KENDALI DINAMIS PARAMETER ALGORITMA GENETIKA MENGGUNAKAN LOGIKA FUZZY

SKRIPSI

Oleh:
RAGIL INTAN NOVITASARI
0610940048-94

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

2011

KENDALI DINAMIS PARAMETER ALGORITMA GENETIKA MENGGUNAKAN LOGIKA FUZZY

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Sains dalam bidang Matematika

Oleh:
RAGIL INTAN NOVITASARI
0610940048-94

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

2011

LEMBAR PENGESAHAN SKRIPSI

KENDALI DINAMIS PARAMETER ALGORITMA GENETIKA MENGGUNAKAN LOGIKA FUZZY

Oleh:
RAGIL INTAN NOVITASARI
0610940048-94

Setelah dipertahankan di depan Majelis Penguji pada tanggal 23 Mei 2011 dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Sains dalam bidang Matematika

Pembimbing I	Pembimbing II
Syaiful Anam, S.Si., MT.	**Prof.Dr. Agus Widodo, M.Kes.**
NIP. 197801152002121003	**NIP. 195305231983031002**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr.Drs. Abdul Rouf Alghofari, M.Sc.
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Ragil Intan Novitasari**
NIM	**: 0610940048-94**
Jurusan	**: Matematika**
Penulis Skripsi berjudul	**: Kendali Dinamis Parameter Algoritma Genetika Menggunakan Logika *Fuzzy***

Dengan ini menyatakan bahwa :

1. **skripsi ini adalah benar-benar karya saya sendiri dan bukan hasil plagiat dari karya orang lain. Karya-karya yang tercantum dalam Daftar Pustaka Skripsi ini, semata-mata digunakan sebagai acuan/referensi,**
2. **apabila di kemudian hari diketahui bahwa isi Skripsi saya merupakan hasil plagiat, maka saya bersedia menanggung akibat hukum dari keadaan tersebut.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 23 Mei 2011
Yang menyatakan,

(Ragil Intan Novitasari)
NIM 0610940048

KENDALI DINAMIS PARAMETER ALGORITMA GENETIKA MENGGUNAKAN LOGIKA *FUZZY*

ABSTRAK

Algoritma genetika merupakan algoritma optimasi yang dapat menemukan solusi yang mendekati nilai eksak untuk masalah-masalah optimasi yang rumit dengan banyak optimum lokal. Algoritma ini bergantung pada penetapan nilai parameternya. Penetapan nilai parameter yang tidak tepat akan menyebabkan kinerja algoritma genetika menjadi jelek. Untuk mengatasi masalah tersebut, diperkenalkan algoritma genetika dengan kendali dinamis. Algoritma genetika dengan kendali dinamis adalah algoritma genetika yang menggunakan pengetahuan *fuzzy* berbasis sistem untuk mengontrol parameter algoritma genetika secara dinamis. Pengetahuan *fuzzy* berbasis sistem adalah sistem basis aturan, yang dibangun pada teori logika *fuzzy* dan himpunan *fuzzy*.

Kinerja algoritma genetika biasa dibandingkan dengan kinerja algoritma genetika dengan parameter dinamis menggunakan beberapa fungsi. Hasil percobaan menunjukkan adanya peningkatan kinerja pada algoritma genetika dengan parameter dinamis jika dibandingkan dengan algoritma genetika biasa.

Kata kunci : algoritma genetika biasa, algoritma genetika dengan parameter dinamis, pengetahuan *fuzzy* berbasis sistem.

DYNAMIC CONTROL OF GENETIC ALGORITHMS USING FUZZY LOGIC

ABSTRACT

Genetic algorithm is an optimization algorithm that can find solutions approaches the exact value for complicated optimization problem which has many local optimum spots. Performance of genetic algorithm depends on the value of parameters. The inappropriate determination of the value of parameters causes bad performance of genetic algorithms. To handle this problem, dynamic control of genetic algorithms is proposed. Genetic algorithm with dynamic control is a genetic algorithm which uses a fuzzy knowledge based system to control genetic algorithm parameters dynamically. Fuzzy knowledge based systems is a rule of based systems, which are built on fuzzy logic and fuzzy set theory.

Ordinary genetic algorithm and genetic algorithm with dynamic control performance are compared for some test functions. Result from the experiments show performance improvement of genetic algorithm with dynamic control over ordinary genetic algorithm.

Keywords : ordinary genetic algorithms, genetic algorithms with dynamic control, fuzzy knowledge based system.

KATA PENGANTAR

Segala puji, syukur, hormat dan kemuliaan kepada Allah SWT yang telah melimpahkan rahmat, hidayah serta inayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "**Kendali Dinamis Parameter Algoritma Genetika Menggunakan Logika *Fuzzy***". Skripsi ini merupakan sebagian persyaratan kelulusan dalam memperoleh gelar kesarjanaan di Fakultas MIPA Jurusan Matematika Universitas Brawijaya.

Banyak pihak yang telah memberikan dukungan baik moral maupun spiritual secara langsung maupun tidak langsung dalam penyelesaian skripsi ini. Pada kesempatan kali ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Syaiful Anam, SSi., MT. selaku pembimbing I atas segala bimbingan, nasihat, motivasi serta kesabaran yang telah diberikan selama penulisan skripsi ini.
2. Prof.Dr. Agus Widodo, M.Kes selaku pembimbing II atas segala bimbingan, nasihat, motivasi serta kesabaran yang telah diberikan selama penulisan skripsi ini.
3. Dr. Wuryansari Muharini K., M.Si., Dr. Sobri Abusini, MT. dan Drs. Marsudi, MS. selaku dosen penguji atas segala saran yang diberikan untuk perbaikan skripsi ini.
4. Dr. Agus Suryanto, M.Sc. selaku dosen pembimbing akademik atas segala bimbingan, nasihat, motivasi serta kesabaran yang telah diberikan.
5. Seluruh dosen Matematika yang telah memberikan bekal dan ilmu pengetahuan serta staf TU Jurusan Matematika atas segala bantuannya.
6. Kedua orang tua dan kakak-kakak yang saya sayangi atas segala semangat dan dukungan serta doa yang tanpa henti diberikan sehingga penulis dapat menyelesaikan skripsi ini.
7. Sahabat-sahabat dan teman seperjuangan Math'06 atas dukungan dan gangguannya.
8. Serta semua pihak yang telah membantu proses penulisan skripsi ini yang tidak dapat disebutkan satu persatu.

Semoga Allah SWT memberikan anugerah dan karunia-Nya kepada semua pihak yang telah membantu menyelesaikan skripsi ini. Penulis menyadari bahwa dalam skripsi ini masih banyak

kekurangan, oleh karena itu segala kritik dan saran dari pembaca sangat diharapkan demi perbaikan selanjutnya melalui email penulis ch4_yi@yahoo.com.

Akhirnya semoga skripsi ini dapat bermanfaat bagi pembaca, khususnya mahasiswa Matematika Universitas Brawijaya.

