PEMILIHAN ARSITEKTUR JARINGAN SYARAF TIRUAN MULTI-LAYER PERCEPTRON MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

oleh :
IKA PRATIWI
0410960025-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2009

PEMILIHAN ARSITEKTUR JARINGAN SYARAF TIRUAN MULTI-LAYER PERCEPTRON MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

oleh :
IKA PRATIWI
0410960025-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2009

LEMBAR PENGESAHAN SKRIPSI

PEMILIHAN ARSITEKTUR JARINGAN SYARAF TIRUAN MULTI-LAYER PERCEPTRON MENGGUNAKAN ALGORITMA GENETIKA

oleh :
IKA PRATIWI
0410960025-96

Telah dipertahankan di depan Majelis Penguji
pada tanggal 7 Januari 2009
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Wayan Firdaus M., SSi., MT	**Agus Wahyu Widodo, ST**
NIP. 132 158 724	**NIP. 132 295 994**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Ika Pratiwi**
NIM	**: 0410960025-96**
Jurusan	**: Matematika**
Program Studi	**: Ilmu Komputer**
Penulis skripsi berjudul	**: Pemilihan Arsitektur Jaringan Syaraf Tiruan Menggunakan Algoritma Genetika.**

Dengan ini menyatakan bahwa :

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

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 7 Januari 2009
Yang menyatakan,

(Ika Pratiwi)
NIM. 0410960025

PEMILIHAN ARSITEKTUR JARINGAN SYARAF TIRUAN MULTI-LAYER PERCEPTRON MENGGUNAKAN ALGORITMA GENETIKA

ABSTRAK

Jaringan syaraf tiruan yang selanjutnya dikenal dengan JST merupakan algoritma pembelajaran mesin yang diilhami dari pengetahuan tentang sel syaraf biologi di dalam otak, mampu untuk menyelesaikan permasalahan yang rumit dan menemukan pola pada data. Kemampuan jaringan syaraf tiruan untuk 'belajar' sangat tergantung pada arsitektur jaringan yang digunakan. Arsitektur jaringan terdiri dari 3 lapisan yang saling terkoneksi, yaitu lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Penentuan jumlah neuron pada lapisan masukan disesuaikan dengan jumlah data masukan diskrit dari permasalahan. Dan pada lapisan keluaran disesuaikan dengan jumlah yang dibutuhkan untuk memodelkan solusi dari permasalahan. Untuk lapisan tersembunyi, jumlah yang diperlukan sangat bervariasi dan biasanya dibutuhkan analisa *heuristik* untuk menentukan jumlah unit yang optimal. Selama ini, dalam menentukan jumlah neuron dalam lapisan tersembunyi dilakukan dengan cara mencoba satu per satu kombinasi antara jumlah neuron masukan, jumlah neuron tersembunyi, dan jumlah neuron keluaran, tanpa adanya aturan yang pasti.

Algoritma genetika sebagai salah satu metode heuristik, dapat memberikan kemungkinan solusi bagi permasalahan pemilihan arsitektur jaringan. Dalam melakukan optimasi arsitektur jaringan, algoritma genetika memilih kombinasi jumlah neuron dan menggunakan parameter jaringan syaraf tiruan sebagai fungsi penghitungan *fitness* untuk menentukan kombinasi yang paling optimal. Hal ini dapat mengurangi kemungkinan mencoba satu per satu kombinasi.

Berdasarkan uji coba yang telah dilakukan, nilai MSE yang dihasilkan arsitektur jaringan syaraf tiruan dengan optimasi lebih kecil daripada arsitektur jaringan syaraf tiruan tanpa optimasi. Sehingga dapat dibuktikan bahwa algoritma genetika mampu melakukan optimasi arsitektur pada jaringan syaraf tiruan.

FINDING MULTI-LAYER PERCEPTRON NEURAL NETWORK ARCHITECTURE USING GENETIC ALGORITHMS

ABSTRACT

Artificial neural network, known as ANN, is learning machine algorithm based on models posed by biological neural system of human brain, able to solve complicated problem and finding data pattern. The artificial neural network's ability to 'learn' determined by the network architecture used. The network architecture consist of 3 connected layers. They are input layer, hidden layers and output layer. The neurons number on input layer based on the discrete input number of the problem. And the neurons number on output layer based on the number of solution models of the problem. The neurons number on hidden layer is variety and often requires heuristic analysis to decide. Producing the number of neurons on hidden layer is usually solved with manual trial of combining the number of neuron between input layer, hidden layer and output layer, without a certain rules.

Genetic algorithms as heuristic method possible to solve the problem of finding an optimal network architectures. To determine the optimal network architecture, genetic algorithm evolve several combination of architectures and using the neural network parameter as the fitness function. Thus, can help eliminate the guess-work in deciding the neural network architecture.

Based on the experiments, MSE yielded neural network which its architectures optimized by genetic algorithm is smaller than neural network without optimization. This indicates that genetic algorithm is capable to find an optimal architectures of neural network.

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur ke hadirat Allah SWT atas segala rahmat dan hidayah yang telah diberikan-Nya, sehingga skripsi yang berjudul "Pemilihan Arsitektur Jaringan Syaraf Tiruan *Multi-Layer* Perceptron Menggunakan Algoritma Genetika " ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Wayan F. Mahmudy, SSi, MT selaku pembimbing utama dan Ketua Program Studi Ilmu Komputer Unibraw Malang. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Agus Wahyu Widodo, ST selaku pembimbing pendamping dalam penulisan skripsi. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
3. Dian Eka Ratnawati, SSi, M.Kom selaku pembimbing akademik.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
5. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya.
6. Orang tua dan keluarga besar penulis atas dukungan materi dan doa restunya kepada penulis.
7. Danu Patria, selaku 'pembimbing 3', terima kasih atas semua 'bimbingan', 'semangat', tawa, dan waktumu untukku.
8. Andika, Arya, Bambang, Desi, Dofi, Gita, Galih dan Ika atas semua pelajaran tentang hidup yang kalian berikan.
9. Teman-teman seperjuangan Ilkom 2004 FMIPA UB yang telah banyak memberikan bantuan demi kelancaran penyusunan skrisi ini. Terima kasih atas senyuman, semangat dan hari-hari kita.
10. Dan semua pihak yang lupa Penulis sebutkan satu per satu.

Akhirnya, penulis sadari bahwa masih banyak kekurangan dalam penyusuan skripsi ini dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Semoga skripsi ini dapat bermanfaat.

Malang, September 2008

Penulis

DAFTAR ISI

Halaman

HALAMAN JUDUL i
LEMBAR PENGESAHAN SKRIPSI iii
LEMBAR PERNYATAAN v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix

BAB I PENDAHULUAN
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi 3
1.7 Sistematika Penulisan 4

BAB II TINJAUAN PUSTAKA
2.1 Jaringan Syaraf Tiruan 7
2.1.1 Model Umum Jaringan Syaraf Tiruan 8
2.1.2 Arsitektur Jaringan Syaraf Tiruan 10
2.1.3 Aplikasi Jeingan Syaraf Tiruan 11
2.1.4 *Multi-Layer* Perceptron 13
2.1.5 *Backpropagation Network* 14
2.2 Algoritma Genetika 15
2.2.1 Representasi Genetik 17
2.2.2 *Fitness Function* 17
2.2.3 Operator Genetik Dasar 18
2.2.3.1 Seleksi *Parent* 18
2.2.3.2 Kawin Silang (*Crossover*) 19
2.2.3.3 Mutasi 20
2.2.4 Parameter Genetika 21

BAB III METODOLOGI DAN PERANCANGAN

3.1 Deskripsi Sistem 23
3.1.1 Batasan Sistem 23
3.1.2 Kebutuhan Perangkat Lunak 24
3.1.3 Kebutuhan Perangkat Keras 24
3.2 Desain Sistem 24
3.2.1 Representasi Genetika 24
3.2.2 Struktur Data 26
3.2.3 Prosedur *Seleksi Parent* 27
3.2.4 Pembentukan Populasi 27
3.2.5 Operasi *Crossover* 27
3.2.5.1 *Single Point Crossover* 28
3.2.5.2 *Two Point Crossover* 30
3.2.6 Operasi Mutasi 32
3.2.6.1 *Mutator : drop node* 32
3.2.6.2 *Mutator : add node* 32
3.2.6.3 *Mutator : number of neurons* 33
3.2.6.4 *Mutator : drop connection* 33
3.2.6.5 *Mutator : add connection* 34
3.2.7 Penghitungan Nilai *Fitness* 34
3.2.8 Parameter Genetik 35
3.2.9 Diagram Alir Algoritma Genetika 35
3.2.10 Perancangan Antar Muka 43
3.2.11 Perancangan Uji Coba 43
3.2.11.1 Skenario Pengujian 44

BAB IV IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi 45
4.1.1 *Input* Data 45
4.1.2 Deskripsi Program 46
4.1.2.1 Inisialisasi Kromosom 46
4.1.2.2 Pembentukan Populasi Awal 47
4.1.2.3 Operasi *Crossover* 55
4.1.2.4 Operasi Mutasi 57
4.1.2.5 Penghitungan Nilai *Fitness* 60
4.1.3 Penerapan Aplikasi 60
4.1.4 Analisa Hasil 61
4.1.4.1 Hasil Uji Coba Perlakuan *Single Point Crossover* 61

4.1.4.2 Hasil Uji Coba Perlakuan *Two Point Crossover* 62
4.1.4.3 Hasil Uji Coba Perlakuan *Drop Node Mutation* 63
4.1.4.4 Hasil Uji Coba Perlakuan *Add Node Mutation* 64
4.1.4.5 Hasil Uji Coba Perlakuan *Swap Number Mutation* 65
4.1.4.6 Hasil Uji Coba Perlakuan *Drop Connection Mutation* 66
4.1.4.7 Hasil Uji Coba Perlakuan *Add Connection Mutation* 67
4.1.4.8 Hasil Uji Coba Pengaruh Algoritma Genetika 70

BAB V KESIMPULAN DAN SARAN
5.1 Kesimpulan 73
5.2 Saran 74

DAFTAR PUSTAKA..... 75

DAFTAR GAMBAR

		Halaman
Gambar 2.1	Struktur unit jaringan syaraf tiruan	7
Gambar 2.2	Model matematis jaringan syaraf tiruan	10
Gambar 2.3	Arsitektur *Multi-Layer* Perceptron	14
Gambar 2.4	Arsitektur *Backpropagation Network*.....	15
Gambar 2.5	*Flowchart* mekanisme kerja algoritma genetika	16
Gambar 2.6	Contoh individu dengan representasi bit	17
Gambar 2.7	Contoh individu dengan representasi *floating Point*	17
Gambar 2.8	Contoh individu dengan representasi integer ...	17
Gambar 2.9	Diagram *Roulette Wheel*.....	19
Gambar 2.10	*Single Point Crossover*.....	20
Gambar 2.11	*Mutasi*	21
Gambar 3.1	Arsitektur jaringan syaraf tiruan.....	25
Gambar 3.2	Representasi kromosom	25
Gambar 3.3	Struktur Data.....	26
Gambar 3.4	*Parent* 1 dan *Parent* 2	28
Gambar 3.5	*Single Point Crossover*.....	29
Gambar 3.6	*Parent* 1 dan *Parent* 2	30
Gambar 3.7	*Two Point Crossover*.....	31
Gambar 3.8	*Mutator : drop node*	32
Gambar 3.9	*Mutator : add node*	32
Gambar 3.10	*Mutator : number of neurons*.....	33
Gambar 3.11	*Mutator : drop connection*	33
Gambar 3.12	*Mutator : add connection*	34
Gambar 3.13	Diagram alir algoritma genetika	35
Gambar 3.14	Diagram alir prosedur *initial population*.....	36
Gambar 3.15	Diagram alir prosedur seleksi *parent*.....	36
Gambar 3.16	Diagram alir prosedur *SinglePointCrossover*.....	37
Gambar 3.17	Diagram alir prosedur *TwoPointCrossover*.....	38
Gambar 3.18	Diagram alir prosedur *dropnodemutation*.....	39
Gambar 3.19	Diagram alir prosedur *addnodemutation*.....	39
Gambar 3.20	Diagram alir prosedur *swapnumbermutation*.....	40
Gambar 3.21	Diagram alir prosedur *dropconnectionmutation.*	40
Gambar 3.22	Diagram alir prosedur *addconnectionmutation*...	41
Gambar 3.23	Diagram alir prosedur *insertchildtopop*.....	41

Gambar 3.24 Diagram alir prosedur *creategeneration*..... 42
Gambar 3.25 Rancangan *interface* 43
Gambar 4.1 Tampilan utama aplikasi 45
Gambar 4.2 Inisialisasi kromosom 46
Gambar 4.3 Penyimpanan variabel ind.kombinasi[i]..... 47
Gambar 4.4 Implementasi arsitektur jaringan 47
Gambar 4.5 Inisialisasi bobot bagian I..... 48
Gambar 4.6 Inisialisasi bobot bagian II..... 49
Gambar 4.7 Normalisasi data 49
Gambar 4.8 Prosedur *feedforward*..... 50
Gambar 4.9 Prosedur Hitung *Error*..... 51
Gambar 4.10 Prosedur *backprop*..... 51
Gambar 4.11 Prosedur *update* bobot bagian I 52
Gambar 4.12 Prosedur *update* bobot bagian II 53
Gambar 4.13 Prosedur *training* bagian I 53
Gambar 4.14 Prosedur *training* bagian II 54
Gambar 4.15 Prosedur *SinglePointCrossover*..... 55
Gambar 4.16 Prosedur *TwoPointCrossover* bagian I..... 56
Gambar 4.17 Prosedur *TwoPointCrossover* bagian II..... 57
Gambar 4.18 Prosedur *dropnodemutation*..... 57
Gambar 4.19 Prosedur *addnodemutation*..... 58
Gambar 4.20 Prosedur *swapnumbermutation*..... 58
Gambar 4.21 Prosedur *dropconnectionmutation*..... 59
Gambar 4.22 Prosedur *addconnectionmutation*..... 59
Gambar 4.23 Prosedur hitung nilai *fitness* 60
Gambar 4.24 Tampilan antarmuka 60
Gambar 4.25 Diagram Perlakuan Aplikasi Peramalan Beban Listrik 68
Gambar 4.26 Diagram Perlakuan Aplikasi Peramalan Saham . 69
Gambar 4.27 Diagram Perlakuan Aplikasi Pengenalan Tulisan Tangan..... 69
Gambar 4.28 Diagram Perlakuan Aplikasi Pengenalan Fonem dalam Bahasa Indonesia 70
Gambar 4.29 Grafik uji coba pengaruh algoritma genetika 71

DAFTAR TABEL

		Halaman
Tabel 3.1	Keterangan struktur data algoritma genetika.....	26
Tabel 3.2	Tabel pengujian pengaruh perlakuan	44
Tabel 3.3	Tabel pengujian pengaruh algoritma genetika...	44
Tabel 4.1	Hasil Uji Coba Perlakuan *Single Point Crossover*	61
Tabel 4.2	Hasil Uji Coba Perlakuan *Two Point Crossover*	62
Tabel 4.3	Hasil Uji Coba Perlakuan *Drop Node Mutation*	63
Tabel 4.4	Hasil Uji Coba Perlakuan *Add Node Mutation* ..	64
Tabel 4.5	Hasil Uji Coba Perlakuan *Swap Number Mutation*	65
Tabel 4.6	Hasil Uji Coba Perlakuan *Drop Connection Mutation*	66
Tabel 4.7	Hasil Uji Coba Perlakuan *Add Connection Mutation*	67
Tabel 4.8	Hasil Uji Coba Pengaruh Algoritma Genetika ..	71

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jaringan syaraf tiruan yang selanjutnya dikenal dengan JST merupakan algoritma pembelajaran mesin yang diilhami dari pengetahuan tentang sel syaraf biologi di dalam otak, mampu untuk menyelesaikan permasalahan yang rumit dan menemukan pola pada data. Menurut Arif Hermawan (2006), jaringan syaraf tiruan merupakan jaringan dari banyak unit pemroses kecil (yang disebut neuron) yang masing-masing melakukan proses sederhana, yang ketika digabungkan akan menghasilkan perilaku yang kompleks. JST dapat digunakan sebagai alat untuk memodelkan hubungan yang kompleks antara masukan (*input*) dan keluaran (*output*) pada sebuah sistem untuk menemukan pola-pola pada data. Tujuan utama jaringan syaraf tiruan adalah bagaimana membuat suatu sistem yang memiliki kemampuan untuk 'belajar', atau dengan kata lain suatu sistem yang adaptif terhadap masalah, suatu sistem yang dapat dilatih seperti halnya manusia (Nurwijaya, 2007).

Kemampuan jaringan syaraf tiruan untuk 'belajar' sangat tergantung pada arsitektur jaringan yang digunakan. Pada umumnya, JST memiliki arsitektur jaringan yang terdiri dari 3 lapisan yang saling terkoneksi, yaitu lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Dimana tiap lapisan terdiri dari beberapa neuron. Jumlah unit neuron pada lapisan masukan bersesuaian dengan jumlah data masukan diskrit dari permasalahan yang dihadapi. Sedangkan jumlah unit neuron pada lapisan keluaran bersesuaian dengan jumlah yang dibutuhkan untuk memodelkan solusi dari permasalahan. Untuk lapisan tersembunyi, jumlah yang diperlukan sangat bervariasi dan biasanya dibutuhkan analisa *heuristik* untuk menentukan jumlah unit yang optimal untuk permasalahan yang dihadapi jaringan syaraf tiruan. Selama ini, dalam menentukan jumlah neuron dalam lapisan tersembunyi dilakukan dengan cara mencoba satu per satu kombinasi antara jumlah neuron masukan, jumlah neuron tersembunyi, dan jumlah neuron keluaran, tanpa adanya aturan yang pasti. Kombinasi paling baik adalah yang memberikan nilai *error* paling minimal. Hal ini tentu saja membutuhkan waktu yang tidak sebentar, tetapi

pemilihan arsitektur jaringan sangat mempengaruhi kemampuan JST dalam menyelesaikan permasalahan, sehingga harus dilakukan.

Algoritma genetika sebagai salah satu metode heuristik, dapat memberikan kemungkinan solusi bagi permasalahan pemilihan arsitektur jaringan. Metode ini dikenal sangat efektif dalam mencari solusi optimal dalam ruang penyelesaian yang luas. Untuk permasalahan optimasi dalam jaringan syaraf tiruan, algoritma genetika dapat digunakan untuk optimasi arsitektur jaringan, bobot koneksi antar lapisan, atau keduanya (Taylor, 1997). Dalam melakukan optimasi arsitektur jaringan, algoritma genetika memilih kombinasi jumlah neuron dan menggunakan parameter jaringan syaraf tiruan sebagai fungsi penghitungan *fitness* untuk menentukan kombinasi yang paling optimal. Hal ini dapat mengurangi kemungkinan mencoba satu per satu kombinasi.

Penelitian sebelumnya dilakukan oleh Christopher M Taylor pada tahun 1997. Pada penelitian tersebut dilakukan pengenalan empat macam barang berdasarkan gambar dan suara menggunakan jaringan syaraf tiruan. Hasilnya menunjukkan bahwa algoritma genetika mampu meningkatkan nilai *fitness* populasi secara keseluruhan. Dan nilai *fitness* tertinggi adalah 0,427 dengan arsitektur jaringan syaraf tiruan 50-44-23-12-4. Penelitian Fiszelew untuk mengklasifikasi data menggunakan jaringan syaraf tiruan dengan perbaikan arsitektur oleh algoritma genetika, menunjukkan bahwa jaringan syaraf tiruan dengan algoritma genetika memiliki persentase tertinggi dalam pengklasifikasian data. Kemudian penelitian oleh Nurwijaya untuk masalah XOR *problem* dan *Double Pole Balancing* telah membuktikan secara empiris bahwa melalui representasi permasalahan dan operator yang sesuai, algoritma genetika dapat melakukan optimasi pada jaringan syaraf tiruan.

