {

if(hasilFuzzifikasiNIPA.get(j).getKategoriLinguistik().equalsIgnor
eCase(semuaAturan.get(i).getKategoriNIPA()))
            tempAtur.add(semuaAturan.get(i));
        }
    }
    for (int i = 0; i < tempAtur.size(); i++) {

if(!tempAtur.get(i).getMinatIPA().equalsIgnoreCase(hasilFuzzifikas
iMinatIPA.getKategoriLinguistik()))
            tempAtur.remove(i);
    }
    ArrayList<Aturan> tempAtur2=new ArrayList<Aturan>();
    for (int i = 0; i < tempAtur.size(); i++) {
        for (int j = 0; j < hasilFuzzifikasiNIPA7030.size(); j++)
{
            if
(tempAtur.get(i).getKategoriNIPA7030().equalsIgnoreCase(hasilFuzzi
fikasiNIPA7030.get(j).getKategoriLinguistik())) {
                tempAtur2.add(tempAtur.get(i));
            }
        }
    }
    for (int i = 0; i < tempAtur2.size(); i++) {
        for (int j = 0; j < hasilFuzzifikasiNIPS.size(); j++) {
            if
(tempAtur2.get(i).getKategoriNIPS().equalsIgnoreCase(hasilFuzzifik
asiNIPS.get(j).getKategoriLinguistik())) {
                aturanDipakai.add(tempAtur2.get(i));
            }
        }
    }
}

    public ArrayList<Aturan> getAturanDipakai() {
        return aturanDipakai;
    }
public void implikasiMin(){
    BilanganFuzzy hasil;
    double min=0;
    for (int i = 0; i < aturanDipakai.size(); i++) {
        for (int j = 0; j < hasilFuzzifikasiNIPA.size(); j++) {

if(aturanDipakai.get(i).getKategoriNIPA().equalsIgnoreCase(hasilFu
zzifikasiNIPA.get(j).getKategoriLinguistik())){

min=hasilFuzzifikasiNIPA.get(j).getDerajatKeanggotaan();
            }
                for      (int      k      =      0;      k      <
hasilFuzzifikasiNIPA7030.size(); k++) {

if(aturanDipakai.get(i).getKategoriNIPA7030().equalsIgnoreCase(has
ilFuzzifikasiNIPA7030.get(k).getKategoriLinguistik())){

if(hasilFuzzifikasiNIPA7030.get(k).getDerajatKeanggotaan()<min){
```

```
min=hasilFuzzifikasiNIPA7030.get(k).getDerajatKeanggotaan();
                        }
                            for    (int    l    =    0;    l    <
hasilFuzzifikasiNIPS.size(); l++) {

if(aturanDipakai.get(i).getKategoriNIPS().equalsIgnoreCase(hasilFu
zzifikasiNIPS.get(l).getKategoriLinguistik())){

if(hasilFuzzifikasiNIPS.get(l).getDerajatKeanggotaan()<min){

min=hasilFuzzifikasiNIPS.get(l).getDerajatKeanggotaan();
                                    }
                        }
                    }
                }
            }

        hasil=new                           BilanganFuzzy("jurusan",
aturanDipakai.get(i).getJurusan(), min);
        hasilImplikasi.add(hasil);
    }

}
public void komposisiAturanMax(){
    ArrayList<BilanganFuzzy> ipa=new ArrayList<BilanganFuzzy>();
    ArrayList<BilanganFuzzy> ips=new ArrayList<BilanganFuzzy>();
    BilanganFuzzy komposisiIPA=new BilanganFuzzy();
    BilanganFuzzy komposisiIPS=new BilanganFuzzy();
    for (int i = 0; i < hasilImplikasi.size(); i++) {
        if
(hasilImplikasi.get(i).getKategoriLinguistik().equalsIgnoreCase("i
pa")) {
            ipa.add(hasilImplikasi.get(i));
        }
         else
             ips.add(hasilImplikasi.get(i));
    }
    if(ipa.size()<1)
        ipa.add(new BilanganFuzzy("jurusan", "ipa", 0));
    if(ips.size()<1)
        ips.add(new BilanganFuzzy("jurusan", "ips", 0));

    double maxIPA=ipa.get(0).getDerajatKeanggotaan();
    double maxIPS=ips.get(0).getDerajatKeanggotaan();
    for (int i = 1; i < ipa.size(); i++) {
        if(ipa.get(i).getDerajatKeanggotaan()>maxIPA)
            maxIPA=ipa.get(i).getDerajatKeanggotaan();
        }

    for (int i = 1; i < ips.size(); i++) {
        if(ips.get(i).getDerajatKeanggotaan()>maxIPA)
            maxIPS=ips.get(i).getDerajatKeanggotaan();
        }
    komposisiIPA=new BilanganFuzzy("jurusan", "ipa", maxIPA);
    komposisiIPS=new BilanganFuzzy("jurusan", "ips", maxIPS);
```