Malang, Mei 2011

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sejak tahun 1960 telah terjadi banyak peningkatan ketertarikan dalam hal meniru pola kehidupan untuk mengembangkan algoritma yang dapat menyelesaikan masalah optimasi yang sulit dengan baik (Gen dan Cheng, 1997). Optimasi merupakan hal yang penting dalam pengambilan keputusan. Secara matematis, optimasi merupakan masalah memaksimumkan atau meminimumkan suatu besaran tertentu, yang disebut dengan fungsi tujuan (Nocedal, 2006). Salah satu algoritma yang dapat menyelesaikan masalah optimasi dengan baik adalah algoritma genetika (Gen dan Cheng, 1997).

Algoritma genetika ditemukan oleh John Holland pada tahun 1960 dan dikembangkan oleh Holland dan murid-murid serta rekannya. Algoritma genetika terinspirasi oleh mekanisme seleksi alam yang mana individu yang lebih kuat menjadi pemenang dalam lingkungan yang berkompetisi. Algoritma genetika relatif lebih mudah dipahami karena komponen-komponen pembentuk algoritma ini mencerminkan kehidupan di alam, seperti seleksi, pindah silang, dan mutasi. Algoritma ini dapat menemukan solusi yang baik untuk masalah-masalah optimasi fungsi yang rumit dengan banyak optimum lokal (Mitchell, 1999).

Untuk meningkatkan kinerja algoritma genetika, beberapa metode telah diselidiki, salah satunya adalah metode untuk menentukan nilai parameter algoritma genetika. Penentuan nilai parameter algoritma genetika biasa, dilakukan di awal dan akan terus digunakan sampai dengan jumlah maksimum generasi. Penentuan nilai parameter algoritma genetika sangat mempengaruhi kinerja algoritma genetika. Penentuan nilai parameter yang tidak tepat akan menyebabkan waktu yang diperlukan untuk mencapai kekonvergenan menjadi lama atau titik yang dicapai belum optimum.

Dalam skripsi ini perbaikan algoritma genetika diperkenalkan, yaitu menggunakan pengetahuan *fuzzy* berbasis sistem untuk menentukan nilai parameter algoritma genetika secara dinamis. Pengetahuan *fuzzy* berbasis sistem adalah sistem basis aturan yang dibangun dalam logika *fuzzy* dan teori himpunan *fuzzy*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka masalah yang akan dibahas dalam skripsi ini adalah sebagai berikut.

1. Bagaimana menentukan nilai parameter algoritma genetika secara dinamis dengan menggunakan logika *fuzzy*?
2. Bagaimana perbandingan kinerja algoritma genetika biasa dan algoritma genetika yang menggunakan logika *fuzzy*?

1.3 Batasan Masalah

Penulisan skripsi ini difokuskan pada pembahasan dengan beberapa batasan masalah, yaitu:

1. parameter algoritma genetika yang dijadikan parameter dinamis adalah peluang pindah silang dan peluang mutasi,
2. hal yang diperbandingkan antara algoritma genetika biasa dan algoritma genetika dengan parameter dinamis adalah waktu komputasi dan titik optimum fungsi.

1.4 Tujuan Penulisan

Berdasarkan masalah di atas, tujuan penulisan skripsi ini adalah untuk:

1. menentukan nilai parameter algoritma genetika secara dinamis dengan menggunakan logika *fuzzy*,
2. membandingkan kinerja algoritma genetika biasa dan algoritma genetika yang menggunakan logika *fuzzy*.

BAB II
TINJAUAN PUSTAKA

Pada bab ini diberikan beberapa definisi dan teori untuk membantu memahami permasalahan yang akan dibahas dan juga digunakan sebagai acuan dalam pembahasan.

2.1 Konsep Dasar Optimasi Fungsi

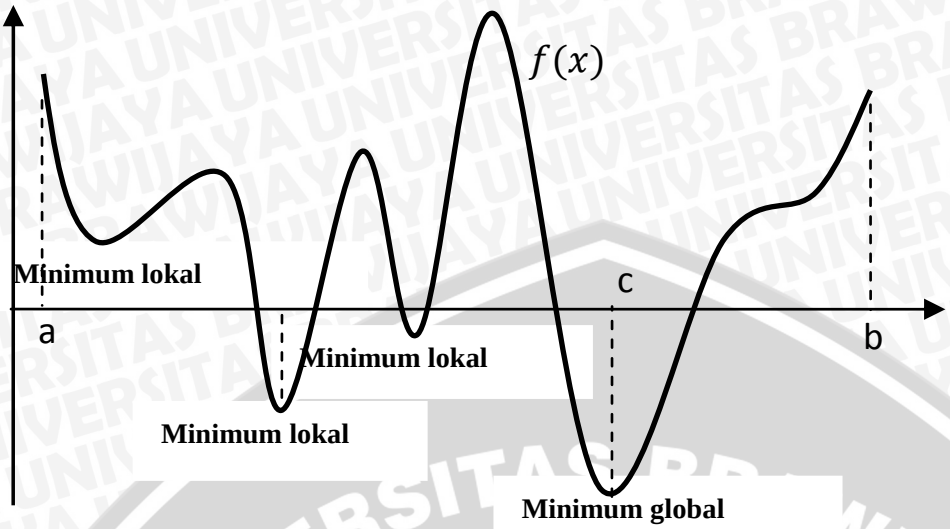
Optimasi merupakan masalah memaksimumkan atau meminimumkan suatu besaran tertentu, yang disebut fungsi tujuan (Nocedal, 2006). Optimasi dibagi menjadi dua, optimasi linear dan nonlinear. Suatu permasalahan optimasi disebut linear jika fungsi tujuan dan atau kendalanya mempunyai bentuk linear dan disebut nonlinear jika fungsi tujuan dan atau kendalanya mempunyai bentuk nonlinier. Optimasi nonlinear dapat dibagi menjadi empat berdasarkan kriteria permasalahannya yaitu optimasi nonlinear satu variabel tanpa kendala, multi variabel tanpa kendala, satu variabel dengan kendala, dan multi variabel dengan kendala (Luknanto, 2000).

Optimasi nonlinear tanpa kendala untuk kasus meminimumkan fungsi tujuan dapat ditulis sebagai berikut:
Diberikan fungsi tujuan $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ akan ditentukan $\boldsymbol{x}^* \in \mathbb{R}^n$ yang memenuhi $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathbb{R}^n$. (Bergh, 2006)

Definisi 2.1 (Minimum Lokal)
Diberikan $\boldsymbol{x}^* \in X$, $\boldsymbol{x}^*$ disebut sebagai titik minimum lokal dari fungsi $f: X \longmapsto \mathbb{R}$ jika $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ untuk semua $\boldsymbol{x}$ yang berdekatan dengan $\boldsymbol{x}^*$.
Jika $X \subseteq \mathbb{R}^n$, maka $\boldsymbol{x}^*$ disebut sebagai titik minimum lokal jika $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})\ \forall \boldsymbol{x} \in X, \|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$ yang mana $\varepsilon > 0$.

Definisi 2.2 (Minimum Global)
Diberikan $\boldsymbol{x}^* \in X$, $\boldsymbol{x}^*$ disebut sebagai titik minimum global dari fungsi $f: X \longmapsto \mathbb{R}$ jika $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in X$.