Algoritma genetika adalah algoritma pencarian yang didasarkan atas mekanisme evolusi biologis. Algoritma genetika dimulai dengan memilih himpunan penyelesaian, yang direpresentasikan dengan kromosom, yang disebut populasi. Solusi dari suatu populasi diambil untuk membentuk populasi baru. Dimana pemilihannya tergantung dari nilai *fitness*. Hal ini diharapkan agar populasi baru yang terbentuk akan lebih baik dari populasi terdahulu. Proses ini dilakukan berulang-ulang sampai terpenuhi kondisi tertentu (Kurnia, 2006).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, rumusan masalah dalam penyusunan skripsi ini adalah:

1. Bagaimana membuat model genetika untuk merepresentasikan arsitektur jaringan dan koneksinya dalam suatu individu.
2. Bagaimana perbandingan nilai *error* jaringan syaraf tiruan dengan optimasi arsitektur dan tanpa optimasi arsitektur jaringan.

1.3 Batasan Masalah

Berdasarkan rumusan masalah yang telah diuraikan sebelumnya, maka batasan masalah pada skripsi ini adalah

1. Algoritma genetika digunakan untuk optimasi arsitektur jaringan.
2. Optimasi digunakan untuk mencari *hidden layer*.
3. Pengujian dilakukan pada aplikasi jaringan syaraf tiruan yang telah dibuat pada penulisan skripsi sebelumnya mengenai jaringan syaraf tiruan *Multi-Layer* perceptron.
4. Informasi yang diambil dari aplikasi jaringan syaraf tiruan adalah arsitektur yang digunakan dan nilai *Mean Square Error*.

1.4 Tujuan

Tujuan dari penyusunan skripsi ini adalah sebagai berikut :

1. Menentukan model genetika untuk merepresentasikan arsitektur jaringan dan koneksinya dalam suatu individu.
2. Menguji penggunaan algoritma genetika untuk optimasi arsitektur jaringan syaraf tiruan dengan membandingkan nilai *error* yang dihasilkan antara jaringan syaraf tiruan dengan optimasi arsitektur dan tanpa optimasi arsitektur jaringan.

1.5 Manfaat

Adapun manfaat yang dapat diambil dari penyusunan skripsi ini adalah membangun perangkat lunak penggunaan algoritma genetika untuk optimasi arsitektur jaringan syaraf tiruan.

1.6 Metodologi

Metodologi yang digunakan dalam penyusunan skripsi ini adalah sebagai berikut :

1. Studi literatur
   Studi ini dilakukan dengan cara mencari sekaligus mempelajari beberapa literatur dan artikel mengenai algoritma genetika dan jaringan syaraf tiruan.
2. Pendefinisian Masalah
   Mendefinisikan masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan model perangkat lunak dan mengimplementasikan hasil rancangan tersebut untuk membuat model perangkat lunak pengimplementasian algoritma genetika untuk optimasi arsitektur jaringan.
4. Uji coba dan analisis hasil implementasi
   Menguji perangkat lunak, dan menganalisis hasil implementasi algoritma genetika untuk optimasi arsitektur jaringan.

1.7 Sistematika Penulisan

Pembahasan dalam skripsi ini terdiri dari beberapa bab yang dijelaskan secara ringkas sebagai berikut :

BAB I : PENDAHULUAN

Dalam bab ini dijelaskan mengenai definisi dan arsitektur jaringan syaraf tiruan pada umumnya, serta penggunaan algoritma genetika sebagai alternatif yang dapat memberikan solusi dalam permasalahan optimasi arsitektur jaringan yang termuat dalam latar belakang. Rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan dalam penyusunan skripsi ini.

BAB II : TINJAUAN PUSTAKA

Dalam bab ini dijelaskan dasar-dasar teori yang digunakan antara lain mengenai jaringan syaraf tiruan, arsitektur jaringan syaraf tiruan, aplikasi dari jaringan syaraf tiruan, algoritma genetika, representasi genetika, cara kerja algoritma genetika dan aplikasi dari algoritma genetika.

BAB III : METODOLOGI DAN PERANCANGAN

Dalam bab ini dijelaskan metode dan perancangan yang digunakan dalam membentuk representasi genetika yang dapat merepresentasikan arsitektur jaringan dan koneksi

antar lapisan, dan cara kerja algoritma genetika dalam mengoptimasi arsitektur jaringan.

BAB IV : IMPLEMENTASI DAN PEMBAHASAN

Dalam bab ini dijelaskan mengenai perancangan perangkat lunak yang mengaplikasikan algoritma genetika untuk optimasi arsitektur jaringan, pengujian dan analisisnya.

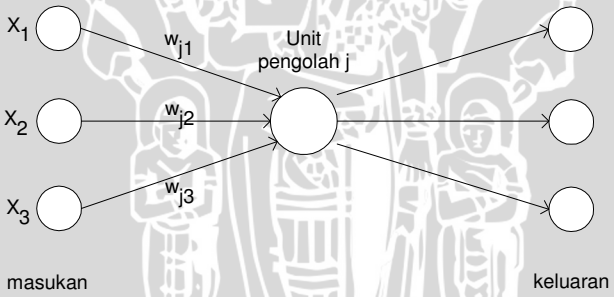
BAB V : KESIMPULAN DAN SARAN

Dalam bab ini berisi kesimpulan berdasarkan implementasi dan pembahasan yang telah dilakukan dan saran yang relevan dengan kesimpulan.

BAB II
TINJAUAN PUSTAKA

2.1 Jaringan Syaraf Tiruan

Arif Hermawan (2006), menyatakan bahwa ”Jaringan Syaraf Tiruan (JST) adalah salah satu algoritma pembelajaran mesin yang meniru cara kerja jaringan syaraf makhluk hidup. Jaringan syaraf tiruan (*artificial neural network*) merupakan jaringan dari banyak unit pemroses kecil (yang disebut neuron) yang masing-masing melakukan proses sederhana, yang ketika digabungkan akan menghasilkan perilaku yang kompleks. Jaringan syaraf tiruan dapat digunakan sebagai alat untuk memodelkan hubungan yang kompleks antara masukan (*input*) dan keluaran (*output*) pada sebuah sistem untuk menemukan pola-pola pada data”. Dan menurut Kristanto (2004), menyatakan bahwa ”Jaringan syaraf tiruan (*artificial neural network*) atau disingkat JST adalah sistem komputasi dimana arsitektur dan operasi diilhami dari pengetahuan tentang sel syaraf biologi di dalam otak. JST dapat digambarkan sebagai model matematis dan komputasi untuk fungsi aproksimasi nonlinear, klasifikasi data, *cluster* dan regresi non parametrik atau sebagai sebuah simulasi dari koleksi model syaraf biologi”. Sehingga dapat diambil suatu kesimpulan bahwa jaringan syaraf tiruan merupakan algoritma pembelajaran mesin yang diilhami dari pengetahuan tentang sel syaraf biologi di dalam otak, mampu untuk menyelesaikan permasalahan yang rumit dan menemukan pola pada data.



Gambar 2.1 Struktur unit jaringan syaraf tiruan

Menurut Yani (2005), berikut ini beberapa keunggulan dari jaringan syaraf tiruan adalah :

1. *Adaptive learning*: Suatu kemampuan untuk melakukan suatu kegiatan yang didasarkan atas data yang diberikan pada saat pembelajaran atau dari pengalaman sebelumnya.
2. *Self-Organisation*: Dapat membuat organisasi sendiri atau me-representasikan informasi yang didapat pada saat pembelajaran.
3. *Fault Tolerance* melalui *Redundant Information Coding*: Kerusakan pada bagian tertentu dari jaringan akan mengakibatkan penurunan kemampuan. Beberapa jaringan mempunyai kemampuan untuk menahan kerusakan besar pada jaringan.
4. Kelebihan jaringan syaraf tiruan terletak pada kemampuan belajar yang dimilikinya. Dengan kemampuan tersebut pengguna tidak perlu merumuskan kaidah atau fungsinya. Jaringan syaraf tiruan akan belajar mencari sendiri kaidah atau fungsi tersebut. Dengan demikian jaringan syaraf tiruan mampu digunakan untuk menyelesaikan masalah yang rumit dan atau masalah yang terdapat kaidah atau fungsi yang tidak diketahui.
5. Kemampuan jaringan syaraf tiruan dalam menyelesaikan masalah yang rumit telah dibuktikan dalam berbagai macam penelitian.

2.1.1 Model Umum Jaringan Syaraf Tiruan

Jaringan syaraf tiruan dibuat berdasarkan model syaraf manusia, tetapi dengan bagian-bagian yang lebih sederhana. Komponen terkecil dari jaringan syaraf tiruan adalah unit atau disebut juga neuron, layaknya neuron pada jaringan syaraf manusia.

Pada jaringan syaraf tiruan, neuron memiliki empat komponen utama, yaitu:

1. Koneksi masukan
   Sumber masukan neuron yang menerima masukan dari neuron-neuron lainnya atau dari luar jaringan. Setiap masukan memiliki bobot yang bersesuaian dengan setiap koneksi antar neuron. Umumnya masukan pada setiap neuron bernilai kontinu dengan rentang nilai antara [0, 1] atau [-1, -1].
2. Fungsi penjumlah
   Fungsi ini menjumlahkan masukan-masukan yang diterima berdasarkan bobot dari masukan tersebut. Masukan yang diterima dikalikan dengan bobotnya lalu hasil seluruh perkalian tersebut

dijumlahkan. Fungsi penjumlah dapat didefinisikan melalui Persamaan 2.1.

$$net = \sum_{i=0}^{n} w_i x_i \qquad (2.1)$$

yang dalam hal ini net adalah hasil keluaran dari fungsi penjumlahan, w_i menyatakan bobot koneksi masukan ke- i, dan x_i menyatakan masukan dari bobot tersebut.

3. Fungsi aktivasi

Fungsi aktivasi adalah fungsi yang menentukan keluaran sebuah neuron dari hasil penjumlahan yang didapat melalui Persamaan 2.1. Fungsi aktivasi ini dilambangkan dengan notasi σ. Terdapat beberapa jenis fungsi aktivasi, yaitu:

a. Fungsi linier

Nilai keluaran neuron sama dengan hasil penjumlahan yang didapat melalui Persamaan 2.2.

$$\sigma(\text{net}) = \text{net} \qquad (2.2)$$

yang dalam hal ini net menyatakan nilai hasil penjumlahan yang didapat melalui Persamaan 2.1.

b. Fungsi ambang (*threshold*)

Nilai keluaran neuron dikeluarkan secara diskrit jika nilai hasil penjumlahan dari Persamaan 2.1 melebihi nilai ambang tertentu. Dalam penggunaan fungsi ambang, biasanya batasan nilai tersebut adalah nol karena nilai batasan telah ikut diperhitungkan dari adanya bobot bias yang dimiliki unit neuron. Fungsi ambang dapat didefinisikan sebagai Persamaan 2.3.

$$\sigma(\text{net}) = \begin{cases} 1 \text{ untuk net} > 0 \\ 0 \text{ untuk net} \le 0 \end{cases} \qquad (2.3)$$

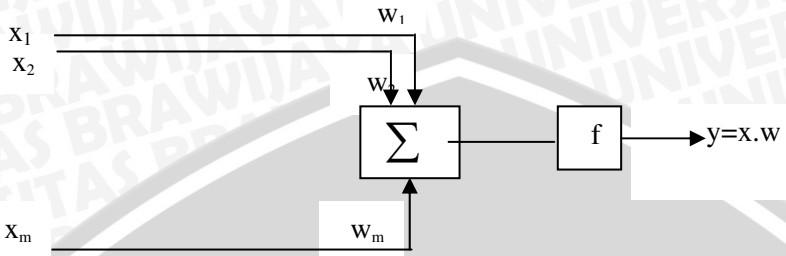
c. Fungsi sigmoid

Nilai keluaran dipetakan dari rentang (-∞,+∞) menjadi bilangan riil dengan rentang antara [0, 1]. Fungsi ini dipilih agar pembelajaran yang menggunakan turunan dari fungsi aktivasi dapat menggunakan fungsi yang kontinu. Fungsi ini dapat didefinisikan melalui Persamaan 2.4.

$$\sigma(\text{net}) = \frac{1}{1 + e^{-\text{net}}} \qquad (2.4)$$

4. Koneksi keluaran
   Koneksi keluaran mengirimkan keluaran neuron ke neuron-neuron lainnya atau sebagai keluaran dari jaringan syaraf tiruan.

Gambar 2.2 menunjukkan model yang disederhanakan dari sebuah sel syaraf (*neuron*) tiruan yang merupakan dasar dari jaringan syaraf tiruan (Kristanto, 2004).



Gambar 2.2 Model matematis jaringan syaraf tiruan

2.1.2 Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan biasanya mempunyai 3 group atau lapisan yaitu unit-unit :

1. Lapisan masukan
   Aktifitas unit-unit lapisan masukan menunjukkan informasi dasar yang kemudian digunakan dalam jaringan syaraf tiruan.
2. Lapisan tersembunyi
   Aktifitas setiap unit-unit lapisan tersembunyi ditentukan oleh aktifitas dari unit-unit masukan dan bobot dari koneksi antara unit-unit masukan dan unit-unit lapisan tersembunyi
3. Lapisan keluaran
   Karakteristik dari unit-unit keluaran tergantung dari aktifitas unit-unit lapisan tersembunyi dan bobot antara unit-unit lapisan tersembunyi dan unit-unit keluaran.

Lapisan masukan terhubung dengan lapisan tersembunyi yang selanjutnya terhubung dengan lapisan keluaran (Yani, 2005).

Terdapat 3 macam arsitektur JST. Yang pertama adalah jaringan lapisan tunggal atau *single layer*, yaitu JST yang hanya memiliki satu lapisan input dan satu lapisan output. Lapisan input terkoneksi langsung dengan lapisan output. Lapisan yang kedua adalah jaringan banyak lapisan atau *multi layer*, yaitu minimal terdiri dari 3 lapisan.

Lapisan input dan output dihubungkan oleh lapisan tersembunyi. Lapisan yang terakhir yaitu lapisan kompetitif atau *reccurent*. Hampir sama dengan lapisan yang lain, bedanya jaringan ini mempunyai umpan balik untuk inputnya (Siang, 2005).

2.1.3 Aplikasi Jaringan Syaraf Tiruan

Jaringan syaraf tiruan memiliki aplikasi yang luas pada berbagai bidang ilmu, misalnya :

1. Pemrosesan sinyal (*Signal Processing*)
   Salah satu aplikasi komersial pertama pada bidang ini adalah untuk menekan *noise* pada saluran telepon (Kristanto, 2004). Aplikasi ini diterapkan pada saluran telepon jarak jauh antar benua melalui satelit, dimana kemungkinan adanya gangguan *noise* sangat besar.
2. Pengenalan pola (*Pattern Recognition*)
   Pengenalan tulisan tangan (angka atau huruf) merupakan salah satu aplikasi ANN yang populer. Variasi yang luas pada ukuran, posisi, bentuk tulisan membuat penegenalan tulisan tangan menjadi masalah yang sulit untuk teknik konvensional. Namun dengan jaringan syaraf tiruan *multi layer* seperti *backpropagation*, dapat digunakan untuk mengenali pola tulisan (Kristanto, 2004).
3. Kedokteran
   Aplikasi dalam bidang kedokteran misalnya untuk menyediakan sistem pintar yang mampu mempelajari satu set data gejala gangguan kesehatan, diagnosa untuk gangguan tersebut, dan pengobatan untuk gangguan tersebut sehingga dapat menyediakan *output* diagnosa dan pengobatan untuk suatu *input* gejala gangguan.
4. *Speech Recognition*
   Aplikasi ANN dalam *speech recognition* misalnya dapat membuat sebuah sistem mampu menerima dan mengenali perintah suara.
5. Bisnis
   ANN memiliki aplikasi sangat luas dalam bidang bisnis, biasanya dalam masalah pembuatan keputusan. Misalnya dalam bidang pertimbangan bidang finansial mengenai keuntungan ketepatan waktu, dan kerugian dari suatu keputusan.

Beberapa aplikasi yang digunakan untuk pengujian kinerja arsitektur adalah

1. Peramalan Harga Saham Menggunakan Jaringan Syaraf Tiruan: *Backpropagation*

   Penelitian yang dilakukan oleh Ruly Dwi Setyaningrum pada tahun 2007 ini, mengimplementasikan jaringan syaraf tiruan untuk memprediksi harga saham berdasarkan data historisnya. Data yang digunakan untuk pelatihan dan pengujian adalah harga penutupan, harga permintaan, harga penawaran, volume penjualan emiten BCA, Bank Mega, Bank Mandiri dan nilai tukar rupiah pada Bursa Efek Jakarta. Uji coba dilakukan dengan menggunakan tipe jaringan *Multi-Layer* perceptron yang memiliki n_1 lapisan input, n_h lapisan tersembunyi yang ditentukan oleh *user* dan n_o lapisan keluaran dengan algoritma pembelajaran *backpropagation.* Pada akhir penelitian menunjukkan bahwa arsitektur yang paling optimal untuk emiten BCA adalah 5-1-1 (5 neuron pada layer input, 1 neuron pada layer tersembunyi, 1 neuron pada layer output) dengan MSE 0, 0131, Bank Mandiri adalah 5-1-1 dengan MSE 0,00644 dan Bank Mega adalah 5-1-1 dengan MSE 0,01577.
2. Perkiraan Penjualan Beban Lisrik Menggunakan Jaringan Syaraf Tiruan *Resilient Backpropagation*

   Penelitian yang dilakukan oleh Apriliyah pada tahun 2008 ini, merupakan implementasi jaringan syaraf tiruan resilient backpropagation untuk memprediksi penjualan beban listrik. Data yang digunakan dalam perkiraan adalah jumlah pelanggan, biaya beban, biaya pemakaian dan biaya kelebihan pemakaian dari unit Blimbing, Dinoyo, Kota dan Kebon Agung periode Januari 2003 – Desember 2007. Pelatihan dan pengujian jaringan syaraf tiruan dilakukan dengan memodifikasi jumlah neuron lapisan tersembunyi dan nilai penyesuaian. Pada akhir penelitian menunjukkan bahwa arsitektur yang paling optimal untuk unit Blimbing dan Dinoyo adalah 4-9-1, Kota 4-8-1, Kebon Agung 4-7-1. Dan nilai MSE untung unit Blimbing 0,00002832, Dinoyo 0,000197, Kota 0,00001836, dan kebon Agung 0,0000875.
3. Pengenalan Tulisan Tangan Menggunakan Algoritma Jaringan Syaraf Tiruan Propagasi Balik (*Backpropagation*)

   Penelitian yang dilakukan oleh Muh. Syaiful Arifin pada tahun 2008 ini menyajikan metode untuk mengenali tulisan tangan

dengan jaringan syaraf tiruan propagasi balik (*backpropagation*) dengan fungsi aktivasi bipolar sigmoid. Metode pengenalan pola huruf tulisan tangan difokuskan pada cara pemisahan (segmentasi) huruf-huruf yang menyusun dalam sebuah kalimat. Segmentasi yang dilakukan terdiri dari segmentasi baris, kata dan karakter. Hasil dari segmentasi karakter ini kemudian dilakukan proses ekstraksi, dimana pixel yang aktif (warna hitam) dikodekan menjadi angka 1, sedangkan pixel yang tidak aktif dikodekan menjadi angka -1 inilah yang dijadikan sebagai input ke dalam jaringan syaraf tiruan. Untuk mendapatkan struktur jaringan syaraf tiruan terbaik, dilakukan pelatihan dengan beberapa parameter diantaranya ukuran citra masukan, jumlah neuron pada layer tersembunyi, laju pembelajaran dan jumlah *max epoch.* Arsitektur jaringan syaraf tiruan yang diperoleh pada akhir penelitian adalah 400 neuron pada layer input, 60 neuron pada layer tersembunyi, dan 400 neuron pada layer output. Dan MSE terkecil 0,0000598788.

