OPTIMASI *FUNGSI KEANGGOTAAN FUZZY INFERENCE SYSTEM TSUKAMOTO* MENGGUNAKAN *ALGORITMA GENETIKA* UNTUK SISTEM DIAGNOSA PENYAKIT TIROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fatimah Nadia Zanzabila
NIM: 125150201111051

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

PENGESAHAN

OPTIMASI *FUNGSI KEANGGOTAAN FUZZY INFERENCE SYSTEM TSUKAMOTO*
MENGGUNAKAN *ALGORITMA GENETIKA*
UNTUK SISTEM DIAGNOSA PENYAKIT TIROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Fatimah Nadia Zanzabila
NIM: 125150201111051

Skripsi ini telah diuji dan dinyatakan lulus pada
13 Mei 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D
NIP: 19720919 199702 1 001

Mengetahui
Ketua Program Studi Informatika

Issa Arwani, S.Kom, M.Sc
NIP: 19830922 201212 1 003

PERNYATAAN ORISINALITAS

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

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

Malang, 13 Mei 2016

Fatimah Nadia Zanzabila

NIM: 125150201111051

KATA PENGANTAR

Assalamu'alaikum wr.wb

Segala puji bagi Allah yang telah senantiasa memberikan penyusun kesehatan serta kemudahan sehingga dapat menyelesaikan proposal ini dengan baik. Tanpa pertolongan-Nya, mungkin penyusun tidak akan sanggup menyelesaikannya dengan baik. Shalawat serta salam semoga tetap terlimpahkan kepada junjungan kita Nabi Agung Muhammad SAW. Penelitian Optimasi Fungsi Keanggotaan *Fuzzy Inference System Tsukamoto* menggunakan Algoritma Genetika untuk Diagnosa Penyakit Tiroid ini disusun untuk memenuhi persyaratan kelulusan dengan mengerjakan mata kuliah wajib yaitu skripsi.

Penyusun juga mengucapkan terima kasih kepada Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D sebagai dosen pembimbing Skripsi karena dengan memenuhi penyusunan peneliti ini, penyusun dapat lebih memahami cara penulisan karya ilmah dan membuat sistem berkaitan dengan kecerdasan buatan yang berguna bagi masyarakat.

Melalui kata pengantar ini, penyusun menyadari bahwa masih banyak kekurangan yang mendasar. Oleh karena itu, penyusun mengharapkan kritik dan saran yang bersifat membangun dari para pembaca.

Terima kasih dan semoga skripsi ini dapat memberikan sumbangsih yang positif bagi kita semua. Amin.

Wassalamu'alaikum wr.wb.

Malang, 13 Mei 2016

Penyusun

fanazbil@gmail.com

ABSTRAK

Kelenjar tiroid merupakan salah satu organ dalam tubuh manusia, fungsi dari kelenjar ini yaitu untuk mengontrol metabolisme dalam tubuh manusia. Ketika kelenjar troid dapat bekerja normal maka jumlah hormon akan dipertahankan secara tepat yang fungsinya untuk menjaga metabolisme dalam tubuh. kelenjar tiroid dapat terkena beberapa gangguan yang dapat merusak fungsinya secara normal. Pada tahun 2009 di Amerika Serikat tedapat sejumlah 37.200 orang mengidap kanker tiroid. Di tahun 2012 sebanyak 20 juta penduduk di Amerika Serikat dari berbagai macam ras dan usia terkena gangguan pada tiroidnya. Pendekatan berbasis komputer dapat menjadi salah satu solusi alat untuk mempermudah diagnosa penyakit tiroid. Metode FIS Tsukamoto dipilih sebagai solusi penyelesaian masalah diagnosa penyakit tiroid karena fleksibel dan sederhana dalam menentukan aturan-aturan yang diberikan. Namun, kekurangan dari metode FIS Tsukamoto adalah penentuan batasan-batasan untuk fungsi keanggotaan yang masih manual sedangkan batasan tersebut berpengaruh pada hasil diagnosa sistem. Oleh karena itu dibutuhkan suatu metode untuk mengoptimasi batasan-batasan pada fungsi keanggotaan tersebut. Metode Algoritma Genetika dipilih sebagai metode yang akan digunakan untuk optimasi fungsi keanggotaan tersebut karena metode ini mampu menyelesaikan berbagai masalah optimasi dengan efisien dan efektif. Masukan yang diberikan pada sistem berupa 5 nilai tes yang diperlukan yaitu *t3-resin* test, total serum *thyroxin*, total serum *triiodothyronine*, *basal thyroid-stimulating hormone* (TSH), maksimal perbedaan dari nilai TSH. Sedangkan keluaran dari sistem berupa 3 kategori hasil diagnosa yaitu normal. *hypertiroidism* dan *hypotiroidism.* Sistem ini memiliki akurasi sebesar 86% data yang digunakan sebanyak 215 data dari UCI *repository*. Parameter Algoritma Genetika yang menghasilkan solusi optimum adalah kombinasi cr=0.7 dan mr=0.3, ukuran populasi sebesar 80 dan jumlah generasi sebanyak 100.

Kata kunci : Algoritma Genetika, optimasi fungsi keanggotaan, FIS Tsukamoto .

ABSTRACT

Thyroid gland is an organ ini human body, the function of this gland is to control the human body metabolism. When thyroid gland can work normally the amount of hormone will be maintained correctly that have function for keep body metabolism. The thyroid gland can affected by some disease that can destroy it's normal function. There is 37.200 people of USA in 2009 affected thyroid cancer. In 2012 there are 20 million people of USA from any age and any race affected thyroid disease. An approach using computer can be a exact solution as a tool for make thyroid disease diagnosis will be easier. FIS Tsukamoto method was chosen as a solution to solve the problem of thyroid disease diagnosis because can be flexible and simple to determine rules that given. But, the weakness of FIS Tsukamoto method is in determining the boundary for membership function that still using manual process, while the boundary affect the result of system diagnosis. So, required a method for optimation the boundaries in that boundary function. Genetic algorithm method was chosen as method that used for membership function optimation. Because this method can solve any optimation problem efficiently and effectively. The input that given in this system have 5 value of test that needed there is t3-resin , total of thyroxin serum, total of triiodothyronine serum, basal thyroid-stimulating hormone (TSH), the maximum differentiation from TSH value. While, the output of the system can be the one of 3 diagnosis category that is normal, *hypertiroidism* dan *hypotiroidism.* This system has accruration as big as 86% data from 215 data from uci repository. The best parameter that can produce optimum solution is combination of cr=0.7 and mr=0.3, population size is 80 and total of generation is 100.

Key word : genetic algorithm, membership function optimation, FIS Tsukamoto .

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL ix
DAFTAR GAMBAR x
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 3
1.6 Sistematika pembahasan 4
BAB 2 TINJAUAN PUSTAKA 5
2.1 Kajian Pustaka 5
2.2 Penyakit Tiroid 6
2.3 Fuzzy Inference System Tsukmoto 7
2.4 Algoritma Genetika 9
2.5.1 Struktur Algoritma Genetika 11
BAB 3 METODOLOGI 13
3.1 Tahapan Penelitian 13
3.3 Algoritma yang Digunakan 14
3.4 Kebutuhan Sistem 14
3.5 Pengujian Algoritma 15
3.5.1 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate* 15
3.5.2 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate* 15
3.5.3 Pengujian Ukuran Populasi 16
3.5.4 Pengujian Ukuran Generasi 16

BAB 4 PERANCANGAN 17
4.1 Formulasi Permasalahan 17
4.1.1 Diagnosa Penyakit Tiroid 19
4.1.2 Uji Coba Menggunakan FIS Tsukamoto 28
4.2 Siklus algoritma yang digunakan 30
4.3 Siklus penyelesaian masalah menggunakan Algoritma Genetika 48
4.3.1 Representasi Kromosom dan Perhitungan *Fitness* 48
4.3.2 Inisialisasi populasi awal 54
4.3.3 Reproduksi 55
4.3.4 Evaluasi dan Seleksi 57
BAB 5 IMPLEMENTASI 62
5.1 Struktur Kelas 62
5.2 Implementasi Kode Program 63
5.2.1 Implemetasi Proses Algortima Genetika 63
5.2.2 Proses FIS Tsukamoto 74
BAB 6 PENGUJIAN 84
6.1 Pengujian Metode Seleksi 84
6.2 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate* 86
6.3 Pengujian Ukuran Populasi 87
6.4 Pengujian Jumlah Generasi 89
6.5 Hasil dan Analisis Akurasi Sistem 90
BAB 7 PENUTUP 92
7.1 Kesimpulan 92
7.2 Saran 92
DAFTAR PUSTAKA 93

DAFTAR TABEL

Tabel 4. 1 Data set penyakit tiroid dari UCI Respository 17
Tabel 4. 2 Nilai input pasien 19
Tabel 4. 3 Derajat keanggotaan 22
Tabel 4. 4 Aturan 24
Tabel 4. 5 Nilai minimal derajat anggota pasien 25
Tabel 4. 6 Perhitungan az 26
Tabel 4. 7 Perbandingan uji coba sistem dengan pakar 28
Tabel 4. 8 Hasil uji dengan satu contoh kromosom 52
Tabel 4. 9 Inisialisai Populasi Awal 54
Tabel 4. 10 Tabel hasil konversi sesuai kegunaan setiap segmen 54
Tabel 4. 11 Hasil pengurutan tiap segmen 54
Tabel 4. 12 Cara mencari Offspring dari 55
Tabel 4. 13 Crossover 56
Tabel 4. 14 Mutasi 57
Tabel 4. 15 Individu Gabungan 57
Tabel 4. 16 Individu gabungan setelah diurutkan dan dikonversi 58
Tabel 4. 17 Populasi baru hasil tournament selection 59
Tabel 4. 18 Hasil pengurutan individu gabungan sesuai fitness 59
Tabel 4. 19 Populasi baru hasil elitism selection 59
Tabel 4. 20 Individu yang telah dihitung nilai prob dan probCum 60
Tabel 4. 21 Individu yang lolos seleksi 61
Tabel 4. 22 Populasi Baru hasil roulette wheel selection 61
Tabel 6. 1 Hasil pengujian metode seleksi 84
Tabel 6.2 Pengujian kombinasi cr dan mr 86
Tabel 6. 3 Pengujian ukuran populasi 88
Tabel 6.4 Pengujian jumlah generasi 89
Tabel 6. 5 Kromosom dengan *fitness* terbaik 90
Tabel 6. 6 Perbandingan hasil diagnosa pakar dengan sistem 90
Tabel 6. 7 Perbandingan hasil diagnosa pakar dengan sistem 91

DAFTAR GAMBAR

Gambar 2. 1 Mapping input dan output 7
Gambar 2. 2 Sistem inferensi fuzzy metode tsukamoto 9
Gambar 3. 1 Diagram Alir Metodologi Pelaksana 13
Gambar 4. 1 Diagram fungsi keanggotaan t3-resin 20
Gambar 4. 2 Diagram fungsi keanggotaan *thyroxin serum* 20
Gambar 4. 3 Diagram fungsi keanggotaan triiodothyronine serum 21
Gambar 4. 4 Diagram fungsi keanggotaan thyroid-stimulating hormone (TSH) 21
Gambar 4. 5 Diagram fungsi keanggotaan 22
Gambar 4. 6 Diagram fungsi keanggotaan hasil 23
Gambar 4. 7 Diagram alir sistem 30
Gambar 4. 8 Diagram alir proses membangkitkan populasi awal 32
Gambar 4. 9 Diagram alir proses *crossover* 33
Gambar 4. 10 Diagram alir proses *extended intermediate crossover* 34
Gambar 4. 11 Diagram alir proses mutasi 35
Gambar 4. 12 Diagram alir *exchange mutation* 36
Gambar 4. 13 Diagram alir proses pembentukan fungasi keanggotaan 37
Gambar 4. 14 Diagram alir proses fuzzifikasi 38
Gambar 4. 15 Diagram alir proses mesin inferensi 40
Gambar 4. 16 Diagram alir defuzzifikasi` 41
Gambar 4. 17 Diagram alir menghitung nilai fitness 43
Gambar 4. 18 Diagram alir proses seleksi 44
Gambar 4. 19 Diagram alir proses elitsm selection 45
Gambar 4. 20 Diagram alir proses tournament selection 46
Gambar 4. 21 Diagram alir metode roulette wheel selection 47
Gambar 4. 22 Representasi Kromosom 48
Gambar 4. 23 contoh kromosom 49
Gambar 4. 24 Kromosom setelah dikonversi sesuai fungsi persegmen 50
Gambar 4. 25 Kromosom yang telah diurutkan persegmen 50
Gambar 4. 26 Grafik fungsi keanggotaan t3-resin 50
Gambar 4. 27 Grafik fungsi keanggotaan thyroxin serum 51
Gambar 4. 28 Grafik fungsi keanggotaan nilai triiodotynine serum 51
Gambar 4. 29 Grafik fungsi keanggotaan nilai TSH 51
Gambar 4. 30 Grafik Fungsi Keanggotaan Perbedaan nilai TSH 52
Gambar 4. 31 Grafik Fungsi Keanggotaan Hasil 52
Gambar 5. 2 Kelas Diagram Sistem 62
Gambar 5.3 Kode program inisialisasi populasi awal 63
Gambar 5.4 Kode Program proses *crossover* 65
Gambar 5. 5 Kode program proses mutasi 66
Gambar 5.6 Kode program proses evaluasi 68
Gambar 5.7 Kode program proses *elitism selection* 70
Gambar 5.8 Kode program proses tournament selection 72
Gambar 5.9 Kode program metode *roulette wheel* 73

Gambar 5. 10 Kode program basis aturan 75
Gambar 5. 11 Kode program proses fuzzifikasi 77
Gambar 5. 12 Kode program proses fuzzifikasi pada parameter 79
Gambar 5. 13 Kode program proses inferensi 81
Gambar 5.14 Kode program proses defuzzifikasi 82
Gambar 6. 1 Grafik hasil uji coba metode seleksi sesuai nilai *fitness* 85
Gambar 6. 2 Hasil pengujian kombinasi cr dan mr 87
Gambar 6. 3 Hasil uji ukuran populasi 88
Gambar 6.4 Pengujian jumlah generasi 89

BAB 1 PENDAHULUAN

1.1 Latar belakang

Kelenjar tiroid merupakan salah satu organ dalam tubuh manusia. Bentuk dari kelenjar ini seperti bentuk kupu-kupu dengan adanya dua sayap yang diwakili oleh dua lobus yaitu lobus tiroid kiri dan kanan (Dogantekin dkk, 2007). Kelanjar tiroid adalah organ yang kecil dan letaknya di tengah-tengah leher bagian bawah, kelenjar ini merupakan salah satu organ yang penting dalam tubuh karena fungsinya yaitu mengontrol metabolisme (Keles, 2008). Kelenjar tiroid menghasilkan dua hormon tiroid aktif yakni *levothyroxine*(T4) dan *triiodothyronine* (*T3*) (Termurtas, 2009). Tiroid yang bekerja secara normal akan mempertahankan jumlah hormon yang diperlukan secara tepat yang berfungsi untuk menjaga metabolisme dalam tubuh (Keles & Keles, 2008).

Kelenjar tiroid ini dapat terkena berbagai macam masalah yang berbeda. Di dunia telah banyak terjadi permasalahan yang berkenaan dengan kelenjar tiroid(Dogantekin dkk, 2007). Pada tahun 2009 di Amerika Serikat jumlah pengidap kanker tiroid semakin bertambah hingga mencapai 37.200 kasus (Acharya dkk, 2012). Pada tahun 2012 dinyakatakan bahwa terdapat sekitar 20 juta penduduk Amerika dari berbagai macam usia dan ras mengidap penyakit tiroid (Liu dkk, 2012). Hal ini menunjukkan hampir setiap orang memiliki resiko yang sama untuk dapat mengidap penyakit tiroid.

Pendekatan berbasis komputer dapat menjadi salah satu solusi dalam permasalahan ini sebagai alat mendeteksi resiko penyakit (Savelonas dkk, 2009). Pada penelitian sebelumnya telah ada sistem pakar untuk membantu mendiagnosa penyakit tiroid pada manusia. Keles dan Keles (2008) melakukan penelitian dengan membuat sistem pakar untuk diagnosa resiko penyakit tiroid secara dini. Penelitian ini menggunakan *fuzzy neuro* dan dataset klasifikasi tiroid yang digunakan diambil dari mesin pembelajaran UCI. Tingkat akurasi yang dihasilkan dari diagnosa penyakit tiroid menggunakan fuzzy neuro memiliki nilai sebesar 95,33%. Selain itu Dogantekin, et al.(2010) juga telah melakukan penelitian dengan membuat sistem yang sama yaitu diagnosa penyakit tiroid. Pada contoh penelitian kedua ini proses sistemnya bekerja menggunakan 3 langkah. Langkah pertama adalah fitur untuk pengurangan variabel yang dibutuhkan menggunakan metode *principle component analysis* (PCA), langkah kedua yaitu mengklasifikasi dengan menggunakan metode klasifikasi least square *machine support machine* (LS-SVM) dan langkah terkahir yaitu evaluasi performa pada sistem diagnosa penyakit tiroid diestimasi dengan menggunakan klasifikasi akurasi, *k-fold cross validation* dan *confusion matrix* secara berurutan. Akurasi pada sistem ini telah mencapai 97,67%.

Sari dan Mahmudy (2015) telah melakukan penelitian dengan menggunakan metode yang sama dengan penelitian ini yaitu FIS Tsukamoto yang digunakan untuk menentukan kelayakan pegawai. Hasil dari pengujian pada

penelitian tersebut berupa perangkingan. Untuk menguji keakurasian sistem digunakan uji korelasi non parametrik *spearman*. Uji korelasinya menghasilkan keakuratan sebesar 0,952.

Untuk mendapatkan hasil yang lebih baik, FIS Tsukamoto dioptimasi dengan salah satu metode optimasi, diantaranya adalah algortima genetika. Sebelumnya terdapat penelitian yang menggunakan Algoritma Genetika sebagai metode untuk mengoptimasi fungsi keanggotaan pada FIS Tsukamoto . Masalah yang digunakan adalah memilih calon mahasiswa yang sesuai untuk menerima beasiswa dan BBP-PPA (Restuputri, et al., 20). Dari hasil peneilitian tersebut dapat dilihat dengan optimasi menggunakan Algoritma Genetika hasil akurasinya lebih tinggi dari penelitian sebelunya yang hanya menggunakan FIS Tsukamoto yaitu untuk penentuan Beasiswa-PPA sebesar 98.9% sedangkan BBP-PPA sebesar 98.7%.

Berdasarkan penjelasan beberapa penelitian yang pernah dilakukan, maka dirancang sebuah aplikasi yang menggabungkan keberhasilan objek penelitian dan keberhasilan metode *Fuzzy Inference System Tsukamoto* yaitu "Optimasi *Fuzzy Inference System* (FIS) *Tsukamoto* menggunakan *Algoritma Genetika* untuk Diagnosa Penyakit Tiroid". Metode *FIS Tsukamoto* dipilih untuk mengembangkan sistem pakar ini karena dinilai lebih fleksibel dan sederhana dalam menentukan aturan-aturan yang diberikan bila dibandingkan dengan metode lainnya. Agar mendapatkan hasil yang sesuai maka optimasi fungsi keanggotaan dapat dilakukan dengan menggunakan Algoritma Genetika. Metode tersebut digunakan karena memiliki tingkat kesuksesan yang tinggi dalam memecahkan masalah optimasi di dalam permasalahan ilmu komputer.

Algoritma Genetika mampu menyelesaikan berbagai masalah optimasi dengan efisien dan efektif(Kaya, et al., 2006). Algoritma ini didasari dengan proses genetika yang ada dalam makhluk hidup, yaitu kembang biak suatu individu dalam suatu populasi yang telah dikenakan dengan seleksi alam. Oleh karena itu algoritma diharapkan ini dapat menyelesaikan permasalahan dalam dunia nyata. Sehingga fungsi keanggotaan yang dihasilkan oleh sistem ini dapat memberikan tingkat akurasi yang tinggi dalam sistem.

Sistem pakar ini akan membantu orang-orang yang bekerja di bidang kesehatan baik dokter umum maupun dokter spesialis, para mahasiswa yang sedang melakukan pelatihan dalam kedokteran, serta intansi-instansi lain yang terkait. Hasil dari sistem ini adalah diagnosa penyakit tiroid yang dibagi menjadi beberapa kategori yakni normal, *hypertiroidism* dan *hypotiroidism*.

1.2 Rumusan masalah

Berdasarkan pemaparan latar belakang, maka rumusan masalah yang bisa dikaji adalah sebagai nerikut :

1. Bagaimana implementasi Algoritma Genetika untuk optimasi fungsi keanggotaan FIS Tsukamoto dalam kasus diagnosa penyakit tiroid?

2. Berapa nilai probabilitas *crossover*, probabilitas mutasi, jumlah populasi dan jumlah generasi agar mendapatkan nilai fitness yang optimal pada Algoritma Genetika untuk optimasi fungsi keangotaan FIS Tsukamoto dalam kasus diagnosa penyakit tiroid?
3. Bagaimana akurasi sistem pada aplikasi optimasi fungsi keanggotaan FIS Tsukamoto menggunakan Algoritma Genetika untuk diagnosa penyakit tiroid dibandingkan dengan perhitungan menggunakan FIS Tsukamoto tanpa di optimasi?

1.3 Tujuan

1. Menerapkan metode Algoritma Genetika untuk optimasi fungsi keanggotaan FIS Tsukamoto dalam kasus diagnosa penyakit tiroid.
2. Mengetahui parameter-parameter terbaik Algoritma Genetika berupa nilai probabilitas *crossover*, probabilitas mutasi, jumlah populasi dan jumlah generasi untuk mendapatkan nilai fitness yang optimal pada Lagoritma Genetika untuk optimasi fungsi keangotaan FIS Tsukamoto dalam kasus diagnosa penyakit tiroid.
3. Membandingkan hasil akurasi sistem pada aplikasi optimasi fungsi keanggotaan FIS Tsukamoto menggunakan Algoritma Genetika untuk diagnosa penyakit tiroid terhadap perhitungan menggunakan FIS Tsukamoto tanpa di optimasi.

1.4 Manfaat

1. Dapat memahami cara mengoptimasi fungsi keanggotaan FIS Tsukamoto terutama pada kasus diagnosa penyakit tiroid.
2. Mengetahui parameter terbaik Algoritma Genetika untuk optimasi fungsi keanggotaan FIS Tsukamoto dalam kasus diagnosa penyakit tiroid.
3. Membantu para praktisi dan akademisi kesehatan untuk mendiagnosa penyakit tiroid.

1.5 Batasan masalah

1. Data yang digunakan di dapat dari UCI *Repository*.
2. Hasil diagnosis penyakit berupa normal, *hypothyroid* dan *hyperthyroid*.
3. Perhitungan dalam pembahasan yang menggunakan metode *fuzzy inference system tsukamoto* yang dipotimasi dengan algortima genetika.
4. Kriteria yang digunakan hanya meliputi tes *T3-resin*, tes *thyroxin serum*, tes *triiodotynine*, tes *basal thyroid-stimulating hormone* dan test perbandingan nilai TSH setelah injeksi.

1.6 Sistematika pembahasan

Sistematika pembahasan pada penelitian ini dibagi dalam tujuh bab, masing-masing diuraikan sebagai berikut:

BAB I Pendahuluan

Membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan yang digunakan untuk merancang penelitian dalam pembuatan sistem diagnosa penyakit tiroid menggunakan metode FIS Tsukamoto.

BAB II Tinjauan Pustaka

Menjelaskan tentang kajian pustaka terkait dengan penelitian yang telah ada seperti penelitian tentang FIS Tsukamoto dan penyakit tiroid. Dasar teori yang dipergunakan untuk penelitian ini adalah sistam pakar, metode FIS Tsukamoto, definisi penyakit tiroid dan Algoritma Genetika.

BAB III Metodologi Penelitian

Membahas tentang metode yang digunakan untuk perancangan aplikasi optimasi fungsi keanggotaan FIS Tsukamoto menggunakan Algoritma Genetika untuk diagnosa penyakit tiroid.

BAB IV Perancangan Sistem

Membahas tentang bagaimana rancangan aplikasi agar menghasilkan diagnosa yang benar sesuai dengan keadaan pasien, yaitu *hypothyroid, hyperthyroid* atau normal.

BAB V Implementasi Program

Membahas tentang implementasi aplikasi sistem pakar agar dapat mendiagnosa penyakit tiroid seorang pasien secara tepat yang didapatkan dari masukan berupa 5 kriteria yaitu tes-tes yang dibutuhkan dengan menggunakan metode FIS Tsukamoto yang fungsi keanggotaanya dioptimasi dengan Algoritma Genetika.

BAB VI Pengujian dan Anlisis

Membahas tentang hasil pengujian akurasi pada pakar, dan analisis terhadap sistem pakar yang telah direalisasikan dan telah memenuhi kriteria sehingga dapat menentukan keadaan seorang pasien secara tepat dengan membandingkan hasil uji pada data set.

BAB VII Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian sistem pakar dalam hal diagnosa penyakit tiroid pada seorang pasien.

BAB 2 TINJAUAN PUSTAKA

Tinjauan pustaka ini akan membahas tentang objek penelitian yang ada dan objek yang akan digunakan, serta teori yang dibutuhkan dalam penyusunan skrispi antara lain sistem pakar, FIS Tsukamoto dan Algoritma Genetika.

2.1 Kajian Pustaka

Dalam menyusun kajian pustaka ini terdapat jurnal yang memiliki judul dan tujuan yang mirip dengan penelitian ini. Terdapat dua jurnal yang digunakan dalam kajian pustaka ini karena kedua jurnal tersebut dinilai paling sesuai dengan tujuan penelitian ini.

Jurnal pertama yaitu diagnosa penyakit tiroid menggunakan sistem pakar(Keles dan Keles, 2008). Sistem ini digunakan untuk mendeteksi penyakit tiroid secara dini dengan menggunakan metode *fuzzy neuro*. Data set yang digunakan berasal dari UCI *Repository* sebanyak 215 data, inputannya berupa hasil nilai tes dari 5 kriteria dan keluarannya berupa 3 kategori yaitu normal, *hypothyroid* dan *hyperthyroid*. Akurasi yang dihasilkan oleh sistem tersebut adalah sebesar 95,33%.

Penelitian kedua adalah sistem diagnosa penyakit tiroid pula (Dogantekin, et al., 2010). Dalam pembuatan sistem ini peneliti menggunakan 3 metode yaitu *Principle Component Analysis* (PCA), klasifikasi *Least Square Machine Support Machine* (LS-SVM) dan klasifikasi akurasi, *k-fold cross validation* dan *confusion matrix*. Metode-metode tersebut digunakan secara berurutan yang pertama metode PCA digunakan sebagai fitur pengurangan variabel, yang kedua metode LS-SVM digunaakan untuk klasifikasi dan yang terakhir klasifikasi akurasi, *k-fold cross validation* dan *confusion matrix* sebagai evaluasi performa pada sistem diagnosis penyakit tiroid. Sistem ini memiliki akurasi sebesar 97,67%.

Sari dan Mahmudy (2015) melakukan penelitian yang menggunakan metode FIS Tsukamoto digunakan untuk menentukan kelayakan calon pegawai. Hasil dari pengujian pada penelitian ini adalah sebuah perangkingan. Untuk menguji keakurasian sistem digunakan uji korelasi non parametrik spearman. Uji korelasinya menghasilkan keakuratan sebesar 0,952.

Penelitian selanjutnya adalah memilih calon pemilihan penerima beasiswa dan BBP-PPA pada PTIIK Universitas Brawijaya (Restuputri, et al., 2015) yang menggunakan metode serta cara optimasi yang sama dengan penelitian ini yaitu optimasi fungsi keanggotaan FIS Tsukamoto dengan Algoritma Genetika. Parameter yang digunakan terdapat 8 yaitu IPK, Penghasilan orang tua, tanggungan orang tua, tagihan telepon, tagihan listrik, tagihan PDAM, pembayaran PBB. Hasil keputusannya adalah peserta didik layak atau tidak untuk mendapatkan beasiswa. Untuk representasi kromosomnya terdapat gen yang mewakili ke-delapan kriteria tersebut dan gen yang mewakili hasil keputusan. Gen-gen pada kromosom tersebut digunakan sebagai fungsi keanggotaan FIS Tsukamoto. Akurasi pada sistem pemilihan calon penerima beasiswa ini adalah 98.9% untuk Beasiswa-PPA dan BBP-PPA sebesar 98.7%.

2.2 Penyakit Tiroid

Kelenjar tiroid adalah kelenjar terbesar di tenggorokan. Terletak pada atas tenggorokan dan dibawah lapisan otot dan kulit, kelenjar tiroid berbentuk seperti kupu-kupu dengan 2 sayap yang direpresentasikan oleh lobus kanan dan kiri yang meyelimuti sekitar trakhea.(Chen, 2011).

Kelenjar tiroid adalah salah satu dari kelenjar endokrin. kelenjar ini membentuk hormon yang berguna sebagai regulasi fungsi fisiologis pada tubuh. Salah satu dari hormon itu adalah hormon tiroid, ini meregulasi laju dimana tubuh membawa fungsi pentingnya. kelenjar tiroid dapat dihinggapai oleh beberapa penyakit yang paling banyak adalah *goiters, solitary thyroid nodules*, *hyperthyroidism, hypotyroidism, thyroiditis* dan lain-lain. (Dogantekin, et al., 2011).

Kebenaran diagnosa penyakit tiroid sangatlah penting karena telah banyak manusia yang menderita penyakit tiroid. Dari hasil yang ada fungsi kelenjar tiroid dapat mempengaruhi kinerja organ penting dalam tubuh (Savelonas, et al., 2011).

Hormon tiroid diproduksi oleh kelenjar tiroid untuk membantu mengontrol metobolisme tubuh. Kelenjar tiroid menghasilkan dua hormon tiroid aktif, levothyroxine (T4) dan triiodothyronine (*T3*). Hormon-hormon ini penting dalam produksi protein, dalam regulasi suhu tubuh, dalam produksi energi secara keseluruhan dan regulasi. Keseriusan gangguan tiroid tidak boleh dianggap remeh seperti penyakit *Hyperthyroidism* dan *myexdema* koma (tahap akhir *Hyperthyroidism* yang tidak bisa diobati) dapat menyebabkan kematian dalam banyak kasus(Temurtas, 2009). *Hyperthyroidism* terjadi ketika tiroid menghasilkan terlalu banyak hormon, yaitu tubuh menggunakan energi lebih cepat dari seharusnya. Sementara *hyporthyroidism* terjadi dibawah kondisi ketika tiroid tidak menghasilkan cukup hormon, yang berarti tubuh menggunakan energi lebih lambat dari yang seharusnya (Chen, et al., 2011).

Ada berbagai alasan mengapa salah satu dari kondisi ini bisa berkembang. Sekarang dikatakan bahwa sekitar 20 juta orang Amerika memiliki beberapa bentuk penyakit tiroid(Chen dkk, 2011).

Penyakit tiroid sulit untuk didiagnosa karena gejalanya mirip dengan kondisi yang lain. Ketika penyakit tiroid terdeteksi dini, pengobatan dapat mengendalikan gangguan yang terjadi bahkan sebelum timbulnya gejala. Jumlah hormon tiroid dalam aliran darah dipantau dan dikendalikan oleh kelenjar hiposfis ketika kelenjar pituitari yang terletak ditengah tengkorak dibawah otak baik kurangnya atau tinggi tingkat hormon tiroid itu akan disesuaikan dan mengirimkannya ke tiroid untuk memberitahu apa yang harus dilakukan (Polat & Gunes, 2006).

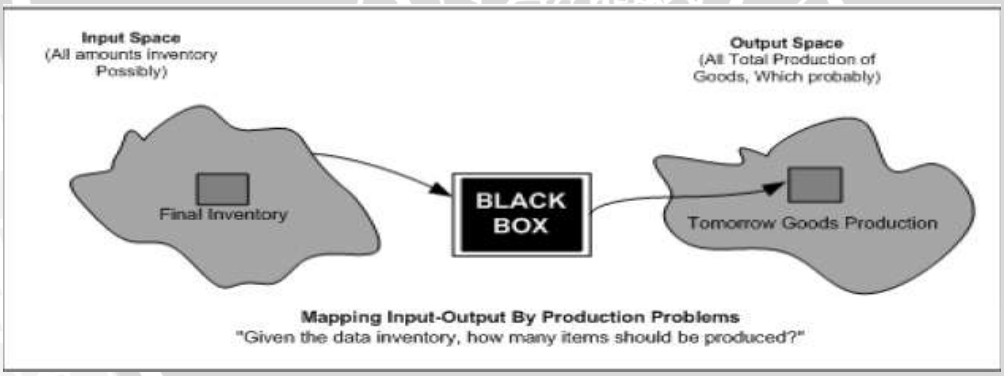
Interpretasi pada data tiroid disamping tes klinik dan pemeriksaan pelengkapnya sudah menjadi hal yang penting untuk mendiagnosa penyakit tiroid. Macam-macam metode baru seperti teknik pengenalan pola, klasifikasi *fuzzy* dan lainnya telah digunakan untuk menganalisa pasien sesuai dengan status definisi yang tepat (Polast dan Gunes, 2006).

Dalam mendiagnosa penyakit tiroid terdapat 5 parameter masukan berupa tes-tes klinik yang dibutuhkan. Kelima kriteria tersebut adalah tes *t3-resin*, total serum *thyroxin*, total serum *triiodothyronine*, *basal thyroid-stimulating hormone* (TSH) dan maksimal perbedaan dari nilai TSH.

2.3 Fuzzy Inference System Tsukmoto

Logika *fuzzy* merupakan suatu alternatif untuk menyampaikan suatu data. Logika *fuzzy* merupakan logika yang mendeskipsikan data atau pengetahuan yang masih tidak pasti, logika ini didasarkan pada logika boolen yang umum digunakan dalam komputasi. Dalam logika *fuzzy* suatu proposisi dapat direpresentasikan dalam derajad kebenaran(Kaya, et al., 2006).

Logika *fuzzy* adalah daerah *soft computing* yang memungkinkan sistem komputer untuk memberi alasan akan ketidakpastian. Logika fuzzy pertama kali diperkenalkan oleh Lotfi A. Zadeh pada tahun 1965. Dalam teori himpunan *fuzzy*, peran derajat keanggotaan sebagai penentu adanya unsur dalam satu set sangat penting. Derajat fungsi keanggotaan sebagai karakteristik utama dari *fuzzy logic* penalaran. Dalam banyak hal, logika *fuzzy* digunakan sebagai cara untuk memetakan masalah input menuju keluaran yang diharapkan. Misalnya, manajer gudang mengatakan kepada manajer produksi berapa banyak persediaan pada akhir pekan ini, maka manajer produksi akan menetapkan jumlah barang yang harus diproduksi besok. Salah satu contoh dari pemetaan masukan dan keluaran dalam bentuk grafik seperti yang ditunjukkan pada gambar berikut:
(Ula, 2014) (Bon & Utami, 2014)



Gambar 2.1 Mapping masukan dan keluaran
Sumber: (Bon & Utami, 2014)

Metode Tsukamoto pertama kali diperkenalkan oleh Tsukamoto pada tahun 1979, yang merupakan salah satu metode pengambilan keputusan. Metode ini berlaku untuk penggunan penalaran aturan *monoton*, maksudnya adalah untuk menggunakan sistem dengan hanya satu aturan. Implikasi dari setiap aturan dalam bentuk implikasi "Sebab Akibat" atau Implikasi "Masukan-keluaran" di mana yang masukan dan keluaran harus berhubungan secara konsekuen. Setiap aturan diwakili menggunakan asosiasi *fuzzy*, dengan fungsi keanggotaan yang monoton. Kemudian, untuk menentukan hasil dari sebuah perusahaan (*Crisp Solution*) digunakan dengan rumus pernyataan (defuzzifikasi)

disebut "berpusat metode rata-rata". dalam inferensinya metode tsukamoto menggunakan tahapan sebagai berikut (Maryaningsih, et al., 2013):

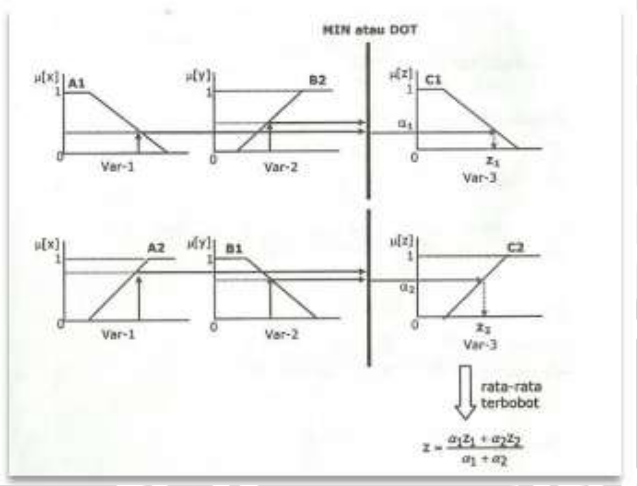
1. Fuzzifikasi.
2. Pembentukan basis pengetahuan *fuzzy* (*Rule* dalam bentuk *IF…. Then*).
3. Mesin inferensi menggunakan fungsi implikasi MIN untuk mendapatkan nilai α-predikat tiap-tiap rule (α1, α2, α3, …… αn). Kemudian nilai α-predikat ini digunakan untuk menghitung keluaran hasil inferensi secara tegas (*crisp*) masing-masing *rule* (z1, z2, z3, ….. zn).
4. Defuzzifikasi.