Contoh minimum lokal dan minimum global dapat dilihat pada Gambar 2.1.



Gambar 2.1 Minimum lokal dan minimum global.

Pada Gambar 2.1 fungsi $f(x)$ dibatasi pada interval [a,b]. Grafik $f(x)$ memiliki beberapa titik minimum lokal dan memiliki satu titik minimum global, yaitu pada titik *c*.

Galat merupakan selisih antara nilai perkiraan dari suatu variabel *x* dengan nilai sebenarnya.

$$\varepsilon = \boldsymbol{x}^* - \boldsymbol{x}$$

Mean Square Error (MSE) merupakan salah satu metode untuk menghitung galat.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^* - \boldsymbol{x}_i)^2$$

yang mana $\boldsymbol{x}^*$ = nilai perkiraan
$\boldsymbol{x}$ = nilai sebenarnya

(Weise, 2007)

2.2 Algoritma Genetika

2.2.1 Latar Belakang Algoritma Genetika

Algoritma genetika didefinisikan sebagai algoritma pencarian yang didasarkan pada mekanisme seleksi alamiah dan genetika alamiah. Algoritma ini pertama kali diperkenalkan oleh John Holland

di Universitas Michigan pada tahun 1960 dan kemudian dikembangkan oleh murid-murid serta rekan-rekannya.

Munculnya algoritma genetika diinspirasi oleh teori-teori dalam ilmu biologi, sehingga banyak istilah dan konsep biologi yang digunakan dalam algoritma genetika, seperti gen, kromosom, individu, reproduksi, *fitness* dan generasi.

Kromosom adalah suatu *string* dari DNA dan bertindak sebagai model suatu organisme. Kromosom dapat dibagi menjadi beberapa gen. Masing-masing gen mengkodekan suatu protein tertentu dan memiliki posisi sendiri di dalam kromosom.

Reproduksi adalah proses untuk membuat keturunan baru. Beberapa proses yang terjadi dalam reproduksi adalah pindah silang dan mutasi. Gen induk dikombinasikan dengan berbagai cara untuk membentuk kromosom baru. Kemudian kromosom baru yang telah terbentuk dapat mengalami proses mutasi. Mutasi berarti mengganti beberapa elemen DNA. Nilai *fitness* suatu organisme diukur dari seberapa sukses organisme tersebut bertahan hidup (Mitchell, 1999).

2.2.2 Garis Besar Algoritma Genetika

Langkah-langkah garis besar algoritma genetika adalah sebagai berikut.

1. Memulai membuat suatu populasi dengan n kromosom secara acak. Kromosom merupakan kandidat solusi untuk masalah.
2. Menghitung nilai *fitness* $f(x)$ masing-masing kromosom x dalam populasi.
3. Mengulang langkah-langkah berikut sampai n keturunan terbentuk:
   a. Memilih dua kromosom dari populasi, yang akan dijadikan induk, pemilihan kromosom didasarkan pada nilai *fitness*-nya yaitu *fitness* yang lebih baik mempunyai kesempatan lebih besar untuk dipiih. Kromosom yang sama dapat dipilih lebih dari sekali untuk menjadi induk.
   b. Berdasarkan probabilitas pindah silang, dilakukan proses persilangan dua induk untuk membentuk keturunan.
   c. Berdasarkan probabilitas mutasi, dilakukan proses mutasi pada keturunan hasil proses pindah silang.
4. Menggantikan populasi lama dengan populasi baru.
5. Kembali ke langkah 2. (Mitchell, 1999)

2.2.3 Skema Pengkodean

Beberapa skema pengkodean yang umum digunakan, yaitu :

1. *Real number encoding*

   Pada skema ini, nilai gen berada dalam interval $[0, R]$, yang mana R adalah bilangan real positif dan biasanya $R = 1$. Contoh dapat dilihat pada Tabel 2.1.

Tabel 2.1 Skema pengkodean *real number encoding*

x_1	x_2	x_3
0.2390	**1.0000**	**0.0131**
g_1	g_2	g_3

2. *Discrete decimal encoding*

   Pada skema ini, setiap gen bernilai salah satu bilangan bulat dalam interval [0,9]. Contoh dapat dilihat pada Tabel 2.2.

Tabel 2.2 Skema pengkodean *discrete decimal encoding*

x_1			x_2			x_3		
2	**3**	**9**	**9**	**9**	**9**	**0**	**1**	**3**
g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9

3. *Binary encoding*

   Pada skema ini, setiap gen hanya bernilai 0 atau 1. Salah satu permasalahan yang menggunakan skema pengkodean *binary encoding* adalah menghitung nilai optimal fungsi. Contoh dapat dilihat pada Tabel 2.3.

Tabel 2.3 Skema pengkodean *binary encoding*

x_1			x_2			x_3		
0	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9

Dekode kromosom adalah mengubah kromosom menjadi bilangan real setelah mengalami pengkodean, kemudian disubstitusikan sebagai nilai dari variabel $x_{i,j}$ untuk dievaluasi ke dalam fungsi yang dicari nilai optimalnya.

Dekode kromosom untuk masing-masing *encoding*, adalah sebagai berikut.

1. *Real number encoding*

$$x_{i,j} = r_b + (r_a - r_b)g_{i,j,k} \quad (2.1)$$

2. *Discrete encoding*

$$x_{i,j} = r_b + (r_a - r_b)(g_{i,j,1}.10^{-1} + g_{i,j,2}.10^{-2} + \cdots + g_{i,j,n}.10^{-n}) \quad (2.2)$$

3. *Binary encoding*

$$x_{i,j} = r_b + (r_a - r_b)(g_{i,j,1}.2^{-1} + g_{i,j,2}.2^{-2} + \cdots + g_{i,j,n}.2^{-n}) \quad (2.3)$$

yang mana $x_{i,j}$ = variabel pada individu ke-i, indeks variabel ke-j
r_a = batas atas interval
r_b = batas bawah interval
i = indeks individu (individu ke-1,2,3, ... , N)
j = indeks variabel
k = indeks gen (gen ke-1,2,3, ... , n)
$g_{i,j,k}$ = nilai gen pada individu ke-i variabel ke- j indeks gen ke-k
n = jumlah gen yang mewakili 1 variabel
(Suyanto, 2005).

2.2.4 Fungsi *Fitness*

Fungsi *fitness* merupakan alat ukur yang digunakan untuk proses evaluasi kromosom. Fungsi inilah yang akan dioptimalkan dan akan menghasilkan nilai *fitness*. Seperti dalam evolusi alam, individu yang bernilai *fitness* tinggi akan bertahan hidup. Sedangkan individu yang bernilai *fitness* rendah akan mati.

Pada masalah optimasi, jika solusi yang dicari adalah memaksimalkan sebuah fungsi h maka fungsi *fitness* yang digunakan adalah

$$f = h \quad (2.4)$$

sedangkan, jika solusi yang dicari adalah meminimumkan sebuah fungsi h maka sama dengan memaksimumkan fungsi g, yang mana $g = -h$ sehingga fungsi *fitness* yang digunakan adalah

$$f = -h \quad (2.5)$$

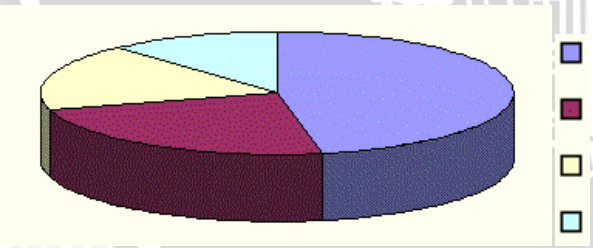
yang mana h = nilai fungsi tujuan (Michalewicz, 1996).