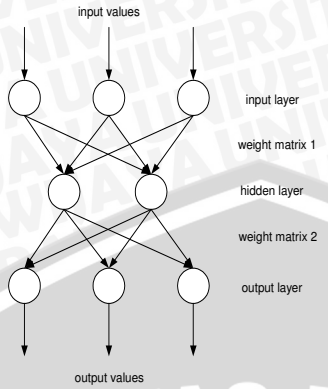
4. Metode *Second Position Asymmetric Windowing* dan Jaringan Syaraf Tiruan Backpropagation untuk Pengenalan Fonem dalam Bahasa Indonesia

   Penelitian yang dilakukan oleh Aldo Septian Praharda pada tahun 2008 ini mengimplementasikan metode *Second Position Asymmetric Windowing* (SPAW) untuk mengatasi masalah perbedaan pengucapan vokal 'e' dan 'o' pada kata dalam Bahasa Indonesia. Data yang digunakan untuk proses pembelajaran adalah kata yang mengandung huruf 'e' atau 'o' sebanyak 500 kata. Sedangkan untuk proses pengujian adalah 50 kata yang mengandung huruf 'e' atau 'o' serta belum pernah dilakukan pembelajaran. Pada proses pembelajaran dilakukan uji coba jumlah unit pada layer tersembunyi, besar *learning rate* dan jumlah *max epoch* untuk menghasilkan model terbaik. Pada akhir penelitian diperoleh arsitektur optimal adalah 189 neuron pada layer input, 80 neuron pada layer tersembunyi, dan 30 neuron pada layer output. Dan MSE terkecil 0,0054.

2.1.4 *Multi-Layer* Perceptron

Multi-Layer Perceptron diperkenalkan oleh M.Minsky dan S. Papert pada tahun 1969, merupakan pengembangan dari perceptron. Arsitektur ini memiliki *hidden layer* (layer tersembunyi) diantara

layer input dan *layer output*. *Multi-Layer* Perceptron telah mampu memecahkan semua masalah gerbang logika termasuk XOR, skema arsitektur ini diperlihatkan pada Gambar 2.3.



Gambar 2.3 Arsitektur *Multi-Layer* Perceptron

2.1.5 *Backpropagation Network*

Backpropagation Network dicetuskan oleh Paul Werbos pada tahun 1974, kemudian dikembangkan lebih lanjut pada tahun 1986 oleh David E. Rumelhart, Geoffrey E. Hinton dan Ronald J. Williams dan disebut sebagai salah satu arsitektur jaringan syaraf tiruan yang memiliki performa paling baik dan mampu menyelesaikan banyak area permasalahan. Strukturnya sama dengan *Multi-Layer* Perceptron, tetapi menggunkan algoritma *learning* yang disebut *backprpagation* membuatnya memiliki performa yang jauh lebih baik dari *Multi-Layer* Perceptron.

Metode *learning backpropagation* terdiri dari tiga tahap yaitu : *feedforward* data input untuk *training*, kalkulasi *error* dan *backpropagation* (penyebaran ke layer sebelumnya) dari hasil kalkulasi error, dan penyesuaian bobot. Algoritma ini secara garis besar memiliki empat tahapan:

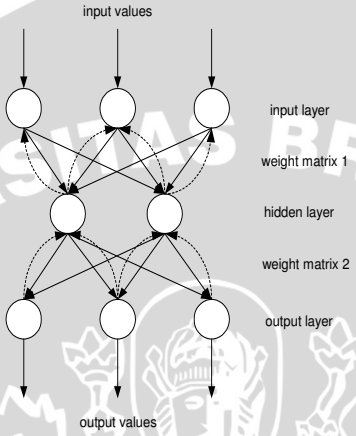
1. Pengkalkulasian nilai keluaran dari data masukan.
2. Perbandingkan nilai keluaran yang didapat dengan nilai keluaran yang diharapkan untuk menentukan tingkat kesalahan.
3. Propagasi balik tingkat kesalahan yang didapat dari lapisan neuron keluaran menuju lapisan neuron masukan.

4. Pengubahan bobot koneksi yang dimiliki setiap neuron sesuai dengan tingkat kesalahan masing-masing neuron tersebut.

Pelatihan dilakukan berulang-ulang dan berhenti jika telah mencapai batas iterasi maksimum yang ditentukan dan nilai *error* kurang dari *Mean Square Error* (MSE). Ketepatan algoritma *backpropagation* ditentukan dengan *Mean Square Error* (MSE). Semakin kecil nilai MSE maka dapat dianggap bahwa arsitektur jaringan semakin baik, demikian pula sebaliknya. MSE dihitung dengan Persamaan 2.5.

$$\text{MSE} = \frac{1}{\mathbf{N}} \sum_{\mathbf{p}} (\mathbf{t_p} - \mathbf{y_p})^2 \qquad (2.5)$$

Skema arsitektur ini diperlihatkan pada Gambar 2.4.



Gambar 2.4 Arsitektur *Backpropagation Network*

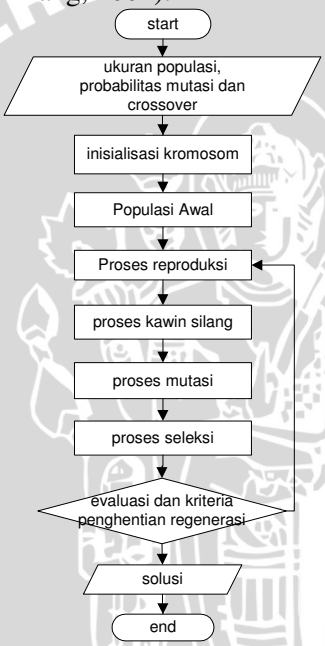
2.2 Algoritma Genetika

Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme seleksi dan genetika alami. Konsep dasar yang mengilhami timbulnya algoritma genetika adalah teori evolusi yang dikemukakan oleh Charles Darwin (Kurnia, 2006). Dalam teori tersebut dijelaskan bahwa pada proses evolusi alami, setiap individu harus melakukan adaptasi terhadap lingkungan di sekitarnya agar dapat bertahan hidup.

Algoritma genetika pertama kali dikembangkan pada tahun 1975 oleh John Holland dan rekan-rekannya dari Universitas Michigan. Sejak itu, algoritma genetika dipelajari, diteliti dan diaplikasikan secara luas pada berbagai bidang ilmu. Algoritma genetika banyak

digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter optimal (optimasi). Keunggulan algoritma genetika telah jelas terlihat dari kemudahan implementasi dan kemampuannya menentukan solusi yang bagus (bisa diterima) secara tepat untuk masalah-masalah berdimensi tinggi (Suyanto, 2005).

Algoritma genetika dimulai dengan membentuk sejumlah alternatif solusi yang disebut sebagai populasi. Pembentukan populasi awal dengan algoritma genetika dilakukan secara random. Dalam populasi tersebut, terdapat anggota populasi yang disebut dengan kromosom yang berisi informasi solusi dari sekian banyak alternatif solusi masalah yang dihadapi. Dimana setiap kromosom memiliki sebuah nilai fungsi obyektif yang bersesuaian dengan parameter masalah yang disebut nilai *fitness* (*fitness value*). Kromosom yang memiliki nilai *fitness* yang tinggi akan memiliki peluang lebih besar untuk terpilih dalam proses seleksi daripada kromosom yang memiliki nilai *fitness* yang kecil. Adapun *flowchart* dari mekanisme kerja algoritma genetika digambarkan pada Gambar 2.5 (Hannawati dan Thiang, 2002).



Gambar 2.5 *Flowchart* mekanisme kerja algoritma genetika

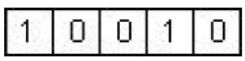
2.2.1 Representasi genetik

Individu-individu yang ada dalam populasi seringkali disebut dengan kromosom (*string*) yang mempunyai panjang yang sama. Setiap kromosom terdiri dari gen-gen yang tersusun secara linier. Setiap gen mempunyai sebuah nilai yang disebut dengan *allele* dan berada pada posisi tertentu dalam kromosom yang disebut dengan *loci*/*locus*. *Allele* tersebut dapat berupa angka biner (0/1) atau bertipe floating point tergantung dari bentuk representasi genetik yang digunakan.

Representasi merupakan permasalahan yang menjadi salah satu kunci keberhasilan dalam algoritma genetika, karena algoritma ini secara langsung memanipulasi kode yang merupakan representasi dari permasalahan, sehingga pemilihan representasi sangat menentukan kualitas hasil penyelesaian.

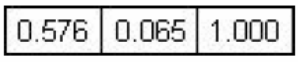
Terdapat beberapa cara dalam merepresentasikan kromosom, yaitu :

a. Representasi bit : gen dalam kromosom hanya dapat bernilai 0 atau 1



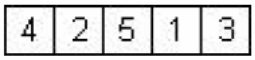
Gambar 2.6 Contoh Individu dengan Representasi Bit

b. Representasi floating point : allele dari setiap gen bertipe floating point.



Gambar 2.7 Contoh Individu dengan Representasi floating point

c. Representasi integer : allele dari setiap gen bertipe integer.



Gambar 2.8 Contoh Individu dengan Representasi Integer

2.2.2 *Fitness function*

Fitness function merupakan suatu fungsi yang digunakan untuk menghitung *fitness* atau tingkat kebaikan suatu individu untuk bertahan hidup. Fungsi ini mengambil parameter berupa individu dan menghasilkan keluaran nilai *fitness* dari individu tersebut. Untuk setiap permasalahan yang akan diselesaikan dengan algoritma

genetika harus terdefinisi untuk suatu *fitness function* (Nurwijaya, 2007).

Di dalam evolusi alam, individu yang mempunyai nilai *fitness* tinggi akan bertahan hidup, sedangkan individu yang mempunyai nilai *fitness* rendah akan mati. Penentuan nilai *fitness* sangat berpengaruh pada performasi algoritma genetika secara keseluruhan.

2.2.3 Operator Genetik Dasar

Setelah representasi permasalahan ditentukan dan *fitness function* didefinisikan, langkah berikutnya adalah mensimulasikan proses evolusi melalui pengeksekusian operator-operator. Terdapat tiga operator yang paling dasar dan paling umum diimplementasikan dalam algoritma genetika yaitu seleksi, kawin silang, dan mutasi. Dalam perkembangannya algoritma genetika beberapa operator lain telah ditambahkan untuk meningkatkan performa algoritma dan meningkatkan kemiripan algoritma dengan proses evolusi di alam.

2.2.3.1 Seleksi *Parent*

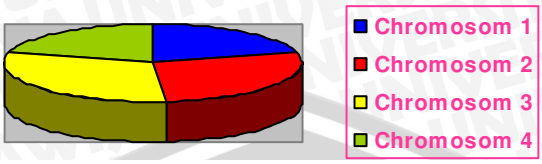
Seleksi *parent* dilakukan dengan melakukan penyeleksian dua buah kromosom yang akan dipindah silangkan untuk mendapatkan generasi baru. Penyeleksian ini dilakukan secara probabilitik berdasarkan nilai *fitness.* Kromosom yang memiliki nilai *fitness* tinggi akan memiliki peluang lebih besar untuk terpilih dalam proses penyeleksian ini. Peluang terpilihnya sebuah kromosom untuk terpilih adalah sebesar nilai *fitness* tersebut dibagi jumlah nilai *fitness* seluruh kromosom dalam populasi tersebut.

Untuk selengkapnya langkah-langkahnya adalah :

1. Hitung nilai *fitness* untuk setiap kromosom
2. Hitung total nilai *fitness* pada populasi
3. Hitung probabilitas seleksi pada setiap kromosom
4. Hitung probabilitas kumulatif untuk setiap kromosom

Ada beberapa metode penyeleksian yang biasa digunakan untuk melakukan seleksi terhadap kromosom, seperti metode *roulette wheel*, Boltzman, *tournament selection, fitness rank selection, elitism*, *steadystage* dan lain-lain. Dari metode-metode tersebut, metode *roulette wheel* merupakan metode penyeleksian yang sering digunakan.

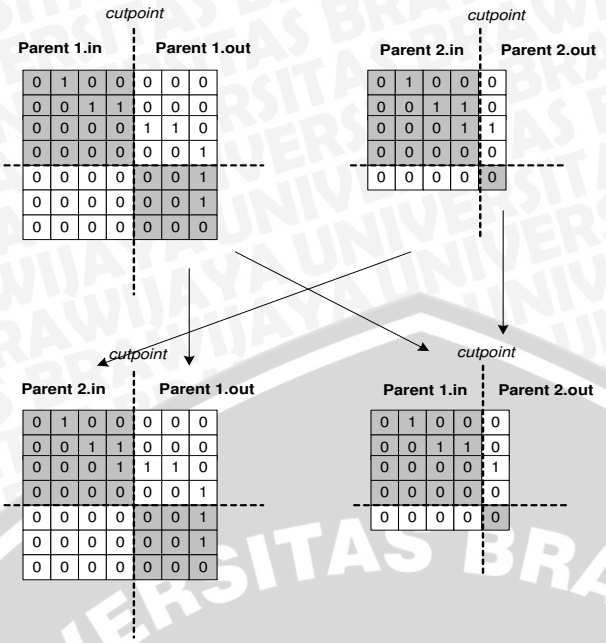
Pada metode *roulette wheel*, *parent* dipilih berdasarkan nilai *fitness*nya. Bila digambarkan pada sebuah *roulette wheel*, ukuran setiap kromosom dalam satu populasi akan seimbang sesuai nilai *fitness*nya (Koza, 1998).



Gambar 2.9 Diagram *Roulette Wheel*

2.2.3.2 Kawin Silang (*Crossover*)

Crossover atau kawin silang merupakan salah satu kunci untuk menghasilakan populasi baru. Operator ini terinspirasi dari perkawinan di alam, kawin silang bertujuan untuk menggabungkan material genetik dari dua individu yang meiliki nilai *fitness* yang tinggi untuk menhasilkan *offspring* atau anak yang memiliki nilai *fitness* yang lebih tinggi lagi dibanding induknya. *Crossover* dilakukan dengan melakukan kawin silang antara dua buah individu yang terpilih sebagai *parent.* Kemudian menukar sebagian informasi pada *parent* pertama dengan informasi pada *parent* kedua. Terdapat beberapa metode *crossover* yang sering digunakan antara lain *Partially Mappes Crossover* (PMX), *Order Crossover* (OX), *position based crossover*, *order based crossover*, *single point crossover* dan *two point crossover*. Sedangkan untuk masalah yang diselesaikan dengan menggunakan matriks dua dimensi dapat menggunakan metode *crossover* dengan melakukan penukaran informasi secara vertikal, horizontal atau kembinasi dari keduanya (Abdelmaguid, 2004).



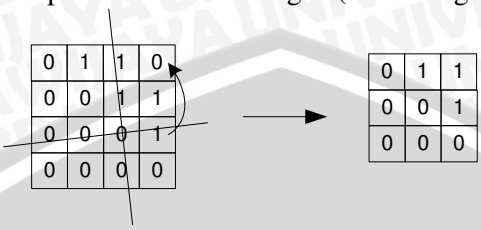
Gambar 2.10 *Single Point* Crossover

Gambar 2.10 merupakan ilustrasi dari proses *crossover* yang dilakukan pada representasi kromosom berupa matrik. Metode yang digunakan pada contoh di atas merupakan metode kombinasi penukaran informasi secara vertikal dan horizontal. Proses tersebut dilakukan dengan menukar sebagian informasi dari *parent* pertama dengan informasi dari *parent* kedua pada baris dan kolom yang sama.

2.2.3.3 Mutasi

Mutasi merupakan suatu metode yang mengimplementasikan fakta bahwa hasil perkawinan di alam tidak selalu dapat diprediksi, terkadang terjadi penyimpangan sifat suatu individu dari induknya. Mutasi berguna untuk mengimbangi penurunan variasi sifat indibidu karena operasi *crossover*. Operasi *crossover* secara alami akan menurunkan variansi sifat individu yang unggul akan mendominasi keseluruhan populasi, dengan mutasi variansi dapat kembali ditingkatkan (Nurwijaya, 2007).

Cara termudah untuk melakukan mutasi adalah dengan mengubah satu atau lebih gen (bagian dari kromosom) dengan probabilitasnya tergantung pada *mutation rate*. Terdapat beberapa metode dalam mutasi antara lain *invertion mutation*, *insertion mutation*, *reciprocal exchange mutation* dan *shif mutation.* Sedangkan untuk masalah yang diselesaikan dengan menggunakan matrik dua dimensi dapat menggunakan metode mutasi dengan melakukan pengurangan dan penambahan nilai pada satu atau lebih gen (Abdelmaguid, 2004).



Gambar 2.11 Mutasi

Gambar 2.11 merupakan ilustrasi dari proses mutasi dengan melakukan pengurangan satu gen pada satu kromosom. Sebelumnya dibangkitkan nilai random untuk menentukan baris dan kolom yang akan dikurangi dari kromosom.

2.2.4 Parameter Genetika

Dalam penerapan algoritma genetika, terdapat bebrapa parameter yang digunakan. Parameter ini menentukan kesuksesan suatu proses optimasi. Adapun parameter-parameter tersebut adalah

1 Ukuran populasi (*pop_size*)
Ukuran populasi sangat mempengaruhi unjuk kerja dan keefektifan algoritma genetika. Jika populasi yang dibentuk terlalu kecil, maka unjuk kerjanya buruk karena populasi tersebut kurang bisa menyediakan ruang solusi yang bisa merepresentasikan keseluruhan ruang persoalan. Jika populasi terlalu banyak, maka membutuhkan waktu komputasi yang lama dalam proses pencariannya.

2 Probabilitas *crossover* (Pc)
C*rossover rate* (Pc) adalah perbandingan jumlah *offspring* (kromosom anak) yang dihasilkan pada tiap-tiap generasi terhadap jumlah kromosom dalam satu populasi (*pop_size*) (Gen and Cheng, 1997). Dalam setiap populasi sebanyak Pc x

pop_size kromosom melakukan kawin silang (*crossover*), dimana Pc adalah nilai probabilitas *crossover* dan *pop_size* adalah jumlah kromosom dalam satu populasi. Semakin besar probabilitas *crossover* akan memungkinkan pencapaian alternatif solusi yang lebih bervariasi dan mengurangi kemungkinan menghasilkan nilai optimum yang tidak dikehendaki. Tetapi bila nilai ini terlalu tinggi, akan mengakibatkan pemborosan waktu untuk melakukan perhitungan di daerah solusi yang tidak menjanjikan hasil yang optimal. Pada umumnya nilai probabilitas *crossover* yang sering digunakan adalah berkisar antar 0,8 sampai 0,95. Akan tetapi pada masalah tertentu, nilai probabilitas *crossover* 0,6 menunjukkan hasil yang lebih baik (Koza, 1998).

3 Probabilitas mutasi (Pm)

Mutasi digunakan untuk meningkatkan variansi populasi. Mutasi dilakukan secara acak, tiap unit dasar (gen) dalam kromosom mempunyai kemungkinan tertentu untuk dipertukarkan berdasarkan besarnya probabilitas mutasi (*mutation rate*). Nilai probabilitas mutasi yang terlalu rendah dapat mengakibatkan gen-gen yang berpotensi tidak dicoba. Sebaliknya jika terlalu tinggi, dapat menghilangkan sifat kemiripan dengan *parent*nya, sehingga proses pencarian bisa salah dan menjauh dari daerah solusi yang diinginkan. Untuk nilai probabilitas mutasi yang sering digunakan adalah berkisar antara 0,005-0,01 (Koza, 1998).

Dalam berbagai jurnal, artikel maupun buku teks tidak ditentukan aturan pasti mengenai ukuran parameter genetik tersebut, hanya saja terdapat suatu kesepakatan seperti peluang *crossover* besar dan peluang mutasi kecil (Basri, 2008).