Menggunakan metode rata-rata (average), formulasi ini adalah untuk menentukan nilai keluaran bilangan *crisp* yang akan menjadi jumlah barang yang diproduksi (Z), dengan mengubah masukan (dalam bentuk *fuzzy set* yang berasal dari komposisi aturan *fuzzy*) menjadi beberapa *fuzzy set* dalam domain. Dengan keterangan Z adalah variabel output, αi adalah nilai α predikat dan zi adalah nilai variabel keluaran. Nilai keanggotaan (α) akan dicari pada setiap aturan dan jika aturan lebih dari satu maka akan dilakukan agregasi semua aturan. Selanjutnya nilai agregasi akan dilakukan *defuzzy* untuk mendapatkan nilai *crisp* sebagai keluaran(Siti dan Soraya, 2014).Berikut persamaannya:

$$Z = \frac{\sum_{i=1}^{n} aizi}{\sum_{i=1}^{n} ai} \qquad (2.1)$$

Sistem inferensi *fuzzy* metode Tsukamoto didasarkan pada konsep penalaran monoton. Karena menggunakan konsep dasar penalaran monoton, pada metode Tsukamoto setiap konsekuensi pada aturan berbentuk jika harus direpresentasikan dengan suatu himpunan fuzzy dengan fungsi keanggotaan yang monoton. Keluaran hasil inferensi dari tiap-tiap aturan diberikan secara tegas berdasarkan α-predikat (*fire strength*) (Gunawan & Rouf, 2013).

Pada metode Tsukamoto, setiap konsekuen pada aturan yang berbentuk *IF-Then* direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot.(Farouq & Miftahus, 2014). Proses pengambilan keputusan merupakan salah satu proses yang penting untuk memecahkan permasalahan yang sedang dihadapi dengan memilih satu dari beberapa alternatif penyelesaian yang dimungkinkan. Dengan menggunakan metode logika Fuzzy Tsukamoto maka akan lebih mudah dalam pembobotan dan pengambilan keputusan (Kumar & Jain, 2012).



Gambar 2.2 Sistem inferensi fuzzy metode tsukamoto

Metode Tsukamoto menggunakan metode deffuzifikasi (penegasan) untuk fungsi keanggotaan monoton secara khusus disebut metode deffuzifikasi dengan rata-rata terpusat (*Center Average Deffuzyfieer*). Ada dua aturan yang digunakan yaitu: (Kodaz, 2009)

[R1] IF (x is A1) and (y is B2) THEN (z is C1) (0.1)

[R2] IF (x is A2) and (y is b1) THEN (z is C2) (0.2)

2.4 Algoritma Genetika

Dalam menentukan suatu solusi dari permasalahan terkadang dibutuhkan formulasi matematika yang kompleks agar mendapatkan solusi yang tepat. Untuk mendapatkan solusi yang optimum pun juga membutuhkan perhitungan yang panjang (Widodo&Mahmudy, 2010). Oleh karena itu dalam mengatasi hal ini dapat digunakan suatu motode heuristik, yaitu metode pencarian yang didasarkan atas intuisi atau aturan-aturan empiris untuk memperoleh solusi yang lebih baik daripada solusi yang telah dicapai sebelumnya (Taha, 2002). Metode heuristik tidak selalu meghasilkan solusi terbaik namun jika dirancang dengan baik akan menghasilkan solusi yang optimum dalam waktu singkat.

Algortima genetika merupakan suatu teknik optimasi berdasarkan genetika alami. Teori evolusi digunakan oleh Algoritma Genetika yaitu suatu individu akan terus-menerus mengalami perubahan gen untuk menyesuaikan dengan lingkungan hidupnya. Konsep algoritma ini pertama kali dikemukakan oleh Joh Holland dan rekannya daru Universitas Michigan (Nugraha, 2008). Tujuan Holland ketika itu mengarah ke studi mengenai fenomena adaptasi di alam dan mencoba menerapkan mekanisme adaptasi alam tersebut ke dalam sistem komputer.

Cara kerja Algoritma Genetika yaitu dengan menggabungkan secara acak berbagai kemungkinan solusi terbaik yang dianalogikan dalam bentuk kromosom, yang mana setiap kromosom memiliki fungsi objektif yang sesuai dengan parameter masalah(Rosyidah, 2015). Fungsi objektif dalam algortima genetika biasa disebut dengan nilai *fitness* (*fitness value*), semakin tinggi nilai

fitness maka kromosom tersebut lebih baik. Pada Algoritma Genetika satu tahapan iterasi, pada setiap generasi mengalami proses evaluasi untuk menetukan populasi baru pada generasi selanjutnya (Berlianty, 2010). Generasi selanjutnya ini menunjukkan populasi yang baru dan lebih baik dari sebelumnya. Dengan melakukan cara tersebut secara berulang-ulang Algoritma Genetika diharapkan bisa menemukan solusi yang lebih baik dari generasi sebelumnya.

Apabila dibandingkan dengan prosedur pencarian dan optimasi biasa, Algoritma Genetika berbeda dalam beberapa hal sebagai berikut :

- Manipulasi dilakukan terhadap kode dari himpunan parameter (biasa disebut kromosom), tidak secara langsung terhadap parameternya sendiri.
- Proses pencarian dilakukan dari beberapa titik dalam satu populasi, tidak dari satu titik saja.
- Proses pencarian menggunakan informasi dari fungsi tujuan.

Pencariannya menggunakan *stochastic operators* yang bersifat probabilistik, tidak menggunakan aturan *deterministic.*

Kelebihan Algoritma Genetika sebagai metode optimasi adalah sebagai berikut:

- Agoritma genetika merupakan algoritma yang berbasis populasi yang memungkinkan digunakan pada optimasi masalah dengan ruang pencarian (*search space*) yang sangat luas dan kompleks. Properti ini juga memungkinkan Algoritma Genetika untuk melompat keluar dari daerah optimum lokal.
- Individu yang ada pada populasi bisa diletakkan pada beberapa sub-populasi yang diproses pada sejumlah komputer secara paralel. Hal ini bisa mengurangi waktu komputasi pada masalah yang sangat kompleks. Penggunaan sub-populasi juga bisa dilakukan pada hanya satu komputer untuk menjaga keragaman populasi & meningkatkan kualitas hasil pencarian.
- Agoritma genetika menghasilkan himpunan solusi optimal yang sangat berguna pada peyelesaian masalah dengan banyak obyektif.
- Agoritma genetika dapat menyelesaikan masalah kompleks dengan banyak variabel (kontinyu, diskrit atau campuran keduanya).
- Agoritma genetika menggunakan kromosom untuk mengkodekan solusi sehingga bisa melakukan pencarian tanpa memperhatikan informasi derivatif yang spesifik dari masalah yang diselesaikan.
- Agoritma genetika bisa diimplementasikan pada bermacam data seperti data yang dibangkitkan secara numerik atau dengan fungsi analitis.
- Agoritma genetika cukup fleksibel untuk dihibridisasikan dengan algoritma lainnya. Beberapa penelitian membuktikan, hybrid

Algoritma Genetika(HGAs) sangat efektif untuk menghasilkan solusi yang lebih baik.

- Agoritma genetika bersifat *ergodic*, sembarang solusi bisa diperoleh dari solusi yang lain dengan hanya beberapa langkah. Hal ini memungkinkan eksplorasi pada daerah pencarian yang sangat luas, yang dapat dilakukan dengan lebih cepat dan mudah.

2.5.1 Struktur Algoritma Genetika

Solusi dari suatu masalah harus dipetakan (*encoding*) menjadi *string* kromosom. *String* kromosom ini tersusun atas sejumlah gen yang menggambarkan variabel-variabel keputusan yang digunakan dalam solusi. Representasi *string* kromosom beserta fungsi *fitness* untuk menilai seberapa bagus sebuah kromosom (untuk menjadi solusi yang layak) dimasukkan ke Algoritma Genetika. Dalam banyak kasus, bagaimana merepresentasikan sebuah solusi menjadi kromosom sangat menentukan kualitas dari solusi yang dihasilkan (Mahmudy, et al., 2012).

Dengan menirukan proses genetika dan seleksi alami maka Algoritma Genetika akan menghasilkan kromosom terbaik setelah melewati sekian generasi. Kromosom terbaik ini harus diuraikan (*decoding*) menjadi sebuah solusi yang diharapkan mendekati optimum.

Dalam siklus perkembangan Algoritma Genetika mencari solusi (kromosom) terbaik terdapat beberapa proses sebagai berikut:

a. Inisialisi

Inisialisasi dilakukan untuk membangkitkan himpunan solusi baru secara acak/*random* yang terdiri atas sejumlah *string* kromosom dan ditempatkan pada peampungan yang disebut populasi. Dalam tahap ini harus dientukan ukuran populasi (*PopSize*). Nilai ini menyatakan banyaknya individu/ kromosom yang ditampung dalam populasi. Panjang setiap *string* kromosom (*StringLen*) dihitung berdasarkan presisi variabel solusi yang kita cari.

b. Reproduksi

Reproduksi dilakukan untuk menghasilkan keturunan dari individu-individu yang ada di populasi.Himpunan keturunan ini ditempatkan dalam penampungan *off-spring*. Dua operator genetika yang digunakan dalam proses ini adalah tukar silang(*crossover*) dan mutasi (*mutation*). Ada banyak metode *crossover* dan mutasi yang telah dikembangkan oleh para peneliti dan biasanya bersifat spesifik terhadap masalah dan representasi kromosom yang digunakan.

Dalam tahap ini harus ditentukan tingkat *cross-over* (*crossover rate*/pc). Nilai ini menyatakan rasio *off-spring* yang dihasilkan proses *crossover* terhadap ukuran populasi sehingga akan dihasilkan *offspring* sebanyak pc x *popSize.*

c. Evaluasi

Evaluasi digunakan untuk menghitung kebugaran (*fitness*) setiap kromosom. Semakin besar *fitness* maka semakin baik kromosom tersebut untuk

dijadikan calon solusi. Kerena sebuah kromosom selalu memiliki nilai *fitness* dan bebrapa properti lain, maka dalam pembahasan berikutnya seringkali digunakan istiah 'individu'. Hal ini bisa dianalogikan dengan seorang manusia sebagai individu. Dia memliki tubuh beserta susunan gen pembentuknya (kromosom), nama, umur, alamat dan properti lainnya.

d. Seleksi

Seleksi dilakukan untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya. Semakin besar nilai *fitness* sebuah kromosom maka semakin besar peluangmya untuk terpilih. Hal ini dilakukan agar terbentuk generasi berikutnya yang lebih baik dari generasi sekarang. Metode seleksi yang sering digunakan adalah *roulette wheel, binary tournament* dan *elitism*.

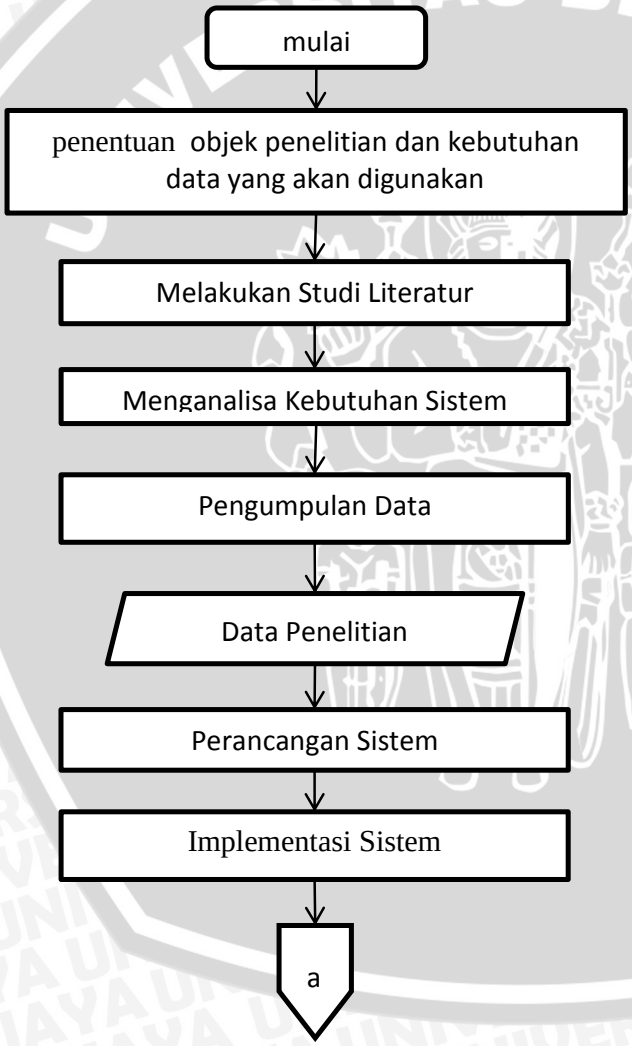
Setelah melewati sekian iterasi (generasi) akan didapatkan individu terbaik. Individu terbaik ini mempunyai susunan kromosom yang bisa dikonversi menjadi solusi yang terbaik (paling tidak mendekati optimum). Dari sini bisa disimpulkan bahwa Algoritma Genetika menghasilkan suatu solusi optimum dengan melakukan pencarian di antara sejumlah alternatif titik optimum berdasarkan fungsi probabilistik (Michalewicz 1996).

BAB 3 METODOLOGI

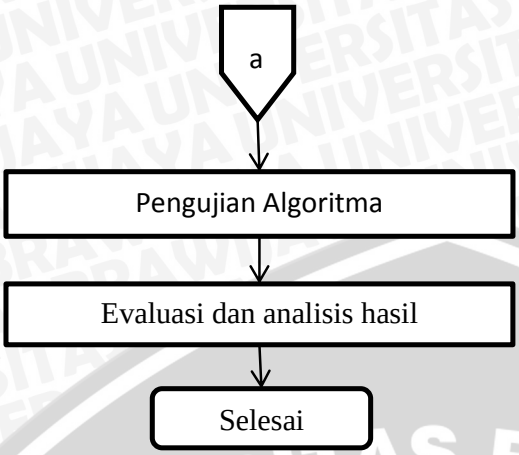
Penelitian ini bisa digolongkan sebagai penelitian implementif dengan pendekatan perancangan (*design*). Selain itu akan menghasilkan sebuah purwarupa (*prototype*) berupa perangkat lunak (*software*) yang bisa digunakan untuk para akademisi maupun praktisi dalam bidang kesehatan agar dapat mendiagnosa seorang pasien dengan tepat tentang konidisi kelenjar tiroid.

3.1 Tahapan Penelitian

Dalam melaksanakan penelitian terdapat beberapa proses dan langkah-langkah yang harus dilakukan diantaranya dapat dilihat pada gambar diagram alir pelaksanaan dibawah ini .



Gambar 3. 1 Diagram Alir Metodologi Pelaksana



Gambar 3.1 Diagram Alir Metodologi Pelaksana

3.2 Teknik Pengumpulan Data

Data yang digunakan dalam penelitian ini berasal dari website UCI *Repository*. Data tersebut berjumlah 215 set yang menunjukkan hasil nilai tes pada kriteria-kriteria yang dibutuhkan dan hasil diagnosanya sesuai yang di alami pasien. 5 kriteria yang dibutuhkan adalah test *T3-resin, test Serum thyroxin, test triiodotynine, test basal thyroid-stimulating hormone* dan test perbandingan nilai *TSH* setelah injeksi.

3.3 Algoritma yang Digunakan

Penelitian ini menggunakan algoritma FIS Tsukamoto yang dioptimasi oleh Algoritma Genetika seperti dijelaskan di Bab 2 yang telah terbukti efektif digunakan untuk pemilihan calon penerima beasiswa dan BBP-PPA (studi kasus: PTIIK Universitas Brawijaya Malang).

Implementasi algoritma menggunakan bahasa pemograman java karena memiliki berbagai fitur dan fungsi untuk mempermudah dalam pembuatan program diagnosa penyakit tiroid ini. Selain itu program juga dapat berjalan dengan cepat jika menggunakan java.

3.4 Kebutuhan Sistem

Agar perangkat lunak dapat berjalan dengan baik maka pengujian perangkat lunak dijalankan pada *personal computer* dengan spesifikasi sebagai berikut:

1. Kebutuhan *hardware*, meliputi:
   - *Personal Computer*
2. Kebutuhan so*ftware,* meliputi:
   - Sistem operasi Windows 7
   - Aplikasi Java Netbeans
   - Bahasa pemrograman java
3. Data yang dibutuhkan
   - Data gejala penyakit tiroid

- Deskripsi penyakit dan pengobatan pada penyakit tiroid

3.5 Pengujian Algoritma

Tahap ini digunakan untuk merancang pengujian algoritma. Pengujian ini digunakan untuk mengetahui tingkat akurasi diagnosis penyakit tiroid menggunakan FIS Tsukamoto yang fungsi keanggotaanya dioptimasi oleh Algoritma Genetika. Pengujian yang akan dilakukan adalah sebagai berikut:

1. Pengujian metode seleksi yang akan digunakan yaitu *elitism, tournament dan roulette wheel selection.*
2. Pengujian dalam mencari kombinasi *crossover rate* dan *mutation rate* yang terbaik.
3. Pengujian dalam menentukan ukuran populasi.
4. Pengujian dalam menentukan ukuran generasi.

3.5.1 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Pengujian ini dilakukan untuk mengetahui metode seleksi terbail untuk kasus dalam penelitian ini. Metode seleksi yang diuji dibatasi dengan hanya 3 metode saja yaitu *elitism, tournament dan roulette wheel selection.* untuk mendapatkan hasil pengujian yang optimum setiap metode dijalankan sebanyak 10 kali dengan parameter uji yang sama. Parameter uji didapatkan dari penelitian sebelumnya yaitu optimasi fungsi keanggotaan pada *FIS Tsukamoto* menggunakan algortima genetika (Restuputri, et al.,2015). Adapun detail parameter yaitu sebanyak 80 kromosom sebagai populasi, generasinya sebanyak 100 kali, nilai cr=0.5 dan mr=0.5. setelah didapatkan hasil *fitness* pada setiap kali pengujian nilai *fitness* di rata-rata pada setiap metode seleksi begitu pula waktu yang digunakan untuk menjalankan aplikasi.

3.5.2 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Pengujian ini digunakan untuk mengetahui kombinasi *crossover rate* dan *mutation rate* yang terbaik untuk menghasilkan solusi yang paling optimal. Pada uji coba ini menggunakan nilai yang berbeda-beda pada parameter Algoritma Genetika *crossover rate* dam *mutation rate*. Nilai yang digunakan adalah antara 0 hingga 1. Sedangkan ukuran populasinya adalah 80 dan ukuran generasi sebesar 100 parameter tersebut didapatkan dari penelitian sebelumnya yang menggunakan metode yang sama yaitu optimasi fungsi keanggotaan pada *FIS Tsukamoto* menggunakan algortima genetika (Restuputri, et al., 2015). Pengujian ini dilakukan mulai dari angka 0 untuk *crossover rate* dan angka 1 untuk *mutation rate* yang diuji coba sebanyak 10 kali percobaan yang kemudian di rata-rata hasil dari 10 percobaan tersebut, yang disusul dengan pengubahan angka *crossover rate* dan *mutation rate* menjadi 0.1 dan 0.9 sebanyak 10 kali pula. Langkah tersebut dilakukan terus dengan mengurangi angka *crossover rate* dengan angka 0.1 dan menambahkan angka pada *mutation rate* dengan angka 0.1 pula dari percobaan sebelumnya hingga mencapai kombinasi angka 1 untuk *crossover rate* dan angka 0 untuk *mutation rate*.

3.5.3 Pengujian Ukuran Populasi

Uji coba ini dilakukan untuk mengetahui berapa ukuran populasi paling optimal untuk mendapatkan solusi terbaik. Pengujian populasi ini menggunakan ukuran populasi mulai dari angka 20 yang diuji sebanyak 10 kali dan hasil *fitness* dari 10 pegujian tersebut di rata-rata. Pengujian seanjutnya dilakukan dengan menambahkan angka populasi sebelumnya dengan angka 20 dan melakukan cara pengujian yang sama yaitu dengan menguji angka populasi sebanyak 10 kali dan merata-rata hasil *fitness*nya per angka populasi yang di pakai. Langkah-langkah tersebut dilakukan hingga mencapai angka populasi sebesar 200 populasi. Untuk parameter cr dan mr menggunakan hasil dari pengujian kombinasi cr dan mr sebelumnya. Sedangkan untuk ukuran generasinya yaitu 100 diambil dari penelitian sebelumnya.

3.5.4 Pengujian Ukuran Generasi

Pengujian ini dilakukan untuk mengetahui berapa ukuran generasi optimal dalam menentukan solusi terbaik untuk fungsi keanggotaan pada *FIS Tsukamoto* agar mencapai akurasi tertinggi pada sistem. Perancangan pengujian banyaknya generasi dilakukan sebanyak 10 angka percobaan untuk jumlah generasi mulai dari 25 hingga 250 yang dilakukan setiap kelipatan angka 25. Setiap percobaan angka generasi dilakukan 10 kali uji coba untuk mendapatkan angka *fitness* rata-rata dari 10 percobaan tersebut. Rata-rata *fitness* terbaik dari setiap percobaan angka untuk jumlah generasi di pakai sebagai solusi optimum dalam menentukan jumlah generasi yang dipakai. Dalam penggunaan parameter lainnya yaitu nilai cr, mr dan ukuran populasi diperoleh dari pengujian sebelumnya.

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

Sub bab ini menjelaskan tentang permasalahan yang akan diselesaikan menggunakan metode FIS Tsukamoto. Mendiagnosa penyakit tiroid merupakan permasalahan yang ada di dunia kesehatan. Diagnosa penyakit ini juga susah ditentukan, karena ciri-ciri yang terjadi pada penderita merupakan hal-hal yang umum terjadi dan tidak ada indikasi di tenggorokan dimana letak kelenjar tiroid berada. Hal ini menyusahkan para praktisi kesehatan untuk mendiagnosa pasien yang mengidap penyakit tiroid. Sistem ini disusun agar dapat menentukan penyakit apa yang terjadi pada penderita apakah mengidap *hypothyroid, hyperthyroid* atau normal. Cara mendapatkan ketiga diagnosa tersebut adalah dengan memasukkan parameter utama yang biasa digunakan untuk menganilis penderita yaitu test *T3-Resin*, tes *thyroxin serum*, tes *triiodothynine serum*, tes *thyroid-stimulating hormone (TSH)* dan perbedaan nilai dari TSH setelah injeksi.

Data yang digunakan di ambil dari UCI *Repository* berupa 215 data dan terdiri dari 6 kolom berupa hasil diagnosis dan kelima parameter yang dibutuhkan. Untuk selanjutnya dapat dilihat pada tabel 4.1 berikut:

Tabel 4. 1 Data set penyakit tiroid dari UCI Respository

No.	Di	a	b	c	d	e
1	1	107	10	2,2	0,9	2,7
2	1	113	9,9	3,1	2	5,9
3	1	127	13	2,4	1,4	0,6
4	1	109	5,3	1,6	1,4	0,6
5	1	105	7,3	1,5	1,5	-0
6	1	105	6,1	2,1	1,5	7
7	1	110	10	1,6	1,6	2,7
8	1	114	9,9	2,4	1,5	5,7
9	1	106	9,4	2,2	1,5	0
10	1	107	13	1,1	0,9	3,1
11	1	106	4,2	1,2	1,6	1,4
12	1	110	11	2,3	0,9	3,3
13	1	116	9,2	2,7	1	4,2
14	1	112	8,1	1,9	3,7	2
15	1	122	9,7	1,6	0,9	2,2
16	1	109	8,4	2,1	1,1	3,6
17	1	111	8,4	1,5	0,8	1,2
18	1	114	6,7	1,5	1	3,5
19	1	119	11	2,1	1,3	1,1
20	1	115	7,1	1,3	1,3	2
21	1	101	7,8	1,2	1	1,7
22	1	103	10	1,3	0,7	0,1

No.	Di	a	b	c	d	e
76	1	109	9,7	1,4	1,1	2,1
77	1	119	13	1,5	1,3	3,6
78	1	101	7,1	1,6	1,5	1,6
79	1	108	10	2,1	1,3	2,4
80	1	117	6,7	2,2	1,8	6,7
81	1	115	15	2,3	2	2
82	1	91	8	1,7	2,1	4,6
83	1	103	8,5	1,8	1,9	1,1
84	1	98	9,1	1,4	1,9	-0
85	1	111	7,8	2	1,8	4,1
86	1	107	13	1,5	2,8	1,7
87	1	119	11	2,3	2,2	1,6
88	1	122	12	2,7	1,7	2,3
89	1	105	8,1	2	1,9	-1
90	1	109	7,6	1,3	2,2	1,9
91	1	105	9,5	1,8	1,6	3,6
92	1	112	5,9	1,7	2	1,3
93	1	112	9,5	2	1,2	0,7
94	1	98	8,6	1,6	1,6	6
95	1	109	12	2,3	1,7	0,8
96	1	114	9,1	2,6	1,5	1,5
97	1	114	11	2,4	2	-0

No.	Di	a	b	c	d	e
151	2	139	16	3,8	1,1	-0
152	2	111	16	2,1	0,9	-0
153	2	113	17	1,8	1	0
154	2	65	25	5,8	1,3	0,2
155	2	88	24	5,5	0,8	0,1
156	2	65	18	10	1,3	0,1
157	2	134	16	4,8	0,6	0,1
158	2	110	20	3,7	0,6	0,2
159	2	67	23	7,4	1,8	-1
160	2	95	11	2,7	1,6	-0
161	2	89	14	4,1	0,5	0,2
162	2	89	24	5,4	0,5	0,1
163	2	88	13	2,7	0,1	0,1
164	2	105	17	1,6	0,3	0,4
165	2	89	20	7,3	1,1	-0
166	2	99	13	3,6	0,7	-0
167	2	80	23	10	0,9	-0
168	2	89	22	7,1	0,7	-0
169	2	99	13	3,1	0,5	-0
170	2	68	15	7,8	0,6	-0
171	2	97	14	3,6	1,5	0,3
172	2	84	22	2,7	1,1	-1

Tabel 4.1 Data set penyakit tiroid dari UCI *Respository*

No.	Di	a	b	c	d	e
23	1	109	10	1,9	0,4	-0
24	1	102	7,6	1,8	2	2,5
25	1	121	10	1,7	1,3	0,1
26	1	100	6,1	2,4	1,8	3,8
27	1	106	9,6	2,4	1	1,3
28	1	116	10	2,2	1,6	0,8
29	1	105	11	2	1	1
30	1	110	10	1,8	1	2,3
31	1	120	8,4	1,1	1,4	1,4
32	1	116	11	2	1,2	2,3
33	1	110	7,8	1,9	2,1	6,4
34	1	90	8,1	1,6	1,4	1,1
35	1	117	12	1,9	1,2	3,9
36	1	117	11	1,4	1,5	2,1
37	1	113	9	2	1,8	1,6
38	1	106	9,4	1,5	0,8	0,5
39	1	130	9,5	1,7	0,4	3,2
40	1	100	11	2,4	0,9	1,9
41	1	121	10	2,4	0,8	3
42	1	110	9,2	1,6	1,5	0,3
43	1	129	12	2,7	1,2	3,5
44	1	121	14	1,5	1,6	0,5
45	1	123	8,1	2,3	1	5,1
46	1	107	8,4	1,8	1,5	0,8
47	1	109	10	1,3	1,8	4,3
48	1	120	6,8	1,9	1,3	1,9
49	1	100	9,5	2,5	1,3	-0
50	1	118	8,1	1,9	1,5	14
51	1	100	11	2,5	0,7	-0
52	1	103	12	1,2	1,3	2,7
53	1	115	8,1	1,7	0,6	2,2
54	1	119	8	2	0,6	3,2
55	1	106	9,4	1,7	0,9	3,1
56	1	114	11	2,1	0,3	1,4
57	1	93	8,9	1,5	0,8	2,7
58	1	120	10	2,1	1,1	1,8
59	1	106	11	1,8	0,9	1
60	1	110	8,7	1,9	1,6	4,4
61	1	103	8,1	1,4	0,5	3,8
98	1	110	8,4	1,4	1	1,9
99	1	120	7,1	1,2	1,5	4,3
100	1	108	11	1,2	1,9	1
101	1	108	8,7	1,2	2,2	2,5
102	1	116	12	1,8	1,9	1,5
103	1	113	12	1,5	1,9	2,9
104	1	105	7	1,5	2,7	4,3
105	1	114	8,4	1,6	1,6	-0
106	1	114	8,1	1,6	1,6	0,5
107	1	105	11	1,1	0,8	1,2
108	1	107	14	1,5	1	1,9
109	1	116	12	1,8	1,4	5,4
110	1	102	9,5	1,4	1,1	1,6
111	1	116	16	0,9	1,3	1,5
112	1	118	11	1,8	1,4	3
113	1	109	8,9	1,7	1	0,9
114	1	110	7	1	1,6	4,3
115	1	104	9,6	1,1	1,3	0,8
116	1	105	8,7	1,5	1,1	1,5
117	1	102	8,5	1,2	1,3	1,4
118	1	112	6,8	1,7	1,4	3,3
119	1	111	8,5	1,6	1,1	3,9
120	1	111	8,5	1,6	1,2	7,7
121	1	103	7,3	1	0,7	0,5
122	1	98	10	1,6	2,3	-1
123	1	117	7,8	2	1	3,9
124	1	111	9,1	1,7	1,2	4,1
125	1	101	6,3	1,5	0,9	2,9
126	1	106	8,9	0,7	1	2,3
127	1	102	8,4	1,5	0,8	2,4
128	1	115	11	0,8	2,1	4,6
129	1	130	10	1,6	0,9	4,6
130	1	101	6,7	1,3	1	5,7
131	1	110	6,3	1	0,8	1
132	1	103	9,5	2,9	1,4	-0
133	1	113	7,8	2	1,1	3
134	1	112	11	1,6	0,9	-0
135	1	118	6,5	1,2	1,2	1,7
136	1	109	9,2	1,8	1,1	4,4
173	2	84	19	4,4	1,1	-0
174	2	98	17	4,3	1,7	0,2
175	2	94	21	1,8	1,4	-1
176	2	99	18	1,9	1,4	0,3
177	2	76	25	4,5	1,2	-0
178	2	110	15	1,9	0,7	-0
179	2	144	22	3,3	1,3	0,6
180	2	105	12	3,3	1,1	0
181	2	88	17	4,9	0,8	0,1
182	2	97	15	1,8	1,2	-0
183	2	106	13	3	1,1	0
184	2	79	19	5,5	0,9	0,3
185	2	92	11	2	0,7	-0
186	3	125	2,1	0,9	17	9,5
187	3	120	6,8	2,1	10	39
188	3	108	3,5	0,6	1,7	1,4
189	3	120	3	2,5	1,2	4,5
190	3	119	3,8	1,1	12	5,7
191	3	141	5,6	1,8	9,2	14
192	3	129	1,5	0,6	13	2,9
193	3	118	3,6	1,5	12	49
194	3	120	1,9	0,7	19	24
195	3	119	0,8	0,7	56	22
196	3	123	5,6	1,1	14	56
197	3	115	6,3	1,2	4,7	14
198	3	126	0,5	0,2	12	8,8
199	3	121	4,7	1,8	11	53
200	3	131	2,7	0,8	9,9	4,7
201	3	134	2	0,5	12	2,2
202	3	141	2,5	1,3	8,5	7,5
203	3	113	5,1	0,7	5,8	20
204	3	136	1,4	0,3	33	8,4
205	3	120	3,4	1,8	7,5	22
206	3	125	3,7	1,1	8,5	26
207	3	123	1,9	0,3	23	22
208	3	112	2,6	0,7	41	19
209	3	134	1,9	0,6	18	8,2
210	3	119	5,1	1,1	7	41
211	3	118	6,5	1,3	1,7	12

Tabel 4.1 Data set penyakit tiroid dari UCI *Respository*

No.	Di	a	b	c	d	e
62	1	101	7,1	2,2	0,8	2,2
63	1	115	10	1,8	1,6	2
64	1	116	10	1,7	1,5	4,3
65	1	117	9,2	1,9	1,5	6,8
66	1	106	6,7	1,5	1,2	3,9
67	1	118	11	2,1	0,7	3,5
68	1	97	7,8	1,3	1,2	0,9
69	1	113	11	1,7	0,8	2,3
70	1	104	6,3	2	1,2	4
71	1	96	9,4	1,5	1	3,1
72	1	120	12	2,4	0,8	1,9
73	1	133	9,7	2,9	0,8	1,9
74	1	126	9,4	2,3	1	4
75	1	113	8,5	1,8	0,8	05

No.	Di	a	b	c	d	e
137	1	116	7,8	1,4	1,1	3,7
138	1	127	7,7	1,8	1,9	6,4
139	1	108	6,5	1	0,9	1,5
140	1	108	7,1	1,3	1,6	2,2
141	1	105	5,7	1	0,9	0,9
142	1	98	5,7	0,4	1,3	2,8
143	1	112	6,5	1,2	1,2	2
144	1	118	12	1,5	1	2,3
145	1	94	7,5	1,2	1,3	4,4
146	1	126	10	1,7	1,2	3,5
147	1	114	7,5	1,1	1,6	4,4
148	1	111	12	2,3	0,9	3,8
149	1	104	6,1	1,8	0,5	0,8
150	1	102	6,6	1,2	1,4	1,3

No.	Di	a	b	c	d	e
212	3	139	4,2	0,7	4,3	6,3
213	3	103	5,1	1,4	1,2	5
214	3	97	4,7	1,1	2,1	13
215	3	102	5,3	1,4	1,3	6,7

Keterangan :

- Di, merupakan hasil diagnosis, untuk angka 1 menunjukkan normal, angka 2 menunjukkan hyperthyroid dan angka 3 menunjukkan hypothyroid.
- Huruf a, Merupakan nilai tes dari *T3-resin.*
- Huruf b, Merupakan nilai dari Tes *thyroxin serum.*
- Huruf c, Merupakan nilai tes *triiodothynine serum.*
- Huruf d, Merupakan nilai tes *thyroid-stimulating hormone (TSH).*
- Huruf e, Merupakan nilai tes perbandingan nilai TSH setelah injeksi.