2.2.5 Operator Algoritma Genetika

2.2.5.1 Seleksi

Dari penjelasan pada *outline* algoritma genetika, dua kromosom dari populasi dipilih untuk dijadikan induk pada proses pindah silang. Masalahnya adalah bagaimana memilih kromosom tersebut. Dalam algoritma genetika kelangsungan hidup suatu individu ditentukan oleh operasi seleksi. Seleksi bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang memiliki nilai *fitness* tinggi. Ada banyak metode untuk memilih individu terbaik tersebut. Beberapa metode adalah sebagai berikut.

1. *Roulette wheel selection*
   Pada metode *roulette wheel*, induk dipilih berdasarkan nilai *fitness*-nya. Kromosom yang baik akan memiliki peluang yang lebih besar untuk terpilih. Bila digambarkan pada sebuah *roulette wheel*, ukuran setiap kromosom dalam suatu populasi akan seimbang sesuai nilai *fitness*-nya seperti ditunjukkan dalam Gambar 2.2.



Gambar 2.2 Contoh penggunaan metode *roulette wheel*

Dari Gambar 2.2 dapat dilihat bahwa kromosom 1 memiliki peluang lebih besar untuk terpilih dalam proses penyeleksian, karena kromosom 1 memiliki nilai *fitness* lebih besar dari kromosom lainnya yang digambarkan dengan bagian yang lebih luas. Adapun langkah-langkah proses *roulette wheel* adalah sebagai berikut:

1. hitung nilai *fitness* untuk setiap kromosom

   $fitness(k)$ (2.6)

2. hitung total nilai *fitness* pada populasi

   $total\ fitness = \sum_{k=1}^{n} fitness(k)$ (2.7)

3. hitung probabilitas seleksi p_k untuk setiap kromosom

   $p_k = \frac{fitness(k)}{total\ fitness}$ (2.8)

4. hitung peluang kumulatif q_k untuk setiap kromosom

   $q_k = \sum_{j=1}^{k} p_j$ (2.9)

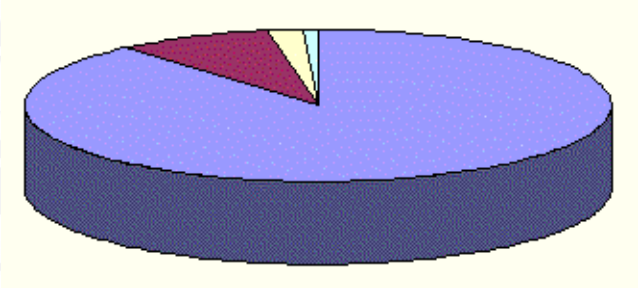
   yang mana $fitness(k)$ = *fitness* kromosom ke k
   k = 1,2,3,..., n
   n = jumlah kromosom dalam populasi

Setelah didapatkan nilai q_k langkah selanjutnya adalah sebagai berikut:

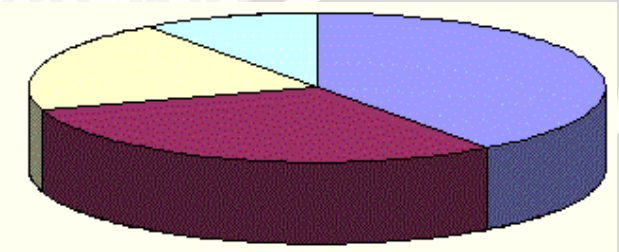
1. bangkitkan bilangan acak r pada selang [0,1]
2. jika $r \leq q_1$, maka pilih kromosom pertama, jika tidak memenuhi, pilih kromosom ke- k $(2 \leq k \leq n)$ yang mana $q_{k-1} < r < q_k$ (Michalewicz, 1996).

2. *Rank selection*

Pada *roulette wheel selection* akan muncul masalah ketika nilai *fitness* terlalu jauh berbeda, misalnya jika kromosom terbaik mempunyai nilai *fitness* 90% dari seluruh *roulette wheel* maka kromosom yang lain akan memiliki kemungkinan sangat kecil untuk terpilih. Dengan *rank selection* populasi diranking, kemudian masing-masing kromosom mendapat nilai *fitness* dari ranking tersebut. *Fitness* yang paling bagus mendapat ranking 1, selanjutnya mendapat ranking 2, dan seterusnya dan yang paling jelek mendapat ranking n (jumlah kromosom dalam populasi). Contoh penggunaan metode *rank selection* dapat dilihat pada Gambar 2.3 dan Gambar 2.4.



Gambar 2.3 Situasi sebelum proses ranking



Gambar 2.4 Situasi setelah proses ranking.

Dengan demikian semua individu mempunyai kesempatan untuk dipilih. Akan tetapi metode ini membutuhkan waktu yang lebih lama untuk mendapatkan solusi optimum, karena individu terbaik tidak terlalu berbeda dengan individu yang lain.

Linear fitness ranking adalah mekanisme yang bertujuan untuk melakukan penskalaan nilai-nilai *fitness*. Metode yang digunakan adalah

$$f(i) = f_{min} + (f_{max} - f_{min})(\frac{rank(i)-1}{N-1}) \qquad (2.10)$$

yang mana i = 1,2,3...,N
f_{min} = *fitness* minimum
f_{max} = *fitness* maksimum
N = jumlah kromosom dalam populasi
$rank(i)$ = ranking individu ke-i
$rank(i)$ = 1, jika i kromosom bernilai *fitness* minimum
$rank(i)$ = N, jika i kromosom bernilai *fitness* maksimum.

3. *Elitism*

Ketika menciptakan suatu populasi baru dengan pindah silang maupun mutasi besar kemungkinan individu yang mempunyai nilai *fitness* terbaik akan hilang. Dalam metode *elitism* individu yang memiliki nilai *fitness* terbaik dijamin tidak akan hilang, yaitu dengan mengkopi individu tersebut ke dalam populasi yang baru (generasi selanjutnya). Sisanya dipilih dengan menggunakan metode yang lain, misal *roulette wheel*. Metode *elitism* dapat meningkatkan *performance* algoritma genetika dengan cepat, karena menghalangi hilangnya solusi terbaik (Mitchell, 1999).

2.2.5.2 Pindah Silang (*Crossover)*

Pasangan-pasangan kromosom dipilih dari populasi untuk dijadikan induk dalam pindah silang. Kemudian beberapa bagian dari dua kromosom ditukar pada titik pindah silang yang dipilih. Titik pindah silang adalah titik terjadinya pertukaran gen antar individu. Pertukaran tersebut akan menghasilkan kromosom-kromosom baru. Kromosom-kromosom baru tersebut merupakan salah satu kemungkinan solusi yang akan dievaluasi nilai *fitness*-nya. Hasil evaluasi digunakan untuk menentukan kromosom mana yang mempunyai kemungkinan terbesar untuk dijadikan solusi.