BAB III
METODOLOGI DAN PERANCANGAN

3.1 Deskripsi Sistem

Perangkat lunak yang akan dibangun merupakan perangkat lunak yang dibuat sebagai implementasi teori yang bertujuan untuk mengetahui bagaimana algoritma genetika dapat memberikan solusi dalam permasalahan pemilihan arsitektur yang seharusnya digunakan pada jaringan syaraf tiruan. Dalam hal ini, arsitektur yang dimaksud adalah jumlah neuron pada *hidden layer*. Perangkat lunak yang dibangun digunakan sebagai alat penguji bagaimana pengaruh perlakuan operator genetika dasar dalam menentukan arsitektur yang sesuai untuk jaringan syaraf tiruan. Terdapat 7 perlakuan yang berbeda dalam pembentukan populasi, akan dijelaskan lebih lanjut pada subbab 3.2.5 untuk operasi *crossover* dan 3.2.6 untuk operasi mutasi. Setiap perlakuan dihitung nilai *fitness* dengan menggunakan Persamaan 3.1. Arsitektur yang sesuai adalah arsitektur jaringan yang menghasilkan nilai *error* paling kecil.

Untuk menguji perangkat lunak yang telah dibangun, dilakukan perbandingan nilai *error* yang dihasilkan antara jaringan syaraf tiruan dengan optimasi arsitektur dan tanpa optimasi arsitektur jaringan. Aplikasi jaringan syaraf tiruan tanpa optimasi arsitektur jaringan adalah aplikasi dari penelitian tugas akhir mengenai jaringan syaraf tiruan *backpropagation* yang telah dilakukan, seperti yang telah dijelaskan pada tinjuan pustaka subbab 2.1.3

3.2.1 Batasan Sistem

Perangkat lunak ini dibangun untuk menyelesaikan permasalahan pemilihan arsitektur jaringan syaraf tiruan yang telah disebutkan di latar belakang. Metode yang digunakan adalah algoritma genetika. Data percobaan yang digunakan adalah data yang dibangkitkan secara *random* dengan tetap memperhatikan syarat dan ketentuan yang berlaku pada pemilihan arsitektur jaringan syaraf tiruan.

3.1.2 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang digunakan dalam pembuatan sistem ini adalah :

- Format file teks adalah *.txt
- Software Borland Delphi 6.0 untuk implementasi proses dan visualisasi.
- Software Microsoft Access 2003
- Menggunakan *personal computer* dengan sistem operasi Windows XPprofessional v 2002 SP2

3.1.3 Kebutuhan Perangkat Keras

Dalam membangun sistem ini, perangkat keras yang digunakan adalah *personal computer* dengan spesifikasi :

- Processor Intel pentium IV 2.40 GHz
- Memory 256 Mb
- Keyboard
- Mouse

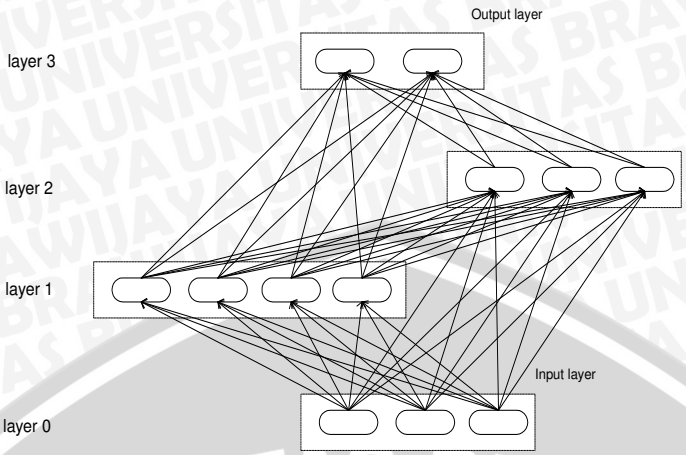
3.2 Desain Sistem

Sistem yang akan dibangun ini menggunakan algoritma genetika. Dalam algoritma genetika terdapat beberapa komponen penyusun yang utama yang digunakan untuk memodelkan permasalahan sekaligus digunakan sebagai alat penyelesaian masalah. Pada subbab berikut akan di bahas masing-masing komponen algoritma genetika, diagram alir dan rancangan antarmuka sistem.

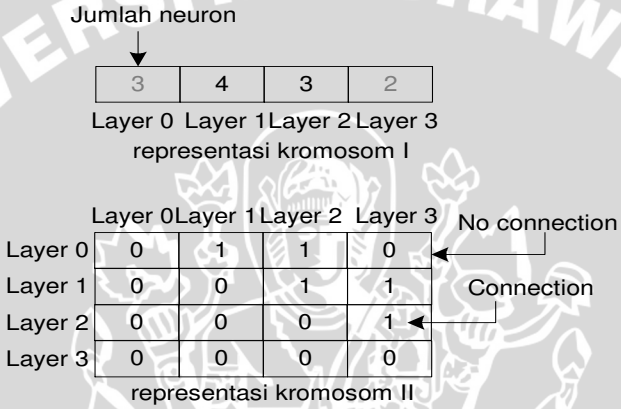
3.2.1 Representasi genetika

Seperti yang dilakukan oleh Matteo Matteuci pada penelitian sebelumnya**,** arsitektur jaringan syaraf tiruan yang terdiri dari jumlah neuron, jumlah *layer*, dan koneksi tiap *layer* direpresentasikan dalam dua bagian kromosom. Bagian pertama berupa representasi integer menunjukkan informasi mengenai jumlah *layer*, dan jumlah neuron tiap *layer*. Dan bagian kedua menunjukkan koneksi jaringan direpresentasikan dalam matrik dua dimensi, dengan baris dan kolom sebanyak *layer* yang digunakan pada arsitektur jaringan syaraf tiruan tersebut. Jumlah neuron pada lapisan pertama (lapisan masukan) dan

lapisan kedua (lapisan keluaran) sudah ditentukan pada tiap aplikasi jaringan syaraf tiruan.



Gambar 3.1 Arsitektur jaringan syaraf tiruan



Gambar 3.2 Representasi kromosom

Arsitektur jaringan syaraf tiruan pada Gambar 3.1 direpresentasikan seperti pada Gambar 3.2. Pada representasi kromosom I menunjukkan bahwa arsitektur jaringan ini terdiri dari 4 *layer*, dengan *layer* 0 sebagai input *layer*, *layer* 1 dan *layer* 2 sebagai hidden *layer* dan *layer* 3 sebagai output *layer*. *Layer* 0 memiliki 3 neuron, *layer* 1 memiliki 4 neuron, *layer* 2 memiliki 3 neuron dan *layer* 3 memiliki 2 neuron.

Pada representasi kromosom II merepresentasikan koneksi antar*layer* pada arsitektur jaringan. *Layer* 0 terhubung dengan *layer* 1 dan *layer* 2, maka pada sel pertemuan baris *layer* 0 dan kolom *layer* 1 diberi nilai 1, yang artinya terhubung. Begitu juga pada sel pertemuan baris *layer* 0 dan kolom *layer* 2 diberi nilai 1 dan seterusnya. Sedangkan sel yang lain diberi nilai 0, yang artinya tidak terhubung.

3.2.2 Struktur Data

Struktur data untuk merepresentasikan kromosom yang telah dirancang dalam penelitian ini dapat dilihat pada Gambar 3.3

```
Tconnection = array[1..5,1..5] of integer;
TArsitektur = record
  kombinasi: array [1..5] of integer;
  connect:tconnection ;
  MSE : double;
  fitness:double
 end;
TLayer = record
  numofneuron : integer;
  connectwith:integer;
  neuron : array[1..400] of TNeuron;
 end;
TNeuron = record
  no,
  start,
  finish :integer;
  weight: real;
 end;
TPop = array [1..200] of TArsitektur;
```

Gambar 3.3 Struktur Data

Struktur data pada Gambar 3.3 dijelaskan dalam Tabel 3.1.

Tabel 3.1 Keterangan struktur data algoritma genetika

Tconnection	Matriks 5x5 untuk meyimpan koneksi antarlayer. Berisi nilai 0 dan 1. Nilai 1 menunjukkan adanya koneksi, dan 0 menunjukkan tidak ada koneksi.
TArsitektur	*record* untuk menyimpan informasi arsitektur jaringan syaraf tiruan, terdiri dari kombinasi yang berisi jumlah neuron masing-masing layer,

	koneksi antarlayer, MSE yang dihasilkan, dan fitness.
TLayer	*record* untuk menyimpan informasi layer, terdiri dari jumlah neuron, jumlah koneksi, dan tiap neuron.
TNeuron	*record* untuk menyimpan informasi neuron, terdiri dari nomor neuron, tujuan koneksi, dan *weight*.
TPop	*Array* untuk menyimpan populasi yang terdiri dari paling banyak 200 individu yang ber-*type* data tarsitektur.

3.2.3 Prosedur Seleksi *Parent*

Terdapat beberapa cara pemilihan *parent* yang digunakan dalam algoritma genetika. Dalam penelitian ini digunakan metode *roulette wheel*, karena metode ini memberikan kesempatan yang besar pada kromosom yang kuat (memiliki nilai *fitness* optimum) untuk dapat disilangkan, dan dari beberapa penelitian sebelumnya metode ini dipandang cukup layak menghasilkan kromosom yang baik (berpeluang menjadi solusi optimal).

3.2.4 Pembentukan Populasi

Pada penelitian ini populasi dibentuk dengan melakukan *crossover* dan mutasi. Diharapkan dengan pemilihan operator *crossover* dan mutasi yang baik akan didapatkan individu-individu yang memiliki nilai *fitness* yang tinggi.

3.2.5 Operasi *crossover*

Operasi *crossover* dilakukan dengan metode *one single point crossover* dan *two point crossover*. Titik potong dipilih secara acak. *Crossover* dilakukan pada dua bagian kromosom dengan menggunakan titik potong yang sama.

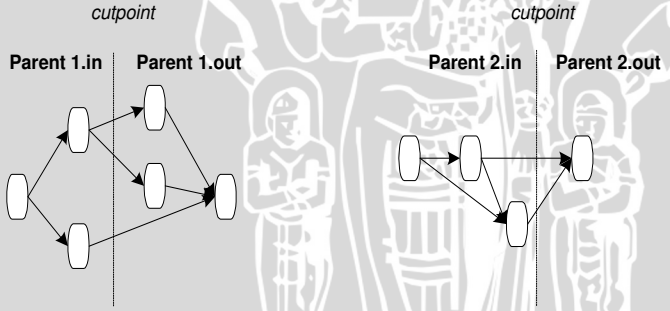
3.2.5.1 *Single point crossover*

Metode ini menggabungkan dua jaringan dengan memotong arsitektur jaringan menjadi dua bagian menggunakan titik potong yang memisahkan lapisan masukan dan lapisan keluaran.

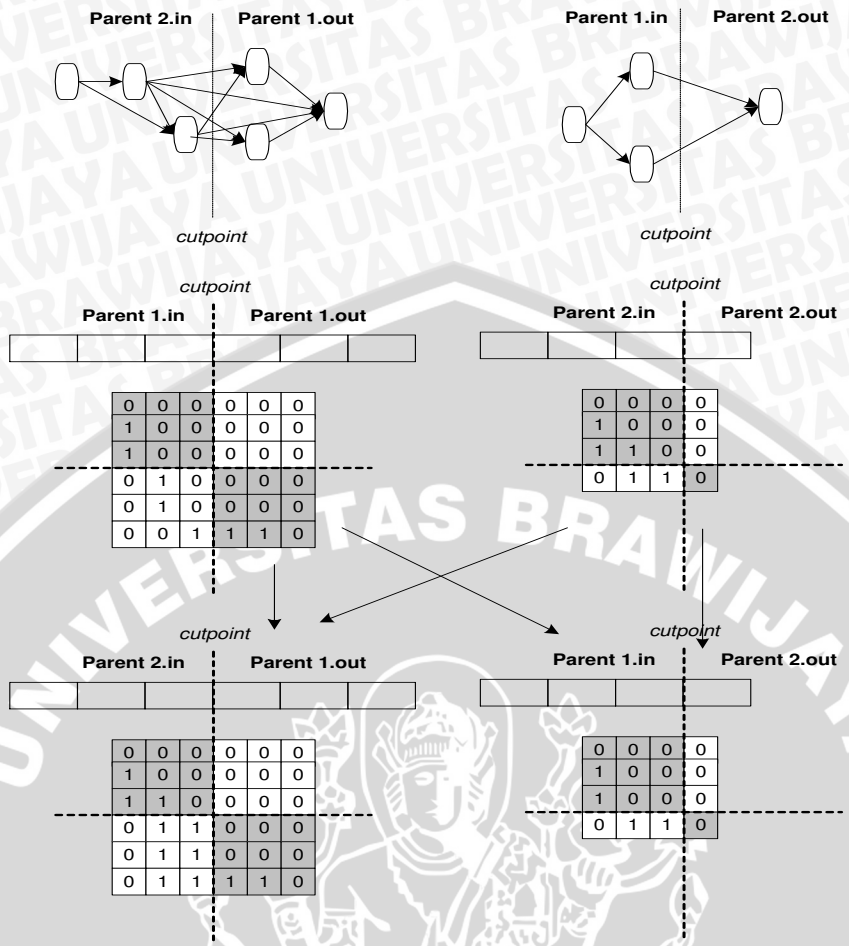
Untuk menjaga koneksi dalam arsitektur jaringan, maka setiap koneksi keluaran dari sebagian koneksi masukan pada arsitektur jaringan pertama harus dihubungkan dengan setiap koneksi masukan pada arsitektur jaringan kedua. Dengan perlakuan ini, jumlah koneksi pada generasi baru dapat lebih banyak daripada jumlah koneksi awal, namun validitas koneksi dapat terjaga.

Gambar 3.5 merupakan ilustrasi proses *single point crossover* dari dua arsitektur jaringan yang berbeda. Dua arsitektur tersebut sebagai *parent* 1 dan *parent* 2, seperti digambarkan pada Gambar 3.4. Titik potong yang dipilih secara acak digunakan pada dua bagian kromosom. Titik potong ini membagi kromosom bagian pertama menjadi dua, dan kromosom bagian kedua menjadi empat. Pada kromosom bagian pertama, bagian *input* dari *parent* 1 digabungkan dengan bagian *output* dari *parent* 2, begitu pula sebaliknya. Pada kromosom bagian kedua, matrik koneksi pada tiap individu baru dibentuk dengan menggabungkan bagian kiri-atas dan bagian kanan-bawah dari *parent* 1. Sedangkan bagian kanan-atas dan bagian kiri-bawah mengikuti *parent* 2. Namun untuk mempertahankan validitas koneksi dari *parent* 1, juga dilakukan penambahan koneksi menyesuaikan dengan keadaan koneksi dari *parent* 1.

Sel (i,j) menggambarkan adanya koneksi antara *layer* i dan *layer* j. *Layer* j menunjukkan *layer* masukan, dan *layer* i adalah *layer* keluaran dari *layer* j.



Gambar 3.4 *Parent* 1 dan *Parent* 2



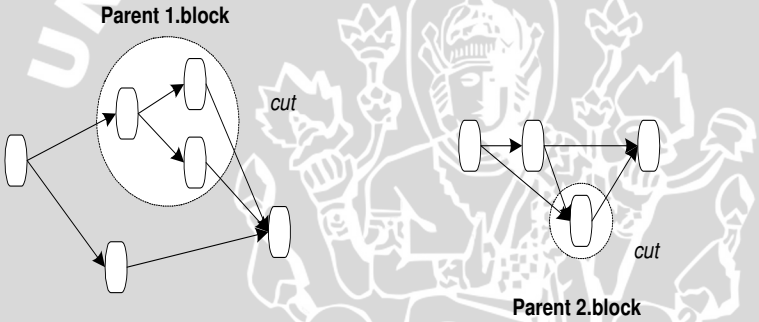
Gambar 3.5 *Single point crossover*

Pada *offspring* 1, dilakukan penyesuaian koneksi pada bagian kiri-bawah. Keadaan koneksi *layer* 1 pada *parent*2.in adalah terhubung dengan *output layer*, maka pada *offspring* 1, *layer* 1 juga harus terhubung dengan *output layer*, yang sebelumnya tidak terhubung. Begitu pula pada *layer* 2, pada *parent*2.*in* terhubung dengan *output layer*, maka pada *offspring* 1 juga harus terhubung.

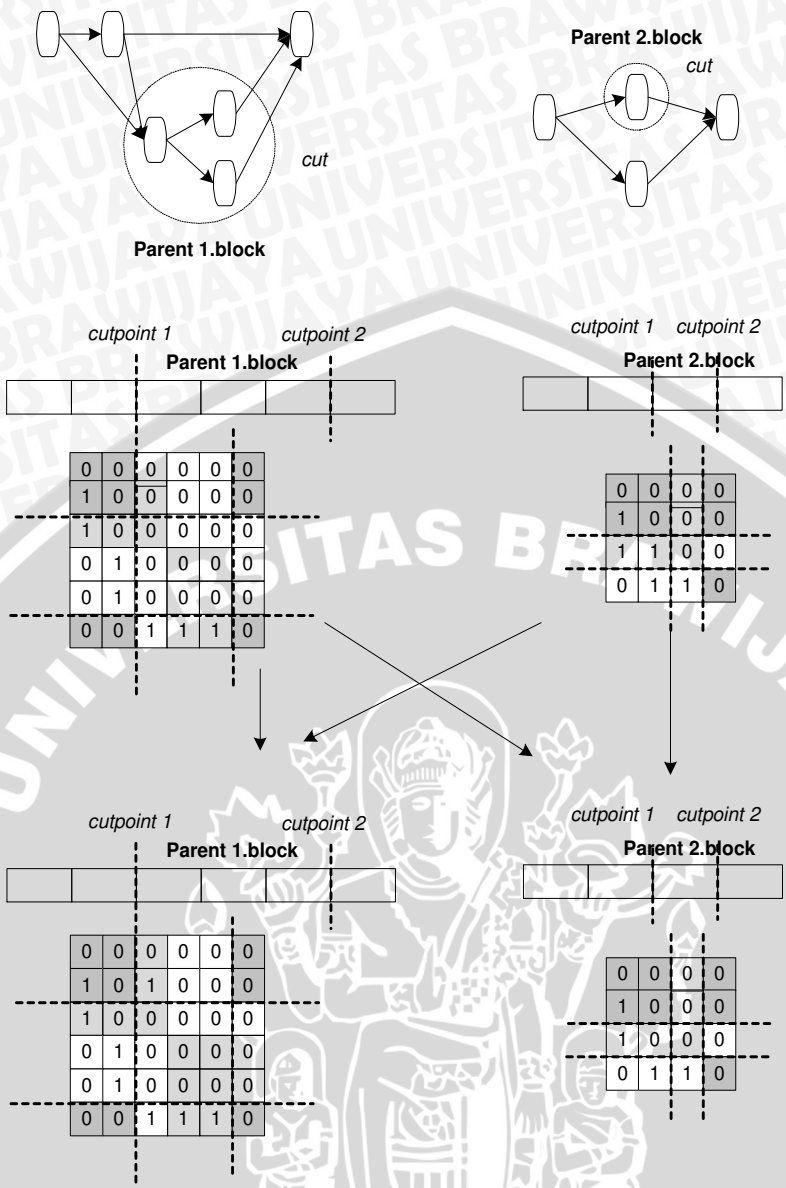
3.2.5.2 *Two point crossover*

Metode ini mempunyai konsep yang sama dengan *single crossover*, perbedaanya pada jumlah titik potong yang digunakan, yaitu sebanyak dua. Dalam menggabungkan dua arsitektur jaringan dilakukan dengan cara memotong tiap subgraf yang terkoneksi dan menukar dua subgraf diantara titik potong.

Gambar 3.7 merupakan ilustrasi proses *two point crossover* dari dua arsitektur jaringan yang berbeda. Dua arsitektur tersebut sebagai *parent* 1 dan *parent* 2, seperti digambarkan pada Gambar 3.6. Dua titik potong yang dipilih secara acak digunakan pada dua bagian kromosom. Titik potong ini membagi kromosom bagian pertama menjadi tiga, dan kromosom bagian kedua menjadi enam. Pada kromosom bagian pertama, bagian *input* dan *output* dari *parent* 2 digabungkan dengan bagian *parent*1.*block*, begitu pula sebaliknya. *Parent1.block* adalah bagian kromosom yang terletak diantara dua titik potong. Pada kromosom bagian kedua, matrik koneksi pada tiap individu baru dibentuk dengan menggabungkan bagian-bagian di luar dua titik potong dari *parent* 2 dengan bagian *parent*1.*block*. Untuk mempertahankan validitas koneksi dari kedua *parent,* dilakukan penambahan koneksi menyesuaikan dengan keadaan koneksi dari *parent*.