4.1.1 Diagnosa Penyakit Tiroid

Setelah mengetahui data langkah selanjutnya adalah melakukan perhitungan tsukamoto dari contoh permasalahan yang ada. Langkah-langkah yang ada di FIS Tsukamoto adalah fuzzifikasi, pebentukan basis aturan, mesin inferensi dan defuzzifikasi. Contoh permasalahan diberikan ketika terdapat nilai tes dari kelima parameter yang ada untuk di diagnosa hasilnya menggunakan FIS Tsukamoto. Contoh permasalahnnya dapat dlihat pada tabel 4.2 berikut ini :

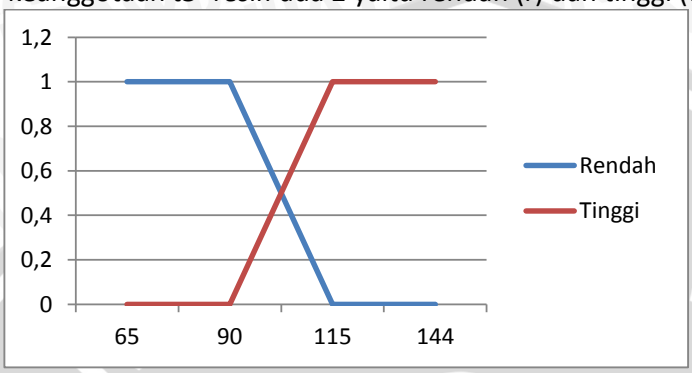
Tabel 4. 2 Nilai input pasien

Kriteria	Nilai
T3- Resin	100
Serum Thyroxin	9
Serum Triiodothyronine	7
Basal Thyroid-stimulatin hormone (TSH)	3
Nilai TSH setelah injeksi	1

1. Fuzzifikasi

Fuzzifikasi yaitu menjadikan bilanga *crisp* menjadi bilangan *fuzzy* yang digunakan dalam perhitungan. Caranya yaitu dengan menyusun fungsi keanggotaan setiap parameter dan diagnosanya. Fungsi keanggotaan setiap parameter dan diagnosa hasilnya dapat dilihat pada Gambar 4.2-4.8.

a. Fungsi keanggotaan *t3- resin* ada 2 yaitu rendah (r) dan tinggi (t).

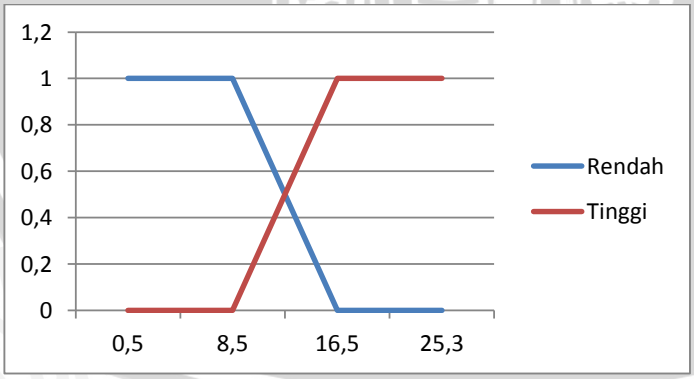


Gambar 4. 1 Diagram fungsi keanggotaan t3-resin

$$\mu_r(a) = \begin{cases} 1 & ; \quad 65 \le a \le 90 \\ \dfrac{115 - a}{25} & ; \quad 90 < a < 115 \\ 0 & ; \quad a \ge 115 \end{cases}$$

$$\mu_t(a) = \begin{cases} 0 & ; \quad a \le 90 \\ \dfrac{a - 90}{25} & ; \quad 90 < a < 115 \\ 1 & ; \quad 115 \le a \le 144 \end{cases}$$

b. Fungsi keanggotaan *thyroxin serum* ada 2 yaitu rendah (r) dan tinggi (t).

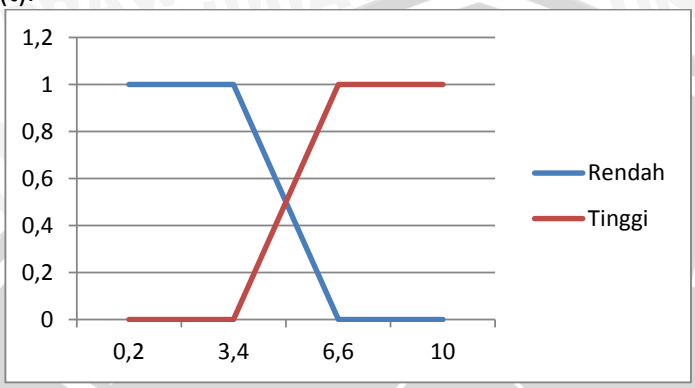


Gambar 4. 2 Diagram fungsi keanggotaan *thyroxin serum*

$$\mu_r(b) = \begin{cases} 1 & ; \quad 0.5 \le b \le 8.5 \\ \dfrac{16.5 - b}{8} & ; \quad 8.5 < b < 16.5 \\ 0 & ; \quad b \ge 16.5 \end{cases}$$

$$\mu_t(b) = \begin{cases} 0 & ; \quad b \leq 8.5 \\ \dfrac{b - 8.5}{8} & ; \quad 8.5 < b < 16.5 \\ 1 & ; \quad 16.5 \leq b \leq 25.3 \end{cases}$$

c. Fungsi keanggotaan *triiodothyronine serum* ada 2 yaitu rendah (r) dan tinggi (t).

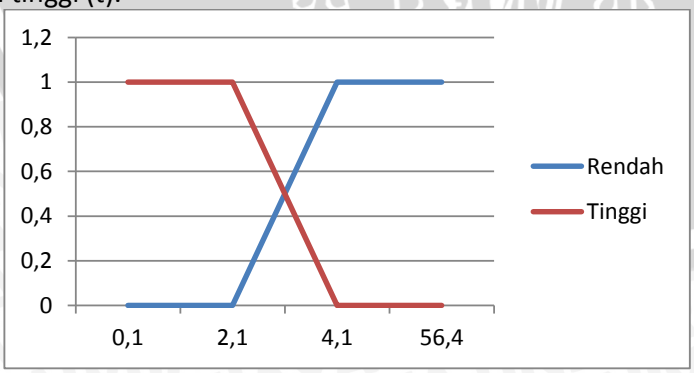


Gambar 4. 3 Diagram fungsi keanggotaan triiodothyronine serum

$$\mu_r(c) = \begin{cases} 1 & ; \quad 0.2 \leq c \leq 3.4 \\ \dfrac{6.6 - c}{3.2} & ; \quad 3.4 < c < 6.6 \\ 0 & ; \quad c \geq 6.6 \end{cases}$$

$$\mu_t(c) = \begin{cases} 0 & ; \quad c \leq 3.4 \\ \dfrac{c - 3.4}{3.2} & ; \quad 3.4 < c < 6.6 \\ 1 & ; \quad 6.6 \leq c \leq 10 \end{cases}$$

d. Fungsi keanggotaan *thyroid-stimulating hormone* (TSH) ada 2 yaitu rendah (r) dan tinggi (t).

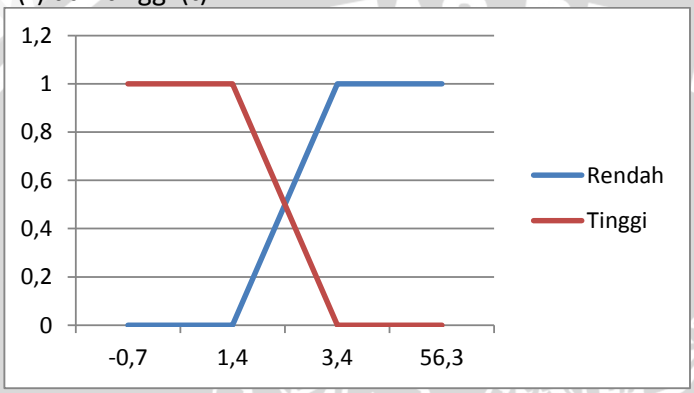


Gambar 4. 4 Diagram fungsi keanggotaan thyroid-stimulating hormone (TSH)

$$\mu_r(d) = \begin{cases} 1 & ; \quad 0.1 \le d \le 2.1 \\ \dfrac{4.1 - d}{2} & ; \quad 2.1 < d < 4.1 \\ 0 & ; \quad d \ge 4.1 \end{cases}$$

$$\mu_t(d) = \begin{cases} 0 & ; \quad d \le 2.1 \\ \dfrac{d - 2.1}{2} & ; \quad 2.1 < d < 4.1 \\ 1 & ; \quad 4.1 \le d \le 56.4 \end{cases}$$

e. Fungsi keanggotaan perbandingan nilai (TSH) setelah injeksi ada 2 yaitu rendah (r) dan tinggi (t).



Gambar 4. 5 Diagram fungsi keanggotaan perbandingan nilai (TSH) setelah injeksi

$$\mu_r(e) = \begin{cases} 1 & ; \quad -0.7 \le e \le 1.4 \\ \dfrac{3.4 - e}{2} & ; \quad 1.4 < e < 3.4 \\ 0 & ; \quad e \ge 3.4 \end{cases}$$

$$\mu_t(e) = \begin{cases} 0 & ; \quad e \le 1.4 \\ \dfrac{e - 1.4}{2} & ; \quad 1.4 < e < 3.4 \\ 1 & ; \quad 3.4 \le e \le 56.3 \end{cases}$$

Setelah ditentukan fungsi keanggotaan masing-masing kriteria maka langkah selanjutnya adalah menentukan nilai derajat keanggotaan untuk tinggi dan rendah dari kasus yang di gunakan. Nilai derajat keanggotaan pada kasus dapat dilihat pada Tabel 4.3 di bawah ini.

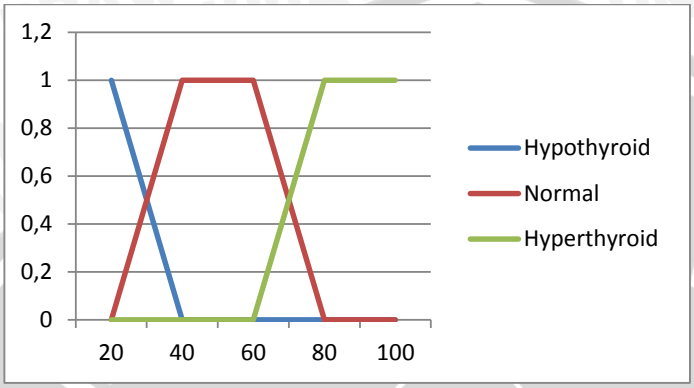
Tabel 4. 3 Derajat keanggotaan

Kriteria	Rendah	Tinggi
Serum thyroxin	0.6	0.4
Serum Triiodothyronine	0.9375	0.0625
Serum Triiodothyronine	1	0

Tabel 4.3 Derajat keanggotaan

Kriteria	Rendah	Tinggi
Basal Thyroid-stimulatin hormone (TSH)	0	1
Nilai TSH setelah injeksi	1	0

f. Fungsi keanggotaan hasil diagnosa



Gambar 4. 6 Diagram fungsi keanggotaan hasil

$$\mu_{Hypo}(e) = \begin{cases} 1 & ; \quad P \leq 20 \\ \dfrac{40-P}{20} & ; \quad 20<P<40 \\ 0 & ; \quad P \geq 40 \end{cases}$$

$$\mu_{Normal}(e) = \begin{cases} \dfrac{P-20}{20} & ; \quad 20 < P < 40 \\ 1 & ; \quad 40 \leq P \leq 60 \\ \dfrac{80-P}{20} & ; \quad 60 \leq P \leq 80 \\ 0 & ; \quad P< 20 \text{ atau } p>80 \end{cases}$$

$$\mu_{Hyper}(e) = \begin{cases} 0 & ; \quad P \leq 60 \\ \dfrac{P-60}{20} & ; \quad 60<P<80 \\ 1 & ; \quad 80 \leq P \leq 100 \end{cases}$$

Nilai batasan pada fungsi keanggotaan untuk masing-masing parameter dan hasilnya didapatkan dari seorang pakar. Namun karena terbatasnya ketersedian seorang pakar yang dapat menentukan nilai batasan secara tepat maka nilai batasan tersebut

dapat dicari dengan menggunakan metode Algoritma Genetika. Seperti yang akan dibahas dalam penelitian ini yaitu mengoptimasi fungsi keanggotaan FIS Tsukamoto dengan menggunakan Algoritma Genetika.

2. Basis Pengetahuan

Basis pengetahuan berisi tentang pengetahuan yang berisi fakta, pemikiran, teori maupun prosedur untuk merumuskan dan memecahkan suatu permasalahan. Basis pengetahuan tersebut terdiri dari dua bentuk pendekatan yaitu pandekatan bersasis aturan yang direpresentasikan dalam bentuk fakta dan pendekatan berbasis kasus tentang solusi yang dipakai sebelumnya dan di turunkan berdasarkan keadaan yang terjadi sekarang.

Tabel 4. 4 Aturan

Rule	*IF*					*THEN*
	t3-resin	*thyroxin serum*	*triiodothyronine serum*	basal (TSH)	TSH setelah injeksi	Diagnosa
1	tinggi	tinggi	tinggi	tinggi	tinggi	*Hypothyroid*
2	tinggi	tinggi	tinggi	tinggi	rendah	*Hyperthyroid*
3	tinggi	tinggi	tinggi	rendah	tinggi	*Hyperthyroid*
4	tinggi	tinggi	tinggi	rendah	rendah	*Hyperthyroid*
5	tinggi	tinggi	rendah	tinggi	tinggi	*Hypothyroid*
6	tinggi	tinggi	rendah	tinggi	rendah	Normal
7	tinggi	tinggi	rendah	rendah	tinggi	Normal
8	tinggi	tinggi	rendah	rendah	rendah	Normal
9	tinggi	rendah	tinggi	tinggi	tinggi	*Hypothyroid*
10	tinggi	rendah	tinggi	tinggi	rendah	Normal
11	tinggi	rendah	tinggi	rendah	tinggi	Hypothyroid
12	tinggi	rendah	tinggi	rendah	rendah	Normal
13	tinggi	rendah	rendah	tinggi	tinggi	Hypothyroid
14	tinggi	rendah	rendah	tinggi	rendah	Hypothyroid
15	tinggi	rendah	rendah	rendah	tinggi	Hypothyroid
16	tinggi	rendah	rendah	rendah	rendah	Normal
17	rendah	tinggi	tinggi	tinggi	tinggi	Normal
18	rendah	tinggi	tinggi	tinggi	rendah	*Hyperthyroid*
19	rendah	tinggi	tinggi	rendah	tinggi	*Hyperthyroid*
20	rendah	tinggi	tinggi	rendah	rendah	*Hyperthyroid*
21	rendah	tinggi	rendah	tinggi	tinggi	*Hypothyroid*

Tabel 4.4 Aturan

Rule	*IF*					THEN
	t3-resin	*thyroxin serum*	*triiodothyronine serum*	basal (TSH)	TSH setelah injeksi	Diagnosa
22	rendah	tinggi	rendah	tinggi	rendah	Normal
23	rendah	tinggi	rendah	rendah	tinggi	Normal
24	rendah	tinggi	rendah	rendah	rendah	*Hypothyroid*
25	rendah	rendah	tinggi	tinggi	tinggi	Normal
26	rendah	rendah	tinggi	tinggi	rendah	Normal
27	rendah	rendah	tinggi	rendah	tinggi	Normal
28	rendah	rendah	tinggi	rendah	rendah	Normal
29	rendah	rendah	rendah	tinggi	tinggi	*Hypothyroid*
30	rendah	rendah	rendah	tinggi	rendah	Normal
31	rendah	rendah	rendah	rendah	tinggi	Normal
32	rendah	rendah	rendah	rendah	rendah	Normal

Pada data basis pengetahuan pada Tabel 4.4 dapat terjadi perubahan sewaktu-waktu sesuai dengan keadaan yang berkembang saat ini. Sehingga pada basis pengetahuan pada Tabel 4.4 dibuat secara dinamis dapat terjadi penambahan aturan baru maupun perubahan gejala pada aturan yang ada.

Tabel 4. 5 Nilai minimal derajat anggota pasien

Rule	*t3-resin*	*Thyroxin serum*	*Triiodothyronine serum*	Basal (TSH)	TSH setelah injeksi
1	0,4	0,0625	0	1	0
2	0,4	0,0625	0	1	1
3	0,4	0,0625	0	0	0
4	0,4	0,0625	0	0	1
5	0,4	0,0625	1	1	0
6	0,4	0,0625	1	1	1
7	0,4	0,0625	1	0	0
8	0,4	0,0625	1	0	1
9	0,4	0,9375	0	1	0
10	0,4	0,9375	0	1	1
11	0,4	0,9375	0	0	0
12	0,4	0,9375	0	0	1

Tabel 4.5 Nilai minimal derajat anggota pasien

Rule	*t3-resin*	*Thyroxin serum*	*Triiodothyronine serum*	Basal (TSH)	TSH setelah injeksi
13	0,4	0,9375	1	1	0
14	0,4	0,9375	1	1	1
15	0,4	0,9375	1	0	0
16	0,4	0,9375	1	0	1
17	0,6	0,0625	0	1	0
18	0,6	0,0625	0	1	1
19	0,6	0,0625	0	0	0
20	0,6	0,0625	0	0	1
21	0,6	0,0625	1	1	0
22	0,6	0,0625	1	1	1
23	0,6	0,0625	1	0	0
24	0,6	0,0625	1	0	1
25	0,6	0,9375	0	1	0
26	0,6	0,9375	0	1	1
27	0,6	0,9375	0	0	0
28	0,6	0,9375	0	0	1
29	0,6	0,9375	1	1	0
30	0,6	0,9375	1	1	1
31	0,6	0,9375	1	0	0
32	0,6	0,9375	1	0	1

3. Mesin Inferensi

Tabel 4. 6 Perhitungan az

Rule	Min(α)	zi	az
1	0	40	0
2	0	60	0
3	0	60	0
4	0	60	0
5	0	40	0
6	0,0625	78,75	4,921875
7	0	80	0
8	0	80	0
9	0	40	0

Tabel 4.6 Perhitungan az

Rule	Min(α)	zi	az
10	0	80	0
11	0	40	0
12	0	80	0
13	0	40	0
14	0,4	32	12,8
15	0	40	0
16	0	80	0
17	0	80	0
18	0	60	0
19	0	60	0
20	0	60	0
21	0	40	0
22	0,0625	78,75	4,921875
23	0	80	0
24	0	40	0
25	0	80	0
26	0	80	0
27	0	80	0
28	0	80	0
29	0	40	0
30	0,6	68	40,8
31	0	80	0
32	0	80	0

4. Defuzzifikasi

Defuzzifikasi adalah pengembalian bilangan *fuzzy* menjadi bilangan *crisp.* Langkah pertama yaitu dengan menjumlahkan semua nilai min(α) dan az yang kemudian mencari nilai Z yaitu hasil pembagian dari jumlah nilai az dengan jumlah min(α).

- Jumlah nilai min(α) = 1,125
- Jumlah nilai az = 63,44375
- $Z = \frac{63,4437575}{1,125} = 56,39444444$

Berdasarkan hasil dari defuzzifikasi dari data pasien 1, dapat disimpulkan bahwa pasien 1 memiliki nilai Z = $56,39444444$ yang masuk ke dalam kriteria normal pada diagnosa penyakit tiroid.

4.1.2 Uji Coba Menggunakan FIS Tsukamoto

Data set yang di ambil dari UCI *Repository* di uji menggunakan sistem ini dan hasil diagnosanya dibandingkan sesuai data yang ada. Sebanyak 215 data di uji menggunakan sistem dengan metode FIS Tsukamoto dan hasil diagnosa yang benar dijumlahkan kemudian dibagi dengan 215 data.Untuk perbandingan hasil uji sistem dengan pakar dapat dilihat pada Tabel 4.7 di bawah ini :

Tabel 4. 7 Perbandingan uji coba sistem dengan pakar

ke-	pkr	stm	ke-	pkr	stm	ke-	pkr	stm	ke-	pkr	stm	ke-	pkr	stm
1	1	1	46	1	1	91	1	1	136	1	1	181	2	1
2	1	1	47	1	1	92	1	1	137	1	1	182	2	1
3	1	1	48	1	1	93	1	1	138	1	1	183	2	1
4	1	1	49	1	1	94	1	1	139	1	1	184	2	1
5	1	1	50	1	1	95	1	1	140	1	1	185	2	1
6	1	1	51	1	1	96	1	1	141	1	1	186	3	1
7	1	1	52	1	1	97	1	1	142	1	1	187	3	1
8	1	1	53	1	1	98	1	1	143	1	1	188	3	1
9	1	1	54	1	1	99	1	1	144	1	1	189	3	1
10	1	1	55	1	1	100	1	1	145	1	1	190	3	1
11	1	1	56	1	1	101	1	1	146	1	1	191	3	1
12	1	1	57	1	1	102	1	1	147	1	1	192	3	1
13	1	1	58	1	1	103	1	1	148	1	1	193	3	1
14	1	1	59	1	1	104	1	1	149	1	1	194	3	1
15	1	1	60	1	1	105	1	1	150	1	1	195	3	1
16	1	1	61	1	1	106	1	1	151	2	1	196	3	1
17	1	1	62	1	1	107	1	1	152	2	1	197	3	1
18	1	1	63	1	1	108	1	1	153	2	1	198	3	1
19	1	1	64	1	1	109	1	1	154	2	1	199	3	1
20	1	1	65	1	1	110	1	1	155	2	1	200	3	1
21	1	1	66	1	1	111	1	1	156	2	1	201	3	1
22	1	1	67	1	1	112	1	1	157	2	1	202	3	1
23	1	1	68	1	1	113	1	1	158	2	1	203	3	1
24	1	1	69	1	1	114	1	1	159	2	1	204	3	1
25	1	1	70	1	1	115	1	1	160	2	1	205	3	1
26	1	1	71	1	1	116	1	1	161	2	1	206	3	1
27	1	1	72	1	1	117	1	1	162	2	1	207	3	1
28	1	1	73	1	1	118	1	1	163	2	1	208	3	1
29	1	1	74	1	1	119	1	1	164	2	1	209	3	1
30	1	1	75	1	1	120	1	1	165	2	1	210	3	1
31	1	1	76	1	1	121	1	1	166	2	1	211	3	1
32	1	1	77	1	1	122	1	1	167	2	1	212	3	1
33	1	1	78	1	1	123	1	1	168	2	1	213	3	1
34	1	1	79	1	1	124	1	1	169	2	1	214	3	1
35	1	1	80	1	1	125	1	1	170	2	1	215	3	1
36	1	1	81	1	1	126	1	1	171	2	1			

Tabel 4.7 Perbandingan uji coba sistem dengan pakar

ke-	pkr	stm
37	1	1
38	1	1
39	1	1
40	1	1
41	1	1
42	1	1
43	1	1
44	1	1
45	1	1

ke-	pkr	stm
82	1	1
83	1	1
84	1	1
85	1	1
86	1	1
87	1	1
88	1	1
89	1	1
90	1	1

ke-	pkr	stm
127	1	1
128	1	1
129	1	1
130	1	1
131	1	1
132	1	1
133	1	1
134	1	1
135	1	1

ke-	pkr	stm
172	2	1
173	2	1
174	2	1
175	2	1
176	2	1
177	2	1
178	2	1
179	2	1
180	2	1

Keterangan nama-nama kolom pada tabel :

- Ke-, menunjukkan urutan data set yang di uji
- pkr, menunjukkan hasil diagnosa pakar
- stm, merupakan hasil diagnosa sistem

Untuk mengetahui tingkat keakurasian sistem diagnosa penyakit tiroid menggunakan metode FIS Tsukamoto dapat di hitung dengan persamaan 4.1.

$$\text{Akurasi}(\%) = \frac{n}{215} * 100\% \quad (4.1)$$

Keterangan :

n adalah data uji yang hasil diagnosanya sesuai dengan diagnosa pakar.

Karena ada 150 prediksi yang benar maka hasil akurasi sistem dapat dihitung menggunakan persaman 4.1.

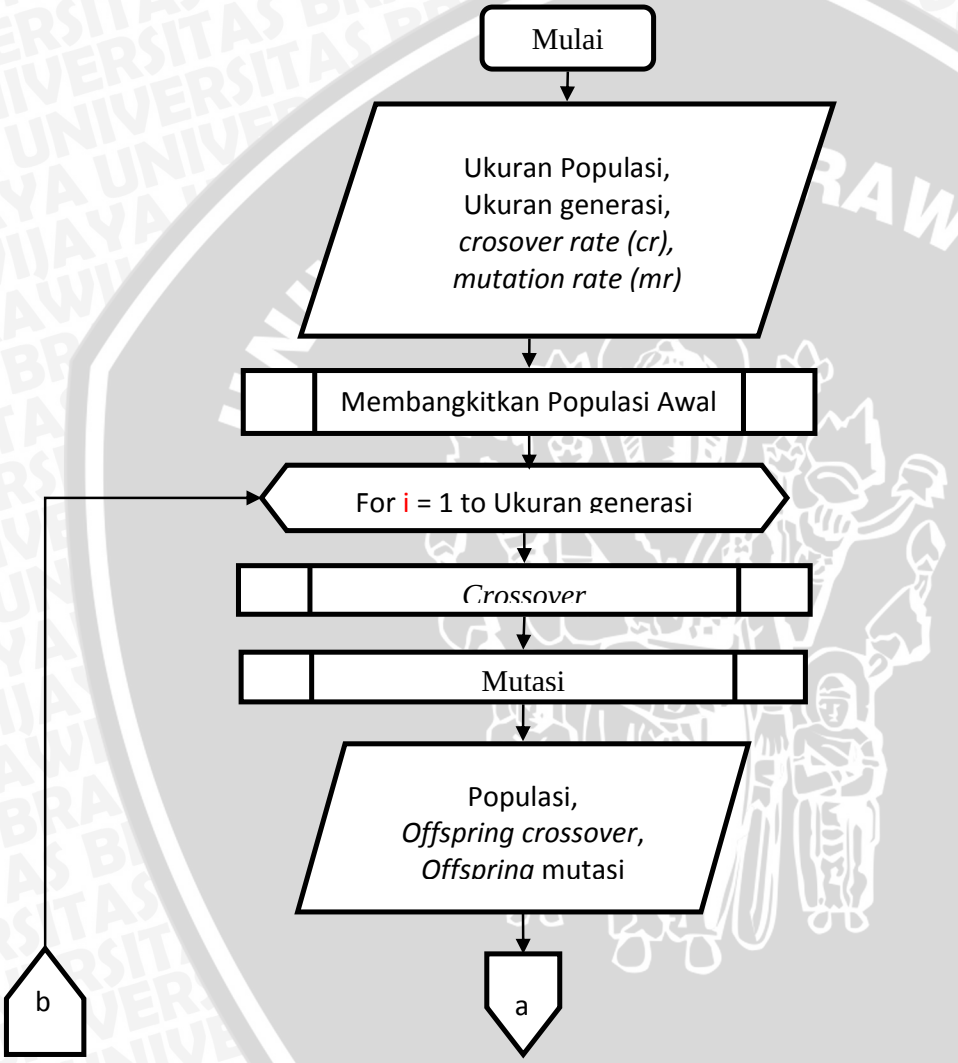
$$\text{Akurasi} = \frac{150}{215} * 100\% = 69{,}7\%$$

Sistem Diagnosa penyakit tiroid jika menggunakan metode FIS Tsukamoto mendapatkan hasil akurasi sebesar 69,7% dari dataset yang di ambil dari UCI *Repository*. Untuk mendapatkan akurasi yang lebih tinggi maka perlu dilakukan optimasi terhadap fungsi keanggotaan *fuzzy.*

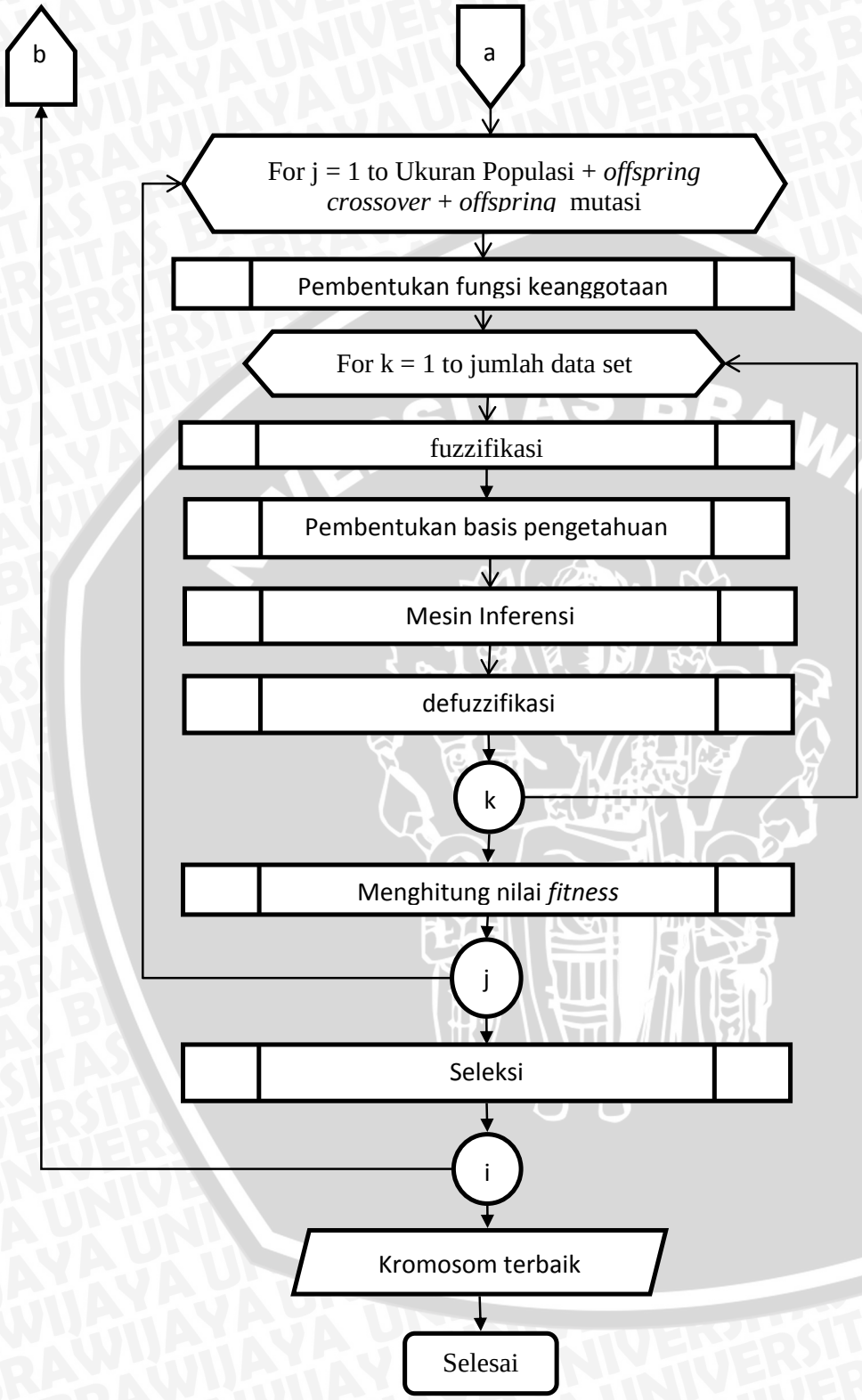
4.2 Siklus algoritma yang digunakan

Sub bab ini menjelaskan tentang gambaran proses algoritma yang dilakukan pada penelitian ini. Pada Gambar 4.1 di bawah ini dapat lihat diagram alir proses perhitungan pada penelitian ini dengan menggunakan algoritma FIS Tsukamoto yang dioptimasi dengan Algoritma Genetika.

1. Diagram alir utama sistem



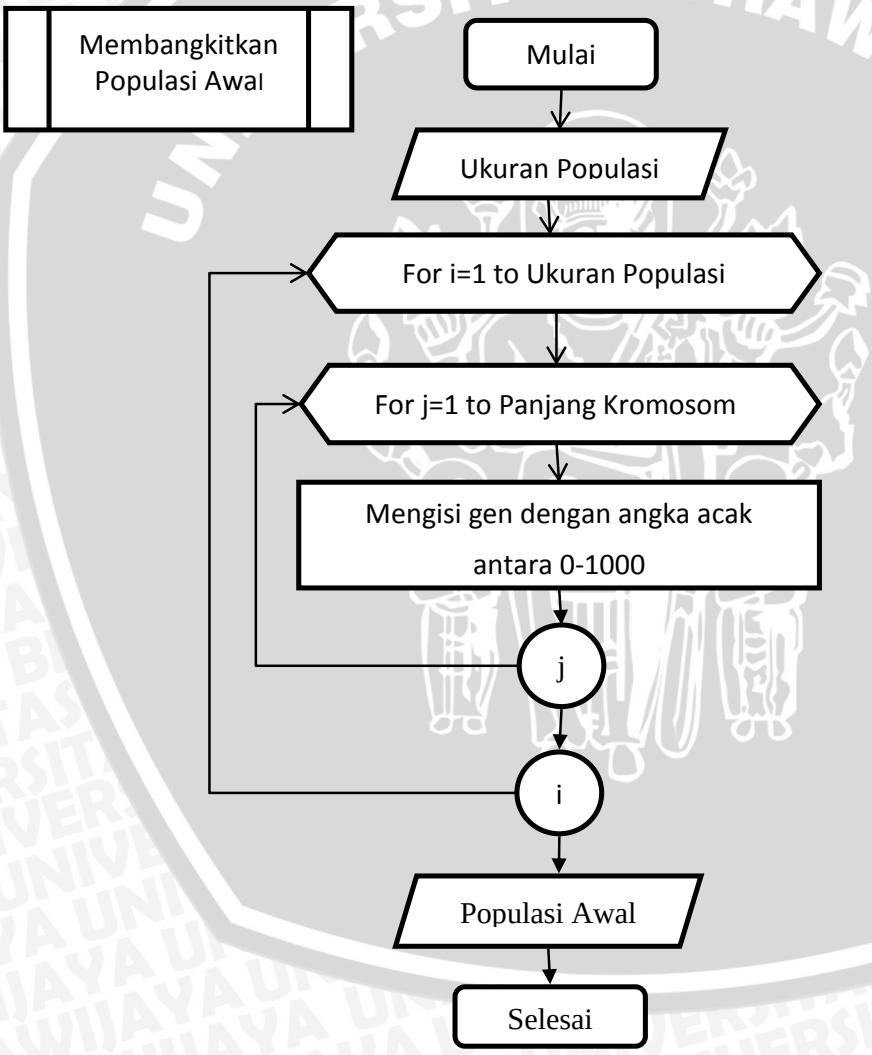
Gambar 4. 7 Diagram alir sistem



Gambar 4.7 Diagram alir sistem

Sistem ini menggunakan proses seperti pada diagram alir diatas, pertama yang dilakukan adalah memasukkan jumlah populasi, jumlah generasi, *crossover rate* dan *mutation rate*. Kemudian dilakukan proses pehitungan algortima genetika mulai dari pembentukan populasi awal yang dilakukan dengan nilai yang acak sebagai induk pertama, kemudian melakukan *crossover* dan mutasi yang menghasilkan *offspring*. Populasi dan hasil *offspring* digunakan untuk pembentukan fungsi keanggotaan pada metode *FIS Tsukamoto* . Proses dari metode tersebut terdiri dari fuzzifikasi, pembentukan basis pengetahuan, mesin inferensi dan defuzzifikasi. Proses pada FIS Tsukamoto diulang pada setiap kromosom sebanyak dataset yang ada, gunanya untuk menghitung *fitness* dari kromosom tersebut.