Beberapa jenis pindah silang untuk *binary encoding*, antara lain:

1. Pindah silang satu titik

   Satu titik pindah silang dipilih secara acak, kemudian *string* biner dari awal sampai titik tersebut dikopi dari satu induk, sisanya diambil dari induk yang lain.

Kromosom induk 1	**11101** | **001000**
Kromosom induk 2	00001 | 010101
Keturunan 1	**11101** | 010101
Keturunan 2	00001 | **001000**

2. Pindah silang dua titik

   Dua titik pindah silang dipilih secara acak, kemudian *string* biner dari awal sampai titik pertama dan dari titik kedua sampai akhir dipilih dari satu induk sisanya diambil dari induk yang lain.

Kromosom induk 1	**11 | 10100 | 1000**
Kromosom induk 2	00 | 00101 | 0101
Keturunan 1	**11 |** 00101 | **1000**
Keturunan 2	00 | **00100 |** 1000

3. Pindah silang seragam

String biner dipilih secara acak dari kedua induk.

Kromosom induk 1	**11 10 10 01 00 0**
Kromosom induk 2	00 00 10 10 10 1
Keturunan 1	**11** 00 **10** 10 **00** 1
Keturunan 2	00 **10** 10 **01** 10 **0**

(Engelbrecht, 2007)

Pemilihan kromosom induk hanya bisa dilakukan dengan suatu peluang tertentu, P_c sebagai berikut :

1. Bangkitkan bilangan acak real (r) pada selang [0,1]
2. Jika $r < P_c$, pilih individu untuk dilakukan pindah silang.

(Michalewicz, 1996)

2.2.5.3 Mutasi

Mutasi merupakan proses mengubah gen-gen pada setiap kromosom untuk mendapatkan kromosom baru. Mutasi digunakan untuk mencegah penemuan solusi terjebak dalam lokal optimum dari masalah yang dipecahkan. Mutasi mengubah keturunan baru secara acak. Terjadinya mutasi dengan peluang yang besar mengakibatkan lebih banyak dilakukan penghapusan, akan tetapi beberapa akan menguntungkan dan dapat memperbaiki gen. Pengontrolan mutasi dilakukan dengan peluang mutasi P_m.

Pada *binary encoding*, mutasi dilakukan dengan cara sebagai berikut:

a. Bangkitkan bilangan acak real (r) pada selang [0,1]
b. Jika $r < P_m$, maka 0 diubah menjadi 1 dan 1 diubah menjadi 0.

Contoh:

Keturunan asli	1**1**001001
Keturunan termutasi	1**0**001001

Pada umumnya P_m dideklarasikan sebagai $\frac{1}{n}$, yang mana n adalah jumlah gen dalam kromosom (Michalewicz, 1996).

2.2.6 Penggantian Populasi

Populasi awal dibangkitkan secara acak. Setelah dilakukan proses seleksi, pindah silang dan mutasi, populasi awal kemudian digantikan oleh populasi hasil proses seleksi, pindah silang dan mutasi (Suyanto, 2005).

2.2.7 Parameter Algoritma Genetika

Terdapat beberapa parameter algoritma genetika, antara lain:

2.2.7.1 Ukuran Populasi

Ukuran populasi atau *population size*, merupakan parameter yang berfungsi untuk menentukan jumlah individu/kromosom yang berada pada populasi (untuk setiap generasi). Semakin banyak dan beragam individu akan memberikan peluang lebih besar untuk menemukan individu yang mendekati sempurna. Jika individu yang ada terlalu sedikit, algoritma genetika hanya memiliki sedikit kemungkinan untuk melakukan pindah silang dan hanya sebagian kecil ruang pencarian yang akan dijelajahi. Namun jika terdapat terlalu banyak individu, maka waktu komputasi algoritma genetika juga akan menjadi lambat. Penelitian menunjukkan bahwa pada batasan tertentu, meningkatkan ukuran populasi akan menjadi sangat tidak berguna, karena pemecahan masalah tidak akan berjalan lebih cepat (Obitko, 1998).

2.2.7.2 Jumlah Generasi

Jumlah generasi mempunyai andil yang besar dalam menemukan individu yang lebih baik, karena semakin banyak generasi maka individu yang dihasilkan akan semakin baik dan sempurna. Tetapi tidak berarti semakin banyak jumlah generasi maka individu yang dihasilkan selalu lebih baik karena ada suatu saat nilai *fitness* semua individu menjadi sama.

2.2.7.3 Peluang Pindah Silang (P_c)

Peluang *crossover* atau peluang pindah silang adalah parameter yang berfungsi untuk mengetahui seberapa sering proses pindah silang dilakukan. Tanpa adanya pindah silang, keturunan

akan sama persis seperti induk. Dengan adanya pindah silang, keturunan dibentuk dari bagian-bagian dari induk. Jika nilai peluang pindah silang adalah 100% maka semua keturunan dihasilkan dari proses pindah silang. Jika nilainya 0% maka semua keturunan adalah kopi dari induk. Dengan adanya pindah silang diharapkan individu yang baru memiliki bagian yang bagus dari individu sebelumnya sehingga akan didapatkan individu yang lebih baik.

Semakin besar peluang pindah silang 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 (Obitko, 1998).

2.2.7.4 Peluang Mutasi (P_m)

Peluang mutasi adalah parameter yang berfungsi untuk mengetahui seberapa sering proses mutasi dilakukan. Jika tidak terjadi mutasi maka semua anak akan berasal dari proses pindah silang. Jika terjadi mutasi maka sebagian individu akan mengalami perubahan. Jika nilai peluang mutasi adalah 100% maka semua individu mengalami mutasi, jika nilainya 0% maka tak satupun individu yang termutasi. Mutasi dibuat untuk menghindari algoritma genetika terjebak pada optimum lokal. Tapi jika terjadi terlalu sering algoritma genetika bisa mencari pada pencarian yang acak.

Nilai peluang mutasi yang terlalu rendah dapat mengakibatkan gen-gen yang berpotensi tidak dicoba. Sebaliknya jika terlalu tinggi, dapat menghilangkan sifat kemiripan dengan induknya, sehingga proses pencarian salah dan menjauh dari daerah solusi yang diinginkan (Obitko, 1998).

2.2.8 Kondisi Berhenti pada Algoritma Genetika

Proses evolusi merupakan proses yang tidak pernah selesai. Demikian pula pada algoritma genetika, proses akan terus berulang hingga kondisi berhenti terpenuhi. Terdapat beberapa macam cara untuk menentukan saat proses dihentikan, sebagai berikut.

1. Proses akan berhenti ketika tidak terjadi perubahan selama beberapa generasi. Hal ini dapat dilihat dari nilai *fitness*-nya. Jika selama beberapa generasi nilai *fitness*-nya tidak mengalami perubahan, maka proses akan dihentikan.
2. Proses akan berhenti ketika solusi telah ditemukan. Misal x merupakan nilai optimum dari fungsi tujuan, x_i merupakan individu terbaik, yang mana $f(x_i) \leq |f(x) - \epsilon|$, maka solusi telah ditemukan (ϵ merupakan *error*).
3. Proses akan berhenti ketika iterasi berulang sebanyak n kali.