Gambar 3.6 *Parent* 1 dan *Parent* 2



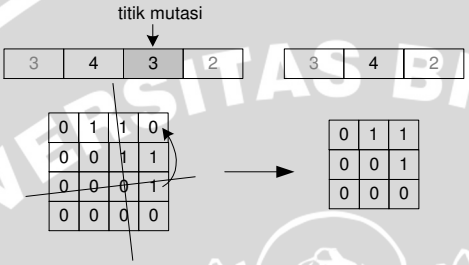
Gambar 3.7 *Two point crossover*

3.2.6 Operasi mutasi

Operasi mutasi dilakukan pada jumlah *layer*, jumlah neuron, dan koneksi. Operator mutasi memilih secara acak sebuah angka untuk mewakili *layer* yang dihapus atau ditambah, atau neuron yang diubah serta koneksi yang dihapus atau ditambah

3.2.6.1 *Mutator : drop node*

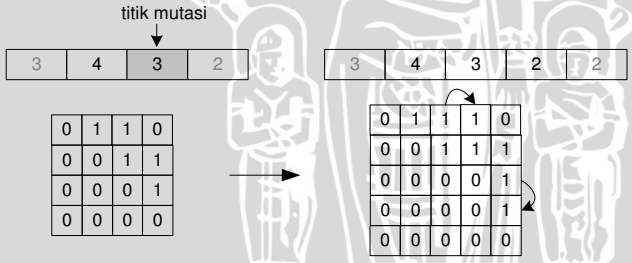
Operasi mutasi memilih secara acak sebuah *layer* dan menghapus *layer* tersebut dari arsitektur jaringan. Sebelum menghapus *layer* tersebut, setiap koneksi masukan dari *layer* tersebut dihubungkan dengan semua tujuan dari koneksi keluaran. Operasi ini menjamin validitas koneksi. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.8.



Gambar 3.8 *Mutator* : *drop node*

3.2.6.2 *Mutator : add node*

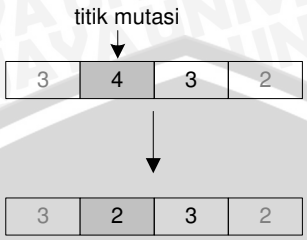
Operasi ini menambah *layer* pada arsitektur jaringan. *Layer* yang sudah ada dipilih secara acak dan diduplikasi. Kemudian jumlah neuron di dalamnya di inisialisasi secara acak. Koneksi dari neuron tersebut mengikuti koneksi *layer* yang terduplikasi. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.9.



Gambar 3.9 *Mutator* : *add node*

3.2.6.3 *Mutator : number of neurons*

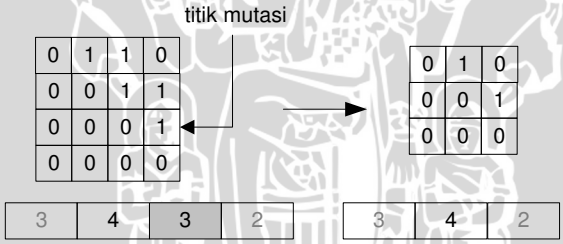
Operasi mutasi ini mengubah jumlah neuron yang ada pada *layer* tertentu pada arsitektur jaringan. Titik mutasi dipilih secara acak dan jumlah neuron pada spesifik *layer* ditukar. Pertukaran jumlah neuron tidak mengubah koneksi, sehingga hanya dilakukan pada kromosom bagian pertama. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Mutator* : *number of neurons*

3.2.6.4 *Mutator : drop connection*

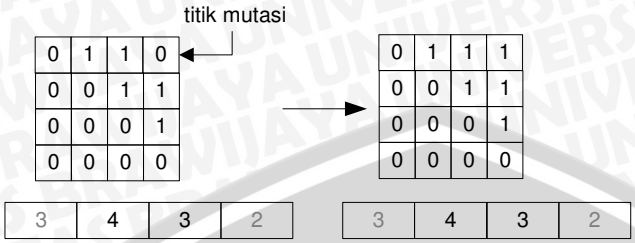
Operasi mutasi ini menghapus koneksi dari matrik koneksi arsitektur jaringan. Ketika koneksi dihapus, maka akan dilakukan pengecekan pada *layer* yang tidak menjadi tujuan dari koneksi masukan dan *layer* yang tidak menjadi output dari arsitektur jaringan. Jika tidak memenuhi kondisi tersebut, penghapusan koneksi dapat merusak validitas koneksi yang telah ada. Dalam kasus *offspring* yang dihasilkan tidak valid, maka dilakukan inisialisasi baru secara acak. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.11.



Gambar 3.11 *Mutator* : *drop connection*

3.2.6.5 *Mutator : add connection*

Operasi mutasi ini menambah koneksi baru pada matrik koneksi arsitektur jaringan. Jika jaringan sudah saling terhubung, maka generasi baru tidak diperbaiki lagi. Operasi ini menjamin validitas koneksi. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.12



Gambar 3.12 *Mutator* : *add connection*

3.2.7 Penghitungan Nilai *Fitness*

Tiap individu yang dihasilkan dari operasi genetika, setelah dilakukan pelatihan jaringan syaraf tiruan akan menghasilkan nilai MSE. Nilai MSE ini kemudian dibandingkan dengan nilai MSE yang diperoleh dari pelatihan jaringan syaraf tiruan tanpa operasi genetika. Nilai perbandingan ini kemudian dihitung sebagai nilai *fitness* dari suatu individu dengan menggunakan Persamaan 3.1.

$$fitness = \frac{MSE\ best}{actual\ MSE} \quad (3.1)$$

MSE best menyatakan nilai MSE yang diperoleh dari pelatihan jaringan syaraf tiruan tanpa operasi genetika, nilai ini telah diperoleh dari aplikasi sebelumnya, yang telah dijelaskan pada subbab 2.1.3. *Actual MSE* menyatakan nilai MSE yang diperoleh dari pelatihan jaringan syaraf tiruan dengan operasi genetika.

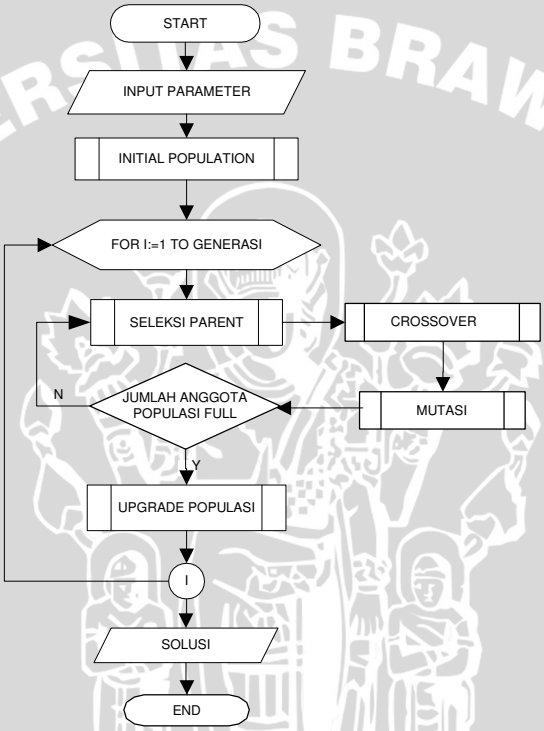
Semakin rendah nilai *actual MSE*, semakin tinggi nilai *fitness* dari individu tersebut. Dan semakin tinggi nilai *fitness*, semakin baik individu tersebut. Hal ini menunjukkan bahwa pelatihan jaringan syaraf tiruan dengan algoritma genetika lebih baik daripada tanpa algoritma genetika.

3.2.8 Parameter Genetik

Parameter genetik digunakan untuk mengendalikan operator-operator genetik, parameter yang digunakan adalah: ukuran populasi, jumlah generasi, *probabilitas crossover*, dan *probabilitas* mutasi. Karenanya ditetapkan ukuran populasi 10, jumlah generasi 10, peluang *crossover* 95%, dan peluang mutasi 5%. Jumlah generasi yang besar berarti semakin banyak iterasi yang dilakukan berakibat pada semakin besar daerah solusi yang dieksploitasi. Sedangkan jumlah populasi yang besar berarti memberikan banyak pilihan untuk melakukan *crossover*.

3.2.9 Diagram Alir Algoritma Genetika

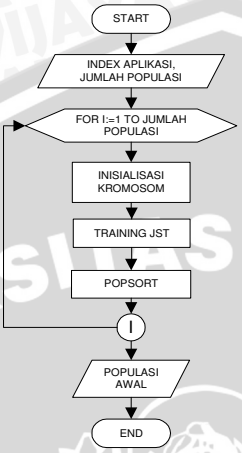
Diagram alir algoritma genetika dapat dilihat pada Gambar 3.13.



Gambar 3.13 Diagram alir algoritma genetika

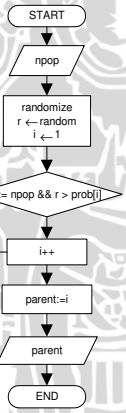
Penjelasan untuk Gambar 3.13 adalah:

1. Sistem dimulai dengan inisialisasi parameter untuk jaringan syaraf tiruan adalah jenis aplikasi yang diuji, jumlah neuron untuk lapisan masukan dan jumlah neuron untuk lapisan keluaran, sedangkan parameter untuk algoritma genetika meliputi jumlah populasi, jumlah generasi, probabilitas crossover, dan probabilitas mutasi.
2. Selanjutnya bangun populasi awal dilakukan untuk memberikan nilai awal pada kromosom secara acak.



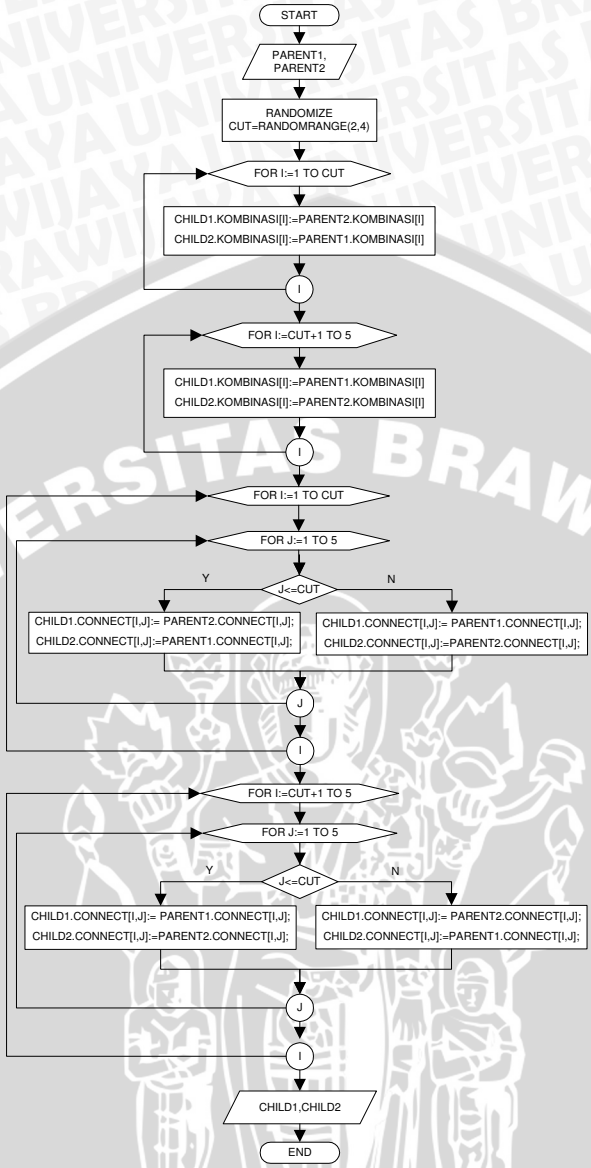
Gambar 3.14 Diagram alir prosedur *initial population*

3. Setelah pembentukan populasi awal, dilakukan seleksi *parent.*

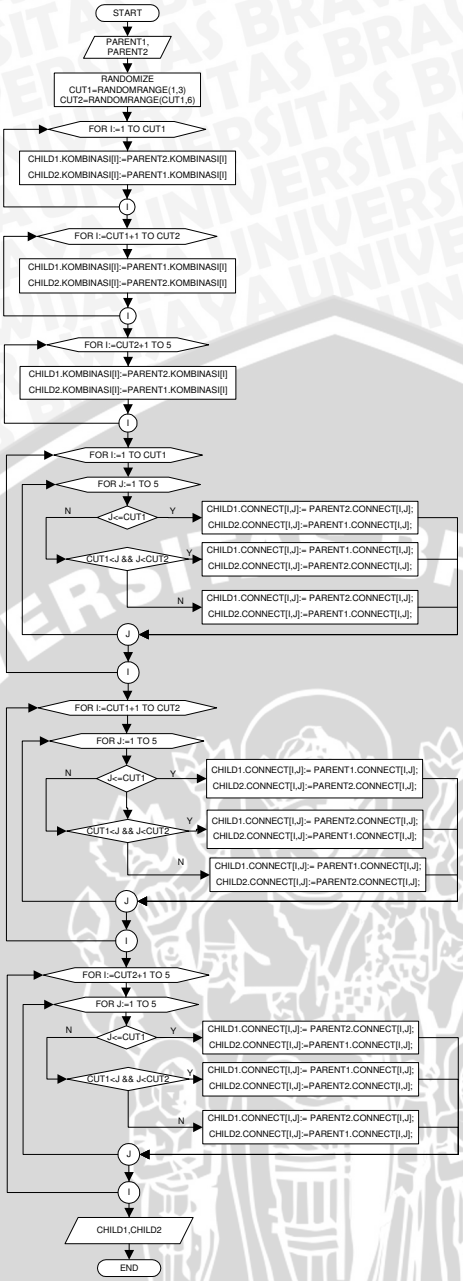


Gambar 3.15 Diagram alir prosedur seleksi *parent*

kemudian dilakukan *crossover*. *Crossover* dilakukan dengan dua metode, *single point crossover* dan *two point crossover*.

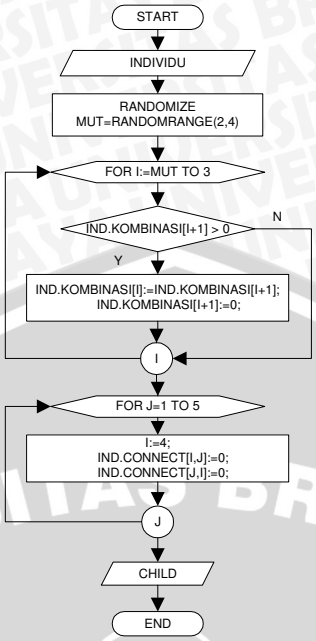


Gambar 3.16 Diagram alir prosedur *single point crossover*

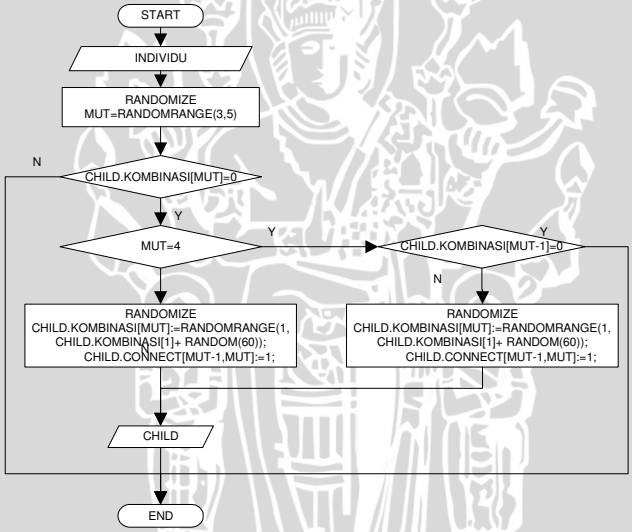


Gambar 3.17 Diagram alir prosedur *two point crossover*

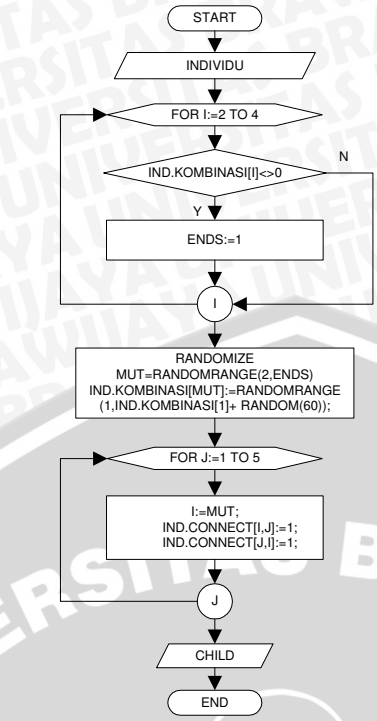
4. Pembentukan generasi baru juga dilakukan dengan mutasi, terdapat 5 metode mutasi.