2. Diagram alir pada fungsi membangkitkan populasi awal

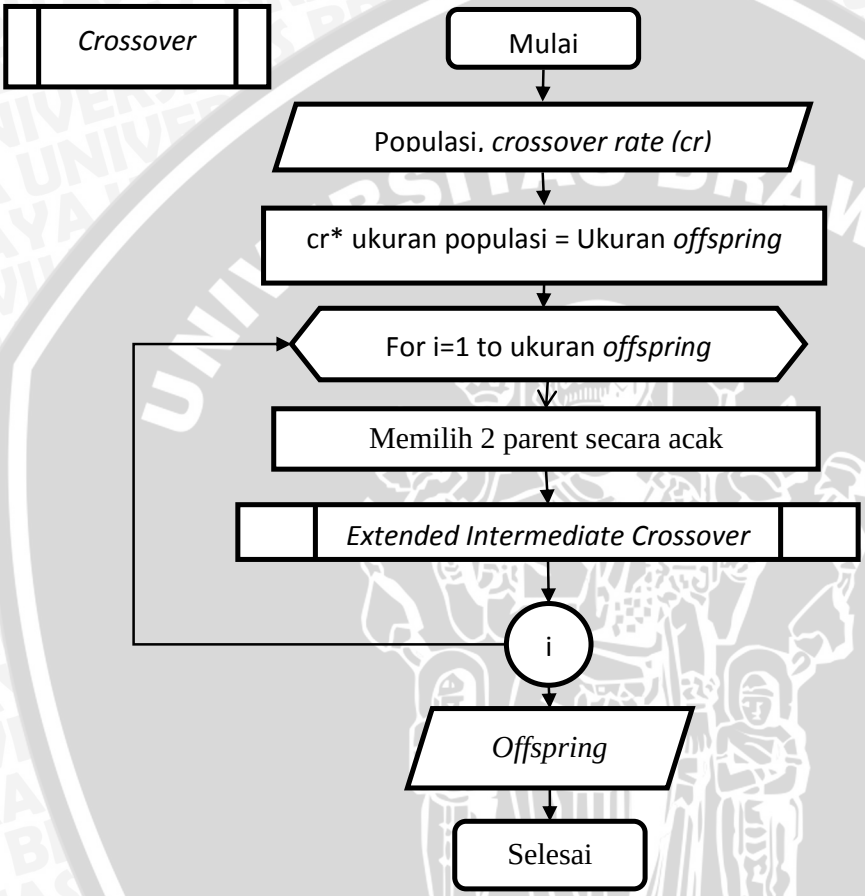


Gambar 4. 8 Diagram alir proses membangkitkan populasi awal

Untuk lebih detail penjelasan diagram fungsi membangkitkan populasi awal, yaitu :

a. Memasukkan ukuran populasi.
b. Mengisi setiap gen pada kromosom sebanyak ukuran populasi dengan angka acak antara 0 sampai 1000.
c. Hasil dari proses ini adalah populasi dengan nilai gen yang acak.

3. Diagram alir pada fungsi *crossover*

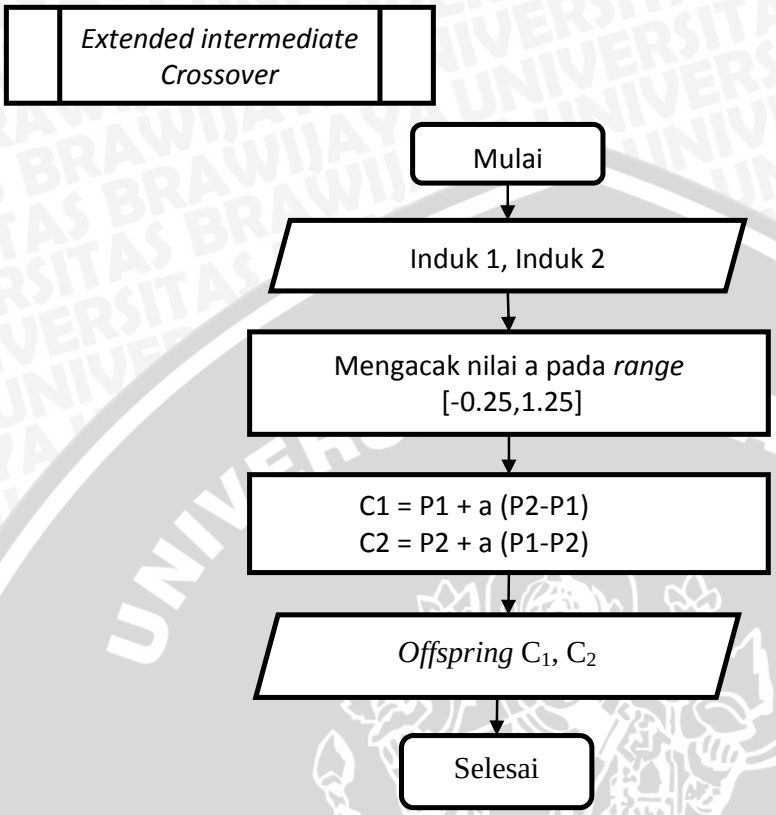


Gambar 4. 9 Diagram alir proses *crossover*

Penjelasan diagram alir di atas dapat dibaca pada pernyataan di bawah ini:

a. Memasukkan populasi dan *crossover rate*
b. Menghitung ukuran *offspring* yang dibutuhkan dari hasil proses *crossover.*
c. Setiap melakukan *crossover* memilih 2 iduk dari populasi secara acak kemudian dilakukan *extended intermediate crossover* sebagai metode *crossover* yang dipilih pada penelitian ini.
d. Hasil dari proses ini berupa *offspring* dari hasil *crossover*.

4. Diagram fungsi *extended intermediate crossover*

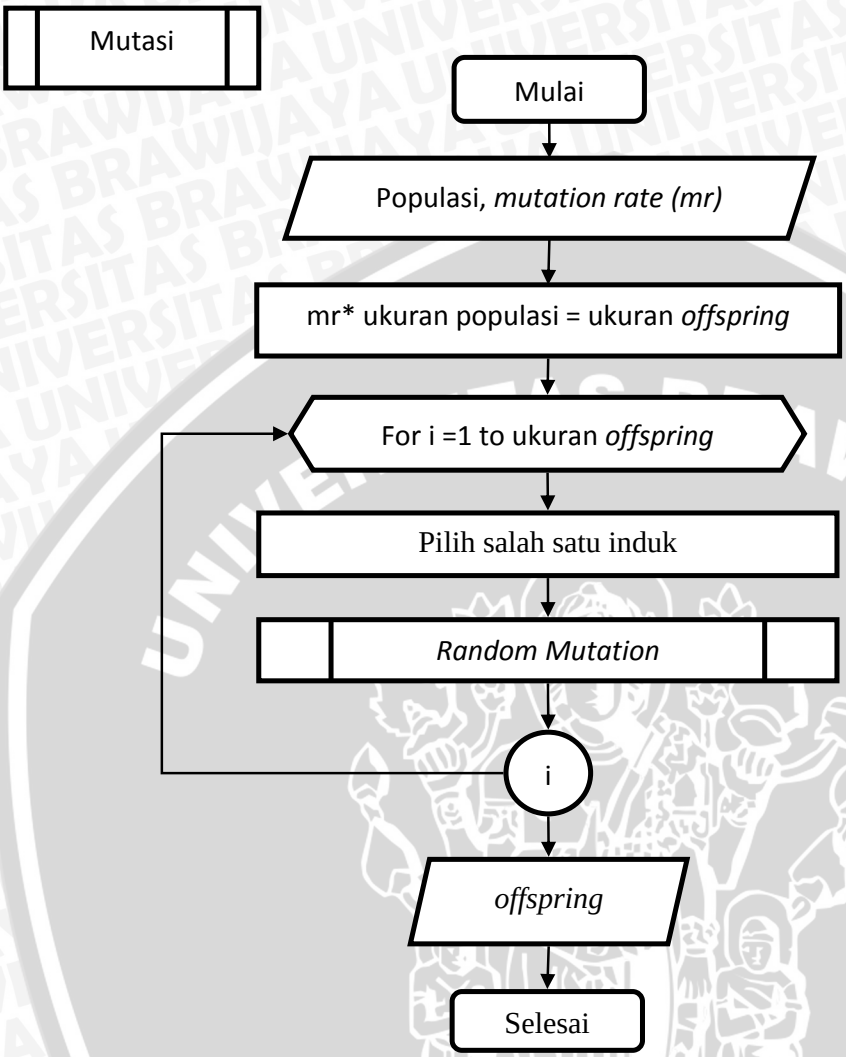


Gambar 4. 10 Diagram alir proses *extended intermediate crossover*

Diagram *extended intermidiate crossover* pada Gambar 4.11 dapat didetailkan pada penjelasan berikut:

a. Masukkan 2 induk terpilih.
b. Mangacak nilai a yaitu nilai pada *range* [-0.25, 1.25].
c. Mencari setiap *offspring* dengan menambahkan nilai pada salah satu gen teprilih dengan hasil kali dari a yang telah dikalikan dengan hasil pengurangan nilai gen terpilih lainnya dengan nilai gen tersebut.
d. Hasilnya adalah 2 *offspring.*

5. Diagram fungsi mutasi

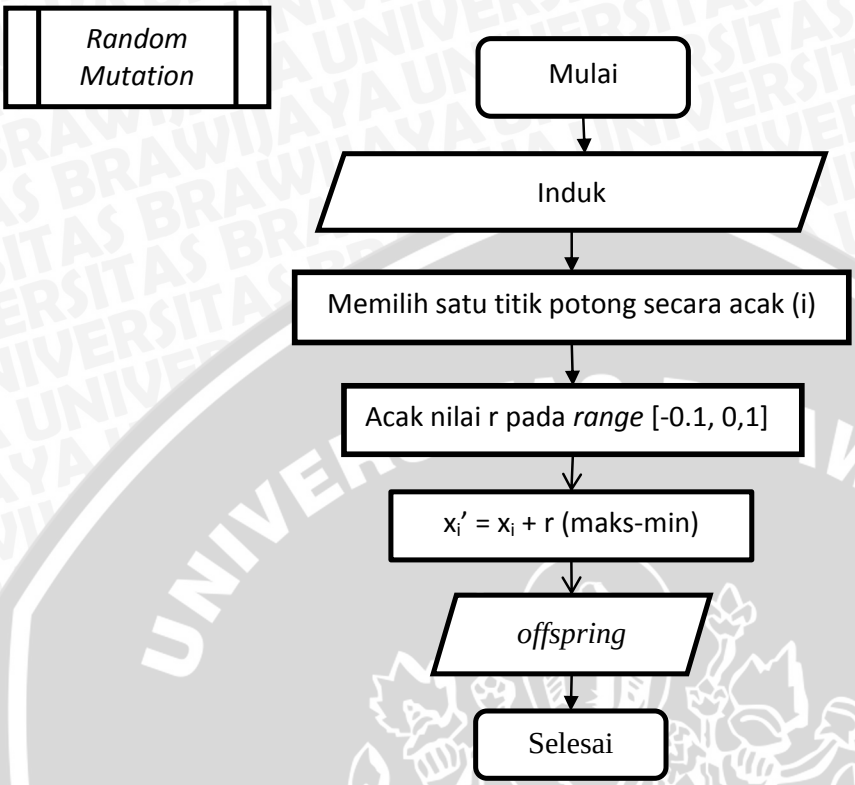


Gambar 4. 11 Diagram alir proses mutasi

Penjelasan diagram mutasi diatas adalah sebagai berikut:

a. Memasukkan populasi dan *mutation rate*.
b. Menghitung banyaknya *offspring* dari proses mutasi dengan mengkalikan populasi dengan *mutation rate.*
c. Memilih salah satu induk dari populasi dan dilakukan mutasi dengan metode *random mutation* pada induk tersebut dan terbentuklah satu *offspring* baru, proses ini dilakukan terus hingga mencapai ukuran populasi.
d. Hasil proses ini berupa individu *offspring* dari mutasi.

6. Diagram fungsi *random mutation*

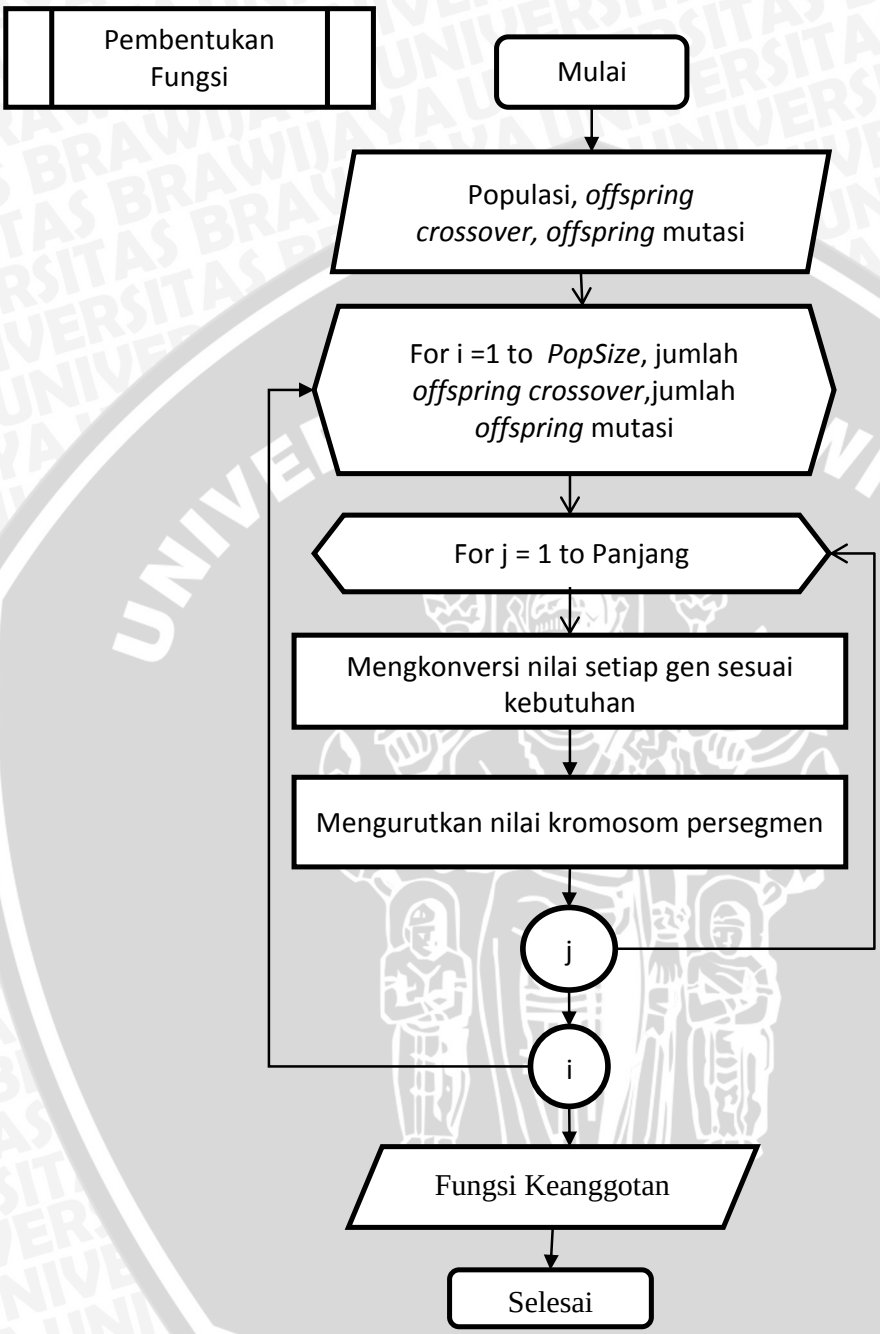


Gambar 4. 12 Diagram alir *exchange mutation*

Untuk penjelasan diagram fungsi *random mutation* dapat dilihat dibawah ini:

a. Masukkan induk terpilih.
b. Memilih satu titik potong pada kromosom secara acak.
c. Mencari nilai r pada *range* [-0.1, 0.1].
d. Menambahkan nilai pada gen terpilih dengan r yang telah dikalikan dengan nilai maksimal dan minimal yang bisa masuk pada gen tersebut.
e. Hasil dari proses ini adalah satu *offspring* hasil *random mutation.*

7. Diagram fungsi keanggotaan

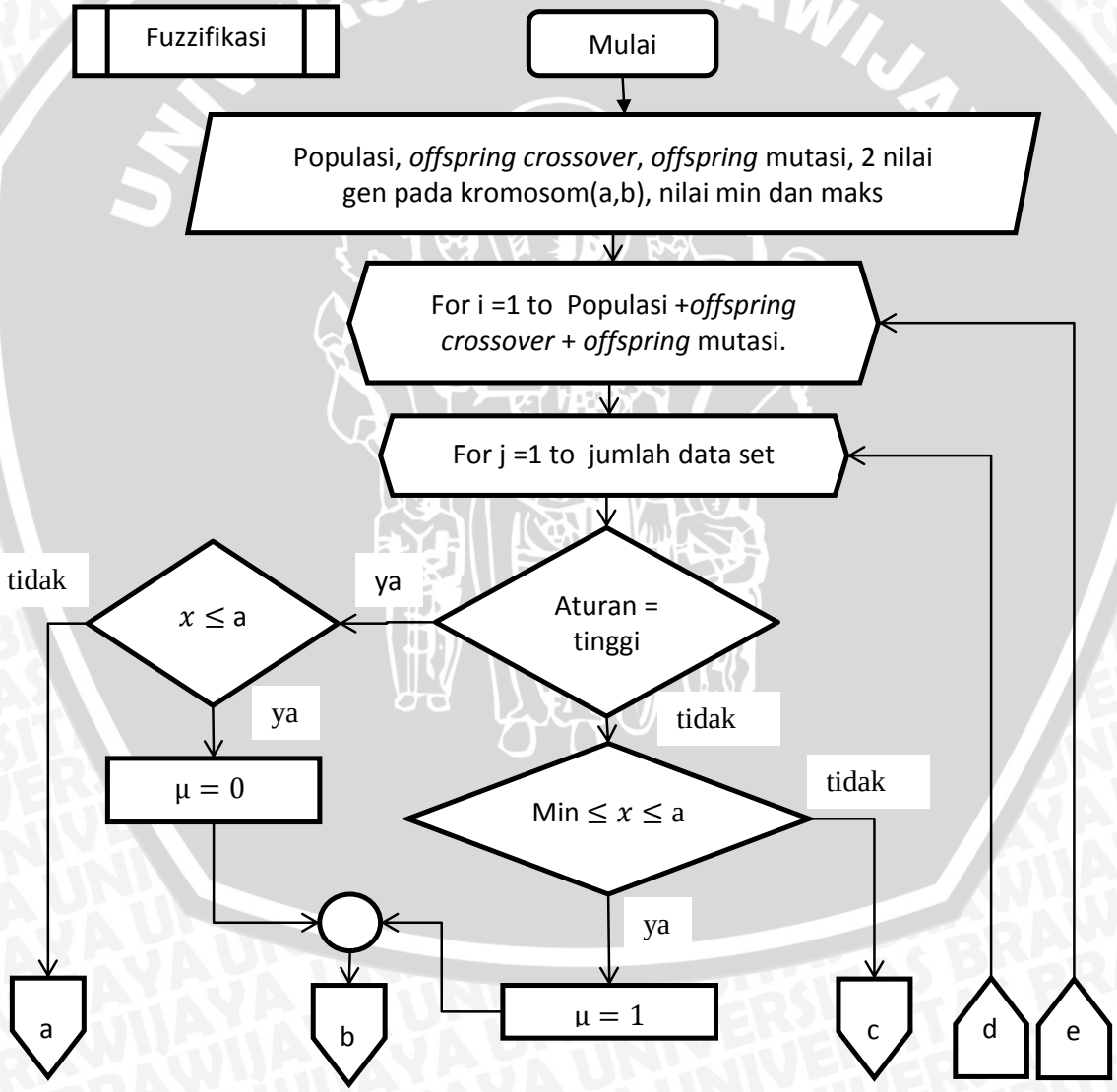


Gambar 4. 13 Diagram alir proses pembentukan fungasi keanggotaan

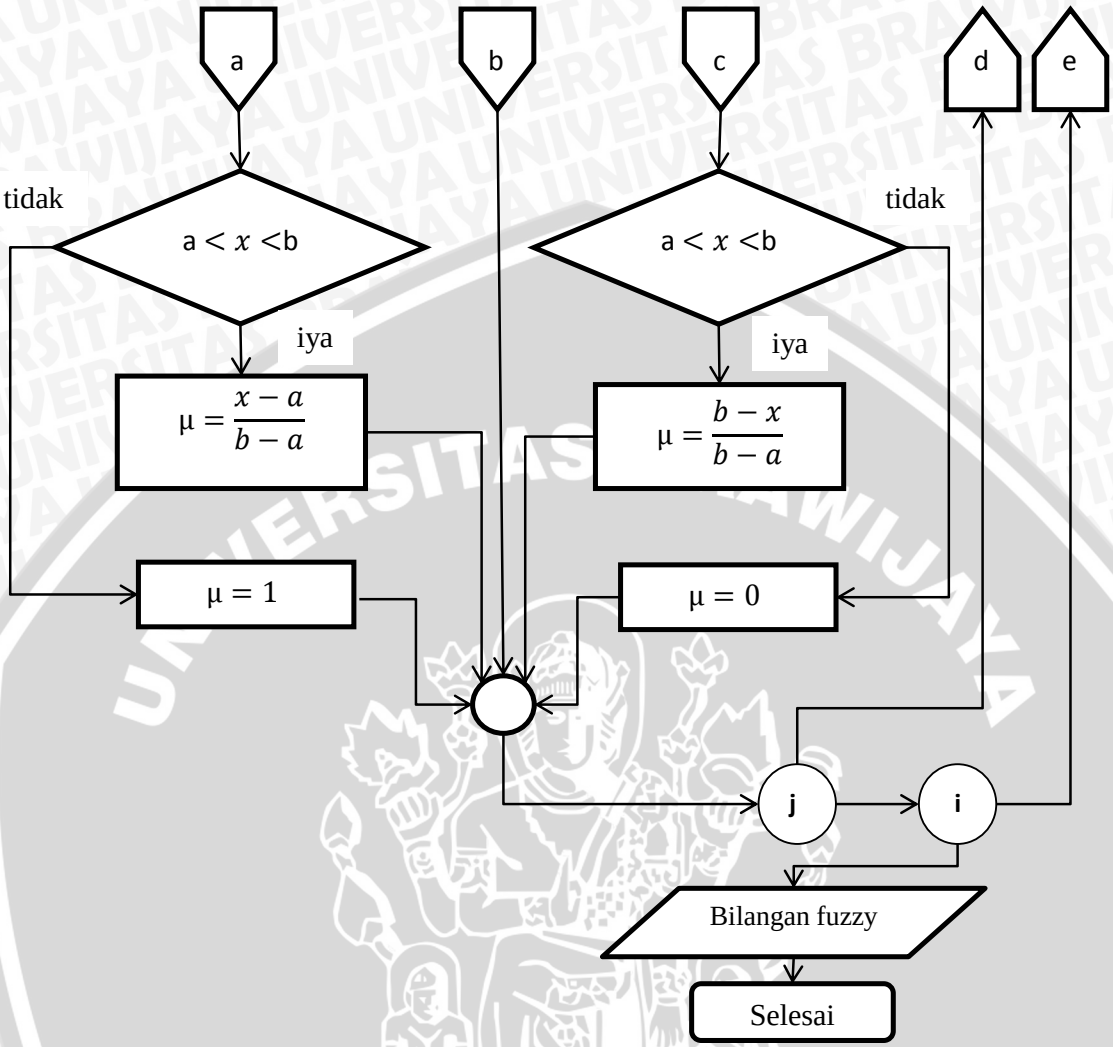
Penjelasan diagram fungsi pembentukan fungsi keangotaan dapat dibaca pada penjelasan berikut:

a. Memasukkan individu gabungan yaitu individu dari populasi, *offspring crossover* dan *offspring* mutasi.
b. Pada setiap kromosom nilai pada gen diurutkan sesuai fungsi persegmennya.
c. Dilakukan pengkonversian nilai pada setiap gen sesuai fungsi pada tiap segmen juga.
d. Nilai pada kromosom yang telah diurutkan dan dikonversi dapat digunakan sebagai batasan pada fungsi keanggotaan.
e. Hasilnya berupa fungsi keanggotaan.

8. Diagram fungsi fuzzifikasi



Gambar 4. 14 Diagram alir proses fuzzifikasi

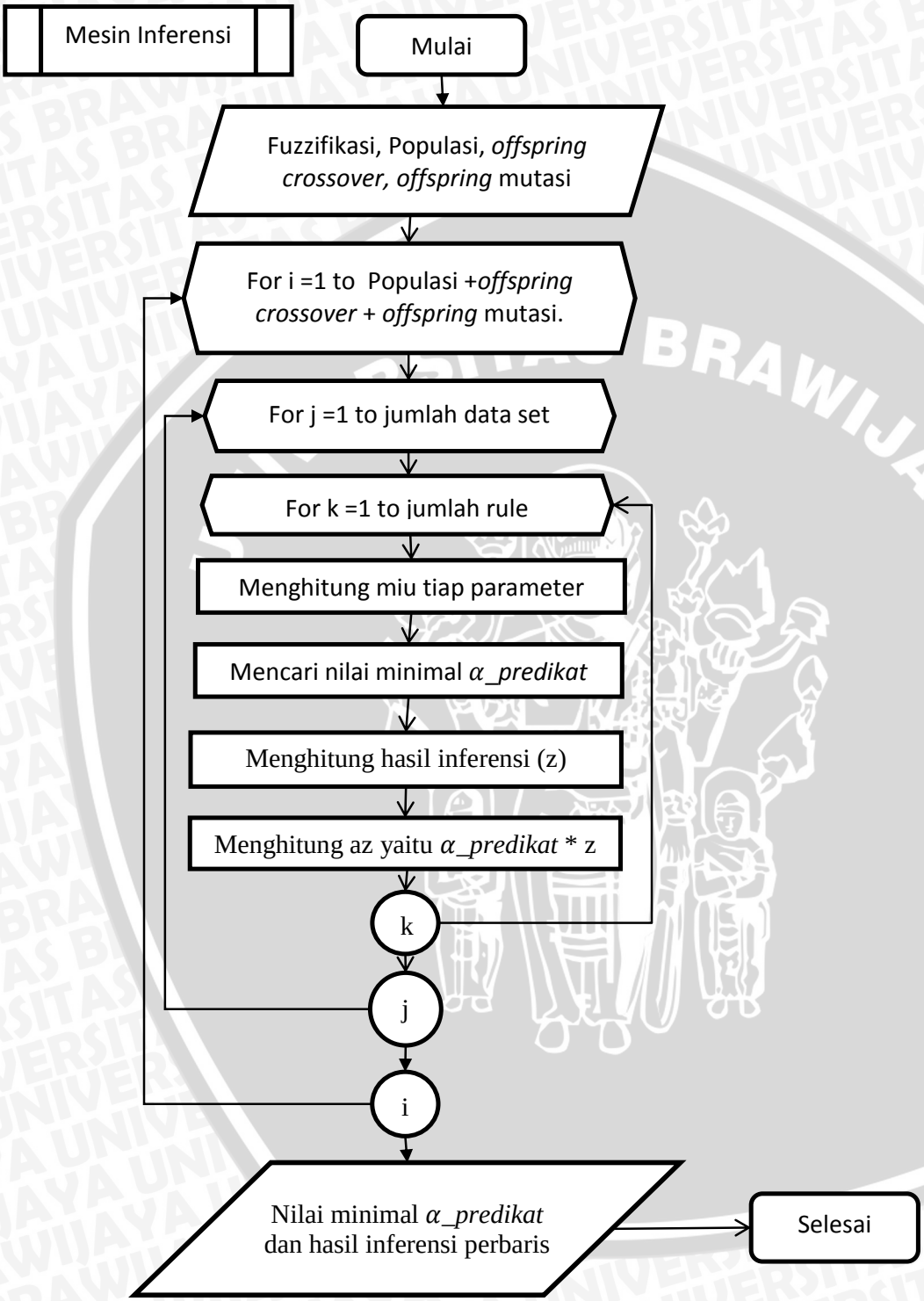


Gambar 4.14 Diagram alir proses fuzzifikasi

Diagram alir fungsi fuzzifikasi diatas dapat dijelaskan lebih detail dengan pernyataan berikut ini:

a. Masukkan Nilai tiap gen pada suatu segmen dan individu gabungan.
b. Setiap kromosom melakukan fuzzifikasi sebanyak dataset yang ada.
c. Persamaan-persamaan diatas digunakan untuk mengkonversi bilangan *crisp* menjadi bilangan *fuzzy* yang selanjutnya akan diinferensi.
d. Hasil dari proses tersebut adalah bilangan *fuzzy.*

9. Diagram alir fungsi mesin inferensi

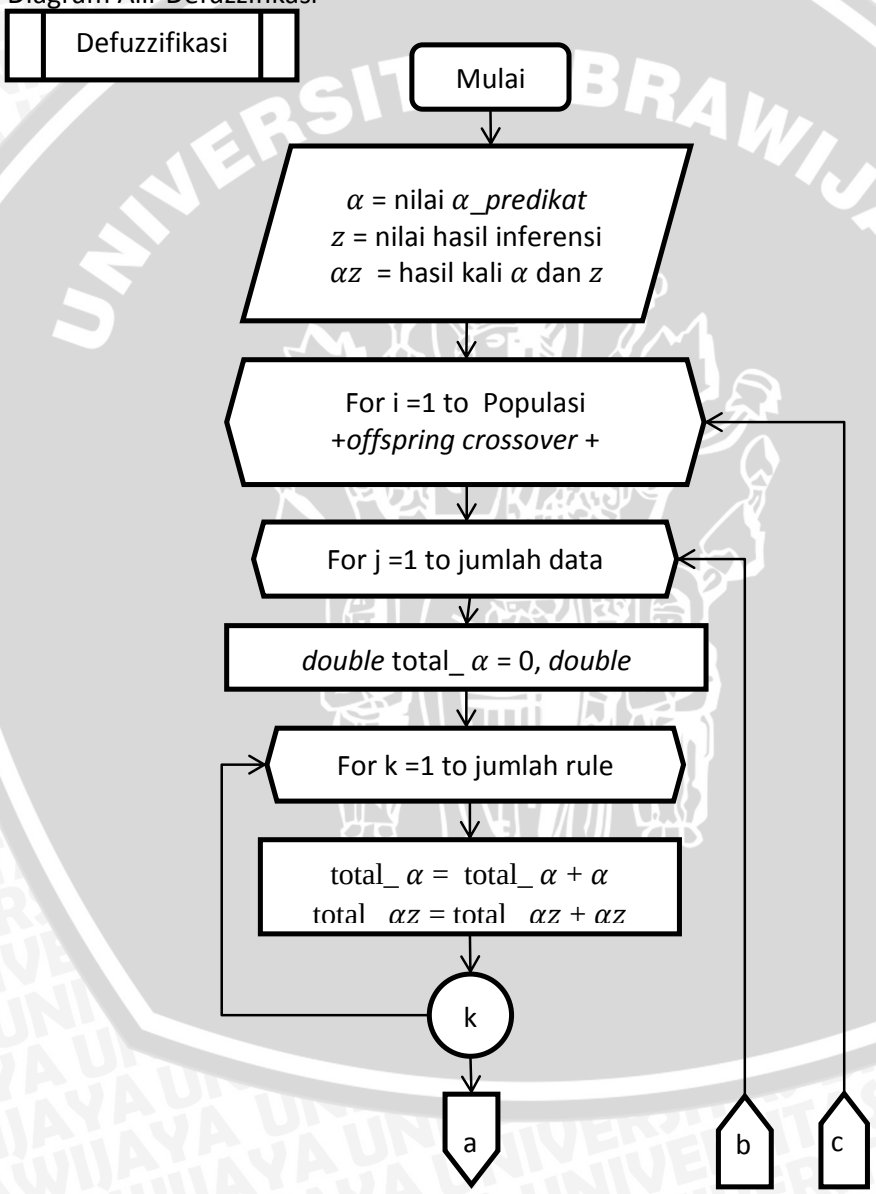


Gambar 4. 15 Diagram alir proses mesin inferensi

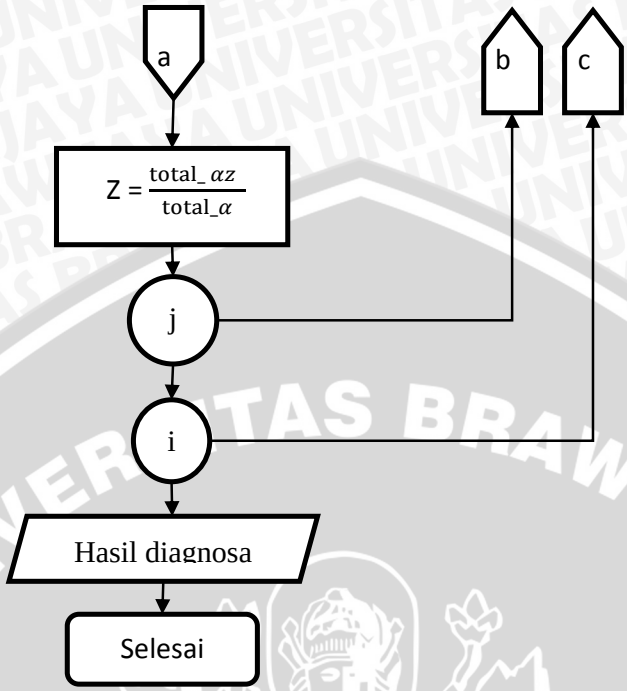
Penjelasan lebih detail tentang diagram alir mesin inferensi adalah sebagai berikut:

a. Masukkan fuzzifikasi dan individu gabungan.
b. Setiap kromosom di coba pada sejumlah dataset yang digunakan, setiap data set memiliki 32 rule pada penelitian ini.
c. Setiap rule nilai minimal $\alpha\_predikat$, di hitung hasil inferensi (z) dan αz yaitu hasil kali antara $\alpha\_predikat$ dan inferensi (z).
d. Hasil dari proses ini adalah nilai minimal $\alpha\_predikat$, hasil inferensi perbaris rule (z) dan αz.

10. Diagram Alir Defuzzifikasi



Gambar 4. 16 Diagram alir defuzzifikasi`

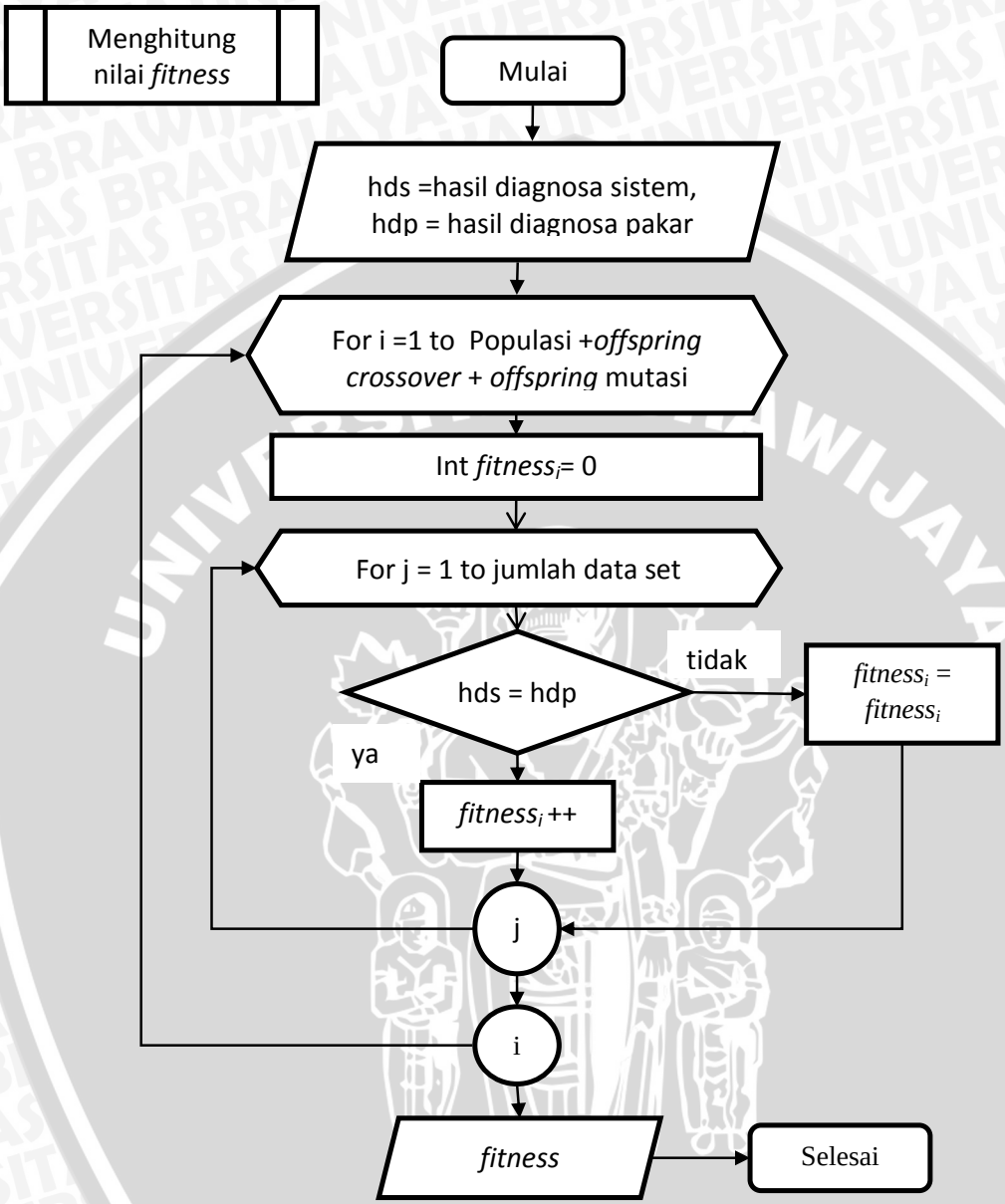


Gambar 4.16 Diagram alir defuzzifikasi

Diagram merupakan proes defuzzifikasi yaitu proes pengubahan bilangan fuzzy menjadi bilangan *crisp* kembali. Penjelasannya lebih lanjut:

a. Masukkan α = nilai $\alpha\_predikat$ z = nilai hasil inferensi.
b. Setiap kromosom melakukan proses defuzzifikasi sebanyak dataset yang ada.
c. Menjumlahkan setiap nilai $\alpha\_predikat$ pada baris aturan.
d. Menjumlahkan setiap nilai αz pada baris aturan.
e. Mencari nilai Z dengan membagi jumlah nilai $\alpha\_predikat$ dengan jumlah αz.
f. Hasil dari proses ini adalah diagnosa sistem dengan menggunakan fungsi keanggotaan pada kromosom tertentu.