```
    hasilKomposisi[0]=komposisiIPA;
    hasilKomposisi[1]=komposisiIPS;
}
public void defuzzifikasi(){
    BatasLinguistik    outputIPA=new    BatasLinguistik("jurusan",
"ipa", 0, 1, 2);
    BatasLinguistik    outputIPS=new    BatasLinguistik("jurusan",
"ips", 1, 2, 3);
    double titikpotong[]=new double[4];
    double z=0;

titikpotong[0]=(hasilKomposisi[0].getDerajatKeanggotaan()*(outputI
PA.getBatasB()-outputIPA.getBatasA()))+outputIPA.getBatasA();

if(hasilKomposisi[0].getDerajatKeanggotaan()>hasilKomposisi[1].get
DerajatKeanggotaan()){
        titikpotong[1]=outputIPA.getBatasC()-
(hasilKomposisi[0].getDerajatKeanggotaan()*(outputIPA.getBatasC()-
outputIPA.getBatasB()));
        titikpotong[2]=outputIPA.getBatasC()-
(hasilKomposisi[1].getDerajatKeanggotaan()*(outputIPA.getBatasC()-
outputIPA.getBatasB()));
    }
    else
if(hasilKomposisi[1].getDerajatKeanggotaan()>hasilKomposisi[0].get
DerajatKeanggotaan()){

titikpotong[1]=hasilKomposisi[0].getDerajatKeanggotaan()*(outputIP
S.getBatasB()-outputIPS.getBatasA())+outputIPS.getBatasA();

titikpotong[2]=hasilKomposisi[1].getDerajatKeanggotaan()*(outputIP
S.getBatasB()-outputIPS.getBatasA())+outputIPS.getBatasA();
    }
    titikpotong[3]=outputIPS.getBatasC()-
(hasilKomposisi[1].getDerajatKeanggotaan()*(outputIPS.getBatasC()-
outputIPS.getBatasB()));
z=((hasilKomposisi[0].getDerajatKeanggotaan()*titikpotong[0])+(has
ilKomposisi[0].getDerajatKeanggotaan()*titikpotong[1])+

(hasilKomposisi[1].getDerajatKeanggotaan()*titikpotong[2])+(hasilK
omposisi[1].getDerajatKeanggotaan()*titikpotong[3]))/

((2*hasilKomposisi[0].getDerajatKeanggotaan())+(2*hasilKomposisi[1
].getDerajatKeanggotaan()));

    BilanganFuzzy hasilIPA=prosesFuzzifikasi(z, outputIPA);
    BilanganFuzzy hasilIPS=prosesFuzzifikasi(z, outputIPS);


if(hasilIPA.getDerajatKeanggotaan()>hasilIPS.getDerajatKeanggotaan
())
        hasilDefuzzifikasi=hasilIPA;
    else
if(hasilIPA.getDerajatKeanggotaan()==hasilIPS.getDerajatKeanggotaa
n()&&minatIPA==1){
            hasilDefuzzifikasi=hasilIPA;
    }
```

```
        else
            hasilDefuzzifikasi=hasilIPS;
    }
        public String getRekomendasiJurusan(){
            String jurusan;
            jurusan=hasilDefuzzifikasi.getKategoriLinguistik();
            return jurusan;
        }
        public double getNilaiKelayakan(){
            double nilai;
            nilai=hasilDefuzzifikasi.getDerajatKeanggotaan();
            return nilai;
        }
        public ArrayList<BilanganFuzzy> getHasilFuzzifikasiNIPA() {
            return hasilFuzzifikasiNIPA;
        }

        public  ArrayList<BilanganFuzzy>  getHasilFuzzifikasiNIPA7030()
    {
            return hasilFuzzifikasiNIPA7030;
        }

        public ArrayList<BilanganFuzzy> getHasilFuzzifikasiNIPS() {
            return hasilFuzzifikasiNIPS;
        }

        public ArrayList<BilanganFuzzy> getHasilImplikasi() {
            return hasilImplikasi;
        }

        public BilanganFuzzy[] getHasilKomposisi() {
            return hasilKomposisi;
        }
```