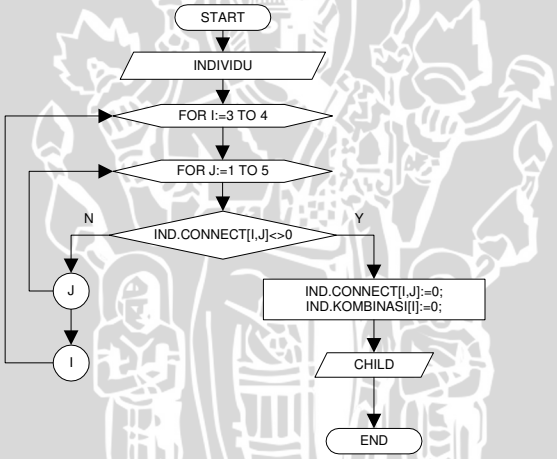
Gambar 3.18 Diagram alir prosedur *drop node mutation*



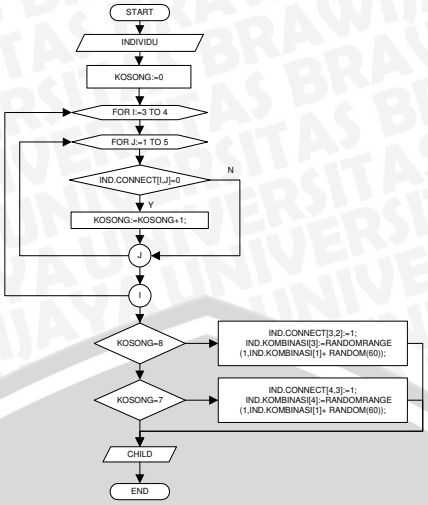
Gambar 3.19 Diagram alir prosedur *add node mutation*



Gambar 3.20 Diagram alir prosedur *swap number mutation*

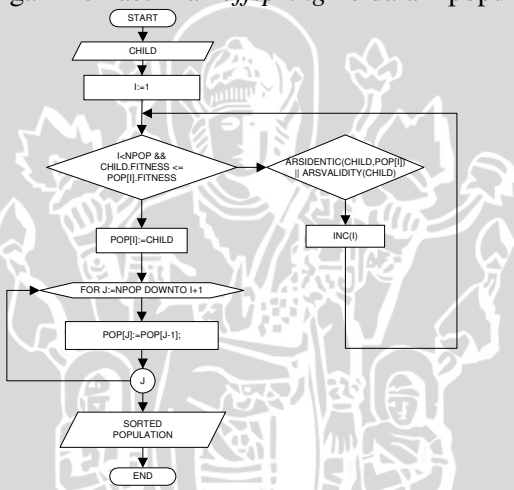


Gambar 3.21 Diagram alir prosedur *drop connection mutation*



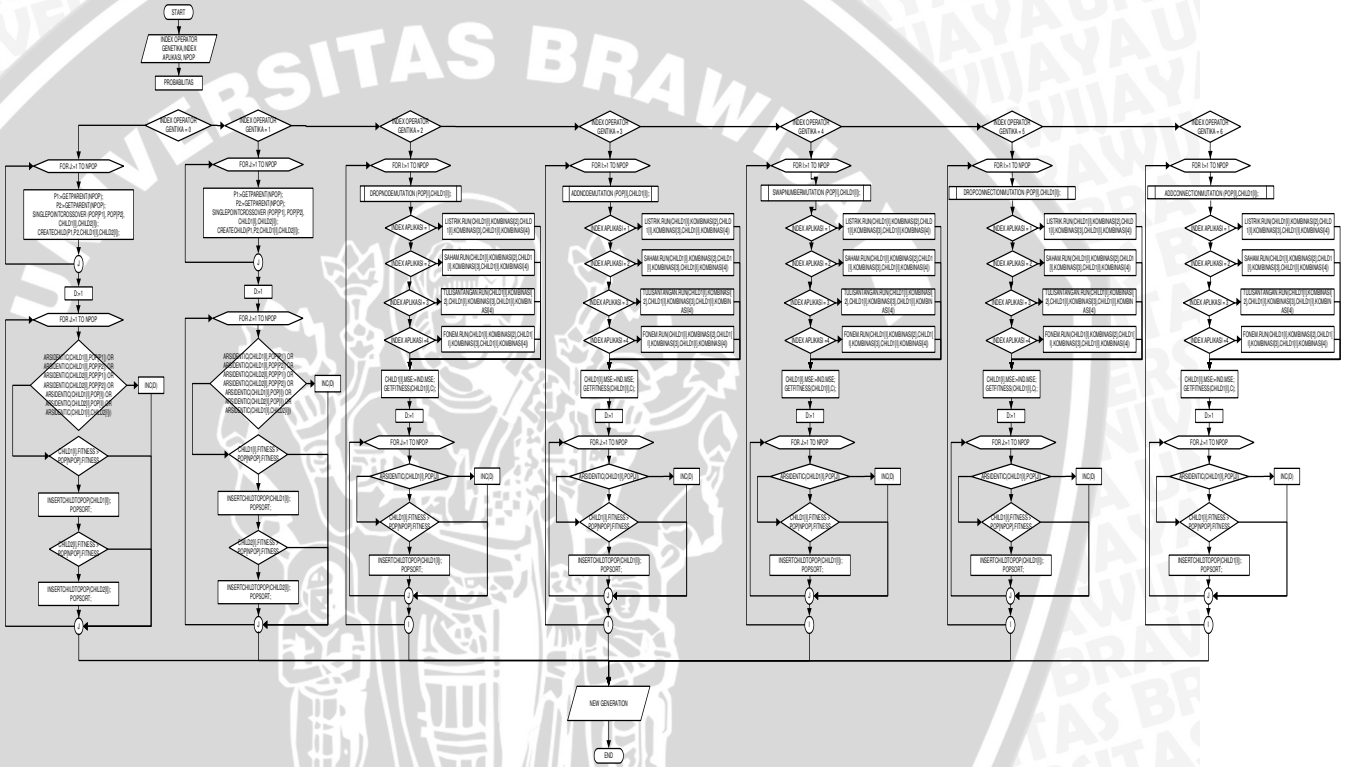
Gambar 3.22 Diagram alir prosedur *add connection mutation*

5. Kemudian dilakukan pengecekan apakah sudah memenuhi jumlah populasi yang diinginkan.
6. Jika sudah memenuhi populasi, kemudian lakukan *upgrade* populasi dengan memasukkan *offspring* ke dalam populasi.



Gambar 3.23 Diagram alir prosedur *insertchildtopop*

7. Selanjutnya dilakukan pengecekan apakah generasi yang dihasilkan sudah sesuai dengan yang dimaksud, jika belum maka langkah pembentukan generasi baru dilakukan lagi.

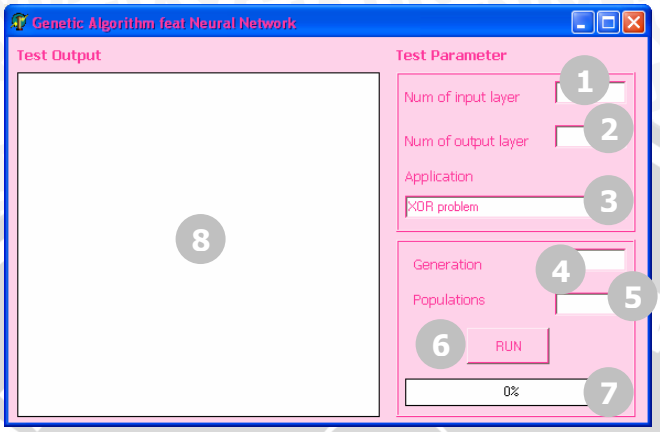


Gambar 3.24 Diagram alir prosedur *creategeneration*

8. Langkah yang terakhir adalah menampilkan hasil dari proses-proses di atas, yang diharapkan merupakan solusi optimal dari permasalahan yang dihadapi.

3.2.10 Perancangan Antar Muka

Rancangan program untuk mensimulasikan rancangan algoritma genetika dan jaringan syaraf tiruan dapat dilihat pada Gambar 3.13



Gambar 3.25 Rancangan *interface*

Keterangan Gambar 3.25

1. Tempat menginputkan jumlah neuron *input layer*.
2. Tempat menginputkan jumlah neuron *output layer*.
3. Pilihan aplikasi yang dioptimasi.
4. Tempat menginputkan jumlah generasi.
5. Tempat menginputkan jumlah populasi untuk tiap generasi.
6. Tombol RUN untuk memulai eksekusi dengan parameter yang telah dimasukkan.
7. Kotak waktu yang menunjukkan posisi proses pencarian solusi yang sedang berjalan.
8. Memo yang akan menunjukkan langkah demi langkah pencarian solusi.

3.2.11 Perancangan Uji Coba

Tujuan dilakukannya uji coba adalah untuk mengetahui pengaruh perlakuan operator genetika yang dapat memberikan solusi terbaik dalam menentukan arsitektur jaringan syaraf tiruan.

3.2.11.1 Skenario Pengujian

Pengujian dilakukan untuk mengetahui pengaruh perlakuan operator genetika dalam memberikan solusi arsitektur optimal untuk jaringan syaraf tiruan. Pengaruh perlakuan tersebut dimuat pada Tabel 3.2.

Tabel 3.2 Tabel pengujian pengaruh perlakuan

No.	Perlakuan	*Best Fitness Value*	*Structure of Network*

Pada Tabel 3.1, kolom perlakuan memuat keterangan cara pembentukan populasi, baik *crossover* maupun mutasi, seperti yang telah dijelaskan pada subbab 3.2.4 dan 3.2.5. Kolom *Best Fitness Value* memuat nilai *fitness* terbaik dari tiap cara pembentukan populasi yang dihitung menggunakan Persamaan 3.1. Kolom *Structure of Network* memuat arsitektur jaringan yang terbentuk ketika mencapai nilai *fitness* terbaik.

Pengujian juga dilakukan untuk mengetahui pengaruh penggunaan algoritma genetika untuk optimasi arsitektur jaringan syaraf tiruan dengan membandingkan nilai *error* yang dihasilkan antara jaringan syaraf tiruan dengan optimasi arsitektur dan tanpa optimasi arsitektur jaringan. Pengujian penggunaan algoritma genetika untuk optimasi arsitektur jaringan syaraf tiruan dimuat pada Tabel 3.3.

Tabel 3.3 Tabel pengujian pengaruh algoritma genetika

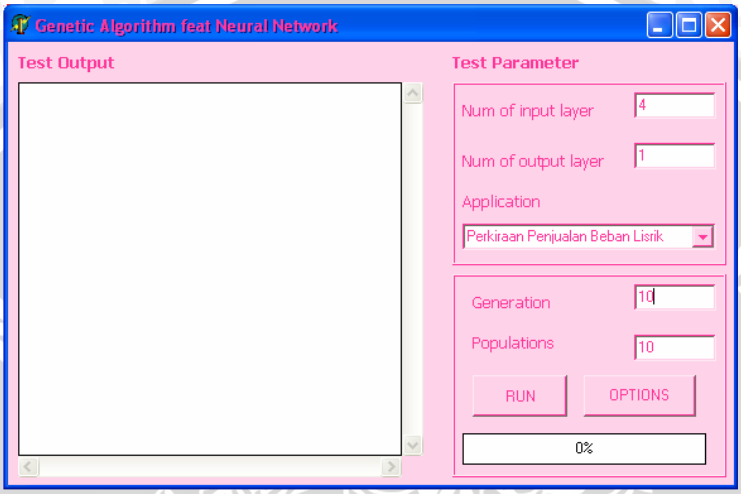
No.	Aplikasi	MSE dg GA	MSE tanpa GA

Pada Tabel 3.3, kolom aplikasi memuat nama aplikasi pengujian, seperti yang telah dijelaskan pada subbab 2.1.3 Kolom MSE dg GA memuat nilai MSE yang dihasilkan jaringan syaraf tiruan dengan optimasi arsitektur. Penghitungan nilai MSE menggunakan Persamaan 2.5. Kolom MSE tanpa GA memuat nilai MSE yang dihasilkan jaringan syaraf tiruan tanpa optimasi arsitektur. Nilai ini telah diperoleh pada penelitian sebelumnya dan telah disebutkan pada subbab 2.1.3.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Implementasi program dibuat dengan menggunakan Borland Delphi 6.0 dan Microsoft Access 2003 sebagai perangkat lunak untuk menyimpan database. Tampilan utama dari aplikasi pemilihan arsitektur jaringan syaraf tiruan multi-layer perceptron menggunakan algoritma genetika dapat dilihat pada Gambar 4.1.



Gambar 4.1 Tampilan utama aplikasi

4.1.1 *Input* Data

Pada tampilan utama, *user* diminta untuk memilih aplikasi jaringan syaraf tiruan yang akan dioptimasi arsistekturnya. Terdapat empat aplikasi yang tersedia, seperti yang telah dijelaskan pada subbab 2.13. Pemilihan aplikasi ini menentukan jumlah neuron pada lapisan masukan dan lapisan keluaran. Untuk aplikasi perkiraan penjualan beban listrik, jumlah neuron untuk lapisan masukan adalah 4 dan lapisan keluaran 1, aplikasi peramalan harga saham, jumlah neuron untuk lapisan masukan adalah 5 dan lapisan keluaran 1,

aplikasi pengenalan tulisan tangan, jumlah neuron untuk lapisan masukan adalah 400 dan lapisan keluaran 400 dan untuk aplikasi pengenalan fonem dalam Bahasa Indonesia, jumlah neuron untuk lapisan masukan adalah 189 dan lapisan keluaran 30.

User juga diminta untuk memasukkan beberapa parameter algoritma genetika, yaitu jumlah generasi, jumlah populasi. Dan klik tombol '*Options*' untuk mengatur probabilitas *crossover* dan mutasi. Perhitungan dilakukan ketika tombol '*RUN'* diklik.

4.1.2 Deskripsi Program

4.1.2.1 Inisialisasi Kromosom

Representasi kromosom seperti yang telah dijelaskan pada subbab 3.2.1 sebelumnya, dibangkitkan menggunakan data random. Kecuali untuk jumlah neuron pada lapisan masukan dan lapisan keluaran. Baris kode yang digunakan untuk membangkitkan jumlah neuron dan jumlah layer pada lapisan tersembunyi ditunjukkan pada Gambar 4.2.

```
procedure init (input,output : integer; var
ind:tarsitektur);
var i, numofhiddenlayer:integer;
                                                    …
randomize;
numofhiddenlayer := randomrange (1,4);
randomize;
for i:=1 to numofhiddenlayer do
layerhidden[i].numofneuron:=randomrange
(1,input+random(60));
…
```

Gambar 4.2 Inisialisasi kromosom

Selanjutnya nilai random yang diperoleh, disimpan dalam variabel ind.kombinasi[i]. Baris kode yang menunjukkan proses tersebut ditunjukkan pada Gambar 4.3.

```
…
//inisialisasi GA
ind.kombinasi[1] := input;
ind.kombinasi[5] := output;
if  numofhiddenlayer =1 then
begin
     ind.kombinasi[2]:= layerhidden[1].numofneuron;
     for i:=3 to 4 do
         ind.kombinasi[i]:= 0;
end
else if  numofhiddenlayer =2 then
begin
     for i:=2 to 3 do
        ind.kombinasi[i]:=layerhidden[i-1].numofneuron;
     ind.kombinasi[4]:= 0;
end
else
begin
   for i:=2 to 4 do
     ind.kombinasi[i]:= layerhidden[i-1].numofneuron;
end;
…
```

Gambar 4.3 Penyimpanan variabel ind.kombinasi[i]

4.1.2.2 Pembentukan populasi awal

Setelah terbentuk inisialisasi kromosom, dibentuk populasi awal dengan merepresentasikan kromosom tersebut menjadi arsitektur jaringan syaraf tiruan dan diimplementasikan terhadap aplikasi yang telah dipilih oleh *user*. Baris kode yang menunjukkan proses tersebut ditunjukkan pada Gambar 4.4.

```
                                procedure popcreate(c:integer);
…
if c=1 then
    listrik.run(pop[i].kombinasi[2],pop[i].kombinasi[3],pop[i]
    .kombinasi[4])
else if c=2 then
    saham.run(pop[i].kombinasi[2],pop[i].kombinasi[3],
    pop[i].kombinasi[4])
else if c=3 then
    tulisantangan.run(pop[i].kombinasi[2],pop[i].kombinasi[3],
    pop[i].kombinasi[4])
else if c=4 then
    fonem.run(pop[i].kombinasi[2],pop[i].kombinasi[3],
    pop[i].kombinasi[4]);
…
```

Gambar 4.4 Implementasi arsitektur jaringan

Untuk masing-masing aplikasi dilakukan proses *training* dengan parameter jaringan syaraf tiruan masing-masing. Parameter ini telah ditentukan pada pembuatan aplikasi sebelumnya. Parameter tersebut adalah *learning rate*, *max epoch*, *target error*. Untuk aplikasi perkiraan penjualan beban listrik terdapat parameter tambahan, yaitu nilai minimum dan nilai maksimum dari laju pembelajaran, dan nilai penaik dan penurun laju pembelajaran. Dan aplikasi pengenalan tulisan tangan memiliki parameter tambahan berupa nilai *threshold* untuk menentukan nilai batasan memisahkan obyek dengan *background.*

Proses *training* diawali dengan inisialisasi bobot. Masing-masing aplikasi memiliki prosedur inisialisasi bobot yang berbeda. Gambar 4.5 dan Gambar 4.6 menunjukkan prosedur inisialisasi bobot pada aplikasi pengenalan tulisan tangan.

```
procedure InitWeights;
var   i, j, k: Integer;
begin
  Randomize;
  //--- Inisialisasi bobot hidden layer
  for j := 0 to FNHidden1Neuron do
    for i := 0 to FNInputNeuron do
      begin
        FHiddenLayer1Weights[j, i] := Random - 0.5;
      end;

  if ((FNHidden2Neuron>0) and (FNHidden3Neuron=0)) then
  begin
     for j := 0 to FNHidden2Neuron do
      for i := 0 to FNHidden1Neuron do
      begin
        FHiddenLayer2Weights[j, i] := Random - 0.5;
      end;

      for k := 1 to FNOutputNeuron do
       for j := 0 to FNHidden2Neuron do
      FOutputLayerWeights[k, j] := Random - 0.5;
  end
  else  if FNHidden3Neuron>0 then
  begin
     for j := 0 to FNHidden2Neuron do
      for i := 0 to FNHidden1Neuron do
      begin
        FHiddenLayer2Weights[j, i] := Random - 0.5;
      end;
```

Gambar 4.5 Inisialisasi bobot bagian I

```
…
       for j := 0 to FNHidden3Neuron do
      for i := 0 to FNHidden2Neuron do
      begin
        FHiddenLayer3Weights[j, i] := Random - 0.5;
      end;

      for k := 1 to FNOutputNeuron do
       for j := 0 to FNHidden3Neuron do
      FOutputLayerWeights[k, j] := Random - 0.5;
  end
  else
  begin
  //--- Inisialisasi bobot output layer
  for k := 1 to FNOutputNeuron do
    for j := 0 to FNHidden1Neuron do
      FOutputLayerWeights[k, j] := Random - 0.5;
  end;
end;
```

Gambar 4.6 Inisialisasi bobot bagian II

Dilanjutkan dengan proses normalisasi data yang bertujuan untuk mempercepat proses pelatihan dan meminimumkan *error*. Masing-masing aplikasi memiliki prosedur normalisasi data yang berbeda. Gambar 4.7 menunjukkan prosedur normalisasi data pada aplikasi pengenalan fonem dalam Bahasa Indonesia.

```
procedure Normalize(inputx:tinput);
var i,iterInput  : integer;
begin
      for i:=1 to N_Input do
      inputx.input[i] := (0.8 * inputx.input[i])+0.1;
end;
```

Gambar 4.7 Normalisasi data

Kemudian dilakukan proses *training*, yang di dalamnya terdapat beberapa prosedur yang dijalankan, yaitu *feedforward*, penghitungan *error, backprop,* penghitungan MSE (*Mean Square Error*) dan perubahan bobot. Baris kode yang menunjukkan prosedur feedforward untuk aplikasi peramalan harga saham dapat dilihat pada Gambar 4.8.

```
procedure
feedforward(inputna:arr;hidden1,hidden2,hidden3:integer);
var i,j :integer;
begin
   for j:=1 to hidden1 do  z[j]:=0;
   y_in:=wawal1[1]*zawal1[1];
   for j:=1 to hidden1 do
   begin
     z_in:=vawal1[1,j]*inputna[1];
     for i:=1 to n_input do
      z_in:=z_in+(vawal1[i,j]*inputna[i]);
     zawal1[j]:=z_in;
     z[j]:=1.7159*tanh(0.66666667*zawal1[j]);
     y_in:=y_in+(z[j]*wawal1[j]);
   end;
   if ((hidden2>0) and (hidden3=0)) then
   begin
      for j:=1 to hidden2 do
      begin
      z_in:=vawal2[1,j]*z[1];
        for i:=1 to hidden1 do
            z_in:=z_in+(vawal1[i,j]*z[i]);
        zawal2[j]:=z_in;
        z1[j]:=1.7159*tanh(0.66666667*zawal2[j]);
        y_in:=y_in+(z1[j]*wawal1[j]);
      end;
      y:=1.7159*tanh(0.66666667*y_in);
   end
   else if hidden3>0 then
   begin
      for j:=1 to hidden2 do
      begin
        z_in:=vawal2[1,j]*z[1];
        for i:=1 to hidden1 do
            z_in:=z_in+(vawal1[i,j]*z[i]);
        zawal2[j]:=z_in;
        z1[j]:=1.7159*tanh(0.66666667*zawal2[j]);
        y_in:=y_in+(z1[j]*wawal1[j]);
      end;
      for j:=1 to hidden3 do
      begin
        z_in:=vawal3[1,j]*z1[1];
        for i:=1 to hidden2 do
            z_in:=z_in+(vawal3[i,j]*z1[i]);
        zawal3[j]:=z_in;
        z2[j]:=1.7159*tanh(0.66666667*zawal3[j]);
        y_in:=y_in+(z2[j]*wawal1[j]);
      end;
       y:=1.7159*tanh(0.66666667*y_in);
   end
   else
   y:=1.7159*tanh(0.66666667*y_in); end;
```

Gambar 4.8 Prosedur *feedforward*

Baris kode yang menunjukkan prosedur penghitungan error untuk aplikasi perkiraan penjualan beban listrik dapat dilihat pada Gambar 4.9.

```
procedure HitungError(target : double);
var selisih :double;
begin
  selisih := target - output;
  error := selisih / 2.0;
  Mse := Mse + (0.5*power(selisih,2));
  deltaoutput:= error*dsigmoid (output);
  inc(TotalMse);
end;
```