11. Diagram fungsi menghitung nilai *fitness*

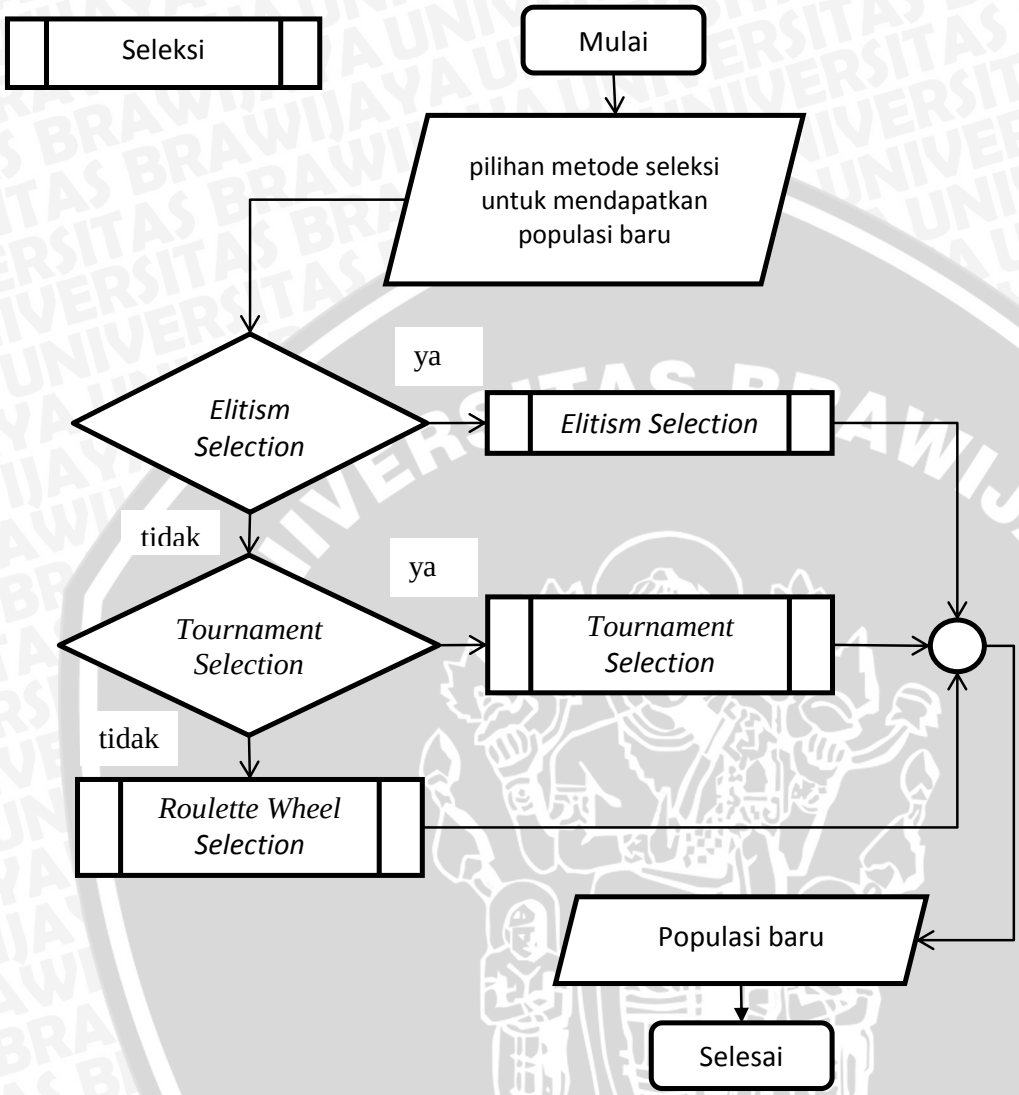


Gambar 4. 17 Diagram alir menghitung nilai fitness

Diagram diatas membahas proses cara menghitung *fitness*, untuk lebih detailnya yaitu:

a. Masukkan hds =hasil diagnosa sistem, hdp = hasil diagnosa pakar.
b. Pada setiap kromosom di individu gabungan apabila hasil diagnosa sistem dengan pakar sama pada setiap data uji.
c. Jika hasilnya sama maka nilai *fitness* bertambah satu, namun juka tidak nilai *fitness* tetap.
d. Hal tersebut dilakukan sampai data uji yang terkahir.
e. Keluaran dari fungsi ini adalah nilai *fitness* pada setiap kromosom.

12. Diagram fungsi seleksi

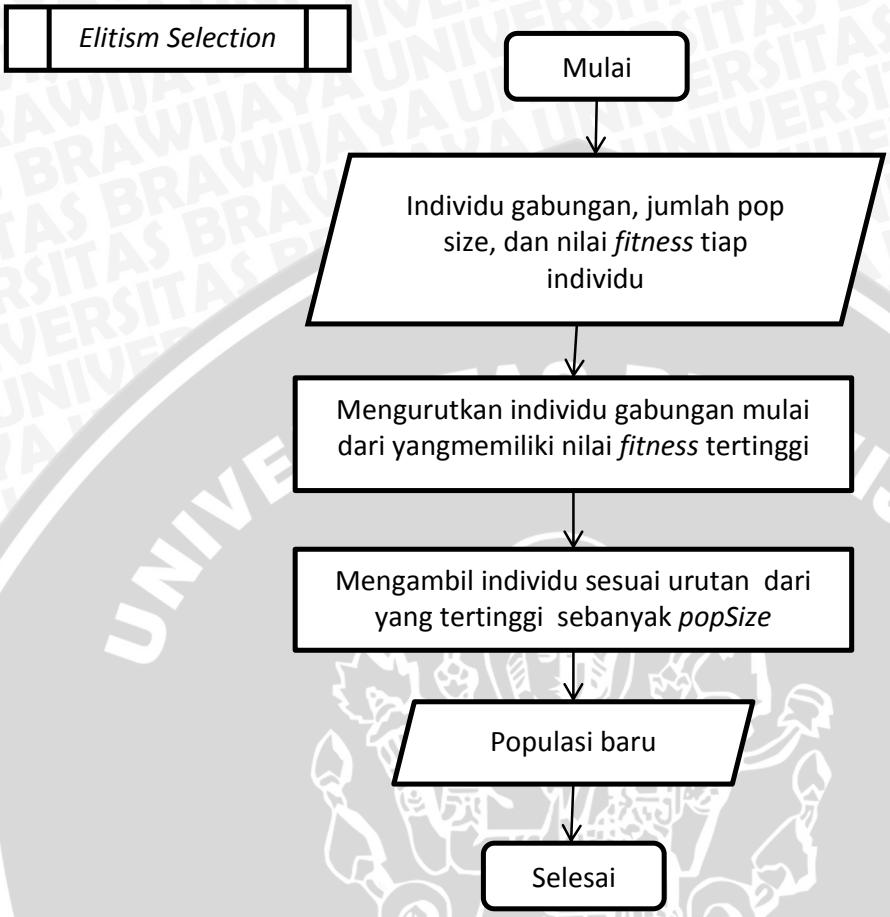


Gambar 4. 18 Diagram alir proses seleksi

Fungsi seleksi dapat dilihat pada diagram pada Gambar 4.18, detail penjelasannya:

a. Masukkan individu gabungan, jumlah *popSize*, nilai *fitness* dan pilihan metode seleksi.
b. Jika pilihannya metode *elitism* maka masuk pada fungsi *elistm selection,* jika pilihannya metode *tournament* maka masuk pada fungsi *tournament selection* dan jika pilihannya *roulette wheel* maka masuk pada fungsi *roulette wheel selection.*
c. Hasilnya berupa populasi baru.

13. Metode *elitism selection.*

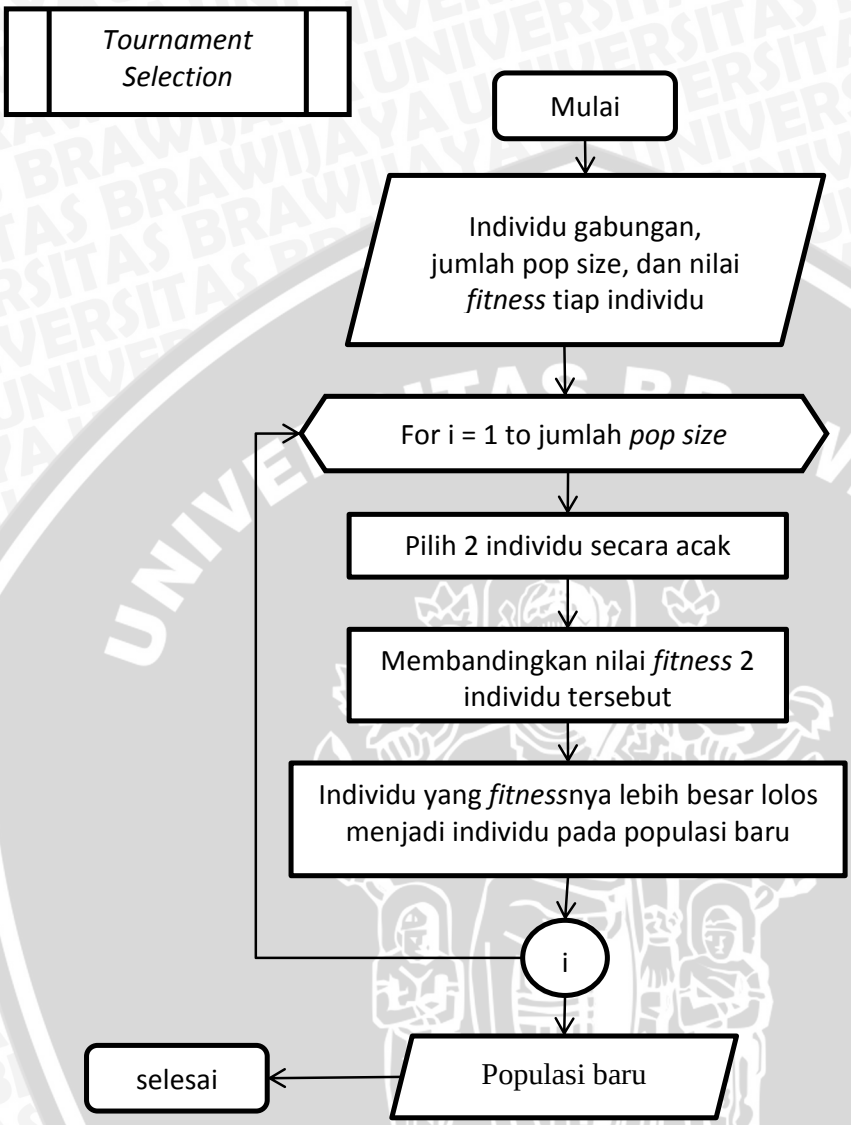


Gambar 4. 19 Diagram alir proses elitsm selection

Penjelasan Gambar 4.19 adalah sebagai berikut:

a. Proses *elitism* membutuhkan masukan barupa kumpulan individu yang akan diselekasi, jumlah *popSize* dan nilai *fitness* masing-masing individu.
b. Langkah pertama dari proses ini adalah mengurutkan kumpulan individu sesuai nilai *fitness*nya.
c. Kemudian mengambil individu sejumlah *popSize* sesuai urutan dari *fitness* terbesar.
d. Terbentuklah populasi baru hasil seleksi menggunakan *elitism.*

14. Diagram alir proses *tournament selection*

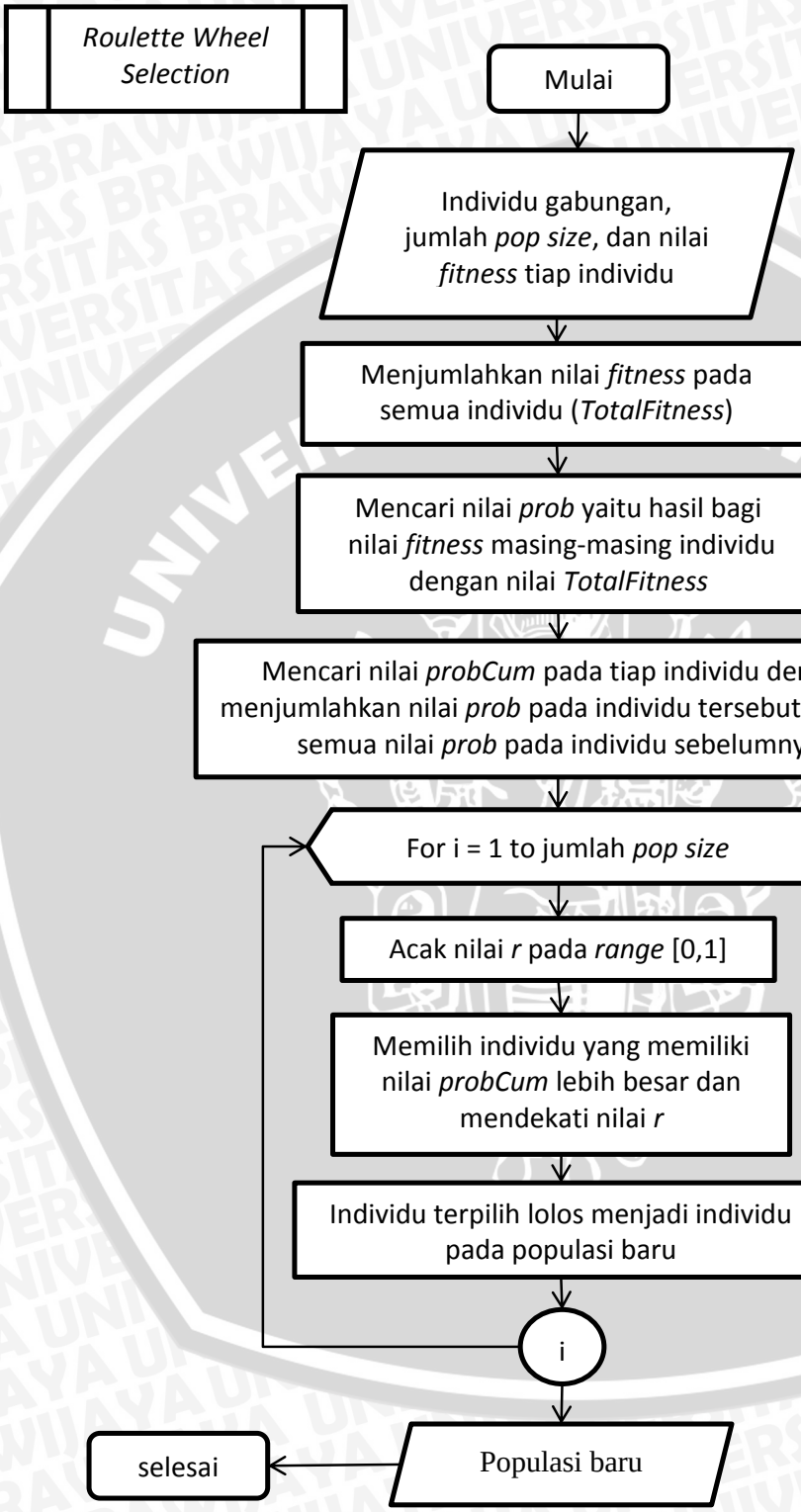


Gambar 4. 20 Diagram alir proses tournament selection

Penjelasan diagram alir pada Gambar 4.20 adalah berikut:

a. Masukan fungsi berupa kumpulan individu yang akan diseleksi, jumlah *popSize* dan *fitness* tiap individu.
b. Proses dilakukan sebanyak ukuran *popSize.*
c. Memilih 2 individu secara acak.
d. Membandingkan nilai *fitness* 2 individu tersebut.
e. Memasukkan kepada populasi yang baru individu yang memiliki nilai *fitness* lebih besar.
f. Hasil berupa terbentuknya populasi baru hasil seleksi *tournament selection.*

15. Diagram alir *roulette wheel selection*



Gambar 4. 21 Diagram alir metode roulette wheel selection

Penjelasan diagram alir pada Gambar 4.21 adalah berikut:

1. Fungsi seleksi membutuhkan masukan berupa kumpulan individu yang akan diseleksi, jumlah *popSize* dan nlai *fitness* masing-masing individu.
2. Menjumlahkan nilai *fitness* pada masing-masing individu sebagai *TotalFitness.*
3. Menghitung nilai *prob* pada masing-masing individu dengan membagi nilai *fitness*nya dengan *TotalFitness.*
4. Menghiung nlai *probCum* pada masing-masing individu dengan menjumlahkan nilai *prob* individu tersebut dengan semua nilai *prob* individu sebelumnya.
5. Melakukan proses *roulette wheel* sebanyak jumlah *popSize* yang diawali dengan mencari nilai *r* secara acak pada *range* [0,1].
6. Memilih salah satu individu yang memiliki nilai *probCum* lebih besar dan mendekati nilai *r.*
7. Individu terpilih lolos menjadi individu pada populasi baru.
8. Sebelum populasi baru mencapai ukuran *popSize* maka ulangi langkah kelima.
9. Terbentuknya populasi baru dari hasil seleksi *roulette wheel.*

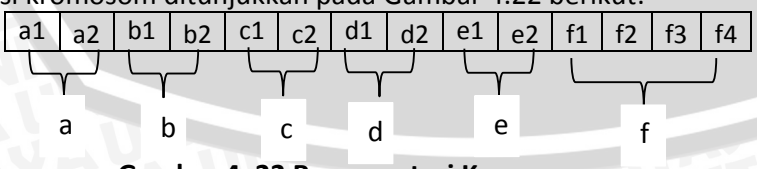
4.3 Siklus penyelesaian masalah menggunakan Algoritma Genetika

Dalam sub bab ini akan dibahas bagaimana cara Algoritma Genetika dalam menyelesaikan malasah optimasi fungsi keanggotaan dari FIS Tsukamoto dalam sistem diagnosa penyakit tiroid.

4.3.1 Representasi Kromosom dan Perhitungan *Fitness*

Kromosom awal dibangitkan secara acak antara bilangan 0 sampai 1000, yang kemudian bilangan tersebut dikonversi sesuai batas nilai pada masing-masing kriteria. Kromosom pada proses ini panjangnya adalah 14. 10 gen pertama digunakan untuk fungsi keanggotaan pada setiap kriteria yang terdiri dari 2 gen karena fungsi keangotaan setiap kriteria terdiri dari 2 yaitu tinggi dan rendah sehingga dibutuhkan 2 gen. Untuk 4 gen selanjutnya adalah fungsi keanggotaan hasil keputusan karena hasil keputusannya terdiri dari 3 yaitu hyperthyroid, normal atau hypothyroid.

Representasi kromosom ditunjukkan pada Gambar 4.22 berikut:



Gambar 4. 22 Representasi Kromosom

Keterangan Gambar 4.22, yaitu :

a. Segmen gen kriteria *t3-resin*
b. Segmen gen kriteria *thyroxin serum*
c. Segmen gen kriteria *triiodotynine serum*

d. Segmen gen kriteria *basal thyroid-stimulating hormone (TSH)*
e. Segmen gen kriteria perbandingan nilai TSH setelah injeksi
f. Segmen gen hasil keputusan

Contoh representasi kromosom untuk bilangan *real* acak antara 0-1000 dapat dilihat pada Gambar 4.23 berikut:

732	730	459	540	432	756	25	190	241	825	555	730	732	459

Gambar 4. 23 contoh kromosom

Langkah awal yang dilakukan adalah mengkonversi nilai setiap segmen sesuai dengan nilai minimal dan maksimal dari setiap kriteria atau nilai batasan hasil.

Cara melakukan konversi dapat dilihat pada persamaan 4.2 berikut :

$$P = \frac{x}{1000} * (maks-min) + min \quad (4.2)$$

Keterangan :

x = nilai input yang akan dikonversi.

maks = nilai maksimal *range* parameter.

min = nilai minimal *range* parameter.

Tes *t3*-resin

$P_1 = \frac{732}{1000} * (144-65) + 65 = 122,828$

$P_2 = \frac{730}{1000} * (144-65) + 65 = 122,67$

Tes *thyroxin serum*

$P_3 = \frac{459}{1000} * (25,3-0,5) + 0,5 = 11,883$

$P_4 = \frac{540}{1000} * (25,3-0,5) + 0,5 = 13,892$

Tes *triiodotynine*

$P_5 = \frac{432}{1000} * (10-0,2) + 0,2 = 4,434$

$P_6 = \frac{756}{1000} * (10-0,2) + 0,2 = 7,609$

Tes *basal thyroid stimulate hormone*

$P_7 = \frac{25}{1000} * (56,4-0,1) + 0,1 = 1,508$

$P_8 = \frac{190}{1000} * (56,4-0,1) + 0,1 = 10,8$

Tes perbandingan nilai TSH

$P_9 = \frac{241}{1000} * (56,3-(-0,7)) + (-0,7) = 13,037$

$P_{10} = \frac{825}{1000} * (56,3-(-0,7)) + (-0,7) = 46,325$

Hasil Diagnosa

$P_{11} = \frac{555}{1000} * (100-0) + 0 = 55,5$

$P_{12} = \frac{730}{1000} * (100-0) + 0 = 73$

$P_{13} = \frac{732}{1000} * (100-0) + 0 = 73,2$

$P_{14} = \frac{459}{1000} * (100-0) + 0 = 45,9$

122,828	122,67	11,883	13,892	4,434	7,609	1,508	10,8	13,037	46,325	55,5	73	73,2	45,9

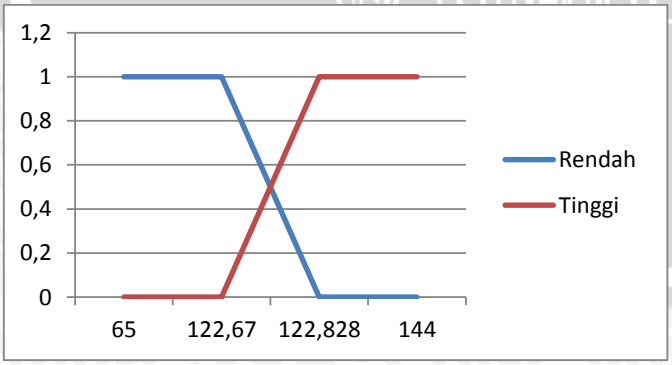
Gambar 4. 24 Kromosom setelah dikonversi sesuai fungsi persegmen

Gambar 4.24 menunjukkan kromosom yang nilai pada gen telah dikonversi menurut fungsinya. Langkah selanjutnya yaitu mengurutkan nilai tiap gen pada segmen pada kromosom.

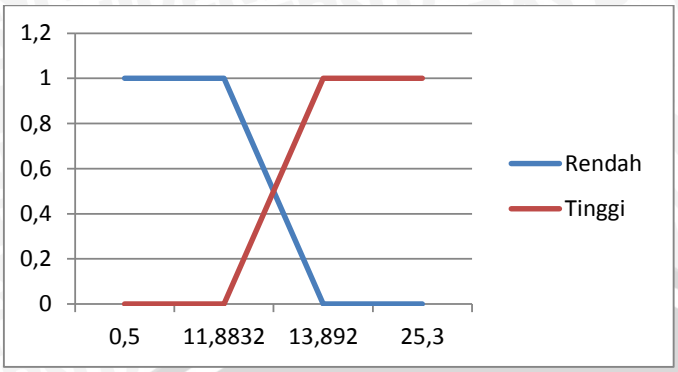
122,67	122,828	11,883	13,892	4,434	7,609	1,508	10,8	13,037	46,325	45,9	55,5	73	73,2

Gambar 4. 25 Kromosom yang telah diurutkan persegmen

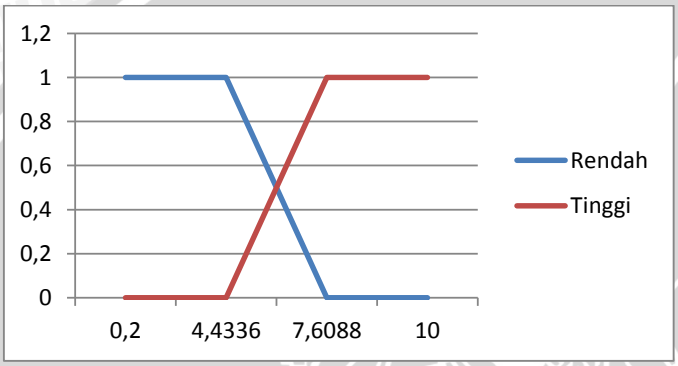
Bilangan-bilangan dalam kromosom tersebut akan dikonversi sesuai nilai minimal dan maksimal masing-masing kriteria. Kemudian digunakan sebagai batasan pada FIS Tsukamoto. Penggunaan batasan tersebut dapat dilihat pada grafik-grafik kriteria pada gambar berikut ini:



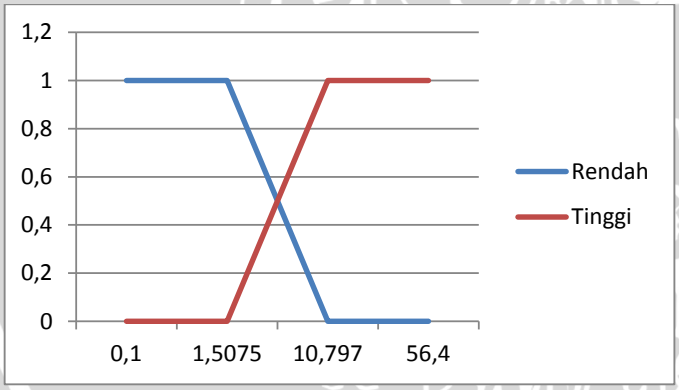
Gambar 4. 26 Grafik fungsi keanggotaan t3-resin



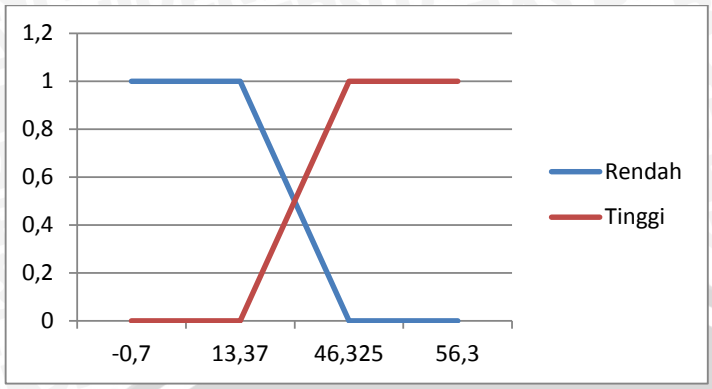
Gambar 4. 27 Grafik fungsi keanggotaan thyroxin serum



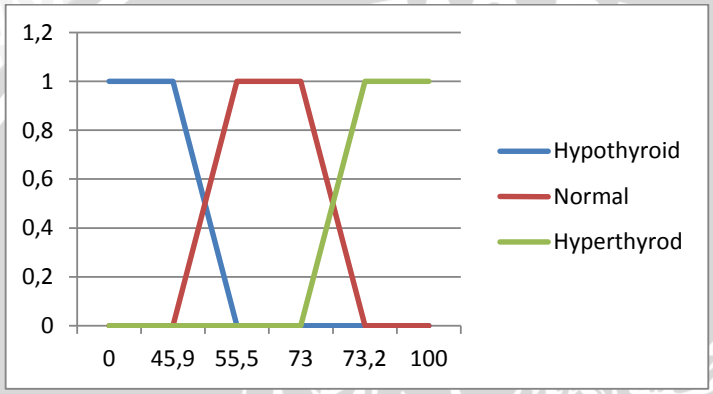
Gambar 4. 28 Grafik fungsi keanggotaan nilai triiodotynine serum



Gambar 4. 29 Grafik fungsi keanggotaan nilai TSH



Gambar 4. 30 Grafik Fungsi Keanggotaan Perbedaan nilai TSH



Gambar 4. 31 Grafik Fungsi Keanggotaan Hasil

Perhitungan *fitness*

Nilai *fitness* didapatkan dengan membandingkan hasil diagnosa pasien dari sistem dengan uji pakar yang didapatkan dari UCI *Repository.* Hasil pembandingan pengujian dapat dilihat pada Tabel 4.8 berikut ini:

Tabel 4. 8 Hasil uji dengan satu contoh kromosom

ke-	pkr	stm	ke-	pkr	stm	ke-	pkr	stm	ke-	pkr	stm	ke-	pkr	stm
1	1	1	46	1	1	91	1	1	136	1	1	181	1	2
2	1	1	47	1	1	92	1	1	137	1	1	182	1	2
3	1	1	48	1	1	93	1	1	138	1	1	183	1	2
4	1	1	49	1	1	94	1	1	139	1	1	184	1	2
5	1	1	50	1	1	95	1	1	140	1	1	185	1	2
6	1	1	51	1	1	96	1	1	141	1	1	186	1	3
7	1	1	52	1	1	97	1	1	142	1	1	187	1	3
8	1	1	53	1	1	98	1	1	143	1	1	188	1	3
9	1	1	54	1	1	99	1	1	144	1	1	189	1	3
10	1	1	55	1	1	100	1	1	145	1	1	190	1	3
11	1	1	56	1	1	101	1	1	146	1	1	191	1	3
12	1	1	57	1	1	102	1	1	147	1	1	192	1	3

Tabel 4.8 Hasil uji dengan satu contoh kromosom

ke-	pkr	stm
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	1
20	1	1
21	1	1
22	1	1
23	1	1
24	1	1
25	1	1
26	1	1
27	1	1
28	1	1
29	1	1
30	1	1
31	1	1
32	1	1
33	1	1
34	1	1
35	1	1
36	1	1
37	1	1
38	1	1
39	1	1
40	1	1
41	1	1
42	1	1
43	1	1
44	1	1
45	1	1

ke-	pkr	stm
58	1	1
59	1	1
60	1	1
61	1	1
62	1	1
63	1	1
64	1	1
65	1	1
66	1	1
67	1	1
68	1	1
69	1	1
70	1	1
71	1	1
72	1	1
73	1	1
74	1	1
75	1	1
76	1	1
77	1	1
78	1	1
79	1	1
80	1	1
81	1	1
82	1	1
83	1	1
84	1	1
85	1	1
86	1	1
87	1	1
88	1	1
89	1	1
90	1	1

ke-	pkr	stm
103	1	1
104	1	1
105	1	1
106	1	1
107	1	1
108	1	1
109	1	1
110	1	1
111	1	1
112	1	1
113	1	1
114	1	1
115	1	1
116	1	1
117	1	1
118	1	1
119	1	1
120	1	1
121	1	1
122	1	1
123	1	1
124	1	1
125	1	1
126	1	1
127	1	1
128	1	1
129	1	1
130	1	1
131	1	1
132	1	1
133	1	1
134	1	1
135	1	1

ke-	pkr	stm
148	1	1
149	1	1
150	1	1
151	1	2
152	1	2
153	1	2
154	1	2
155	1	2
156	1	2
157	1	2
158	1	2
159	1	2
160	1	2
161	1	2
162	1	2
163	1	2
164	1	2
165	1	2
166	1	2
167	1	2
168	1	2
169	1	2
170	1	2
171	1	2
172	1	2
173	1	2
174	1	2
175	1	2
176	1	2
177	1	2
178	1	2
179	1	2
180	1	2

ke-	pkr	stm
193	1	3
194	1	3
195	1	3
196	1	3
197	1	3
198	1	3
199	1	3
200	1	3
201	1	3
202	1	3
203	1	3
204	1	3
205	1	3
206	1	3
207	1	3
208	1	3
209	1	3
210	1	3
211	1	3
212	1	3
213	1	3
214	1	3
215	1	3

Dari Tabel 4.8 dapat disimpulkan bahwa hasil *fitness* pada kromosom tersebut ialah sebesar 150, angka tersebut di dapat karena hasil dari sistem yang sesuai dengan hasil uji pakar adalah sebanyak 150 data dari 215 data yang ada.

4.3.2 Inisialisasi populasi awal

Seperti yang telah dijelaskan sebelumnya proses ini menggunakan kromosm yang panjangnya 14 gen, 10 gen untuk 5 kriteria dan 4 gen untuk fungsi keanggotaannya hasil keputusan. Jika *pop-size* yang digunakan adalah 5 maka dapat dihasilkan populasi awal sebagai berikut :

Tabel 4. 9 Inisialisai Populasi Awal

ind	*t3-resin*		*thyroxin serum*		*Triiodothyronine Serum*		*thyroid-stimu latin hormone*		maksimal perbedaan TSH		Hasil Diagnosa			
p1	314	234	609	354	632	567	433	368	372	159	89	726	313	53
p2	368	427	814	132	850	69	347	601	529	337	249	92	63	138
p3	432	163	54	682	355	152	259	714	965	458	198	809	622	368
p4	630	618	147	365	173	365	63	92	917	439	783	807	286	871
p5	165	743	59	119	956	233	880	629	786	477	347	324	475	168

Keterangan :
ind = individu

Setelah diakukan inisialisasi langkah selanjutnya adalah mengkonversi dan mengurutkan populasi awal tersebut, hasilnya dapat dilihat pada Tabel 4.10 dan 4.11 berikut :

Tabel 4. 10 Tabel hasil konversi sesuai kegunaan setiap segmen

ind	*t3-resin*		*thyroxin serum*		*Triiodo thyronine Serum*		*thyroid-stimu latin hormone*		maksimal perbedaan TSH		Hasil Diagnosa			
p1	89,806	83,486	15,603	9,279	6,394	5,757	24,478	20,818	28,204	16,063	8,9	72,6	31,3	5,3
p2	94,072	98,733	20,687	3,774	8,530	0,876	19,636	33,936	37,153	26,209	24,9	9,2	6,3	13,8
p3	99,128	77,877	1,839	17,414	3,679	1,690	14,682	40,298	62,005	33,106	19,8	80,9	62,2	36,8
p4	114,770	113,822	4,146	9,552	1,895	3,777	3,647	5,280	59,269	32,023	78,3	80,7	28,6	87,1
p5	78,035	123,697	1,963	3,451	9,569	2,483	49,644	35,513	51,802	34,189	34,7	32,4	47,5	16,8

Tabel 4. 11 Hasil pengurutan tiap segmen

ind	*t3-resin*		*thyroxin serum*		*Triiodo thyronine Serum*		*thyroid-stimu latin hormone*		maksimal perbedaan TSH		Hasil Diagnosa				*Fitness*
p1	83,486	89,806	9,279	15,603	5,757	6,394	20,818	24,478	16,063	28,204	5,3	8,9	72,6	31,3	30
p2	94,072	98,733	3,774	20,687	0,876	8,530	19,636	33,936	26,209	37,153	9,2	6,3	13,8	24,9	30
p3	77,877	99,128	1,839	17,414	1,690	3,679	14,682	40,298	3,106	62,005	19,8	36,8	62,2	80,9	150
p4	113,822	114,770	4,146	9,552	1,895	3,777	3,647	5,280	32,023	59,269	28,6	78,3	80,7	87,1	149
p5	78,035	123,697	1,963	3,451	2,483	9,569	35,513	49,644	34,189	51,802	16,8	32,4	34,7	47,5	150

4.3.3 Reproduksi

Reproduksi bertujuan untuk menghasilkan kromosom-kromosom baru (*offspring*) melalui proses *crossover* dan *mutasi*.