(Engelbrecht, 2007)

2.3 *Fuzzy*

2.3.1 Latar Belakang *Fuzzy*

Teori *fuzzy* diperkenalkan oleh Lotfi A. Zadeh pada tahun 1965 dalam tulisannya "*fuzzy Set*". Ide awal dari teori *fuzzy* adalah karena teori kontrol klasik terlalu banyak menekankan ketepatan, sehingga tidak dapat menangani sistem yang kompleks.

Pada tahun 1973 Zadeh mempublikasikan sebuah tulisan berjudul "*Outline of a new approach to the analysis of complex system and decision processes*" yang menetapkan dasar dari *fuzzy control*. Dalam tulisan ini diperkenalkan konsep dari variabel linguistik dan aturan *if-then*. Pada tahun 1975 Mamdani dan Assilian membuat kerangka dasar *fuzzy controller* dan mengaplikasikannya untuk mengontrol mesin uap. Pada tahun-tahun berikutnya kontrol *fuzzy* makin berkembang pesat dalam berbagai bidang kehidupan (Wang, 1997).

2.3.2 Logika *Fuzzy*

Logika *fuzzy* adalah peningkatan dari logika Bool yang berhadapan dengan konsep kebenaran sebagian. Logika klasik menyatakan bahwa segala hal dapat diekspresikan dalam istilah biner (0 atau 1, hitam atau putih, ya atau tidak), logika *fuzzy* menggantikan kebenaran Bool dengan tingkat kebenaran.

Logika *fuzzy* memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan", dan

"sangat". *Fuzzy* berhubungan dengan himpunan *fuzzy*, teori kemungkinan dan sistem *fuzzy* (Wang, 1997).

2.3.3 Himpunan *Fuzzy*

Himpunan *fuzzy*, merupakan suatu grup yang mewakili suatu kondisi atau keadaan dalam suatu variabel *fuzzy*. Variabel *fuzzy*, merupakan variabel yang dibahas dalam suatu sistem *fuzzy*, seperti umur, temperatur, permintaan. Himpunan *fuzzy* memiliki dua atribut, yaitu:

1. linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti Kecil, Sedang, Besar,
2. numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran suatu variabel, misalkan 40, 25, 50.

Sebagai contoh variabel umur terbagi menjadi 3 himpunan *fuzzy*, yaitu Muda, Dewasa, dan Tua.

Definisi 2.3

Jika X adalah semesta pembicaraan dan x menyatakan elemennya, maka himpunan *fuzzy* A dalam X dinyatakan sebagai

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{2.11}$$

dengan $\mu_A(x)$ disebut fungsi keanggotaan dari x dalam A. Fungsi keanggotaan memetakan setiap elemen dari X ke nilai keanggotaan antara 0 dan 1, secara matematis dinyatakan sebagai

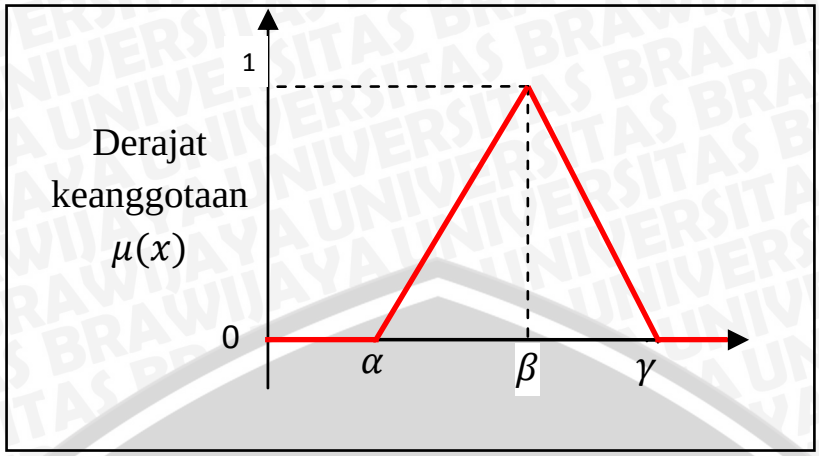
$$\mu_A(x): X \rightarrow [0,1] \tag{2.12}$$

Beberapa fungsi keanggotaan adalah sebagai berikut.

1. Fungsi keanggotaan segitiga (*triangular membership function*).

$$\mu_A(\alpha, \beta, \gamma) = max\left(min\left(\frac{x-\alpha}{\beta-\alpha}, \frac{\gamma-x}{\gamma-\beta}\right), 0\right) \tag{2.13}$$

Bentuk fungsi keanggotaan ini dapat dilihat pada Gambar 2.5, dengan α menyatakan batas bawah, γ menyatakan batas atas, dan β menyatakan titik tengah domain.

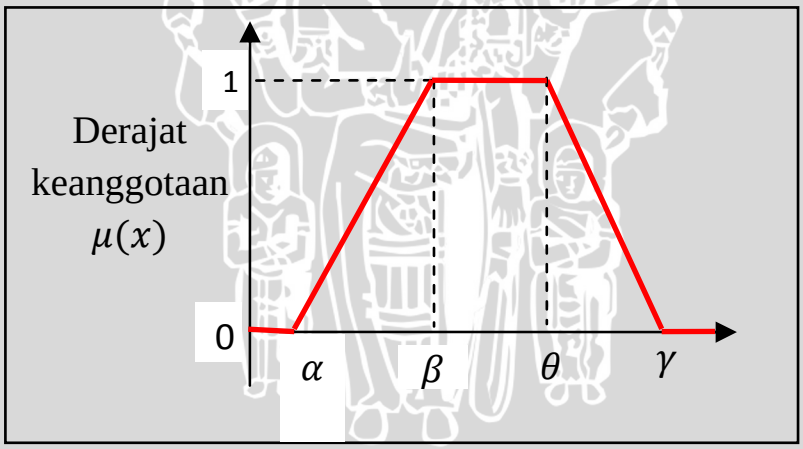


Gambar 2.5 Fungsi keanggotaan segitiga

2. Fungsi keanggotaan trapesium (*trapezoidal membership function*).

$$\mu_A(\alpha,\beta,\theta,\gamma) = max\left(min\left(\frac{x-\beta}{\beta-\alpha},1,\frac{\gamma-x}{\gamma-\theta}\right),0\right) \quad (2.14)$$

Fungsi keanggotaan trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1. Lebih jelasnya lihat Gambar 2.6.