Gambar 4.9 Prosedur Hitung *Error*

Baris kode yang menunjukkan prosedur *backprop* untuk aplikasi pengenalan tulisan tangan dapat dilihat pada Gambar 4.10.

```
//--- Backpropagation of error
//- Hitung informasi error bobot output layer
for k := 1 to FNOutputNeuron do
begin
   Error := FTrainingSet[l].TargetPattern[k] -
            FOutputLayer[k];
   if Abs(Error) >= FErrorThreshold then
            FNNeuronError := FNNeuronError + 1;
   FTrainingError := FTrainingError + (Error * Error);
   OutputErrors[k]:=Error*
                    BipolarSigmoidDerivation(FOutputLayer[k
                    ]);
   if FNHidden3Neuron>0 then
   begin
      for j := 0 to FNHidden3Neuron do
          OutputWeightsCorrection[k, j] := FLearningRate *
          OutputErrors[k] * FHiddenLayer3[j];
   end
    else if((FNHidden3Neuron=0)and(FNHidden2Neuron>0))then
   begin
      for j := 0 to FNHidden2Neuron do
        OutputWeightsCorrection[k, j] := FLearningRate *
          OutputErrors[k] * FHiddenLayer2[j];
   end
   else
   begin
      for j := 0 to FNHidden1Neuron do
       OutputWeightsCorrection[k, j] := FLearningRate *
       OutputErrors[k] * FHiddenLayer1[j];
   end;
end;
```

Gambar 4.10 Prosedur *backprop*

Baris kode yang menunjukkan prosedur perubahan bobot untuk aplikasi peramalan harga saham dapat dilihat pada Gambar 4.11 dan Gambar 4.12.

```
procedure updatebobot( hidden1,hidden2,hidden3:integer);
var i,j:integer;
begin
    for j:=1 to hidden1 do
    begin
        for i:=1 to n_input do
         begin
            vbaru1[i,j]:=vawal1[i,j]+SukuPerubahanBobotInput[
                    i,j]+(momentum*(vawal1[i,j]-
                    vlama1[i,j]));
          vlama1[i,j]:=vawal1[i,j];
          vawal1[i,j]:=vbaru1[i,j];
        end;
    end;
     if ((hidden2>0) and (hidden3=0)) then
    begin
         for j:=1 to hidden2 do
         begin
          for i:=1 to hidden1 do
          begin
           vbaru2[i,j]:=vawal2[i,j]+SukuPerubahanBobotInput1
                    [i,j]+(momentum*(vawal2[i,j]-
                    vlama2[i,j]));
          vlama2[i,j]:=vawal2[i,j];
          vawal2[i,j]:=vbaru2[i,j];
          end;
        end;

        for j:=1 to hidden2 do
         begin
         wbaru[j]:=wawal1[j]+SukuPerubahanBobotHL1[j]+(momen
                    tum*(wawal1[j]-wlama[j]));
        wlama[j]:=wawal1[j];
        wawal1[j]:=wbaru[j];
         end;
    end
    else if hidden3>0 then
    begin
        for j:=1 to hidden2 do
         begin
          for i:=1 to hidden1 do
          begin
           vbaru2[i,j]:=vawal2[i,j]+SukuPerubahanBobotInput1
                    [i,j]+(momentum*(vawal2[i,j]-
                    vlama2[i,j]));
          vlama2[i,j]:=vawal2[i,j];
           vawal2[i,j]:=vbaru2[i,j];
      end;
```

Gambar 4.11 Prosedur *update* bobot bagian I

```
…
          end;

          for j:=1 to hidden3 do
          begin
           for i:=1 to hidden2 do
            begin
             vbaru3[i,j]:=vawal3[i,j]+SukuPerubahanBobotInput2
                          [i,j]+(momentum*(vawal3[i,j]-
                          vlama3[i,j]));
           vlama3[i,j]:=vawal3[i,j];
           vawal3[i,j]:=vbaru3[i,j];
           end;
         end;

          for j:=1 to hidden3 do
          begin
             wbaru[j]:=wawal1[j]+SukuPerubahanBobotHL2[j]+(mom
                          entum*(wawal1[j]-wlama[j]));
           wlama[j]:=wawal1[j];
           wawal1[j]:=wbaru[j];
          end;
     end
     else
     begin
     for j:=1 to hidden1 do
     begin
       wbaru[j]:=wawal1[j]+SukuPerubahanBobotHL[j]+(momentum*
                 (wawal1[j]-wlama[j]));
      wlama[j]:=wawal1[j];
      wawal1[j]:=wbaru[j];
     end;
     end;
end;
```

Gambar 4.12 Prosedur *update* bobot bagian II

Sehingga secara lengkap, proses *training* untuk aplikasi pengenalan fonem dalam Bahasa Indonesia ditunjukkan pada baris kode pada Gambar 4.13 dan 4.14.

```
procedure Learning(_input: tinput;hidden1,hidden2,hidden3 :
integer);
var  dEtaMin,dEtaDecay  : double ;
     i,step,kata,start,jumhuruf, index, sample: integer;
     inputx : tinput;
begin
     iteration :=0;
     m_vPatternKata[m_iPrevId].jumhuruf := m_iJumHuruf;
     dEtaDecay:=0.7941833;
     dEtaMin:=0.00005;
```

Gambar 4.13 Prosedur *training* bagian I

```
…
   Normalize(_input);

   for i:=1 to m_ijumkata do
      m_vorder[i]:=i;

   m_dMinMSE := 99;
   m_dMinErrAbs := 99;

   step :=0;
   sample:=1;
   while ((iteration < MaxEpoh) and (m_dMSE>TargetError)) do
   begin
     if m_bstop = true then
     begin
       m_vWeightHidden2Output := m_vBestWeightHidden2Output;
       m_vWeightInput2Hidden := m_vBestWeightInput2Hidden;
     end;

     if((step mod 120000=0) and (step<>0)) then
     begin
         m_dEta:=dEtaDecay;
         if(m_dEta<dEtaMin) then
            m_dEta := dEtaMin;
     end;
     if (sample = m_iJumKata) then
     begin
         TotalError(hidden1,hidden2,hidden3 );
         sample := 0;
          inc(Iteration);
         if m_dMSE>TargetError then
           m_dMinMSE := m_dMSE;
          Randomizex(hidden1,hidden2,hidden3 );
     end;
     kata := m_vOrder[sample];
     start := m_vPatternKata[kata].start;
     jumhuruf := m_vPatternKata[kata].jumhuruf;

     ResetDeltaWeight(hidden1,hidden2,hidden3);

     for index:=start to(start+jumhuruf) do
     begin
         FeedForward(_input,hidden1,hidden2,hidden3 );
          CalculateError(_input,hidden1,hidden2,hidden3 );
         inc(step);
        UpdateWeight(hidden1,hidden2,hidden3);
       end;
       inc(sample);
       inc(iteration);
     end;
     genetic.ind.Mse :=m_dMSE;
end;
```

Gambar 4.14 Prosedur *training* bagian II

4.1.2.3 Operasi *Crossover*

Operasi *crossover* dilakukan dengan dua metode, seperti telah dijelaskan pada subbab 3.2.5. Kedua metode dikerjakan untuk mengetahui pengaruh perlakuan terhadap nilai MSE yang dihasilkan. Baris kode yang menunjukkan operasi *singlepointcrossover* dapat dilihat pada Gambar 4.15.

```
Procedure                          singlepointcrossover(const
parent1,parent2:tarsitektur; var child1,child2:tarsitektur);
var i,j,cut :integer;
begin
 randomize;  cut :=randomrange (2,4);
 for i:=1 to cut do
 begin
    child1.kombinasi[i]:=parent2.kombinasi[i];
    child2.kombinasi[i]:=parent1.kombinasi[i];
 end;
 for i:=cut+1 to 5 do
 begin
    child1.kombinasi[i]:=parent1.kombinasi[i];
    child2.kombinasi[i]:=parent2.kombinasi[i];
 end;
 for i:=1 to cut do
  for j:=1 to 5 do
  begin
      if j<= cut then
      begin
      child1.connect[i,j]:= parent2.connect[i,j];
      child2.connect[i,j]:=parent1.connect[i,j];
      end
      else
      begin
       child1.connect[i,j]:= parent1.connect[i,j];
      child2.connect[i,j]:=parent2.connect[i,j];
      end;
  end;
  for i:=cut+1 to 5 do
  for j:=1 to 5 do
  begin
      if j<= cut then
      begin
      child1.connect[i,j]:= parent1.connect[i,j];
      child2.connect[i,j]:=parent2.connect[i,j];
      end
      else
      begin
       child1.connect[i,j]:= parent2.connect[i,j];
      child2.connect[i,j]:=parent1.connect[i,j];
      end;
  end; end;
```

Gambar 4.15 Prosedur *singlepointcrossover*

Dan baris kode yang menunjukkan operasi *twopointcrossover* dapat dilihat pada Gambar 4.16 dan Gambar 4.17.

```
procedure                                twopointcrossover(const
     parent1,parent2:tarsitektur;                          var
     child1,child2:tarsitektur);
var i,j,cut1,cut2 :integer;
begin
 randomize;
 cut1:=randomrange(1,3);
 cut2:=randomrange(cut1,6);
 while cut1=cut2 do
 begin
     randomize;
     cut2:=randomrange(cut1,6);
 end;
 for i:=1 to cut1 do
 begin
    child1.kombinasi[i]:=parent2.kombinasi[i];
    child2.kombinasi[i]:=parent1.kombinasi[i];
 end;

 for i:=cut1+1 to cut2 do
 begin
    child1.kombinasi[i]:=parent1.kombinasi[i];
    child2.kombinasi[i]:=parent2.kombinasi[i];
 end;

 for i:=cut2+1 to 5 do
 begin
    child1.kombinasi[i]:=parent2.kombinasi[i];
    child2.kombinasi[i]:=parent1.kombinasi[i];
 end;

 for i:=1 to cut1 do
  for j:=1 to 5 do
  begin
      if j<= cut1 then
      begin
      child1.connect[i,j]:= parent2.connect[i,j];
      child2.connect[i,j]:=parent1.connect[i,j];
      end
      else if ((cut1<j) and (j<cut2)) then
      begin
          child1.connect[i,j]:= parent1.connect[i,j];
          child2.connect[i,j]:=parent2.connect[i,j];
      end
      else
      begin
          child1.connect[i,j]:= parent2.connect[i,j];
          child2.connect[i,j]:=parent1.connect[i,j];
      end;
  end;
```

Gambar 4.16 Prosedur *twopointcrossover* bagian I

```
for i:=cut2+1 to 5 do
  for j:=1 to 5 do
  begin
      if j<= cut1 then
      begin
      child1.connect[i,j]:= parent2.connect[i,j];
      child2.connect[i,j]:=parent1.connect[i,j];
      end
      else  if ((cut1<j) and (j<cut2)) then
      begin
          child1.connect[i,j]:= parent1.connect[i,j];
          child2.connect[i,j]:=parent2.connect[i,j];
      end
      else
      begin
         child1.connect[i,j]:= parent2.connect[i,j];
          child2.connect[i,j]:=parent1.connect[i,j];
      end;  end;
    end;
```

Gambar 4.17 Prosedur *twopointcrossover* bagian II

4.1.2.4 Operasi Mutasi

Operasi mutasi dilakukan dengan lima metode, seperti telah dijelaskan pada subbab 3.2.6. Kelima metode dikerjakan untuk mengetahui pengaruh perlakuan terhadap nilai MSE yang dihasilkan. Baris kode yang menunjukkan operasi *dropnodemutation* dapat dilihat pada Gambar 4.18.

```
procedure dropnodemutation (var ind :tarsitektur);
var i,mut,j:integer;   temp:integer;
begin
  randomize;
  mut:=randomrange(2,4);
  for i:=mut to 3 do
  begin
      if ind.kombinasi[i+1]<>0 then
      begin
      ind.kombinasi[i]:=ind.kombinasi[i+1];
      ind.kombinasi[i+1]:=0;
      end;
  end;

  for j:=1 to 5 do
  begin
   i:=4;
   ind.connect[i,j]:=0;
   ind.connect[j,i]:=0;
  end;
end;
```

Gambar 4.18 Prosedur *dropnodemutation*

Baris kode untuk operasi *addnodemutation* dapat dilihat pada Gambar 4.19.

```
procedure addnodemutation (var ind :tarsitektur);
var i,j,mut:integer;
begin
  randomize;
  mut:=randomrange(3,5);  child:=ind;
  if child.kombinasi[mut]=0 then
  begin
      if mut=4 then
      begin
       if child.kombinasi[mut-1]=0  then  exit
       else
       begin
          randomize;
         child.kombinasi[mut]:=randomrange
         (1,child.kombinasi[1]+ random(60));
      child.connect[mut-1,mut]:=1;
       end
      end else
      begin
      randomize;
     child.kombinasi[mut]:=randomrange  (1,child.kombinasi[1]+
     random(60));
      child.connect[mut-1,mut]:=1;
      end;
  end;
end;
```

Gambar 4.19 Prosedur *addnodemutation*

Baris kode untuk operasi *swapnumbermutation* dapat dilihat pada Gambar 4.20.

```
procedure swapnumbermutation (var ind :tarsitektur);
var i,j,mut,ends:integer;
begin
  for i:=2 to 4 do
  begin
      if ind.kombinasi[i]<>0 then
            ends:=i;
  end;
  randomize;   mut:=randomrange(2,ends);
  ind.kombinasi[mut]:=randomrange          (1,ind.kombinasi[1]+
random(60));
  for j:=1 to 5 do
  begin
        i:=mut;  ind.connect[i,j]:=1;
        ind.connect[j,i]:=1;
  end;
end;
```

Gambar 4.20 Prosedur *swapnumbermutation*

Baris kode untuk operasi *dropconnectionmutation* dapat dilihat pada Gambar 4.21.

```
procedure dropconnectionmutation (var ind :tarsitektur);
var i,mut,j:integer;   temp:integer;
begin
  for i:=3 to 4 do
  for j:=1 to 5 do
  begin
      if ind.connect[i,j]<>0 then
      begin
          ind.connect[i,j]:=0;
          ind.kombinasi[i]:=0;
          exit;
      end;
  end;
end;
```

Gambar 4.21 Prosedur *dropconnectionmutation*

Baris kode untuk operasi *addconnectionmutation* dapat dilihat pada Gambar 4.22.

```
procedure addconnectionmutation (var ind :tarsitektur);
var i,mut,j:integer;   kosong:integer;
begin
  kosong:=0;
  for i:=3 to 4 do
  for j:=1 to 5 do
  begin
      if ind.connect[i,j]=0 then
      begin
          kosong:=kosong+1;
      end;
  end;

  if kosong=8 then
  begin
     ind.connect[3,2]:=1;
     ind.kombinasi[3]:=randomrange        (1,ind.kombinasi[1]+
random(60));
  end
  else if kosong=7 then
  begin
     ind.connect[4,3]:=1;
     ind.kombinasi[4]:=randomrange        (1,ind.kombinasi[1]+
random(60));
  end;
end;
```

Gambar 4.22 Prosedur *addconnectionmutation*

4.1.2.5 Penghitungan nilai *fitness*

Nilai fitness dihitung menggunakan Persamaan 3.1. Baris kode yang menunjukkan proses tersebut dapat dilihat pada Gambar 4.23.

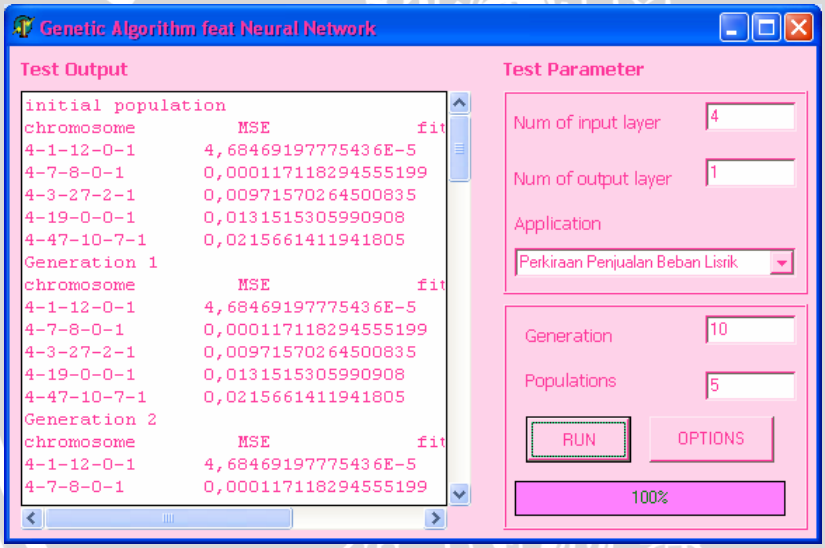
```
procedure getfitness(var ind :tarsitektur; a:integer);
begin
     case a of
     1 : msebest:= 0.00002832;
     2 : msebest:= 0.0131;
     3 : msebest:= 0.0000598788;
     4 : msebest:= 0.0054;
     end;

     ind.fitness:=(msebest / ind.MSE );
end;
```

Gambar 4.23 Prosedur hitung nilai *fitness*

4.1.3 Penerapan Aplikasi

Aplikasi diterapkan dengan memasukkan data sesuai dengan keinginan *user*. Misal aplikasi yang dipilih adalah aplikasi peramalan penjualan beban listrik, dengan generasi sebanyak 10 dan tiap generasi menghasilkan 5 populasi kromosom. Maka tampilan program setelah tombol '*RUN*' diklik dapat dilihat pada Gambar 4.24.



Gambar 4.24 Tampilan antarmuka

4.1.4 Analisa Hasil

Berdasarkan skenario uji coba yang telah dijelaskan pada subbab 3.2.11.1, maka dilakukan beberapa uji coba, dengan perlakuan yang berbeda pada tiap aplikasi. Dan analisa hasil untuk tiap perlakuan dan tiap aplikasi akan dijelaskan pada subbab 4.1.4.1 sampai subbab 4.1.4.8.