4.3.3.1 Pindah Silang (Crossover)

Metode yang diplih dalam permasalahan pada paenelitian ini adalah metode *extended intermediate crossover* (Muhlenbein dan Schlierkamp-Voosen, 1993). Metode ini dapat menghasilkan dua *offspring* dalam sekali penggunaan yang dikombinasikan oleh dua induk yang berbeda. Cara mengimplementasikan metode ini yaitu mengmabil dua induk secara acak terlebih dahulu, missal kedua induk yang terpilih tersebut diinisialisasi menjadi P_1 dan p_2 maka *offspring* C_1 dan C_2 di bangkitkan dengan persamaan 4.3 berikut ini :

$$C1 = P1 + a\ (P2-P1) \qquad (4.3)$$

$$C2 = P2 + a\ (P1-P2)$$

Keterangan :

a = nilai yang dipilih secara acak pada *interval* [-0.25,1.25].

contoh cara memperoleh *offspring crossover* dengan menggunakan metode *extended intermediate crossover* :

- pilih dua induk secara acak, misal induk terpilih adalah P2 dan P4.
- mencari nilai a misal 0.5375 yang diperoleh dengan mengacak angka antara nilai -0.25 hingga 1.25.
- maka diperoleh C1 dan C2 dengan dengan menggunakan persamaan 4.3

Tabel 4. 12 Cara mencari Offspring dari metode *Extended Intermediate Crossover*

C1 (*Offspring* Pertama)	C2 (*Offspring* Kedua)
368 + 0.5375 (630 - 368) = 508,825	630 + 0.5375 (368 - 630) = 489,175
427 + 0.5375 (618 - 427) = 529,6625	618 + 0.5375 (427 - 618) = 515,3375
814 + 0.5375 (147 - 814) = 455,4875	147 + 0.5375 (814 - 147) = 505,5125
132 + 0.5375 (365 - 132) = 257,2375	365 + 0.5375 (132 – 365)= 239,7625
850 + 0.5375 (173 - 850) = 486,1125	173 + 0.5375 (850 - 173) = 536,8875
69 + 0.5375 (365 - 69) = 228,1	365 + 0.5375 (95 - 365) = 219,875
347 + 0.5375 (63-347) = 194,35	63 + 0.5375 (347 – 63) = 215,65
601 + 0.5375(92-601) = 327,4125	92 + 0.5375 (601 - 92)= 365,5875
529 + 0.5375(917 - 529) = 737,55	917 + 0.5375 (529 – 917)= 708,45
337 + 0.5375(439 - 337) = 391,825	439 + 0.5375 (337 - 439) = 384,175
249 + 0.5375 (783 - 249) = 536,025	783 + 0.5375 (249 - 783)= 495,975

Tabel 4.12 Cara mencari Offspring dari metode *Extended Intermediate Crossover*

C1 (Offspring Pertama)	C2 (Offspring Kedua)
92 + 0.5375(807 - 92) = 476,3125	807 + 0.5375 (92 - 807) = 422,6875
63 + 0.5375 (286 - 63) = 182,8625	286 + 0.5375 (63 - 286) = 166,1375
138 + 0.5375 (871 - 138) = 531,9875	871 + 0.5375 (138 - 871) = 477,0125

Sebelumnya telah ditentukan bahwa *crossover rate*(cr)nya adalah sebesar 0,4 dan *population size* sebesar 5 maka *offspring* dari hasil *crossover* adalah sebanyak 2 individu baru yang diperoleh dari cr**population size* yaitu 0,4*5 =2 . Setiap *crossover* yang dilakukan maka akan menghasilkan 2 individu baru maka harus dilakukan 1 kali *crossover* dalam satu iterasi untuk mendapatkan 2 *offspring.* Contoh satu kali proses *crossover* dengan menggunakan induk P2 dan P4 dapat dilihat pada tabel 4.13 dibawah ini:

Tabel 4. 13 Crossover

ind	Kromosom													
P1	368	427	814	132	850	69	347	601	529	337	249	92	63	138
P2	630	618	147	365	173	365	63	92	917	439	783	807	286	871
C1	508, 825	529, 6625	455, 4875	257, 2375	486, 1125	228, 1	194, 35	327, 4125	737, 55	391, 825	536, 025	476, 3125	182, 8625	531, 9875
C2	489, 175	515, 3375	505, 5125	239, 7625	536, 8875	219, 875	215, 65	365, 5875	708, 45	384, 175	495, 975	422, 6875	166, 1375	477, 0125

4.3.3.2 Mutasi

Setelah melakukan *crossover* maka masuk ke dalam tahap berikutnya yaiu reproduksi dengan cara mutasi. Metode yang digunakan kali ini adalah *random mutation*. Cara kerja dari metode ini adalah memilih salah satu induk kemudian mengambil secara acak salah satu nilai gen pada salah satu titik kromosom yang terpilih. Nilai gen tersebut akan ditambah atau dikurangi dengan dengan bilangan *random* yang kecil. Metode ini juga menggunakan nilai minimal serta maksimal dari suatu gen terpilih. Misalkan x_i merupakan gen terpilih, maka cara mencari nilai gen baru sesuai teori *random mutation* adalah pada persamaan 4.4 berikut:

$$x_i' = x_i + r\,(maks-min) \qquad (4.4)$$

keterangan :
x = nilai dari salah satu gen terpilih
r = nilai random pada *range* [-0.1 , 0.1]
maks = nilai maksimal sesuai kegunaan gen terpilih
min = nilai minimal sesuai kegunaan gen terpilih

Contoh cara memperoleh *offspring* mutasi dengan menggunakan metode *random mutation* :

- pilih satu induk secara acak, misal induk terpilih adalah P3.
- Indeks gen pada kromosom yang terpilih adalah pada indeks ke 5.
- Karena indeks ke- 5 merupakan nilai gen untuk segmen *triiodothyronine Serum* maka nilai maks = 10 dan min = 0.2.
- mencari nilai r misal 0.0956 yang diperoleh dengan mengacak angka antara nilai -0.1 hingga 0.1.
- maka diperoleh C3 dengan dengan menggunakan persamaan 4.4

$x_{5} =$ 355 + 0.0956(10-0.2) = 355,93688

Ukuran *offspring* diperoleh dari perkalian antara population size dan *mutation rate*, sehingga nilai *offspring* yang dihasilkan adalah 0,2 * 5 = 1. Dalam melakukan mutasi diperlukan 1 hasil *offspring* sehingga membutuhkan 1 kali proses mutasi dalam sekali iterasi. Contoh satu kali proses mutasi dengan menggunakan induk p3 dapat dilihat pada Tabel 4.16 dibawah ini:

Tabel 4. 14 Mutasi

ind	Kromosom													
P3	432	163	54	682	355	152	259	714	965	809	198	458	622	368
C3	432	163	54	682	355,93688	152	259	714	965	809	198	458	622	368

4.3.4 Evaluasi dan Seleksi

Pada proses evaluasi semua individu dari populasi awal, hasil dari *crossover* dan juga hasl dari *mutasi* digabungkan, selain itu setiap kromosom dari individu diberi nilai *fitness* sehingga dapat diseleksi sesuai nilai *fitness* masing-masing.

Tabel 4. 15 Individu Gabungan

ind	Kromosom													
p1	314	234	609	354	632	567	433	368	372	159	89	726	313	53
p2	368	427	814	132	850	69	347	601	529	337	249	92	63	138
p3	432	163	54	682	355	152	259	714	965	458	198	809	622	368
p4	630	618	147	365	173	365	63	92	917	439	783	807	286	871
p5	165	743	59	119	956	233	880	629	786	477	347	324	475	168
p6	508,825	529,6625	455,4875	257,2375	486,1125	228,1	194,35	327,4125	737,55	391,825	536,25	476,3125	182,8625	531,9875
p7	489,175	515,3375	505,5125	239,7625	536,8875	219,875	215,65	365,5875	708,45	384,175	495,975	422,6875	166,1375	477,125
p8	432	163	54	682	355,93688	152	259	714	965	809	198	458	622	368

Tabel 4. 16 Individu gabungan setelah diurutkan dan dikonversi

ind	*t3-resin*		*thyroxin serum*		*Triiodo thyronine Serum*		*thyroid-stimu latin hormone*		maksimal perbedaan TSH		Hasil Diagnosa				*Fit-ness*
p1	83,49	89,81	9,28	15,60	5,76	6,39	20,82	24,48	16,06	28,20	5,3	8,9	72,6	31,3	30
p2	94,07	98,73	3,77	20,69	0,88	8,53	19,64	33,94	26,20	37,15	9,2	6,3	13,8	24,9	30
p3	77,88	99,13	1,84	17,41	1,69	3,68	14,68	40,3	3,11	62,01	19,8	36,8	62,2	80,9	150
p4	113,82	114,77	4,14	9,55	1,9	3,78	3,65	5,28	32,02	59,27	28,6	78,3	80,7	87,1	149
p5	78,04	123,70	1,96	3,45	2,49	9,57	35,51	49,64	34,19	51,80	16,8	32,4	34,7	47,5	150
p6	105,2	106,84	6,88	11,8	2,43	4,96	11,04	18,53	29,33	49,04	18,29	47,63	53,2	53,63	150
p7	103,64	105,71	6,45	13,04	2,36	5,46	12,24	20,68	28,90	47,38	16,61	42,27	47,71	49,6	150
p8	77,88	99,13	1,84	17,41	1,69	3,69	14,68	40,3	53,11	62,01	19,8	36,8	45,8	62,2	150

Setelah diketahui semua nilai *fitness* dari masing-masing kromosom maka langkah selanjutnya adalah memfilter individu gabungan tersebut sejumlah *popSize* dalam kasus ini sebesar 5 individu untuk populasi baru.

Seleksi berguna untuk memfilter populasi gabungan yang dimasuki oleh *offspring* dari *crossover* dan mutasi dipilih sebanyak jumlah populasi dengan metode tertentu untuk mendapatkan keturunan yang lebih baik lagi. Pada penelitian kali ini metode yang digunakan adalah *tournament selection, rouette wheel selection* dan *elitism selection* yang nantinya akan dibandingkan dengan menguji setiap seleksi tersebut.

Proses dari *tournament selection* adalah dengan melombakan 2 kromosom dari individu gabungan dari populasi, *offspring crossover* dan *offspring mutasi.* Kedua kromosom itu dilombakan dengan mancri nilai *fitness* yang lebih tinggi. Individu yang memiliki *fitness* yang lebih tinggi dari lawannya akan lolos menjadi individu yang akan dipakai untuk populasi yang baru. Hal ini dilakukan hingga tercapai ukuran populasi atau disebut *popSize.*

Cara seleksi dengan metode *tournament selection* dapat dilihat dibawah ini,individu yang menang dalam tournament akan dimasukkan dalam populasi baru :

- P3 dan P5 = 150 vs 150
- P1 dan P7 = 30 vs 150
- P4 dan P5 = 149 vs 150
- P6 dan P9 = 150 vs 150
- P8 dan P6 = 149 vs 149

Hasil seleksi

Tabel 4. 17 Populasi baru hasil tournament selection

ind	kromosom														*Fitness*
p1	432	163	54	682	355	152	259	714	965	458	198	809	622	368	150
p2	489, 175	515, 3375	505, 5125	239, 7625	536, 8875	219, 875	215, 65	365, 5875	708, 45	384, 175	495, 975	422, 6875	166, 1375	477, 0125	150
p3	165	743	59	119	956	233	880	629	786	477	347	324	475	168	150
p4	508, 825	529, 6625	455, 4875	257, 2375	486, 1125	228, 1	194, 35	327, 4125	737, 55	391, 825	536, 025	476, 3125	182, 8625	531, 9875	150
p5	432	163	54	682	355, 93688	152	259	714	965	809	198	458	622	368	150

Untuk proses dari *elitism selection* yaitu dengan mengurutkan individu dari yang memiliki nilai *fitness* tertinggi ke individu yang memiliki nilai *fitness* terendah. Setelah disorting individu-individu terbaik diambil sebanyak ukuran populasi.

Tabel 4. 18 Hasil pengurutan individu gabungan sesuai fitness

ind	*t3-resin*		*thyroxin serum*		*Triiodo thyronine Serum*		*thyroid-stimu latin hormone*		maksimal perbedaan TSH		Hasil Diagnosa				Fitness
p3	77,88	99,13	1,84	17,41	1,69	3,68	14,68	40,3	3,11	62,01	19,8	36,8	62,2	80,9	150
p5	78,04	123,70	1,96	3,45	2,49	9,57	35,51	49,64	34,19	51,80	16,8	32,4	34,7	47,5	150
p6	105,2	106,84	6,88	11,8	2,43	4,96	11,04	18,53	29,33	49,04	18,29	47,63	53,2	53,63	150
p7	103,64	105,71	6,45	13,04	2,36	5,46	12,24	20,68	28,90	47,38	16,61	42,27	47,71	49,6	150
p8	77,88	99,13	1,84	17,41	1,69	3,69	14,68	40,3	53,11	62,01	19,8	36,8	45,8	62,2	150
p4	113,82	114,77	4,14	9,55	1,9	3,78	3,65	5,28	32,02	59,27	28,6	78,3	80,7	87,1	149
p1	83,49	89,81	9,28	15,60	5,76	6,39	20,82	24,48	16,06	28,20	5,3	8,9	72,6	31,3	30
p2	94,07	98,73	3,77	20,69	0,88	8,53	19,64	33,94	26,20	37,15	9,2	6,3	13,8	24,9	30

Tabel 4. 19 Populasi baru hasil elitism selection

Ind	*t3-resin*		*thyroxin serum*		*Triiodo thyronine Serum*		*thyroid-stimu latin hormone*		maksimal perbedaan TSH		Hasil Diagnosa				Fitness
p1	77,88	99,13	1,84	17,41	1,69	3,68	14,68	40,3	3,11	62,01	19,8	36,8	62,2	80,9	150
p2	78,04	123,70	1,96	3,45	2,49	9,57	35,51	49,64	34,19	51,80	16,8	32,4	34,7	47,5	150
p3	105,2	106,84	6,88	11,8	2,43	4,96	11,04	18,53	29,33	49,04	18,29	47,63	53,2	53,63	150
p4	103,64	105,71	6,45	13,04	2,36	5,46	12,24	20,68	28,90	47,38	16,61	42,27	47,71	49,6	150
p5	77,88	99,13	1,84	17,41	1,69	3,69	14,68	40,3	53,11	62,01	19,8	36,8	45,8	62,2	150

Sedangkan proses dari *roulette wheel selection* yaitu dengan menjumlahkan semua *fitness* dari kromosom pada individu gabungan. Misal *finess*(P_k) merupakan nilai *fitness* individu ke-k maka persamaan untuk menghitung total *fitness* adalah sebagai berikut:

$$TotalFitness = \sum_{k=1}^{PopSize} fitness(P_k)$$

Kemudian membagi nilai *fitness*(P_k) pada setiap kromosom dengan *totalFitness* seluruh kromosom yang disebut sebagai nilai probabilitas seleksi $Prob_k$. caranya dapat dilihat pada persamaan berikut ini:

$$Prob_k = \frac{fitness}{totalFitness}, \qquad k = 1,2, \dots, popSize$$

Setelah mendapatkan nilai $Prob_k$ langkah selanutnya yaitu menghiung nilai probabilitas kumulatif (*ProbCum*) dengan mengguankan persamaan sebagai berikut :

$$probCum_k = \sum_{j=1}^{k} prob_j, \quad k = 1,2, \dots, popSize$$

hasil perhitungan *totalFitness,* $Prob_k$ dan $probCum_k$ pada setiap individu dapat dilihat pada tabel berikut :

Tabel 4. 20 Individu yang telah dihitung nilai prob dan probCum

K	individu	*Fitness*	*prob*	*probCum*
1	P1	30	0.0313	0.0313
2	P2	30	0.0313	0.0626
3	P3	150	0.1564	0.219
4	P4	149	0.1554	0.3744
5	P5	150	0.1564	0.5308
6	C1	150	0.1564	0.6872
7	C2	150	0.1564	0.8438
8	C3	150	0.1564	1
Jumlah *fitness*		959		

Untuk memilih individu yang akan diloloskan sebagai individu pada populasi baru dengan menggunakan roulette wheel selection dilakukan beberapa langkah berikut :

- Membangkitkan bilangan random untuk *r* dengan *range* [0,1], misal *r* = 0.5254
- Cek dan pilih nilai probCum pada setiap k yang memenuhi persyaratan $r \leq probCum_k$, dengan nilai *r* = 0.5254 maka k yang terpilih adalah k = 5. Ini berarti bahwa individu ke-5 tepilih menjadi individu yang masuk pada populasi baru.

Langkah-langkah tersebut dilakukan sebanyak ukuran *popSize*, sehingga terdapat 5 kali pencarian nilai *r* pada ukuran *popSize* = 5. Hasil seleksi menggunakan metode *roulette wheel selection* dapat dilihat pada Tabel 4.21 berikut:

Tabel 4. 21 Individu yang lolos seleksi

r	*probCum*	K	individu
0.5254	0.5308	5	P5
0.3200	0.3744	4	P4
0.2004	0.219	3	P3
0.0433	0.0626	2	P2
0.9592	1	8	C3

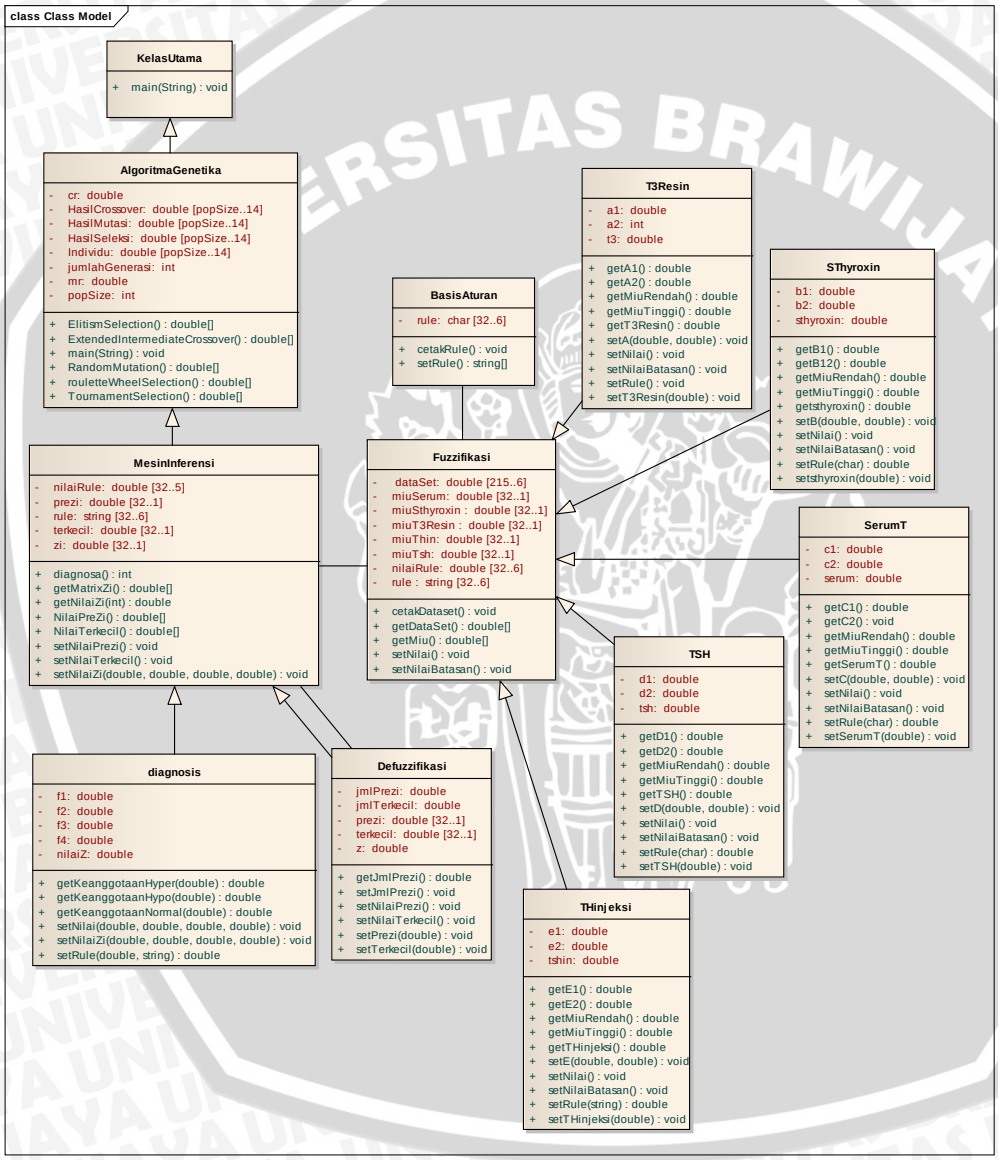
Tabel 4. 22 Populasi Baru hasil roulette wheel selection

Ind baru	kromosom														*Fitness*
P1	165	743	59	119	956	233	880	629	786	477	347	324	475	168	150
P2	630	618	147	365	173	365	63	92	917	439	783	807	286	871	149
P3	432	163	54	682	355	152	259	714	965	458	198	809	622	368	150
P4	368	427	814	132	850	69	347	601	529	337	249	92	63	138	30
P5	432	163	54	682	355, 93688	152	259	714	965	809	198	458	622	368	150

BAB 5 IMPLEMENTASI

Pada bab 5 ini akan dijelaskan struktur kelas pada program dan kode program untuk menjalankan fungsi-fungsi utama sistem diagnosa penyakit tiroid dengan menggunakan FIS Tsukamoto yang fungsi keanggotaanya telah di optimasi dengan Algoritma Genetika.

5.1 Struktur Kelas



Gambar 5. 1 Kelas Diagram Sistem

Sistem ini dijalankan dari beberapa kelas yang disesuaikan dengan jalan dari algoritma. Pada program terdapat *source code* untuk FIS Tsukamoto dan *source* untuk Algoritma Genetika. Proses FIS Tsukamoto dilakukan oleh kelas-kelas berikut yaitu BasisAturan.java, Fuzzifikasi.java, MesinInferensi.java dan

Defuzzifikasi.java. Pada kelas defuzzifikasi memiliki beberapa turunan yang merupakan parameter tes yang dibutuhkan yaitu kelas T3Resin, Sthyroxin, SerumT, TSH dan THInjeksi. Sedangkan pada kelas MesinInferensi menurunkan kelas diagnosis dan Defuzzifikasi.

Pada pemograman untuk Algoritma Genetika di implementasikan pada kelas AlgoritmaGenetika, kelas ini menurunkan kelas MesinInferensi sebagai evaluasi metode algortima genetika. Dalam menjalankan program terdapat kelas utama yaitu kelasUtama.java yang menurunkan kelas Algoritma Genetika.

5.2 Implementasi Kode Program

Sub bab ini menjelaskan tentang kode program pada fungsi-fungsi utama yang digunakan untuk menjalankan sistem beserta keterangannya.

5.2.1 Implemetasi Proses Algortima Genetika

Pada sub bab ini menjelaskan kode program yang digunakan untuk melakukan proses Algoritma Genetika. Didalamnya terdapat proses inisialisasi awal, reproduksi dan seleksi. Pada proses reproduksi dibagi menjadi dua proses lagi yaitu *crossover* yang menggunakan metode *extended intermediate crossover* dan mutasi yang menggunakan metode *random mutation.* Sedangkan untuk proses Seleksi menggunakan tiga metode yaitu *elitism selection, tournament selection* dan *rhoulette wheel selection* yang akan diuji optimasinya jika digunakan untuk kasus diagnosa penyakit menggunakan metode FIS Tsukamoto yang fungsi keanggotaanya dioptimasi menggunakan metode Algoritma Genetika.

5.2.1.1 Implementasi Proses Inisialisasi Awal

Inisialisasi populasi awal ini dilakukan dengan mengisi nilai gen pada kromosom secara acak pada nilai antara 0 hingga 1000. Inisialisasi awal ini mengacak nilai tersebut sejumlah panjang kromosom dan jumlah populasi. Panjang kromosom jumlahnya sesuai kebutuhan pada penelitian ini yaitu 14 gen pada satu kromosom. Sedangkan jumlah populasi sebanyak masukan dari pengguna. Contoh kode program inisialisasi awal ada pada Gambar 5.2 berikut:

```
1  Random random = new Random();
2  System.out.println("");
3          for (int i = 0; i < Pop_Size ; i++) {
4              for (int j = 0; j < pj; j++) {
5                  int nilaiTerendah=0, nilaiTertinggi=1000;
6   Individu[i][j]    =    random.nextInt(nilaiTertinggi    -
7  nilaiTerendah + 1) + nilaiTerendah;
8              }
9          }
```

Gambar 5.2 Kode program inisialisasi populasi awal

Penjelasan baris kode program pada Gambar 5.1 sebagai berikut:

1. Baris 1 membangkitkan fungsi *random.*

2. Baris 3-10 membangkitkan nilai random antara 0 hingga 1000, yang dilakukan berulang-ulang sehingga setiap gen pada kromosom yang jumlahnya sebanyak populasi terisi oleh nilai acak tersebut.

5.2.1.2 Implementasi Proses Reproduksi

Menjelaskan kode program yang digunakan untuk proses *crossover* yang menggunakan metode *Extended Intermediate Crossover* dan proses mutasi yang menggunakan metode *Random Mutation.*

5.2.1.2.1 Implemetasi Proses *Crossover*

Pada penelitian ini metode *crossover* yang diapakai adalah *extended intermediate crossover*. Proses ini memnutuhkan 2 kromosom dari 2 individu yang berbeda danmenghasilkan 2 *offspring* baru. Cara kerja metode ini adalah dengan mencari nilai yang berinisial a secara acak dengan *range* [-0.25, 1.25]. *offspring* pertama didapatkan dengan cara menambahkan nilai pada setiap gen pada kromosom dari individu yang terpilih pertama dengan hasil kali a dengan nilai gen pada kromosom dari indivdidu yang terpilih kedua yang dikurangi dengan nilai gen pada kromosom dari individu yang terpilih pertama. Kode program untuk metode *extended intermediate crossover* dapat dilihat pada Gambar 5.3 berikut:

```
1     Random random = new Random();
2           int panjangkromsosom = Individu[0].length;
3           int popsize = Individu.length;
4           int a[]=new int [panjangkromsosom];
5   double aKonversi[] = new double [panjangkromsosom];
6   double[][]Hasil = new double[2][panjangkromsosom];
7   double[][]IndukPertama=new double[1][panjangkromsosom];
8   double[][]IndukKedua = new double[1][panjangkromsosom];
9           int nilaiTerendah = 0;
10          int nilaiTertinggi = popsize - 1;
11          boolean flag = true;
12          int IndeksIndukPertama = 0,IndeksIndukKedua = 0;
13          while (flag) {
14  IndeksIndukPertama = random.nextInt(nilaiTertinggi -
15  nilaiTerendah + 1) + nilaiTerendah;
16  IndeksIndukKedua = random.nextInt(nilaiTertinggi -
17  nilaiTerendah + 1) + nilaiTerendah;
18              if (IndeksIndukPertama != IndeksIndukKedua) {
19                  flag = false;
20               }
21          }
22  System.out.print("P"+ (IndeksIndukPertama + 1)+"= ");
23          for (int j = 0; j < panjangkromsosom; j++) {
24  System.out.print(Individu[IndeksIndukPertama][j] + " ");
25          }
26        System.out.println();
27        System.out.print("P" +(IndeksIndukKedua + 1)+"=");
28          for (int j = 0; j < panjangkromsosom; j++) {
29  System.out.print(Individu[IndeksIndukKedua][j] + " ");
30          }
```

```
31          System.out.println();
32  for (int k=0;k<panjangkromsosom;k++){
33           int nilaiTerendahA= 0, nilaiTertinggiA = 1000;
34          a[k]= random.nextInt(nilaiTertinggiA -
35  nilaiTerendahA + 1) + nilaiTerendahA;
36          //konversi menjadi -0,25-1,25
37          aKonversi[k]=((a[k]/1000)*1.5)-0.25;
38          for (int i = 0; i < panjangkromsosom; i++) {
39       IndukPertama[0][i]=Individu[IndeksIndukPertama][i];
40          }
41          for (int i = 0; i < panjangkromsosom; i++) {
42       IndukKedua[0][i] = Individu[IndeksIndukKedua][i];
43          }
44          for (int i = 0; i < panjangkromsosom; i++) {
45              Hasil[0][i] = IndukPertama[0][i] +
46  (aKonversi[k]*(IndukKedua[0][i]-IndukPertama[0][i]));
47              Hasil[1][i] = IndukKedua[0][i] +
49  (aKonversi[k]*(IndukPertama[0][i]-IndukKedua[0][i]));
50          }
51  }
52          // menampilkan hasil crossover
53          for (int i = 0; i < 2; i++) {
54              System.out.print("C" + (i + 1) + " = ");
55              for (int j = 0; j < panjangkromsosom; j++) {
56                  System.out.print(Hasil[i][j] + " ");
57               }
58               System.out.println();
```

Gambar 5.3 Kode Program proses *crossover*

Penjelasan baris kode program pada Gambar 5.3 sebagai berikut:

1. Baris 1 membangkitkan fungsi *random.*
2. Baris 2 dan 3 inisialisasi *integer* untuk mengetahui panjang kromosom dan jumlah populasi dari parameter yang dimasukkan.
3. Baris 4 inisialisasi *array integer* 1 dimensi untuk menyimpan nilai a acak pada *range* 0-1000.
4. Baris 5 inisialisasi *array double* 1 dimensi untuk menyimpan hasil konversi nilai a pada *range* -0,25 dan 1,25.
5. Baris 6 inisialisasi *array double* 2 dimensi untuk menyimpan hasil proses *crossover.*
6. Baris 7 inisialisasi *array double* 2 dimensi untuk menyimpan induk pertama terpilih.
7. Baris 8 inisialisasi *array double* 2 dimensi untuk menyimpan induk kedua terpilih.
8. Baris 9 dan 10 inisialisasi variabel *integer* sebagai batas atas dan bawah dalam pencarian induk yang akan dipilih.
9. Baris 11-21 mencari dua induk secara acak dari populasi.
10. Baris 22-30 mencetak induk yang terpilih.
11. Baris 34 dan 35 mengacak nilai a antara 0 hingga 1000.
12. Baris 37 mengkonversi nilai a sesuai *range* -0.25 hingga 1.25.

13. Baris 38-43 mengkopi nilai pada induk terpilih kedalam *array* yang disediakan.
14. Baris 44-50 melakukan *crossover* dengan metode *extended intermediate.*
15. Baris 53-58 mencetak hasil *crossover.*

5.2.1.2.1 Implementasi proses mutasi

Metode *random mutation* membutuhkan satu induk dan satu titik potong pada kromosom yang dipilih secara acak untuk membuat *offspring* baru. Prosesnya adalah dengan mencari nilai berinisial *r* terlebih dahulu, nilai *r* merupakan nilai kecil yang dipilh secara acak pada *range* [-0.1, 0,1]. *Offspring* dibentuk dengan nilai gen pada titik potong yang terpilih pada kromosom dengan *r* yang telah dikalikan dengan hasil pengurangan dari nilai maksimal dan minimal pada gen tersebut. Kode program proses mutasi dapat dilihat pada Gambar 5.3 berikut:

```
1   Random random = new Random();
2   int panjangKromosom = Individu[0].length;
3   int popsize = Individu.length;
4   double[][] Hasil = new double[1][panjangKromosom];
5  int nilaiTerendahTitik = 0,
6  nilaiTertinggiTitik = panjangKromosom-1 ;
7          int posisi=0;
8          posisi = random.nextInt(nilaiTertinggiTitik -
9  nilaiTerendahTitik + 1) + nilaiTerendahTitik;
10         int Random = posisi+1;
11 System.out.println("Titik Point Terpilih = " + Random);
12         int nilaiTerendah = 0;
13         int nilaiTertinggi = popsize - 1;
14         int IndeksInduk = random.nextInt(nilaiTertinggi
15 - nilaiTerendah + 1) + nilaiTerendah;
16       System.out.print("P"  + (IndeksInduk + 1) + "= ");
17          for (int j=0; j<panjangKromosom; j++) {
18        System.out.print(Individu[IndeksInduk][j] + " ");
19         System.out.println();
20         int nilaiTerendahR= 0, nilaiTertinggiR = 1000;
21         int rAwal= random.nextInt(nilaiTertinggiR -
22 nilaiTerendahR + 1) + nilaiTerendahR;
23         double r =((rAwal/1000)*0.2)-0.1;
24             for (int j = 0; j < panjangKromosom ; j++) {
25 Hasil[0][j]=Individu[IndeksInduk][j]; }
26         double maks=1000;
27         double min=0;
28         double HasilGen=
29 Individu[IndeksInduk][posisi]+(r*(maks-min));
30          Hasil [0][posisi]= HasilGen;
31          System.out.print("C = ");
32          for (int j = 0; j < panjangKromosom ; j++) {
33              System.out.print(Hasil[0][j] + " ");}
```

Gambar 5. 4 Kode program proses mutasi

Penjelasan baris kode program pada Gambar 5.3 sebagai berikut:

1. Baris 1 membangkitkan fungsi *random*.
2. Baris 2 dan 3 inisialisasi *integer* untuk mengetahui panjang kromosom dan jumlah populasi dari parameter yang dimasukkan.
3. Baris 4 inisialisasi *array double* 2 dimensi untuk menyimpan hasil mutasi.
4. Baris 5-6 inisialisasi variabel *integer* sebagai batas atas dan bawah dalam pencarian titik kromosom yang akan dipilih.
5. Baris 7-9 memilih titik pada kromosom.
6. Baris 11 menampilkan titik yang terpilih.
7. Baris 12-15 mengacak individu yang akan dipilih menjadi induk pada proses mutasi.
8. Baris 16 menampilkan indeks individu yang terpilih.
9. Baris 17-19 menampilkan kromosom individu yang terpilih.
10. Baris 20-23 mencari nilai *r* dengan mengacak nilai antara -0.1 dan 0.1.
11. Baris 24-25 mengkopi individu terpilih pada *array* yang telah disediakan.
12. Baris 26-29 melakukan proses *random* mutasi.
13. Baris 30 mengkopi nilai gen yang telah dimutasi dalam *array* hasil.
14. Baris 31-33 mencetak hasil mutasi.