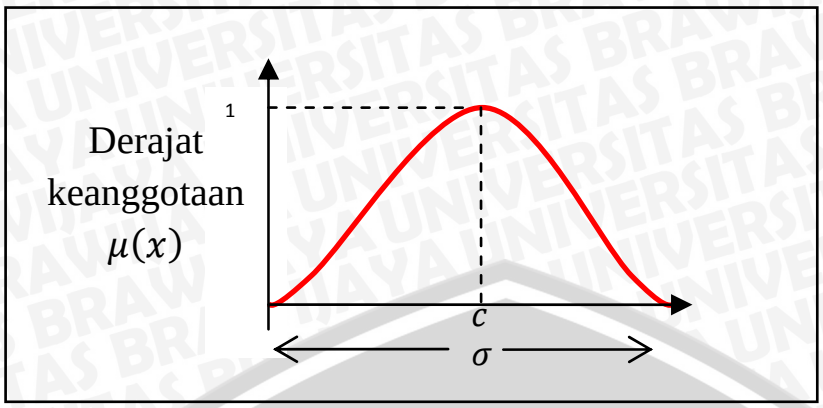


Gambar 2.6 Fungsi keanggotaan trapesium

3. Fungsi keanggotaan Gaussian (*Gaussian membership function*).

$$\mu_A(c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)} \quad (2.15)$$

Fungsi keanggotaan Gaussian menggunakan 2 parameter yaitu c yang menunjukkan nilai domain pada pusat kurva dan σ yang menunjukkan lebar kurva. Lebih jelasnya lihat Gambar 2.7.



Gambar 2.7 Fungsi keanggotaan Gaussian

2.3.4 Operator Himpunan *Fuzzy*

Terdapat 2 jenis operator himpunan *fuzzy* yang digunakan di dalam operasi himpunan *fuzzy*, yaitu operator *t-norm* dan *t-conorm* (*s-norm*).

Definisi 2.4 *t-norm*

t-norm adalah operasi biner $t: [0,1] \times [0,1] \rightarrow [0,1]$, yang memenuhi aturan-aturan sebagai berikut:

a. komutatif :$atb = bta$
b. asosiatif :$at(btc) = (atb)tc$
c. monoton : jika $b \leq c$, maka $atb \leq atc$
d. tertutup :$at1 = a$
$at0 = 0$

yang mana $a, b, c \in [0,1]$.

Untuk himpunan, *t-norm* menghasilkan hasil yang sama seperti halnya yang dihasilkan oleh operasi interseksi pada himpunan, yaitu $A \cap X = A, A \cap \emptyset = \emptyset$.

Definisi 2.5 *t-conorm*

t-conorm (*s-norm*) adalah operasi biner $s: [0,1] \times [0,1] \rightarrow [0,1]$, yang memenuhi aturan-aturan sebagai berikut:

a. komutatif : $asb = bsa$
b. asosiatif : $as(bsc) = (asb)sc$
c. monoton : jika $b \leq c$, maka $asb \leq asc$
d. tertutup : $as0 = a$
$as1 = 1$

yang mana $a, b, c \in [0,1]$ (Pedrycz, 2007).

2.3.5 Variabel Linguistik dan Aturan *Fuzzy*

Dalam kehidupan sehari-hari, kata-kata sering digunakan untuk menggambarkan variabel. Sebagai contoh, saat kita mengatakan ‘hari ini panas’ artinya sama dengan ‘temperatur hari ini tinggi’, kita menggunakan kata ‘tinggi’ untuk menggambarkan ‘temperatur hari ini’. Yang mana ‘temperatur hari ini’ sebagai variabel dan ‘tinggi’ adalah nilainya. Ketika suatu variabel mempunyai bilangan sebagai nilainya maka akan mudah untuk memformulasinya, akan tetapi ketika nilai tersebut berupa kata-kata kita tidak memiliki kerangka formal untuk memformulasikannnya dalam teori klasik. Untuk menyediakan kerangka formal, konsep variabel linguistik diperkenalkan.

Definisi 2.6

Jika suatu variabel dapat mengambil kata-kata dalam bahasa alami sebagai nilainya, maka disebut variabel linguistik, yang mana kata-kata tersebut digolongkan oleh himpunan *fuzzy* yang didefinisikan dalam semesta pembicaraan dimana variabel tersebut didefinisikan (Wang, 1997).

Secara umum, aturan dalam sistem *fuzzy* terbentuk dari variabel-variabel linguistik. Aturan-aturan tersebut berdasar pada pengetahuan dan pengalaman dari tenaga ahli dalam bidangnya masing-masing. Bentuk aturan *fuzzy* secara umum adalah sebagai berikut:

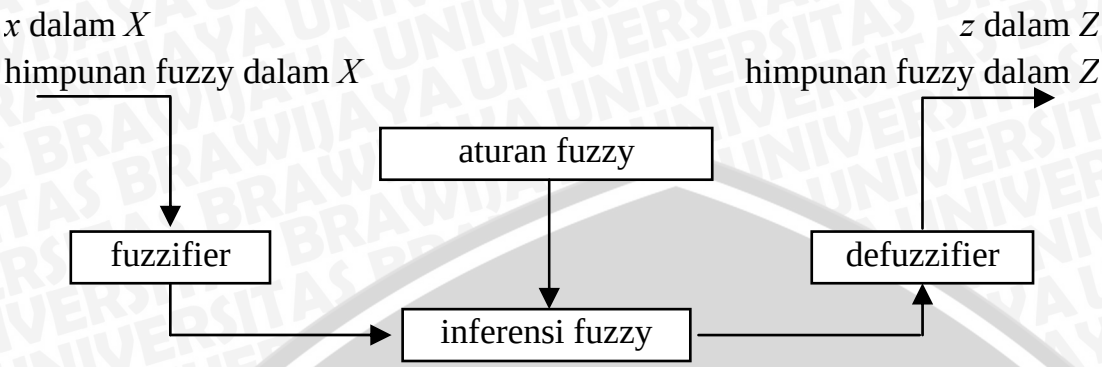
jika *antecedent* **maka** *consequent*

Antecendent dan *consequent* merupakan pernyataan *fuzzy* yang mengandung variabel linguistik. Pernyataan *fuzzy* bisa dalam bentuk pernyataan tunggal ataupun gabungan dari beberapa pernyataan (Engelbrecht, 2007).

2.3.6 Sistem *Fuzzy*

Sistem *fuzzy* adalah sebuah sistem yang dibangun oleh seperangkat aturan berdasarkan logika *fuzzy*. Salah satu aplikasi sistem *fuzzy* adalah perancangan pengontrol berbasis logika *fuzzy* yang disebut sebagai *fuzzy logic controller*. Sistem *fuzzy* dapat mengolah masukan yang berupa titik *crisp* menjadi keluaran yang juga berupa titik *crisp* dengan melalui tahap-tahap tertentu.

Gambaran umum tahap-tahap tersebut dapat dilihat pada Gambar 2.8.