4.1.4.1 Hasil Uji Coba Perlakuan *Single Point Crossover*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Uji coba perlakuan *Single Point Crossover*

No.	Aplikasi	Uji Coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	0,6045221358091370	4-1-12-1
		2	0,0134396728791792	4-33-1
		3	0,8501112962534000	4-9-15-1
		4	1,0848542821792300	4-15-14-1
		5	1,2703844966743100	4-7-6-1
2.	Peramalan harga saham	1	9,7905065669348	5-6-1
		2	44,6956199583068000	5-10-26-40-1
		3	59,7901259090344	5-33-1
		4	39,8038050600836	5-22-1
		5	17,0754036776673	5-5-1
3.	Pengenalan tulisan tangan	1	0,0701976114229159	400-248-400
		2	0,332969894997789	400-181-414-400
		3	0,0766266689796065	400-141-400
		4	0,434621203799595	400-60-143-400
		5	0,103418679391136	400-151-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	1,9263943029587300	189-188-30
		2	0,58710598077514	189-173-30
		3	0,613237063083271	189-204-30
		4	0,583597187033269	189-210-30
		5	0,450825577686512	189-82-43-30

4.1.4.2 Hasil Uji Coba Perlakuan *Two Point Crossover*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Uji coba perlakuan *Two Point Crossover*

No.	Aplikasi	Uji Coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	0,7829614260610950	4-9-5-1
		2	0,796614461614341	4-37-7-1
		3	1,04296629144523	4-7-6-1
		4	2,4461691180042900	4-10-16-1
		5	0,630688713693684	4-6-5-1
2.	Peramalan harga saham	1	286,1398490164410000	5-51-1
		2	23,2859045676926	5-37-1
		3	11,796217597873	5-6-37-1
		4	30,4007196256979	5-9-1
		5	368,09887786997	5-10-1
3.	Pengenalan tulisan tangan	1	0,140409557137469	400-355-33-146-400
		2	0,493889677353628	400-67-161-400
		3	0,508479679639482	400-283-230-400
		4	0,58994445767616	400-169-400
		5	0,443488075897162	400-394-61-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	0,68686971166471	189-128-30
		2	0,577013349863847	189-172-30
		3	0,451277852653644	189-196-58-30
		4	0,686346756395318	189-184-30
		5	0,666225925641702	189-200-30

4.1.4.3 Hasil Uji Coba Perlakuan *Drop Node Mutation*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Uji coba perlakuan *Drop Node Mutation*

No.	Aplikasi	Uji Coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	1,3334003446612300	4-6-4-1
		2	1,04965498222337	4-21-10-1
		3	0,559177745426984	4-32-11-1
		4	1,59479885673797	4-21-16-1
		5	2,05304026862605	4-23-10-1
2.	Peramalan harga saham	1	223,296349780274000	5-14-1
		2	48,824601913808	5-4-1
		3	99,4411926355833	5-7-1
		4	101,142122913029	5-18-1
		5	85,9927169818792	5-41-1
3.	Pengenalan tulisan tangan	1	0,0817766437389425	400-135-92-400
		2	2,3328047191664	400-412-216-400
		3	4,17224171167411	400-288-382-400
		4	4,09422168632462	400-384-277-400
		5	3,25020857997501	400-10-383-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	0,68038866027871	189-90-30
		2	0,591780625455505	189-158-30
		3	0,617942096637932	189-16-30
		4	0,664251660963458	189-161-30
		5	0,688768192698227	189-79-30

4.1.4.4 Hasil Uji Coba Perlakuan *Add Node Mutation*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Uji coba perlakuan *Add Node Mutation*

No.	Aplikasi	Uji Coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	0,3759135169010320	4-9-11-1
		2	0,78826584547905	4-14-16-1
		3	0,68354571657039	4-8-15-1
		4	0,3098841474443871	4-10-31-1
		5	0,198607411927801	4-29-18-1
2.	Peramalan harga saham	1	18,075054094665400	5-3-19-14-1
		2	18,8173041330967	5-13-26-19-1
		3	152,189788181476	5-12-40-45-1
		4	7,71216436557047	5-9-25-27-1
		5	35,9295054024499	5-4-8-27-1
3.	Pengenalan tulisan tangan	1	0,1040281270181020	400-302-153-301-400
		2	0,141292698151746	400-62-145-270-400
		3	0,165474224680328	400-76-400
		4	1,56932033346765	400-354-338-400
		5	2,14485630500229	400-198-281-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	0,450862726100222	189-11-50-30
		2	0,565941658925574	189-127-30
		3	0,684926764863543	189-180-30
		4	0,580344446060797	189-152-30
		5	0,663466064091664	189-170-30

4.1.4.5 Hasil Uji Coba Perlakuan *Swap Number Mutation*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Uji coba perlakuan *Swap Number Mutation*

No.	Aplikasi	Uji Coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	1,1932105428213200	4-10-27-1
		2	0,824839893534064	4-5-20-1
		3	2,75388915271145	4-43-17-1
		4	1,1708973850687	4-45-16-1
		5	0,124706410606825	4-1-12-3-1
2.	Peramalan harga saham	1	70,92485775887220	5-8-7-31-1
		2	5,33489354262497	5-2-40-1
		3	225,055176803738	5-15-1
		4	56,8061853749702	5-12-1
		5	399,626097906354	5-17-1
3.	Pengenalan tulisan tangan	1	6,4473906213471	400-385-358-400
		2	0,671010079076643	400-202-182-271-400
		3	0,0662962864178212	400-386-14-400
		4	0,424518942484567	400-191-168-400
		5	1,11293973245773	400-99-325-234-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	0,688471288650543	189-221-30
		2	0,700949248106285	189-118-30
		3	0,675170994222726	189-213-30
		4	0,450223421615999	189-71-85-10-30
		5	0,69737392274025	189-182-30

4.1.4.6 Hasil Uji Coba Perlakuan *Drop Connection Mutation*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Uji coba perlakuan *Drop Connection Mutation*

No.	Aplikasi	Uji Coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	0,8081606172717180	4-14-6-1
		2	0,0161170040000202	4-11-1
		3	0,381833736174524	4-11-18-1
		4	0,311643733524975	4-2-14-1
		5	0,123266287172372	4-25-20-1
2.	Peramalan harga saham	1	227,7613166089000000	5-10-1
		2	59,7187719029728	5-4-1
		3	16,3571554490504	5-4-1
		4	64,1664104855316	5-12-1
		5	36,4919498701637	5-4-1-50-1
3.	Pengenalan tulisan tangan	1	0,0765229382612725	400-390-31-144-400
		2	0,0740082577089182	400-112-400
		3	0,294926740270569	400-138-97-400
		4	0,237735198859907	400-85-128-400
		5	0,44602622799614	400-333-143-127-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	0,69737392274025	189-182-30
		2	0,698553017364537	189-133-30
		3	0,557236413904838	189-88-30
		4	0,604267572770167	189-156-30
		5	0,685493319228129	189-88-30

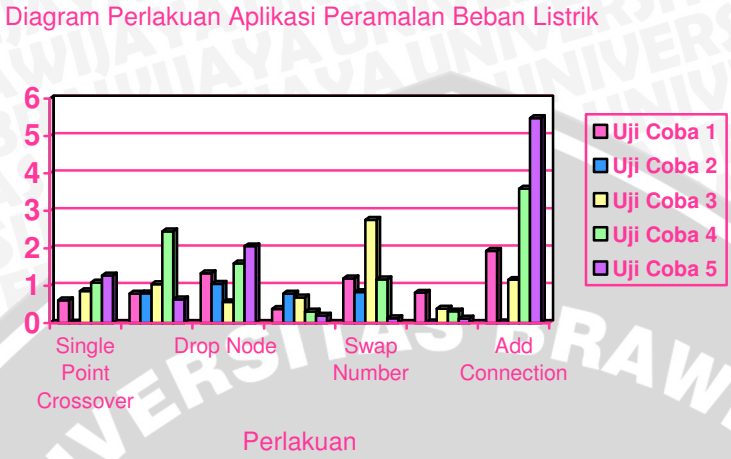
4.1.4.7 Hasil Uji Coba Perlakuan *Add Connection Mutation*

Uji coba dilakukan sebanyak 5 kali pada tiap aplikasi, dengan parameter genetika, jumlah generasi 10 dan populasi sebanyak 5. Hasil uji coba dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil Uji coba perlakuan *Add Connection Mutation*

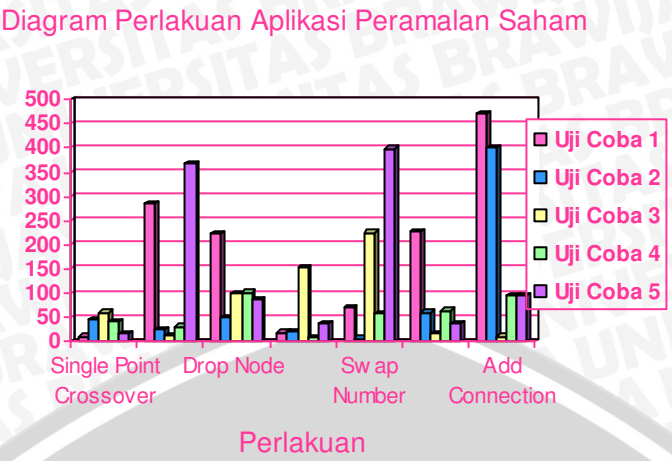
No.	Aplikasi	Uji coba ke-	Nilai *fitness*	Arsitektur Jaringan
1.	Perkiraan penjualan beban listrik	1	1,9318976557167400	4-18-3-1
		2	0,0176623692207313	4-25-8-22-1
		3	1,157210856049	4-5-14-1
		4	3,59146444869084	4-9-17-1
		5	5,47776600089546	4-1-7-19-1
2.	Peramalan harga saham	1	470,5794030080350000	5-22-1
		2	401,966890939856	5-16-1
		3	9,74211287623378	5-1-1
		4	93,870653566001	5-10-1
		5	94,7091997415435	5-14-1
3.	Pengenalan tulisan tangan	1	0,0725164334063642	400-242-163-400
		2	0,179797966385104	400-307-281-312-400
		3	0,179416946916577	400-291-126-232-400
		4	1,92235206253636	400-237-265-400
		5	0,104319595726917	400-5-345-107-400
4.	Pengenalan fonem dalam Bahasa Indonesia	1	0,587476905358354	189-5-30
		2	0,687439824379812	189-89-30
		3	0,665200994865256	189-109-30
		4	0,450325661313863	189-62-143-24-30
		5	0,673714836752985	189-51-30

Berdasarkan uji coba yang telah dilakukan, tiap perlakuan yang dilakukan pada tiap aplikasi menghasilkan nilai *fitness* yang berbeda. Hal ini sangat tergantung pada proses inisialisasi bobot awal yang diambil dari nilai acak (random)**.** Perilaku masing-masing aplikasi pada tiap perlakuan dapat dilihat pada Gambar 4.25, 4.26, 4.27 dan 4.28.



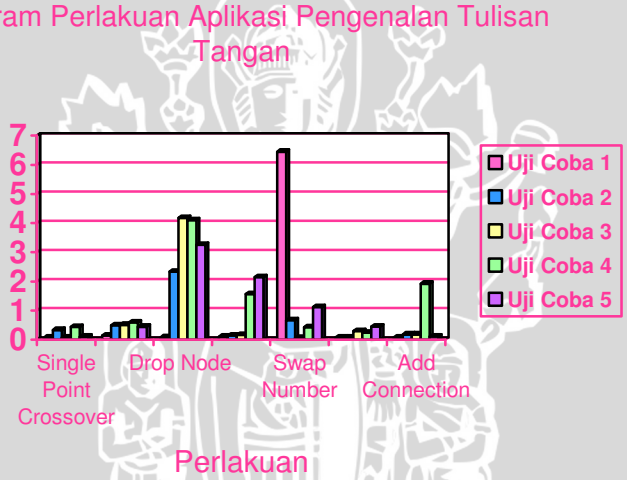
Gambar 4.25 Diagram Perlakuan Aplikasi Peramalan Beban Listrik

Untuk aplikasi perkiraan penjualan beban listrik, nilai *fitness* terbaik diperoleh pada perlakuan *Add Connection Mutation* dengan arsitektur jaringan 4-1-7-19-1.



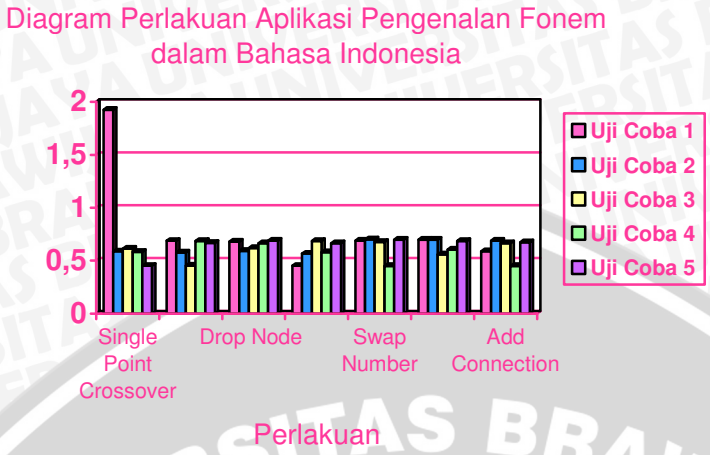
Gambar 4.26 Diagram Perlakuan Aplikasi Peramalan Saham

Untuk aplikasi peramalan harga saham, nilai *fitness* terbaik diperoleh pada perlakuan *Add Connection Mutation* dengan arsitektur jaringan 5-22-1.



Gambar 4.27 Diagram Perlakuan Aplikasi Pengenalan Tulisan Tangan

Untuk aplikasi pengenalan tulisan tangan, nilai *fitness* terbaik diperoleh pada perlakuan *Swap Number Mutation* dengan arsitektur jaringan 400-385-358-400.



Gambar 4.28 Diagram Perlakuan Aplikasi Pengenalan Fonem dalam Bahasa Indonesia

Untuk aplikasi pengenalan fonem dalam Bahasa Indonesia, nilai *fitness* terbaik diperoleh pada perlakuan *Single Point Crossover* dengan arsitektur jaringan 189-188-30.

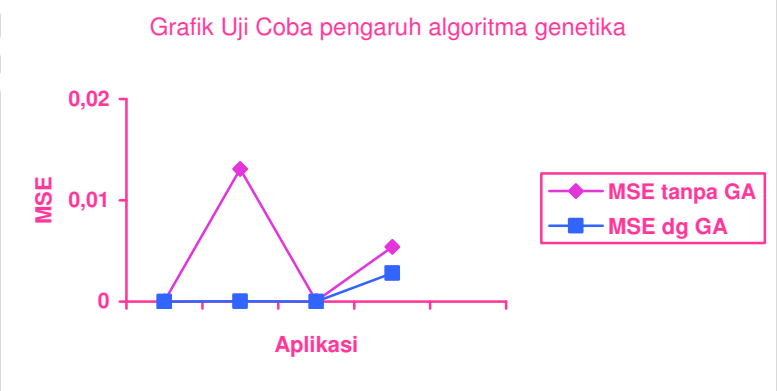
4.1.4.8 Hasil Uji Coba Pengaruh Algoritma Genetika

Hasil perbandingan nilai MSE yang dihasilkan antara jaringan syaraf tiruan tanpa optimasi arsitektur jaringan dan dengan optimasi arsitektur seperti yang dilakukan pada uji coba sebelumnya, dapat dilihat pada Tabel 4.8 dan disajikan dalam bentuk grafik pada Gambar 4.29.

Tabel 4.8 Hasil uji coba pengaruh algoritma genetika

No.	Aplikasi	MSE tanpa GA	MSE dg GA
1.	Perkiraan penjualan beban listrik	0,00002832	0,0000051699908312
2.	Peramalan harga saham	0,0131	0,0000278380224809
3.	Pengenalan tulisan tangan	0,0000598788	$9,28729210259779 \times 10^{-6}$
4.	Pengenalan fonem dalam Bahasa Indonesia	0,0054	0,0028031644361210

Berdasarkan grafik perbandingan nilai MSE antara jaringan syaraf tiruan tanpa optimasi arsitektur jaringan dan dengan optimasi arsitektur pada Gambar 4.29 menunjukkan bahwa nilai MSE yang dihasilkan lebih kecil. Sehingga dapat dibuktikan bahwa algoritma genetika mampu melakukan optimasi arsitektur pada jaringan syaraf tiruan.



Gambar 4.29 Grafik uji coba pengaruh algoritma genetika

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil uji coba yang telah dilakukan, maka dapat diambil kesimpulan antara lain :

1. Berdasarkan nilai MSE yang dihasilkan, model algoritma genetika (kromosom, *crossover* dan mutasi yang dirancang), berhasil menyelesaikan permasalahan optimasi arsitektur jaringan syaraf tiruan.
2. Hasil pelatihan *neural network* baik dengan dan tanpa algoritma genetik sangat tergantung pada proses inisialisasi yang diambil dari nilai acak (random) sehingga pengaruh perlakuan pada nilai MSE yang dihasilkan akan berlainan setiap kali melakukan pelatihan.
3. Perlakuan genetika yang menghasilkan nilai fitness paling baik untuk aplikasi peramalan beban listrik adalah *Add Connection Mutation* dengan nilai *fitness* sebesar 0,0000051699908312untuk arsitektur jaringan 4-1-7-19-1. Untuk aplikasi peramalan harga saham adalah *Add Connection Mutation* dengan nilai *fitness* sebesar 470,5794030080350000 untuk arsitektur jaringan 5-22-1. Untuk aplikasi pengenalan tulisan tangan adalah *Swap Number Mutation* dengan nilai *fitness* sebesar 6,4473906213471untuk arsitektur jaringan 400-385-358-400. Untuk aplikasi pengenalan fonem dalam Bahasa Indonesia adalah *Single Point Crossover* dengan nilai *fitness* sebesar 1,9263943029587300 untuk arsitektur jaringan 189-188-30.
4. Nilai MSE yang dihasilkan arsitektur jaringan syaraf tiruan dengan optimasi lebih kecil daripada arsitektur jaringan syaraf tiruan tanpa optimasi. Sehingga dapat dibuktikan bahwa algoritma genetika mampu melakukan optimasi arsitektur pada jaringan syaraf tiruan.

5.2 Saran

Saran untuk pengembangan lebih lanjut adalah

1. Sebaiknya jenis aplikasi untuk uji coba diperbanyak, sehingga memberi banyak alternatif ragam arsitektur jaringan syaraf tiruan.
2. Bentuk koneksi arsitektur jaringan sebaiknya diberi pilihan *full connection* dan *random connection*, untuk mengetahui pengaruh bentuk arsitektur terhadapa nilai MSE yang dihasilkan.
3. Sebaiknya lama waktu pelatihan dipertimbangkan selain nilai MSE yang dihasilkan.

DAFTAR PUSTAKA

Abdelmaguid, T.F and Dessouley, M.M.2004. *A Genetic Algorithm Approach to The Integrated Inventory-Distribution Problem*. University of Southern California. California http://www.rcf.usc.edu/~paged/publication/GAinventoryrouting.pdf, tanggal akses : 14 Juli 2008

Apriliyah. 2008. *Perkiraan Penjualan Beban Lisrik Menggunakan Jaringan Syaraf Tiruan Resilient Backpropagation*. Universitas Brawijaya. Malang

Arifin, M.S., 2008. *Pengenalan Tulisan Tangan Menggunakan Algoritma Jaringan Syaraf Tiruan Propagasi Balik (Backpropagation)*. Universitas Brawijaya. Malang

Basri, H. 2008. *Pengaruh Waktu Penambahan Job Baru Pada Penjadwalan Job Shop Dinamis Menggunakan Algoritma Genetik*. Universitas Brawijaya. Malang

Fiszelew, A., Britis, P., Ochoa A., Merlino, H.,Fernandez E., Garcia-Martinez. R *Finding Optimal Neural Network Architecture Using Genetic Algorithms*. Buenos Aires Institute of Technology

Gen, M and Runwei C. 1997. *Soft and Engine Ring Design*. John Willey and Sons

Hannawati, Anies dan Thiang. 2002. *Pencarian Rute Optimum Menggunakan Algoritma Genetika*. Universitas Kristen Petra. Surabaya.http://www.petra.ac.id/~puslit/journals/articles.php?PublishedID=ELK02020205, tanggal akses : 16 Juli 2008

Hermawan, A. 2006. *Jaringan Syaraf Tiruan: Teori dan Aplikasi*. Andi.Yogyakarta

Koehn, P. 1994. *Combining Genetic Algorithms and Neural Networks: The Encoding Problem*.The University of Tennessee. Knoxville

Kristanto, A. 2004. *Jaringan Syaraf Tiruan (Konsep Dasar,Algoritma dan Aplikasi)*. Gava Media.Yogyakarta

Kurnia, M. N. 2006. *Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika*. Universitas Brawijaya. Malang

Matteucci, M. *ELeaRNT : Evolutionary Learning of Rich Neural Network Topologies*. School of Computer Science Carnegie Mellon University

Nurwijaya, 2007. *Analisis Penggunaan Algoritma Genetika Untuk Optimalisasi Jaringan Syaraf Tiruan*. Institut Teknologi Bandung.Bandung

Praharda, A.S., 2008. *Metode Second Position Asymmetric Windowing dan Jaringan Syaraf Tiruan Backpropagation untuk Pengenalan Fonem dalam Bahasa Indonesia*. Universitas Brawijaya. Malang

Setyaningrum, R.D., 2007. *Peramalan Harga Saham Menggunakan Jaringan Syaraf Tiruan: Backpropagation*. Universitas Brawijaya. Malang

Suyanto, 2005. *Algoritma Genetika Dalam Matlab*.Andi offset. Yogyakarta

Taylor, Christopher M. 1997. *Selecting Neural Network Topologies : A Hybrid Approach Combining Genetic Algorithms and Neural Networks*. Southwest Missouri State University

Eliyani, 2005. *Pengantar Jaringan Syaraf Tiruan*.www.materikuliah.com