5.2.1.3 Implemetasi Evaluasi (Mencari Nilai *Fitness*)

Nilai *fitness* didapatkan dari suatu kromosom dengan menguji kromosom tersebut ketika dimasukkan sebagai batasan pada fungsi keanggotaan sesuai kegunaan setiap segmen pada kromosom. Caranya dengan membandingkan hasil diagnosa pada sistem dengan hasil diagnosa dari pakar yang didapatkan dari UCI *repository* dan berjumlah 215 data. Semakin besar data yang memiliki hasil diagnosa sistem yang sama dengan hasil diagnosa pakar maka semakin besar nilau *fitness*nya. Kode program dapat dilihat pada Gambar 5.5 berikut:

```
1           int panjangkromosom= 14;
2           int angka  = 0;
3           int diagnosis [] = new int[215];;
4           double[][] dataSet = new double[215][6];
5
6           MesinInferensi MI = new MesinInferensi();
7           Fuzzifikasi fuz = new Fuzzifikasi();
8           dataSet=fuz.getDataSet();
9           for(int j=0; j<215; j++){
10              diagnosis[j]=MI.diagnosa(dataSet[j][1],
11  dataSet[j][2], dataSet[j][3], dataSet[j][4],
12  dataSet[j][5], Individu [0], Individu [1], Individu [2],
13  Individu [3], Individu [4], Individu [5], Individu [6],
14  Individu [7], Individu [8], Individu [9], Individu [10],
15  Individu [11], Individu [12], Individu [13]);
16          }
17                  for (int j=0;j<215;j++ ){
18              if (diagnosis[j]==dataSet[j][0]){
19                  angka ++;
20              }
21          }
```

```
22                System.out.println("");
23            return angka;
24        }
```

Gambar 5.5 Kode program proses evaluasi

Penjelasan baris kode program pada Gambar 5.5 sebagai berikut:

1. Baris 1 inisialisasi *integer* untuk mengetahui panjang kromosom dari parameter yang dimasukkan.
2. Baris 2 inisialisasi *integer* untuk menyimpan hasil *fitness*.
3. Baris 3 inisialisasi *array integer* 1 dimensi untuk menyimpan hasil diagnosa sistem.
4. Baris 4 inisialisasi *array double* 2 dimensi untuk menyimpan data set yang ada.
5. Baris 8 mengambil niai dataset dari fungsi yang ada di kelas fuzzfikasi.
6. Baris 10-16 mengambil hasil diagnosa sebanyak dataset dari metode FIS Tsukamoto yang fungsi keanggotaannya memakai nilai pada tiap kromosom yang diuji.
7. Baris 17-21 mencari nilai *fitness* dengan menambahkan variabel angka dengan 1 jika hasil diagnosa sistem sama dengan hasil diagnosa pakar.

5.2.1.4 Implemetasi Seleksi

Cara menyeleksi induk dan hasil *offspring* pada setiap iterasi dalam penelitian ini menggunakan 3 metode berbeda yang hasilnya masing-masing akan dibandingkan sebegai metode terbaik untuk kasus mencari kromosom terbaik yaitu kromosom yang nilai gennya dapat digunakan sebagai batasan pada sistem diagnosa penyakit tiroid menggunakan metode FIS Tsukamoto dengan waktu yang cepat.

Ketiga metode seleksi yang akan diuji adalah *elitism selection, tournament selection dan rhoulette wheel selection.* Pada metode *elitism selection* individu dari induk dan *offspring* digabungkan dan di urutkan dari individu yang memiliki *fitness* terbesar kemudian diambil sebanyak jumlah *popSize* dari nilai *fitness* tertinggi. Untuk metode *tournament selection* induvidu dari induk dan *offspring* digabungkan juga kemudian di ambil 2 individu secara acak yang nilai *fitness*nya akan dibandingkan, dari kedua individu tersebut yang memilki nilai *fitness* kebih besar lolos menjadi individu terpilih pada populasi baru, proses ini dilakukan sebanyak jumlah *popSize.* Metode seleksi yang terakhir yang akan digunakan adalah meode *rhoulette wheel*, langkah pertama dalam metode ini adalah menjumlahkan semua *fitness* pada setiap induk dan *offspring* hasilnya dinamai dengan *totalFitness,* kemudian nilai *fitness* pada setiap individu dibagi dengan *totalFitness* hasilnya dinamakan $prob_k$ yang mana nilai k sebagai indikasi indeks, dan terakhir yaitu dengan memberi nilai $procCum_k$ yaitu nilai $prob_k$ dari individu tersebut ditambah dengan nilai semua $prob_k$ pada individu sebelumnya. Sehingga nilai *probCum* pada individu terkahir adalah 1. Cara mendapatkan individu baru pada metode ini adalah dengan mencari nilai *r* secara acak antara 0 hingga 1. Kemudian mencocokkannya dengan nilai

probCum$_k$ yang lebih kecil dan mendekati nilai *r* tersebut, individu yangmemiliki nilai *probCum* tersebut lolos menjadi individu pada populasi baru. Nilai *r* dicari sebanyak *popSize*.

- Metode *elitism selection*

Kode program proses seleksi menggunakan metode *Elitism Selection* dapat dilihat pada Gambar 5.6 berikut:

```
1  public double [][] SeleksiElitismSimpleFx(double [][]
2  IndividuGabungan, int Pop_Size, int []angka) {
3          AlgenRealCode myGAs = new AlgenRealCode();
4          int panjangkromosom =
5  IndividuGabungan[0].length;
6  AlgenRealCode myGAs = new AlgenRealCode();
7  int panjangkromosom = IndividuGabungan[0].length;
8  int popsize = IndividuGabungan.length;
9
10 int[] IndeksAwalIndividuGabungan = new int[popsize];
11         for (int i = 0; i < popsize; i++) {
12             IndeksAwalIndividuGabungan[i] = i;
13         }
14         for (int i = 0; i < popsize; i++) {
15             int TempNilaiFx = angka[i];
16 int TempIndeksAwalIndividuGabungan =
17 IndeksAwalIndividuGabungan[i];
18 double[][] TempHasil=new double[1][panjangkromosom];
19
20             for (int j = (i + 1); j < popsize; j++) {
21                 if (angka[j] > TempNilaiFx) {
22                     TempNilaiFx = angka[j];
23                     angka[j] = angka[i];
24                     angka[i] = TempNilaiFx;
25                     TempIndeksAwalIndividuGabungan =
26 IndeksAwalIndividuGabungan[j];
27                     IndeksAwalIndividuGabungan[j] =
28 IndeksAwalIndividuGabungan[i];
29                     IndeksAwalIndividuGabungan[i] =
30 TempIndeksAwalIndividuGabungan;
31                   for(int k=0; k<panjangkromosom; k++){
32               TempHasil[0][k] = IndividuGabungan[j][k];
33                     }
34                   for(int k=0; k<panjangkromosom; k++){
35           IndividuGabungan[j][k]=IndividuGabungan[i][k];
36                     }
37                   for (int k=0; k<panjangkromosom; k++){
38 IndividuGabungan[i][k]=TempHasil[0][k];
39                     }
40               }
41             }
42         }
43 // mengambil individu gabungan yang terurut sebanyak
44 popSize
45         for (int i = 0; i < Pop_Size; i++) {
```

```
46              for (int j = 0; j < panjangkromosom; j++) {
47                  Hasil[i][j] = IndividuGabungan[i][j];
48              }
49          }
50          return Hasil;
51      }
```

Gambar 5.6 Kode program proses *elitism selection*

Penjelasan baris kode program pada Gambar 5.6 sebagai berikut:

1. Baris 1-2 judul fungsi untuk mendapatkan populasi baru dari hasil seleksi *elitism selection* berupa *array* 2 dimensi bertipe *double* dengan 3 parameter sebagai masukan yaitu individu gabungan induk dan *offspring,* jumlah *popSize* dan nilai *fitness* per individu.
2. Baris 10 membuat *array* satu dimensi sepanjang nilai popsize untuk menyimpan indeks pada tiap individu.
3. Baris 11-13 menyimpan angka indeks dari 1 hingga jumlah *popSize* pada *array integer* yang telah disesiakan sebelumnya.
4. Baris 14-43 kode program untuk mengurutkan individu sesuai nilai *fitness* nya.
5. Baris 15 membuat variable bertipe *integer* untuk menyimpan nilai *fitness* sementara pada indeks ke-i.
6. Baris 16-17 membuat variabel bertipe *integer* untuk menyimpan no indeks sementara pada individu ke-i.
7. Baris 18 membuat *array* 2 dimensi bertipe *double* untuk menyimpan kromosom sementara.
8. Baris 20 membuat indeks j yang nilai nya i+1 hingga seukuran populasi.
9. Baris 21 apabila *fitness* pada indeks ke j lebih besar dari isi dari variabel penyimpanan sementara untuk *fitness* ke-i.
10. Baris 22 nilai dari variabel sementara menjadi *fitness* ke-j.
11. Baris 23 *fitness* ke-j menjadi *fitness* ke-i.
12. Baris 24 nilai *fitness* ke-i didapat dari variabel sementara tersebut.
13. Baris 25-26 variabel sementara untuk indeks menjadi indeks ke-j.
14. Baris 26-27 indeks ke-j nilainya mengambil nilai dari indeks ke-i.
15. Baris 27-28 indeks ke-i mendapatkan nilai dari variabel indeks semetara.
16. Baris 31-33 kromosom pada hasil sementara menjadi kromosom individu gabungan ke-j.
17. Baris 34-36 kromosom pada individu gabungan ke-j diuhah sesuai kromosom dari individu gabungan ke-i.
18. Baris 37-9 nilai pada individu gabungan ke-i mengambil nilai dari hasil sementara.
19. Baris 45-49 mengambil individu gabungan yang telah di urutkan sebanyak ukuran *popSize*.
20. Baris 50 mengaembailkan hasil seleksi *elitsm selection.*

- Metode *Tournament Selection*

Kode program proses seleksi untuk metode *tournament selection* dapat dilihat pada Gambar 5.7 berikut:

```
1  public double [][] TournamentSelection(double [][]
2  Individu, int Pop_Size, int [] angka) {
3          AlgenRealCode algen = new AlgenRealCode();
4          Random random = new Random();
5          int panjangkromosom = Individu[0].length;
6          int popsize = Individu.length;
7
8  double[][] Hasil=new double[Pop_Size][panjangkromosom];
9
10         for(int i=0; i<Pop_Size; i++){
11 double[] IndukPertama = new double[panjangkromosom];
12 double[] IndukKedua = new double[panjangkromosom];
13         int nilaiTerendah = 0;
14         int nilaiTertinggi = popsize-1;
15
16         //2 induk terpilih untuk dilombakan
17         boolean flag = true;
18        int IndeksIndukPertama = 0, IndeksIndukKedua = 0;
19         while (flag) {
20             IndeksIndukPertama =
21 random.nextInt(nilaiTertinggi - nilaiTerendah + 1) +
22 nilaiTerendah;
23             IndeksIndukKedua =
24 random.nextInt(nilaiTertinggi - nilaiTerendah + 1) +
25 nilaiTerendah;
26             if (IndeksIndukPertama != IndeksIndukKedua){
27                 flag = false;
28             }
29         }
30         System.out.println("Indeks IndukPertama = " +
31 (IndeksIndukPertama + 1) + ", Indeks IndukKedua = " +
32 (IndeksIndukKedua + 1));
33
34         // menampilkan hasil induk terpilih
35         System.out.print("Individu ke-
36 "+(IndeksIndukPertama + 1) + " ");
37         for (int j = 0; j < panjangkromosom; j++) {
38 System.out.print(Individu[IndeksIndukPertama][j] + " ");
39         }
40         System.out.println();
41
42 System.out.print("Individu ke-"+(IndeksIndukKedua + 1)+
43 " ");
44         for (int j = 0; j < panjangkromosom; j++) {
45 System.out.print(Individu[IndeksIndukKedua][j] + " ");
46         }
47         System.out.println();
48
49         for (int j = 0; j < panjangkromosom; j++) {
```

```
50  IndukPertama[j]=Individu[IndeksIndukPertama][j];
51          }
52          for (int j = 0; j < panjangkromosom; j++) {
53  IndukKedua[j]=Individu[IndeksIndukKedua][j];
54          }
55  System.out.println("Fitness Individu ke- "+
56  (IndeksIndukPertama+1)+ "="+angka[IndeksIndukPertama]);
57  System.out.println("Fitness Individu ke- "+
58  (IndeksIndukKedua+1)+ " = "+angka[IndeksIndukKedua]);
59              System.out.println("");
60  if (angka[IndeksIndukPertama]>angka[IndeksIndukKedua]){
61              for (int j = 0; j < panjangkromosom; j++) {
62                  Hasil[i][j]=IndukPertama[j];
63          }
64                   }
65              else {
66              for (int j = 0; j < panjangkromosom; j++) {
67                  Hasil[i][j]=IndukKedua[j]; }
68              }
69                   }
70          return Hasil; }
```

Gambar 5.7 Kode program proses tournament selection

Penjelasan baris kode program pada Gambar 5.7 sebagai berikut:

1. Baris 1-2 nama fungsi untuk seleksi individu gabungan dengan parameter individu gabungan yang akan dimasukkan ke sistem, jumlah *popSize* dan nilai *fitness* pada masing-masing individu.
2. Baris 8 membuat *array* 2 dimensi bertipe *double* sebagai tempat hasil seleksi.
3. Baris 10 mengulang proses sebanyak *popSize*.
4. Baris 11 membuat *array* untuk menyimpan kromosom pada induk pertama.
5. Baris 12 membuat *array* untuk menyimpan kromosom pada induk kedua.
6. Baris 13-14 inisialisasi batas *range* angka yang akan di random.
7. Baris 17-29 merandom 2 angka dengan *range* ukuran populasi, jika 2 angka yang dipilih sama maka proses acak diulang lagu namun jika tidak maka proses keluar dari while.
8. Baris 30-32 menampilkan nilai indeks pada individu terpilih.
9. Baris 35-47 menampilkan kromosom pada setiap individu terpilih.
10. Baris 49-54 mengcopy kromosom pada induk ke dalam *array* yang tersedia untuk menyimpan 2 individu terpilih tersbut.
11. Baris 55-59 menampilkan nila *fitness* masing-masing individu terpilih.
12. Baris 60-70 apabila nilai *fitness* pada individu pertama lebih besar dari individu kedua maka individu pertama lilos sebagai populasi selanjutnya dengan meletakkan kromosom pada individu pertama pada *array* hasil, begitu juga seballiknya jika yang lebih besar pada individu kedua maka kromosom pada individu kedua diletakkan pada *array* hasil.
13. Baris 71 mengembalikan hasil seleksi.

- Metode *Rhoulette Wheel Selection*

Kode program untuk proses seleksi menggunakan metode *roulette wheel selection* dapat dilihat pada Gambar 5.8 berikut:

```
1   public double[][] SeleksiRouletteWheelPrecisionFx(double
2   [][] IndividuGabungan, int [] fitness, int PopSize) {
3           AlgenRealCode algen = new AlgenRealCode();
4   int panjangkromosom=IndividuGabungan[0].length;
5   int popsize = IndividuGabungan.length;
6   double[][] Hasil=new double[popsize][panjangkromosom];
7           Random random = new Random();
8   int[] IndeksAwalIndividuGabungan = new int[popsize];
9           for (int i = 0; i < popsize; i++) {
10              IndeksAwalIndividuGabungan[i] = i;
11          }
12  double TotalFitness = 0.0;
13          for (int i = 0; i < popsize; i++) {
14              TotalFitness = TotalFitness + fitness[i];
15          }
16          double[] Prob = new double[popsize];
17          double[] ProbCum = new double[popsize];
18          for (int i = 0; i < popsize; i++) {
19              Prob[i] = fitness[i] / TotalFitness;
20              if (i == 0) {
21                  ProbCum[i] = Prob[i];
22              } else {
23                  ProbCum[i] = ProbCum[i - 1] + Prob[i];
24              }
25          }
26          double[] r = new double[PopSize];
27          double nilaiTerendahD = 0, nilaiTertinggiD = 1;
28          for (int i = 0; i < PopSize; i++) {
29              r[i] = nilaiTerendahD + (random.nextDouble()
30  * (nilaiTertinggiD - nilaiTerendahD));
31          }
32          int[] IndeksK = new int[PopSize];
33          for (int i = 0; i < PopSize; i++) {
34              for (int j = 0; j < popsize; j++) {
35                  if (r[i] < ProbCum[j]) {
36                      IndeksK[i] = j;
37                for (int k = 0; k < panjangkromosom;k++) {
38                     Hasil[i][k] = IndividuGabungan[j][k];
39                      }
40                     break;
41                  }}}}
42          return Hasil; }
```

Gambar 5.8 Kode program metode *roulette wheel*

Penjelasan baris kode program pada Gambar 5.8 sebagai berikut:

1. Baris 1-2 judul fungsi *roulette wheel* dengan parameter individu gabungan yang akan di seleksi, jumlah *popSize* dan nilai *fitness* tiap individu.

2. Baris 6 menyediakan *array* 2 dimensi bertipe *double* untuk tempat menyimpan hasil seleksi.
3. Baris 8 menyediakan *array* untuk menyimpan indeks pada tiap individu.
4. Baris 9-11 inisialisasi no indeks pada tiap individu.
5. Baris 12 inisialilsasi total*Fitness*.
6. Baris 13-15 menghitung total*Fitness* pada semua individu yang akan diseleksi.
7. Baris 16-17 menyediakan *array* 1 dimensi untuk menyimpan hasil *prob* dan *probCum* pada tiap indiviu.
8. Baris 20-25 menghitung nilai *prob* dan *probCum* pada tiap individu.
9. Baris 26-31 mencari nilai r secara acak pada *range* [0,1] selama belum mencapai *popSize*.
10. Baris 32-43 mencari *proCum* pada individu gabungan yang nilainya lebih besar dan mendekati nilai r dan menjadikan individu tersebut sebagai individu yang lolos untuk populasi baru.
11. Baris 44 mengembalikan hasil seleksi.

5.2.2 Proses FIS Tsukamoto

Sub bab ini menjelaskan kode program untuk menjalankan proses dari metode *Fuzzy Inference System Tsukamoto.* Proses pada metode ini terdiri dari 4 macam yaitu proses Pembentukan fungsi keanggotaan, proses Fuzzifikasi, proses Mesin Inferensi dan proses Defuzzifikasi.

5.2.2.1 Basis Aturan

Baris aturan disini digunakan untuk menampung aturan-aturan yang digunakan dalam perhitungan FIS Tsukamoto . Karena terdapat 5 parameter dan 2 kemungkinan pada tiap parameter maka baris aturan yang ada berjumlah 32 baris dan kolomnya berjumlah 6 yaitu 5 parameter yang ada dan satu hasil diagnosa. Untuk detail *source code* yang digunakan dapat dilihat pada Gambar 5.

```
1     int banyakAturan =32;
2     int panjangAturan= 6;
3     public String[][] rule = new
4  String[banyakAturan][panjangAturan];
5     public BasisAturan() {
6     }
7     public String[][] setRule() {
8         try {
9             InputStream in =
10 this.getClass().getResourceAsStream("Rule.txt");
11            BufferedReader br = new BufferedReader(new
12 InputStreamReader(in));
13            String line;
14            int j = 0;
15            while ((line = br.readLine()) != null) {
16                int start = line.indexOf("");
17                String sub = line.substring(start);
18                String[] vertex = sub.split(" ");
```

```
19
20                  rule[j][0] = vertex[0];
21                  rule[j][1] = vertex[1];
22                  rule[j][2] = vertex[2];
23                  rule[j][3] = vertex[3];
24                  rule[j][4] = vertex[4];
25                  rule[j][5] = vertex[5];
26                  j++;
27              }
28              br.close();
29          } catch (Exception e) {
30              System.out.println("Error: " +
31 e.getMessage());
32          }
33          return rule;
34      }
35      public void cetakRule() {
36          this.rule=setRule();
37          for (int j = 0; j < 32; j++) {
38              System.out.println(rule[0][j] + " " +
39 rule[1][j] + " " + rule[2][j] + " " + rule[3][j] + " "
40                      + rule[4][j] + " " + rule[5][j]);
41          }
42      }
```

Gambar 5. 9 Kode program basis aturan

Penjelasan baris kode program pada Gambar 5.7 sebagai berikut:

1. Baris 1 dan 2 inisialisasi *integer* untuk jumlah baris dan kolom aturan
2. Baris 3 dan 4 inisialisasi *array* string 2 dimensi untuk menyimpan aturan
3. Baris 7-34 fungsi setRule untuk mendapatkan aturan yang ada dari data pada txt.
4. Baris 5-42 fungsi untuk mencetak aturan

5.2.2.2 Proses Fuzzifikasi

Proses selanjutnya dalam programan FIS Tsukamoto adalah fuzzifikasi. Proses ini bertanggung jawab untuk menjadikan bilangan *crisp* yang dimasukkan dalam sistem menjadi bilangan *fuzzy* yang akan digunakan dalam perhitunan selanjutnya. Fuzzifikasi dilakukan pada tiap baris aturan yang ada. Dalam sistem ini proses fuzzifikasi dilakukan oleh kelas pada tiap parameter dan satu kelas induk yaitu fuzzifikasi.java. Kode program pada kelas fuzzifikasi.java. dapat dilihat pada Gambar 5.10. Berikut sedangkan kode program pada salah satu dari 5 kriteria dapat dilihat pada Gambar 5.10.

```
1      int banyakAturan =32;
2      int panjangAturan= 6;
3      int banyakData= 215;
4      public double[][] dataSet = new
5  double[banyakData][panjangAturan];
6  public double[] miuT3Resin = new double[banyakAturan];
7  public double[] miuSthyroxin = new double[banyakAturan];
```

```
8       public double[] miuSerum = new double[banyakAturan];
9       public double[] miuTsh = new double[banyakAturan];
10      public double[] miuThin = new double[banyakAturan];
11      public double[] terkecil = new double[banyakAturan];
12      String [][] rule = new
13  String[banyakAturan][panjangAturan-1];
14      double [][] nilaiRule = new
15  double[banyakAturan][panjangAturan-1];
16      T3Resin t3resin = new T3Resin();
17      SThyroxin sthyroxin = new SThyroxin();
18      SerumT serum = new  SerumT();
19      TSH tsh = new  TSH();
20      THinjeksi thin = new THinjeksi() ;
21      BasisAturan r = new BasisAturan();
22       public double [][] getDataSet() {
23           try {
24               InputStream in =
25  this.getClass().getResourceAsStream("uci.txt");
26               BufferedReader br = new BufferedReader(new
27  InputStreamReader(in));
28               String line;
29               int j = 0;
30               while ((line = br.readLine()) != null) {
31                   int start = line.indexOf("");
32                   String sub = line.substring(start);
33                   String[] vertex = sub.split(" ");
34  dataSet[j][0]=0+Double.parseDouble(vertex[0]);
35  dataSet[j][1]=0+Double.parseDouble(vertex[1]);
36  dataSet[j][2]=0+Double.parseDouble(vertex[2]);
37  dataSet[j][3]=0+Double.parseDouble(vertex[3]);
38  dataSet[j][4]=0+Double.parseDouble(vertex[4]);
39  dataSet[j][5]=0+Double.parseDouble(vertex[5]);
40                   j++;
41               }
42               br.close();
43           } catch (Exception e) {
44               System.out.println("Error: " +
45  e.getMessage());
46           }
47           return dataSet;
48       }
49      public void cetakDataSet() {
50          this.dataSet=getDataSet();
51           for (int j = 0; j < banyakData; j++) {
52               System.out.println(dataSet[j][0] + " " +
53  dataSet[j][1] + " " + dataSet[j][2] + " " +
54  dataSet[j][3] + " "+ dataSet[j][4] + " " +
55  dataSet[j][5]);
56           }
57       }
58      public void setNilai(double nT3resin, double
59  nThyroxin,  double nSerum, double nTsh,  double nThin) {
60           t3resin.setT3Resin(nT3resin);
```

```
61          sthyroxin.setsthyroxin(nThyroxin);
62          serum.setSerumT(nSerum);
63          tsh.setTSH(nTsh);
64          thin.setTHinjeksi(nThin);
65      }
66      public void setNilaiBatasan( double a1, double a2,
67  double b1, double b2, double c1, double c2, double d1,
68  double d2, double e1, double e2) {
69          t3resin.setA(a1, a2);
70          sthyroxin.setB(b1, b2);
71          serum.setC(c1, c2);
72          tsh.setD(d1, d2);
73          thin.setE(e1, e2);
74      }
75    public double[][] getMiu()  {
76          this.rule=r.setRule();
77          for (int i = 0; i < banyakAturan; i++) {
78              miuT3Resin[i] =
79  t3resin.setRule(this.rule[i][0]);
80              miuSthyroxin[i] =
81  sthyroxin.setRule(this.rule[i][1]);
82              miuSerum[i] =
83  serum.setRule(this.rule[i][2]);
84              miuTsh[i] = tsh.setRule(this.rule[i][3]);
85              miuThin[i] = thin.setRule(this.rule[i][4]);
86              nilaiRule[i][0]= miuT3Resin[i];
87              nilaiRule[i][1]= miuSthyroxin[i];
88              nilaiRule[i][2]= miuSerum[i];
89              nilaiRule[i][3]= miuTsh[i];
90              nilaiRule[i][4]= miuThin[i];
91          }
92          return nilaiRule;
93      }
```

Gambar 5. 10 Kode program proses fuzzifikasi

Penjelasan baris kode program pada Gambar 5.10 sebagai berikut:

1. Baris 1 dan 2 inisialisasi variabel *integer* untuk jumlah baris dan kolom aturan.
2. Baris 3 inisialisasi variabel *integer* untuk jumlah dataset yang ada.
3. Baris 4 dan 5 inisialisasi *array double* 2 dimensi untuk menyimpan data set yang ada.
4. Baris 6-10 inisialisasi *array double* 1 dimensi untuk menyimpan nilai fuzzy tiap parameter.
5. Baris 11 inisialisasi *array double* 1 dimensi untuk menyimpan nilai min alpa tiap baris aturan.
6. Baris 12-13 inisialisasi *array* string 2 dimensi untuk menyimpan aturan.
7. Baris 14-15 inisialisasi *array* string 2 dimensi untuk menyimpan nilai bilangan fuzzy pada tiap baris aturan.
8. Baris 22-48 fungsi untuk mendapatkan nilai dataset yang ada.
9. Baris 49-57 fungsi untuk mencetak dataset.
10. Baris 58-65 fungsi untuk memasukkan nilai hasil tes pada seorang pasien tiap parameter.

11. Baris 66-74 fungsi untuk memasukkan nilai batasan pada fungsi keanggotaan tiap parameter.
12. Baris 75-93 fungsi untuk mendapatkan nilai fuzzy pada tiap baris aturan.

Contoh pembentukan fungsi keanggotaan pada suatu kelas untuk kriteria test *T3*-Resin sebagaimana Gambar 5.11. untuk kriteria lainnya memilki kode program yang sama.

```
1   public class T3Resin {
2
3       private double t3,a1,a2;
4       public T3Resin() {
5       }
6       public void setT3Resin(double t3) {
7           this.t3 = t3;
8       }
9       public double getT3Resin() {
10          return t3;
11      }
12      public void setA(double a1, double a2) {
13          this.a1 = a1;
14          this.a2 = a2;
15      }
16      public double A1() {
17          return a1;
18      }
19      public double A2() {
20          return a2;
21      }
22      public double getMiuRendah() {
23         if(t3 >= 65 && t3 <= a1){
24              return 1;
25          }else if(t3 > a1 && t3 < a2){
26              return ((a2-t3)/(a2-a1));
27          }else {
28              return 0;
29          }
30      }
31     public double getMiuTinggi() {
32          if(t3 >= a2 && t3 <= 144){
33              return 1;
34          }else if(t3 > a1 && t3<a2){
35              return ((t3-a1)/(a2-a1));
36          }else {
37              return 0;
38          }
39      }
40          public double setRule(String rule) {
41          if (rule.equalsIgnoreCase("rendah")) {
42              return getMiuRendah();
43          }
44         else {
45              return getMiuTinggi();
46          }
```

```
47          }
48      }
```

Gambar 5. 11 Kode program proses fuzzifikasi pada parameter

Penjelasan baris kode program pada Gambar 5.11 sebagai berikut:

1. Baris 3 membuat variabel bertipe *double* untuk menyimpan nilai batasan dan masukkan hasil test pasien.
2. Baris 4-5 merupakan konstraktor.
3. Baris 6-8 mengeset nilai hasil test pasien yang didaptkan dari masukan pengguna.
4. Baris 9-11 mengambil nilai test pasien.
5. Baris 12-15 mengeset nilai batasan untuk fungsi keanggotaan.
6. Baris 12-17 mengambil nilai batasan pada fungsi keanggotaan.
7. Baris 22-30 mereturn fungsi keanggotaan rendah sesuai nilai hasil test dan batasan pada fungsi keanggotaan.
8. Baris 31-39 mereturn fungsi keanggotaan tinggi sesuai nilai hasil test batasan pada fungsi keanggotaan.
9. Baris 40-49 mengembalikan nilai hasil fungsi keanggotaan rendah atau tingginya sesuai kolom pada tabel aturan.

5.2.2.3 Proses Mesin Inferensi

Pada proses mesin inferensi langkah-langkah yang dilakukan adalah mencari nilai *α_predikat* kemudian mencari nilai z yang didapatkan dari perhitungan fungsi keanggotaan hasil diagnosa dengan *α_predikat.* Hasil inferensi diapatkan dengan mengkalikan *α_predikat* tiap aturan dengan nilai z nya. Kode program untuk proses mesin inferensi dapat dilihat pada Gambar 5.12 berikut:

```
1       int banyakAturan =32;
2       int panjangAturan= 6;
3       public double[] zi = new double[banyakAturan];
4       public double[] prezi = new double[banyakAturan];
5       public double[] terkecil = new double[banyakAturan];
6       String [][]rule = new
7   String[banyakAturan][panjangAturan];
8       double [][]nilaiRule = new
9   double[banyakAturan][panjangAturan-1];
10
11      Fuzzifikasi fuz = new Fuzzifikasi();
12      Diagnosis diagnosis = new Diagnosis();
13      BasisAturan r = new BasisAturan();
14      Defuzzifikasi defuz = new Defuzzifikasi();
15      public double []  NilaiTerkecil() {
16          this.nilaiRule=fuz.getMiu();
17           for (int i = 0; i < banyakAturan; i++) {
18               terkecil[i] =
19  Math.min(Math.min(nilaiRule[i][0], nilaiRule[i][1]),
20  Math.min(Math.min(nilaiRule[i][2],nilaiRule[i][3] ),
21  Math.min(nilaiRule[i][4], nilaiRule[i][4])));
```

```
22          }
23          return terkecil;
24      }
25      public void setNilaiZi(double f1, double f2, double
26  f3, double f4) {
27          this.terkecil=NilaiTerkecil();
28          this.rule=r.setRule();
29      diagnosis.setNilai(f1, f2,   f3,   f4);
30          for (int i = 0; i < banyakAturan; i++) {
31              zi[i] = diagnosis.setRule(this.rule[i][5],
32  this.terkecil[i]);
33          }
34      }
35      public  double [] NilaiPreZi() {
36          this.terkecil=NilaiTerkecil();
37          for (int i = 0; i < banyakAturan; i++) {
38              prezi[i] = this.terkecil[i] * zi[i];
39          }
40          return prezi;
41      }
42          public void setNilaiTerkecil() {
43          defuz.setTerkecil(terkecil);
44      }
45          public void setNilaiPrezi() {
46          defuz.setPrezi(prezi);
47      }
48        public double getNilaiZi(int index) {
49          return zi[index];
50      }
51        public double[] getMatrixZi() {
52          return zi;
53      }
54     public int diagnosa (double t3Resin,double thyroxin,
55  double triiodothyronine, double TSH, double
56  perbedaanTSH, double a1, double a2, double b1, double
57  b2, double c1, double c2, double d1, double d2,
58  double e1, double e2, double f1, double f2, double f3,
59  double f4){
60           int hasil;
61  fuz.setNilai( t3Resin,thyroxin, triiodothyronine, TSH,
62  perbedaanTSH);
63  fuz.setNilaiBatasan(a1, a2, b1, b2, c1, c2, d1, d2, e1,
64  e2);
65      fuz.getMiu();
66      setNilaiZi(f1, f2, f3, f4);
67      setNilaiTerkecil();
68      setNilaiPrezi();
69      double [] printTerkecil = new double [banyakAturan];
70      double [] printPrezi = new double [banyakAturan];
71      double [] printZi = new double [banyakAturan];
72      printTerkecil=NilaiTerkecil();
73      printPrezi=NilaiPreZi();
74      printZi=getMatrixZi();
```

```
75      defuz.setJmlTerkecil();
76      defuz.setJmlPrezi();
77      defuz.setNilaiZ();
78
79  /**mengambil hasil keputusan sistem sesuai nilai batasan
80  *yang diambil dari kromosom dan disesuaikan dengan
81  *penyimpanan hasil pada dataset yang berupa angka.
82  */
83      if (defuz.getKeputusan()=="Normal"){
84          hasil =1;
85      }
86          else if (defuz.getKeputusan()=="Hyperthyroid"){
87           hasil= 2;
88                  }
89                  else{
90                  hasil=3;
91                  }
92      return hasil;
93      }
```

Gambar 5. 12 Kode program proses inferensi

Penjelasan baris kode program pada Gambar 5.12 sebagai berikut:

1. Baris 1 dan 2 inisialisasi variabel *integer* untuk jumlah baris dan kolom aturan
2. Baris 3 inisialisasi *array double* 1 dimensi untuk menyimpan nilai z
3. Baris 4 inisialisasi *array double* 1 dimensi untuk menyimpan nilai z inferensi
4. Baris 5 inisialisasi *array double* 1 dimensi untuk menyimpan nilai alpa tiap baris
5. Baris 6-7 inisialisasi *array* string 2 dimensi untuk menyimpan baris aturan
6. Baris 8-9 inisialisasi *array* string 2 dimensi untuk menyimpan nilai funzzy pada baris aturan
7. Baris 15-24 fungsi untuk mencari nilai minimal alpa pada tiap baris aturan
8. Baris 25-34 fungsi untuk mendapatkan nilai z pada baris diagnosa
9. Baris 35-41 fungsi untuk mendapatkan nilai z inferensi yaitu dengan mengkalikan nilai minimal alfa dengan z
10. Baris 42-44 fungsi untuk mengirim nilai minimal alpa pada kelas defuzzifikasi
11. Baris 45-47 fungsi untuk mengirim nilai z inferensi pada kelas defuzzfikasi
12. Baris 48-50 fungsi untuk mendapatkan nilai z pada indeks tertentu
13. Baris 51-53 fungsi untuk mendapatkan matriks z
14. Baris 54-89 fungsi untuk mendapatkan nilai diagnosa sistem dengan masukan nilai tes tiap parameter dan batasan pada fungsi keanggotaan tiap parameter dan kolom diagnosa

5.2.2.4 Proses Defuzzifikasi

Proses defuzzifikasi merupakan pengembalian bilangan fuzzy menjadi bilangan crisp. Langkah pertama yang dilakukan adalah menjumlahkan semua nilai *α_predikat* pada tiap baris pada tabel aturan dan juga menjumlahkan semua nilai z* α, yang kemudian membagi total nilai z* α dengan jumlah total nilai *α_predikat.* Hasilnya dimasukkan pada aturan keputusan.

```
1       int banyakAturan =32; int panjangAturan= 6;
2       String keputusan;
3       public double[] terkecil = new double[banyakAturan];
4       public double[] prezi = new double[banyakAturan];
5       public double Z, jmlPrezi = 0, jmlTerkecil = 0;
6       public void setTerkecil(double [] nilaiTerkecil) {
7           this.terkecil = nilaiTerkecil;
8       }
9           public void setPrezi(double [] nilaiPrezi) {
10          this.prezi = nilaiPrezi;
11      }
12         public void setJmlPrezi() {
13          for (int i = 0; i < banyakAturan; i++) {
14              jmlPrezi += prezi[i];
15          }
16      }
17      public double getJmlPrezi() {
18          return jmlPrezi;
19      }
20      public void setJmlTerkecil() {
21          for (int i = 0; i < banyakAturan; i++) {
22              jmlTerkecil +=terkecil[i];
23          }
24      }
25      public double getJmlTerkecil() {
26          return jmlTerkecil;
27      }
28      public void setNilaiZ() {
29          Z = getJmlPrezi() / getJmlTerkecil();
30      }
31      public double getNilaiZ() {
32          return Z;
33      }
34      public String getKeputusan(){
36          if(getNilaiZ() <= 33){
37              return keputusan = "Hypothyroid";
38          }else if(getNilaiZ() >33 && getNilaiZ() <=67){
39              return keputusan = "Normal";
40          }
41          else{
42              return keputusan ="Hyperthyroid";
43          }}
```

Gambar 5.13 Kode program proses defuzzifikasi

Penjelasan baris kode program pada Gambar 5.13 sebagai berikut:

1. Baris 1 inisialisasi variabel *integer* untuk jumlah baris dan kolom aturan
2. Baris 2 inisialisasi variabel string untuk menyimpan keputusan
3. Baris 3 inisialisasi *array double* 1 dimensi untuk menyimpan minimal alpa
4. baris 4 inisialisasi *array double* 1 dimensi untuk menyimpan nilai z inferensi
5. Baris 5 inisialisasi varibel yang dibutuhkan dalam perhitungan mencari nilai Z
6. Baris 6-8 fungsi untuk set nilai terkecil yang diambil dari kelas MesinInferensi.java
7. Baris 9-11 fungsi untuk set nilai z inferensi yang dari kelas MesinInferensi.java
8. Baris 12-16 fungsi untuk menjumlahkan nilai z inferensi pada tiap baris aturan
9. Baris 17-19 fungsi untuk mendapatkan jumlah nilai z inferensi
10. Baris 20-24 fungsi untuk menjumlahkan nilai minimal alpa pada tiap baris aturan
11. Baris 25-27 fungsi untuk mendapatkan jumlah nilai minimal alpa
12. Baris 28-30 fungsi untuk menghitung nilai Z
13. Baris 31-33 fungsi untuk mendapatkan nilai Z
14. Baris 34-43 fungsi untuk mendapatkan hasil diagnosa

BAB 6 PENGUJIAN

Pada bab ini membahas tentang pengujian algoritma seperti yang telah di bahas bab sebelumnya sesuai dengan perancangan pengujian. Scenario pengujian pada penelitian ini adalah pengujian 3 metode seleksi, pengujian mencari nilai *crossover rate* dan *mutation rate,* pengujian banyaknya populasi dan yangterlahir adalah pengujian banyaknya generasi untuk bisa mendapatkan rata-rata *fitness* terbaik.

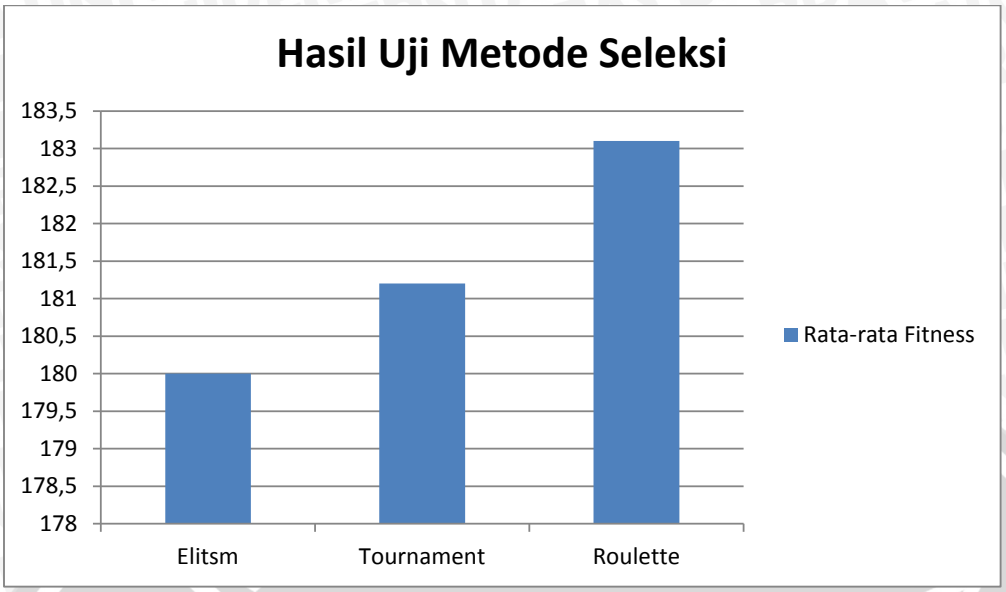
6.1 Pengujian Metode Seleksi

Pegujian ini dilakukan untuk mengetahui metode seleksi terbaik dari ketiga metode yang di uji yaitu metode *Elitism Selection, Tournament Selection* dan *Roulette Wheel* dalam menentukan batasan fungsi keanggotaan untuk sistem Diagnosa penyakit menggunakan metode *Fuzzy Inference System Tsukamoto*.

Uji coba ini dilakukan dengan nilai cr = 0.5 dan mr = 0.5, ukuran populasi yang di pakai adalah 80 dan generasinya sebanyak 100 sesuai dengan parameter terbaik dari penelitian menggunakan FIS Tsukamoto yang fungsi keanggotaanya dioptimasi dengan Algoritma Genetika sebelumnya (Restuputri dkk, 2015). Hasil pengujian ketiga metode seleksi menggunakan parameter yangtelah disebutkan dalam dilihat pada Tabel 6.1.

Tabel 6. 1 Hasil pengujian metode seleksi

No.	Elitism		Tornament		Roulette Wheel	
	fitness	waktu	*fitness*	waktu	*fitness*	waktu
1	180	0:17:41	180	0:17:57	180	0:18:17
2	180	0:17:38	180	0:18:08	180	0:18:08
3	180	0:18:04	184	0:17:48	180	0:18:10
4	180	0:17:29	184	0:17:53	185	0:18:14
5	180	0:17:54	180	0:18:05	185	0:18:04
6	180	0:17:37	184	0:18:10	184	0:17:58
7	180	0:17:49	180	0:17:47	184	0:17:54
8	180	0:18:03	180	0:18:01	184	0:18:11
9	180	0:17:45	180	0:17:58	185	0:18:04
10	180	0:17:40	180	0:17:42	184	0:18:03
Rata-rata *Fitness*	180		181,2		183,1	
Rata-rata Waktu		0:17:46		0:17:57		0:18:06
Standart Deviasi	0		1,932 18357		2,183 26972	0



Gambar 6. 1 Grafik hasil uji coba metode seleksi sesuai nilai *fitness*

Dari hasil percobaan dapat dilihat bahwa rata-rata hasil *fitness* dari 10 percobaan, metode *roulette wheel selection* memiliki nilai lebih tinggi dibandingkan 2 metode lainnya. Metode *roulette wheel selection* menjadi metode seleksi dengan rata-rata nilai *fitness* rata-rata terbaik dalam penelitian ini dibandingkan metode *elitism* dan *tournament* dengan nilai yang didapatkan sebesar 183,1 nilai ini disusul oleh metode *tournament* dengan nilai rata-rata sebesar 181,1 dan hasil rata-rata terendah ketika menggunakan metode *elitism* yaitu sebesar 180.

Dapat dilihat grafik pada Gambar 6.1 nilai *fitness* dengan menggunakan metode *elitism* selalu stabil pada nilai *fitness* 180 dalam 10 kali percobaan seingga nilai standart deviasinya bernilai 0. Sedangkan untuk metode *tournament* dan *roulette wheel* memiliki nilai setandart deviasi sebesar 1,932 18357dan 2,18326972, hal in menunjukkan nilai *fitness* yang muncul pada tiap percobaan bervariasi tidak selalu sama. Namun, hasil *fitness* terendah dari percobaan dengan menggunakan kedua metode tersebut yaitu *tournament* dan *roulette wheel* adalah 180 sama dengan nilai yang muncul pada setiap percbaan dengan menggunakan metode *elitsm*. Ketika dilihat rata-rata waktu yang dibutuhkan untuk menjalankan sistem dengan parameter yang sama metode *elitism* membutuhkan waktu paling rendah yaitu 17 menit 46 detik. Untuk rata-rata waktu pada metode lainnya yaitu tournament dan roulette wheel adalah 17 menit lebih 57 detik dan 18 menit lebih 6 detik. Walaupun metode *roulette wheel* relatif lebih lama dibandingkan kedua metode lainnya namun hasil rata-rata *fitness*nya lebih baik dari kedua metode tersebut.

Dengan menggunakan metode *roulette wheel* memungkinkan individu dengan nilai *fitness* rendah terpilih menjadi individu pada generasi selanjutnya, terbukti pada beberapa penelitian bahwa individu yang memiliki nilai *fitness* yang kecil memungkinkan menempati area yang dekat dengan titik optimum. Hal

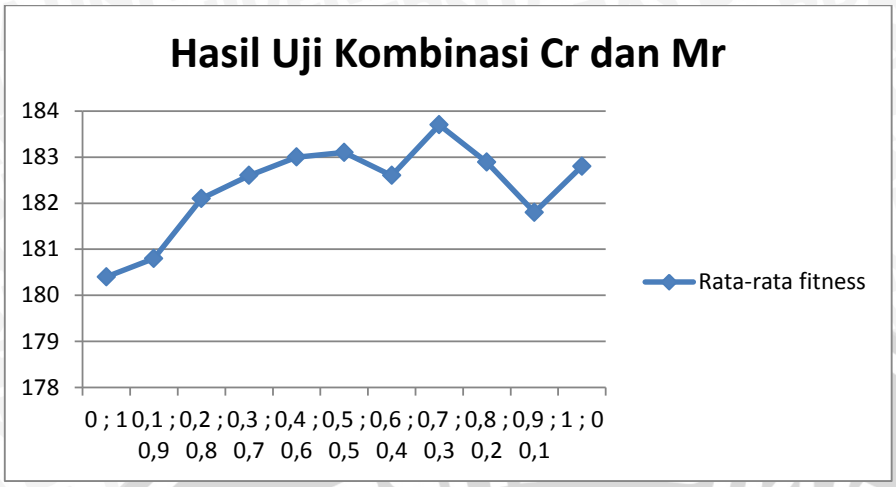
ini dapat terjadi ketika optimasi fungsi memiliki banyak titik optimum lokal (Gen & Cheng, 1997).

6.2 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Dari hasil percobaan metode seleksi sebelumnya di dapat bahwa seleksi menggunakan metode *roulette wheel* sebagai metode terbaik untuk penelitian ini. Pengujian ini dilakukan untuk mengetahui kombinasi antar *crossover rate* dan *mutation rate* paling optimal yang dapat diimplemetasikan pada program untuk menemukan solusi terbaik. Cara pengujiannya yaitu dikombinasikan nilai *crossover rate* dan *mutation rate* sehingga nilai totalnya sebesar 1. Setiap kombinasi yang ada diujikan sebanyak 10 kali. Setiap kombinasi yang ada di cari nilai *fitness* rata-rata dari 10 percobaan yang telah dilakukan. Kombinasi yang memiliki nilai rata-rata terbesar terpilih menjadi kombinasi *crossover rate* dan *mutation rate* terbaik untuk mendapatkan solusi optimum. Nilai parameter lainnya yang digunakan adalah populasi sebanyak 80 individu dan generasinya sebanyak 100 sesuai hasil parameter terbaik dari penelitian sebelumnya yang menggunakan metode yang sama (Restuputri dkk, 2015). Hasil percobaan dapat dilihat pada Tabel 6.2 berikut:

Tabel 6.2 Pengujian kombinasi cr dan mr

Nilai Kombinasi		Percobaan ke-										Rata-rata
Cr	*Mr*	1	2	3	4	5	6	7	8	9	10	
0	1	180	180	184	180	180	180	180	180	180	180	180,4
0,1	0,9	185	180	180	182	181	180	180	180	180	180	180,8
0,2	0,8	180	181	180	184	180	185	185	184	182	180	182,1
0,3	0,7	180	184	184	183	185	185	180	182	180	183	182,6
0,4	0,6	184	185	185	180	180	184	184	185	180	183	183
0,5	0,5	180	180	180	185	185	184	184	184	185	184	183,1
0,6	0,4	180	183	183	184	180	185	181	180	185	185	182,6
0,7	0,3	184	184	183	184	180	185	185	184	183	185	183,7
0,8	0,2	183	180	182	184	185	180	184	183	185	185	182,8889
0,9	0,1	180	180	180	183	184	185	180	181	180	185	181,8
1	0	184	180	180	184	185	182	183	183	183	184	182,8
Nilai rata-rata terbesar :												183,7



Gambar 6. 2 Hasil pengujian kombinasi cr dan mr

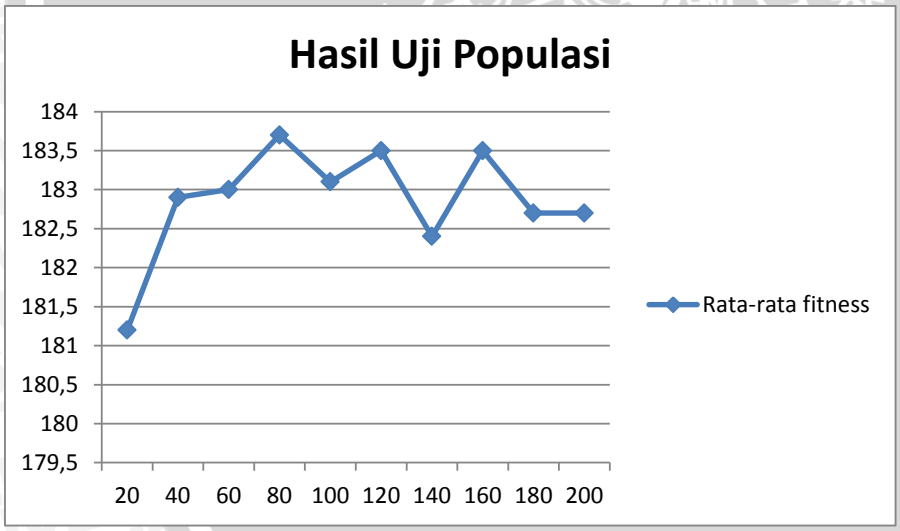
Pada Gambar 6.2 dapat dilihat nilai rata-rata *fitness* terbaik ada pada kombinasi nilai *crossover rate* sebesar 0,3 dan nilai *mutation rate* sebesar 0,7 yaitu sebesar 183,7. Nilai rata-rata terendah yaitu 180,4 diperoleh dari hasil uji kombinasi *crossover rate* sebesar 0 dan *mutation rate* sebesar 1. Ketika menggunakan *mutation rate* yang terlalu tinggi sedangkan *crossover rate* yang rendah hasilnya kurang optimum, walaupun dengan *mutation rate* yang tinggi Algoritma Genetika dapat mempunyai tingkat eksplorasi dan diversitas populasi yang tinggi namun *crossover rate* yang rendah akan menyebabkan algoritma tidak bisa belajar dengan efektif dari generasi sebelumnya, sehingga mengakibatkan pencariannya tidak dapat secara efektif tereksploitasi ke daerah lain untuk mendapatkan kemungkinan solusi yang lebih baik. Begitu pula jika nilai *crossover rate* yang digunakan tinggi sedangkan nilai *mutation rate* nya rendah maka keturunannya akan memliki kemiripan yang tinggi dengan induknya erjadi konvergensi secara dini hanya dalam nilai generasi yang rendah dan tidak dapat mengeksplorasi area lain (Mahmudy, 2013).

6.3 Pengujian Ukuran Populasi

Pengujian populasi ini dilakukan untuk mengetahui berapa ukuran populasi yang dapat menghasilkan kromosom terbaik. Parameter yang digunakan pada uji coba yaitu meggunakan nilai kombinasi cr dan mr sebesar 0.7 dan 0.3, dan metode seleksi yang digunakan adalah metode *roulette wheel*. Cara melakukan percobaan ini yaitu dengan menggunakan variasi angka kelipatan 20 dari angka 20 hingga 200 sehingga ada 10 angka percobaan, setiap angka dilakukan percobaan sebanyak 10 kali kemudian di cari rata-rata tiap angka percobaan. Angka percobaan dengan nilai rata-rata terbesar terpilih sebagai jumlah populasi optimum untuk sistem ini. Detail nilai tiap percobaan dapat dilihat pada Tabel 6.3 berikut:

Tabel 6. 3 Pengujian ukuran populasi

Ukuran populasi	Percobaan ke-										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
20	180	180	185	180	180	180	180	181	183	183	181,2
40	180	183	184	180	180	185	185	183	184	185	182,9
60	185	185	182	184	180	180	184	185	181	184	183
80	184	184	183	184	180	185	185	184	183	185	183,7
100	184	185	180	180	185	182	185	184	183	183	183,1
120	185	184	183	185	183	182	185	184	180	184	183,5
140	180	184	180	185	180	184	185	184	180	182	182,4
160	185	180	180	185	185	185	184	185	185	181	183,5
180	181	183	185	185	185	180	180	184	180	184	182,7
200	182	185	180	183	180	180	185	184	183	185	182,7
Nilai rata-rata tertinggi :											183,7



Gambar 6. 3 Hasil uji ukuran populasi

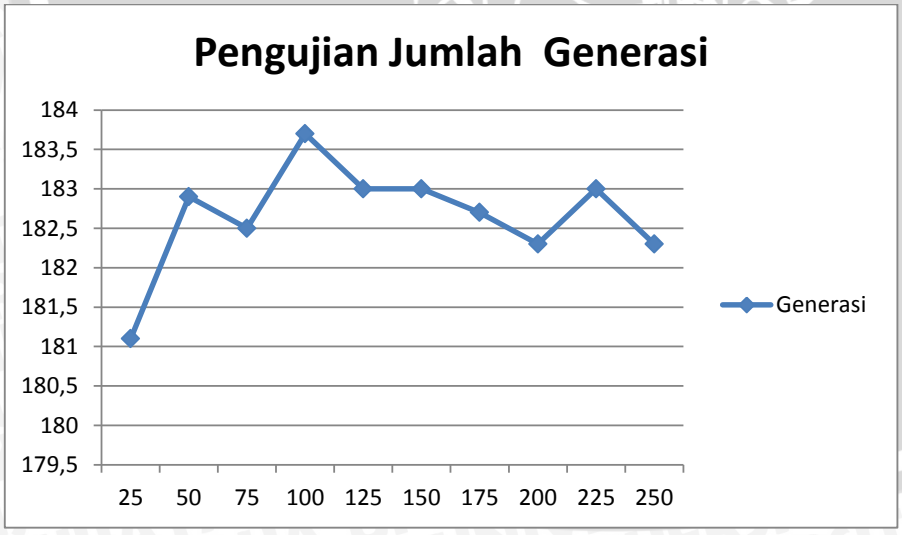
Dari Grafik pada Gambar 6.3 dapat dilihat bahwa nilai rata-rata pengujian terbesar didapatkan pada angka 80 yaitu 183,7. Sedangkan rata-rata nilai *fitness* terkecil adalah pada angka populasi sebesar 20 yaitu 181,2. Dari hasil pengujian populasi tersebut ketika ukuran populasi yang digunakan besar maka semakin besar pula peluang mendapatkan kromosom optimal. Namun, dengan menggunakan ukuran populasi yang terlalu besar dapat mengakibatkan waktu eksekusi Algoritma Genetika lebih lama dan belum pasti mendapatkan kromosom yang lebih baik (Pratiwi, 2014).

6.4 Pengujian Jumlah Generasi

Pengujian ini dilakukan untuk mendapatkan jumlah generasi yang tepat agar bisa menghasilkan solusi optimal. Pengujian ini dilakukan dengan menggunakan 10 nilai ukuran yang berbeda antara 25 hingga 250 yang, setiap nilai pengujian didapatkan dengan menambahkan angka 25 ke pengujian sebelumnya. Untuk parameter lainnya seperti kombinasi nilai cr, mr dan ukuran populasi didapatkan dari hasil pengujian sebelumnya yaitu cr = 0.7, mr = 0.2 dan ukuran populasi = 80. Hasil pegujian ukuran generasi dapat dilihat pada Tabel 6.3.

Tabel 6.4 Pengujian jumlah generasi

Jumlah Generasi	1	2	3	4	5	6	7	8	9	10	Rata-rata
25	182	180	184	180	180	183	180	182	180	180	181,1
50	183	180	185	184	180	185	184	184	180	184	182,9
75	183	184	183	183	180	184	183	182	180	183	182,5
100	184	184	183	184	180	185	185	184	183	185	183,7
125	180	185	184	180	184	180	185	183	185	184	183
150	185	180	183	180	185	184	184	181	184	184	183
175	182	180	185	185	180	181	184	184	182	184	182,7
200	184	183	181	180	182	185	185	182	181	180	182,3
225	183	185	183	185	185	184	183	180	182	180	183
250	184	182	184	180	183	180	180	185	180	185	182,3



Gambar 6.4 Pengujian jumlah generasi

Pada Gambar 6.4 terlihat bahwa ukuran generasi paling baik agar dapat menghasilkan solusi optimum yaitu pada ukuran 100 generasi dengan nilai rata-rata sebesar 183,7. Dari gambar tersebut juga terlihat kaetika ukuran generasi

tinggi dan dapat menghasilkan solusi optimum, maka dengan penambahan ukuran generasi rata-rata *fitness* cenderung stabil.

6.5 Hasil dan Analisis Akurasi Sistem

Dari hasil pengujian masing-masing parameter pada subbab-subbab sebelumnya kemudian dilakukan pengujian akurasi sistem. Akurasi sistem didapatkan dengan membandingkan hasil diagnosa pakar dengan sistem. Untuk nilai parameter yang digunakan didapatkan dari hasil pengujian scenario kombinasi cr dan mr, ukuran populasi dan jumlah generasi. Dari hail pengujian sebelumnya didapatkan kombinasi terbaik yaitu dengan nilai cr=0.7 dan mr=0.3, ukuran populasi sebesar 80 dan jumlah generasi sebanyak 100 kali. Data hasil diagnosa pakar yang ada adalah sebanyak 215, setiap data yang ada dibandingkan dengan hasil diagnosa sistem. Ketika hasil diagnosa pakar dan sistem sama sistem akan menambah sebuah variabel nilai *integer* dengan angka 1 kemudian jika semua data telah di bandingkan maka didapatkan hasil akurasi dengan mengkalikan jumlah hasil yang benar dengan 100 dan membaginya dengan angka 215. Hasil akurasi sistem ini adalah sebesar 86%. Untuk kromosom yang menghasilkan akurasi tertinggi dapat dilihat pada Tabel 6.5.

Tabel 6. 5 Kromosom dengan *fitness* terbaik

t3-resin		*thyroxin serum*		*Triiodothyronine Serum*		*thyroid-stimulatin hormone*		maksimal perbedaan TSH		Hasil Diagnosa				*Fitness*
81.432	106.791	14.7104	19.695	2.835	15.5	185	15.5	15.5	40.46	15.5	27.8	65.5	84.9	185

Tabel 6. 6 Perbandingan hasil diagnosa pakar dengan sistem

ke-	pkr	stm
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	1	1
9	1	1
10	1	1
11	1	1
12	1	1
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	1

ke-	pkr	stm
46	1	1
47	1	1
48	1	1
49	1	1
50	1	1
51	1	1
52	1	1
53	1	1
54	1	1
55	1	1
56	1	1
57	1	1
58	1	1
59	1	1
60	1	1
61	1	1
62	1	1
63	1	1
64	1	1

ke-	pkr	stm
91	1	1
92	1	1
93	1	1
94	1	1
95	1	1
96	1	1
97	1	1
98	1	1
99	1	1
100	1	1
101	1	1
102	1	1
103	1	1
104	1	1
105	1	1
106	1	1
107	1	1
108	1	1
109	1	1

ke-	pkr	stm
136	1	1
137	1	1
138	1	1
139	1	1
140	1	1
141	1	1
142	1	1
143	1	1
144	1	1
145	1	1
146	1	1
147	1	1
148	1	1
149	1	1
150	1	1
151	2	2
152	2	2
153	2	2
154	2	2

ke-	pkr	stm
181	2	2
182	2	2
183	2	2
184	2	2
185	2	2
186	3	2
187	3	2
188	3	2
189	3	2
190	3	2
191	3	2
192	3	2
193	3	2
194	3	2
195	3	1
196	3	1
197	3	1
198	3	1
199	3	1

Tabel 6. 7 Perbandingan hasil diagnosa pakar dengan sistem

ke-	pkr	stm
20	1	1
21	1	1
22	1	1
23	1	1
24	1	1
25	1	1
26	1	1
27	1	1
28	1	1
29	1	1
30	1	1
31	1	1
32	1	1
33	1	1
34	1	1
35	1	1
36	1	1
37	1	1
38	1	1
39	1	1
40	1	1
41	1	1
42	1	1
43	1	1
44	1	1
45	1	1

ke-	pkr	stm
65	1	1
66	1	1
67	1	1
68	1	1
69	1	1
70	1	1
71	1	1
72	1	1
73	1	1
74	1	1
75	1	1
76	1	1
77	1	1
78	1	1
79	1	1
80	1	1
81	1	1
82	1	1
83	1	1
84	1	1
85	1	1
86	1	1
87	1	1
88	1	1
89	1	1
90	1	1

ke-	pkr	stm
110	1	1
111	1	1
112	1	1
113	1	1
114	1	1
115	1	1
116	1	1
117	1	1
118	1	1
119	1	1
120	1	1
121	1	1
122	1	1
123	1	1
124	1	1
125	1	1
126	1	1
127	1	1
128	1	1
129	1	1
130	1	1
131	1	1
132	1	1
133	1	1
134	1	1
135	1	1

ke-	pkr	stm
155	2	2
156	2	2
157	2	2
158	2	2
159	2	2
160	2	2
161	2	2
162	2	2
163	2	2
164	2	2
165	2	2
166	2	2
167	2	2
168	2	2
169	2	2
170	2	2
171	2	2
172	2	2
173	2	2
174	2	2
175	2	2
176	2	2
177	2	2
178	2	2
179	2	2
180	2	2

ke-	pkr	stm
200	3	1
201	3	1
202	3	1
203	3	1
204	3	1
205	3	1
206	3	1
207	3	1
208	3	1
209	3	1
210	3	1
211	3	1
212	3	1
213	3	1
214	3	1
215	3	1

Dari Tabel 6.6 dapat dilihat bahwa hasil diagnosa sistem (**stm**) yang sama dengan hasil diagnosa dari pakar (**pkr**) adalah sebanyak 185 data diagnosa pakar dari 215 data yang telah didiagnosa oleh pakar sebelumnya. Angka 1 pada tabel dia atas menunjukkan hasil diagnosa normal, angka 2 menunjukkan hasil diagnosa hyperthyroid dan angka 3 menunjukkan hasil diagnosa hypothyroid. Jika dijadikan persentase maka akurasi sistem adalah sebesar 86%.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis sebelumnya maka dapat disimpulkan bahwa:

1. Penelitian ini menggunakan representasi kromosom yang berbasis real-coded, digunakan sebagai nilai batasan pada fungsi keanggotaan FIS Tsukamoto . Proses reproduksi yang digunakan dalam Algoritma Genetika menggunakan proses reproduksi yang digunakan untuk real-coded yaitu *extended intermediate crossover* untuk proses *crossover*nya dan *random mutation* untuk proses mutasi. Dengan demikian permasalahan dalam mengoptimasi fungsi keanggotaan dapat diselesaikan.
2. Berdasarkan hasil pengujian yang telah dilakukan dapat disimpulkan bahwa nilai tiap parameter yang dibutuhkan dalam proses Algoritma Genetika mempengaruhi hasil kromosom yang optimal sebagai pengisi batasan pada fungsi keanggotaan. Dengan menggunakan ukuran populasi dan jumlah generasi yang besar maka kemungkinan mendapatkan hasil yang optimal juga semakin besar pula. Namun ketika kedua jumlah parameter tersebut terlalu besar maka juga membutuhkan waktu yang lama dan belum tentu akan didapatkan hasil yang lebih optimal.
3. Parameter pada algortima genetika yang dapat menghasilkan rata-rata *fitness* paling tinggi dalam pengujiannya yaitu dengan kombinasi cr sebesar 0.7 dan mr sebesar 0.3, ukuran populasi sebesar 80 dan jumlah generasi sebesar 100.
4. Hasil akurasi sistem diagnosa penyakit thyroid ini yaitu sebesar 86% dengan menggunakan FIS Tsukamoto yang fungsi keanggotaannya dioptimasi dengan Algoritma Genetika. Hasil ini terbukti lebih besar daripada hanya menggunakan FIS Tsukamoto saja pada bab perancangan sebelumnya yaitu sebesar 69,7%.

7.2 Saran

Penelitian ini dapat dikembangkan lagi sehingga diperolah hasil yang lebih baik, saran untuk penelitian selanjutnya adalah:

1. setiap parameter masukan pada FIS Tsukamoto dalam penelitian ini hanya terdiri dari dua keanggotaan yaitu tinggi dan rendah, dengan menambahkan keanggotaan lagi diharapkan akan menghasilkan sistem yang lebih akurat lagi.
2. Optimasi aturan dan batas nilai Z sebagai diagnosa sistem juga dapat dioptimasi agar mendapatkan hasil yang lebih baik.
3. Metode pada proses reproduksi maupun seleksi dapat dilakukan dengan metode lainnya untuk membandingkan metode terbaik dalam kasus optimasi sistem diagnosa penyakit thyroid menggunakan metode FIS Tsukamoto menggunakan Algoritma Genetika.

DAFTAR PUSTAKA

Acharya, U.R., dkk, 2012. ThyroScreen system: High resolution ultrasound thyroid image characterization into benign and malignant classes using novel combination of texture and discrete wavelet transform. Computer Methods And Program In Bio Medicine, pp. 233-241.

Berlianty,Intan dan Miftahol, 2010. Teknik-Teknik Optimasi Heuristik. Yogyakarta: Graha Ilmu.

Bon , A.T. dan Utami, S.V., 2014. An analytical hierarchy process and fuzzy inference system tsukamoto for production planning: a review and conceptual research. The Business & Management Review, vol. 5..

Chen, H.L., 2011. A Three-Stage Expert System Based on Support Vector Machines for Thyroid Disease Diagnosis. pp. 1953-1963, 2011.

Chen, H.L., dkk., 2011. Design of an Enhanced Fuzzy k-nearest Neighbor Classifier Based Computer Aided Diagnostic System for Thyroid Disease. pp. 3243-3254.

Dogantekin, E., Dogantekin A. dan Derya, A., 2007. An Automatic diagnosis system based on thyroid gland: ADSTG. Expert System with Applications, pp. 6368-6372.

Dogantekin, E., Dogantekin, A., dan Avci, D., 2011. An expert system based on Generalized Discriminant Analysis and Wavelet Support Vector Machine for diagnosis of thyroid diseases. Expert Systems with Applications, pp. 146-150.

Farouq, K. dan Miftahus, S., 2014. Penerapan Fuzzy Tsukamoto Dalam Pengangkatan Jabatan Pegawai Di Bkd Lamongan. Jurnal Teknika, vol. 6.

Gen, M & Cheng, R 1997, Genetic Algorithms and Engineering Design, John Wiley & Sons, Inc., New York.

Gunawan, N.H. dan Rouf A., 2013. Purwarupa Sistem Kendali Kecepatan Mobil Berdasarkan Jarak dengan Sistem Inferensi Fuzzy Tsukamoto. pp. 117-126.

Kaya, Mehmet dan Alhajj, R., 2006. Utilizing Genetic Algiruthm to Optimize Membership Function for Fuzzy Weighted Association Rules Mining. Firat University Turkey And Calgary University Canada, Springer Science an Business Media inc. Netherlands.

Keles, A. dan Keles, A., 2008. ESTDD: Expert System for Thyroid Disease Diagnosis. Expert Systems with Applications 32, pp. 242-246.

Kodaz, H., dkk, 2009. Medical application of information gain based artificial immune recognition system (AIRS): Diagnosis of thyroid disease. Expert Systems with Applications, pp. 3086-3092.

Kumar, Y. dan Jain, J., 2012. Research Aspects of Expert System. International Journal of Computing & Business Research.

Liu, D.Y., dkk, 2012. Design of an Enhanced Fuzzy k-nearest Neighbor Classifier Based Computer Aided Diagnostic System for Thyroid Disease. pp. 3243-3254.

Mahmudy, WF 2013, 'Optimisation of Integrated Multi-Period Production Planning and Scheduling Problems in Flexible Manufacturing Systems (FMS) Using Hybrid Genetic Algorithms ', School of Engineering, University of South Australia

Mahmudy, W.F., Marian, R.M. dan Luong, L.H.S., 2012, Solving part type selection and loading problem in flexible manufacturing system using real coded genetic algorithms – Part II: optimization. International Conference on Control, Automation and Robotics, Singapore, 12-14 September, World Academy of Science, Engineering and Technology, pp. 706-710.

Maryaningsih, Siswanto dan Mesterjono, 2013. Metode logika fuzzy tsukamoto dalam sistem pengambilan keputusan penerimaan beasiswa. Jurnal Media Infotama, vol. 9.

Michalewicz, Z., 1996, Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Heidelberg.

Nugraha, Ivan. 2008. Algoritma Genetika untuk Optimasi Penjadwalan Kegiatan Belajar Mengajar. Makalah IF2251 Strategi Algoritmik 1.

Perera , E. D. P. dan Lahat, L., 2014. Fuzzy logic based flood forecasting model for the Kelantan River basin, Malaysia. Journal of Hydro-environment Research, pp. 1-12.

Polat, K .dan Gunes, S., 2006. A Hybrid medical decision making system based on principles components analysis, k-NN based weighted pre-processig anda adaptive neuro fuzzy inference system. Digital Signal Processing, pp. 913-921.

Pratiwi, M.I., Mahmudy, W.F. & Dewi, C. 2014, "Implementasi Algoritma Genetika pada optimasi biaya pemenuhan kebutuhan gizi", DORO: Repositor Jurnal Mahasiswa PTIIK Universitas Brawijaya, vol. 4, no. 6.

Restuputri, B.A., Mahmudy, W.F. dan Colissodin, I., 2015. Optimasi fungsi keanggotaan fuzzy Tsukamoto dua tahap menggunakan Algoritma Genetika pada pemilihan calon penerima beasiswa dan BBP-PPA (studi kasus: PTIIK Universitas Brawijaya Malang). DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, 5.

Rizzi, L., dkk, 2003. Simulation of ECB decisions and forecast of short term Euro *rate* with an adaptive fuzzy expert system. European Journal of Operational Research, pp. 363-381.

Savelonas, M., 2009. A computer-aided system for malignancy risk assessment of nodules in thyroid US images based on boundary features. computer methods and programs in biomedicine, pp. 25-32.

Savelonas, M., Maroulis D., dan Manolis, S., 2009. A computer-aided system for malignancy risk assessment of nodules in thyroid US images based on boundary features. computer methods and programs in biomedicine, pp. 25-32.

Soraya dan Siti., 2014. Implementasi metode tsukamoto dalam menentukan jumlah pembibitan kelapa sawir berdasarkan data persediaan dan jumlah permintaan (Studi kasus: Pusat penelitian Kelapa Sawit). Pelita Informatika Budi Darma, vol. VII.

Taha, H A., 2002. Operations Research-An Introduction 6th ed. Upper Saddle River NJ 07458: Prentice Hall.

Temurtas F., 2009. A comparative study on thyroid disease diagnosis using neural networks. Expert Systems with Applications, pp. 944-949.

Termurtas F., 2009. A Comparative Study On Thyroid Disease diagnosis using neural networks. Expert System with Applications, pp. 944-949.

Thamrin, F., 2012. Studi Inferensi Fuzzy Tsukamoto Untuk Penentuan Faktor Pembebanan Trafo PLN. Tesis.

Ula M., 2014. Implementasi Logika Fuzzy Dalam Optimasi Jumlah Pengadaan Barang Menggunakan Metode Tsukamoto(Studi Kasus : Toko Kain My Text). Jurnal Ecotipe, vol. 1.

Widodo, A.W. dan Mahmudy, W.F., 2010, Penerapan Algoritma Genetika Pada Sistem Rekomendasi Wisata Kuliner. Kursor, vol. 5, no. 4, pp. 205-211.