Gambar 2.8 Blok diagram sistem *fuzzy*

Sistem *fuzzy* terdiri dari 4 komponen yaitu:

1. Basis aturan *fuzzy*
   Basis aturan *fuzzy* terdiri dari himpunan aturan *fuzzy if-then*
2. Sistem penarikan kesimpulan *fuzzy*
3. Fuzzifikasi
   Fuzzifikasi adalah proses memetakan titik *crisp x* di *X* ke dalam himpunan *fuzzy* A yang dinyatakan sebagai derajat keanggotaan.
4. Defuzzifikasi
   Defuzzifikasi adalah proses memetakan kembali besaran yang berupa himpunan *fuzzy* menjadi titik *crisp*. Defuzzifikasi dibutuhkan dalam penerapan sistem *fuzzy* karena yang digunakan dalam aplikasi adalah besaran *crisp* (Engelbrecht, 2007). Beberapa metode yang digunakan dalam defuzzifikasi yaitu:
   a. *Centroid of Area* (*COA*) / *Center of Gravity*, yang dinyatakan sebagai

$$COA = \frac{\int_Z \mu_A(z) z dz}{\int_Z \mu_A(z) dz} \qquad (2.16)$$

   dengan *z* adalah semesta pembicaraan himpunan keluaran *Z*.
   b. *Bisector of Area* (*BOA*), yang dinyatakan secara matematis dalam bentuk

$$\int_{\alpha}^{zBOA} \mu_A(z) dz = \int_{zBOA}^{\beta} \mu_A(z) dz \qquad (2.17)$$

α = min ($z|z \in Z$), dan β = max ($z|z \in Z$), yaitu garis vertikal z = BOA membagi daerah antara $z = \alpha$, $z = \beta$, $y = 0$ dan $y = \mu_A(z)$ ke luas daerah yang sama.

c. *Mean of Maximum* (*MOM*), yaitu z maksimum rata-rata pada saat fungsi keanggotaan menjadi maksimum dan dirumuskan sebagai

$$MOM = \frac{\int_Z z dz}{\int_Z dz} \tag{2.18}$$

yang mana $Z' = \{z|\mu_A(z) = \mu^*\}$.

Dalam skripsi ini defuzzifikasi yang dipakai adalah *Centroid of Area* (Jang dkk, 1997).

2.3.7 Model *Fuzzy* Mamdani

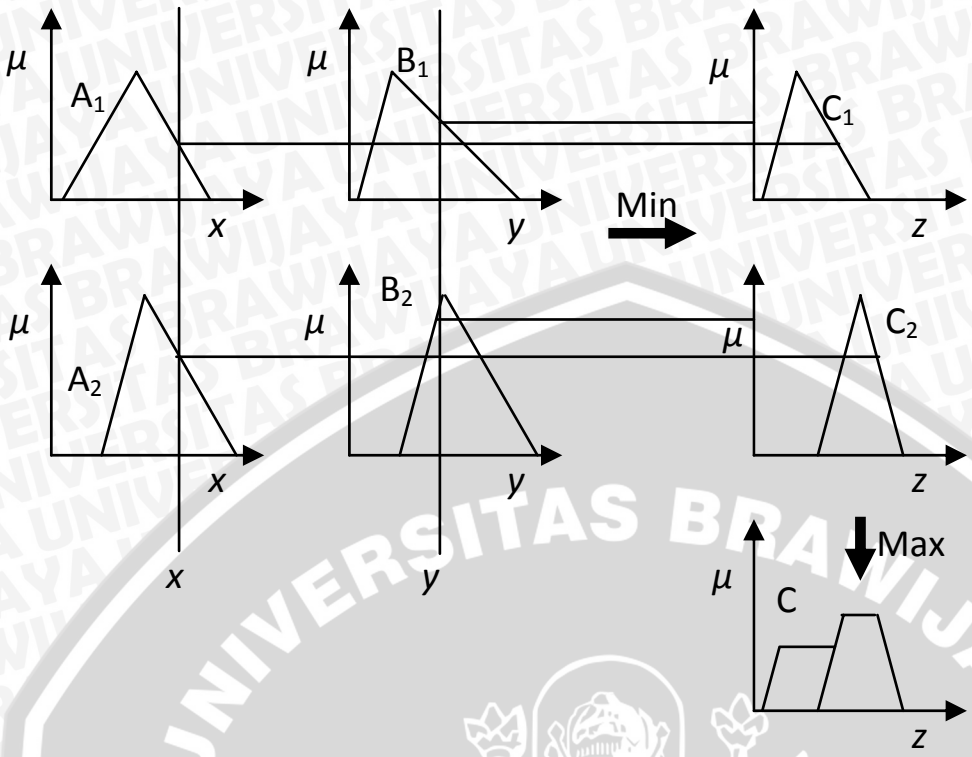
Salah satu sistem penarikan kesimpulan *fuzzy* yang paling banyak digunakan sebagai pengontrol adalah sistem penarikan kesimpulan *fuzzy* Mamdani. Tahap-tahap untuk mendapatkan keluaran dengan menggunakan sistem *fuzzy* Mamdani, yaitu:

1. menentukan suatu himpunan dari aturan *fuzzy*,
2. melakukan fuzzifikasi pada masukan menggunakan fungsi keanggotaan variabel masukan,
3. melakukan operasi inferensi sesuai aturan *fuzzy* dengan menggunakan operator *t-norm* atau *s-norm*,
4. menentukan keluaran masing-masing aturan *fuzzy* yang disesuaikan dengan fungsi keanggotaan variabel keluaran,
5. melakukan operasi *t-norm* atau *s-norm* terhadap keluaran yang diperoleh masing-masing aturan, dan
6. melakukan defuzzifikasi (Engelbrecht, 2007).

Pada Gambar 2.9 diilustrasikan bagaimana sistem inferensi Mamdani memberikan penalaran terhadap masukannya.

Aturan 1 : jika x adalah A_1 dan y adalah B_1 maka $z = C_1$

Aturan 2 : jika x adalah A_2 dan y adalah B_2 maka $z = C_2$



Gambar 2.9 Mekanisme inferensi *fuzzy* Mamdani

Gambar 2.9 memperlihatkan bagaimana dua aturan sistem inferensi *fuzzy* Mamdani menghasilkan keluaran *z* ketika diberikan dua masukan *crisp x* dan *y*, empat himpunan *fuzzy* A_1, A_2, B_1, dan B_2, dengan menggunakan metode *min-max* masing-masing untuk operator *t-norm* dan *s-norm* (Jang dkk, 1997).

BAB III
PEMBAHASAN

3.1 Algoritma Genetika dengan Parameter Dinamis

Pada dasarnya algoritma genetika dengan parameter dinamis sama dengan algoritma genetika biasa. Perbedaan algoritma genetika dengan parameter dinamis dan algoritma genetika biasa terletak pada nilai parameter algoritma genetika tersebut. Pada algoritma genetika biasa nilai parameter ditentukan di awal dan akan terus digunakan untuk seluruh generasi, sedangkan pada algoritma genetika dengan parameter dinamis, nilai parameter terus berubah-ubah pada setiap generasi.

Algoritma genetika dengan parameter dinamis adalah algoritma genetika yang menggunakan pengetahuan *fuzzy* berbasis sistem untuk mengontrol parameter algoritma genetika secara dinamis. Pengetahuan *fuzzy* berbasis sistem adalah suatu aturan berbasis sistem yang dibangun berdasarkan logika *fuzzy* dan teori himpunan *fuzzy*.

Algoritma genetika mempunyai parameter yang akan menentukan keberhasilan algoritma tersebut dalam mencapai solusi optimum, yaitu ukuran populasi, jumlah generasi, peluang pindah silang, dan peluang mutasi. Dalam skripsi ini parameter algoritma genetika yang akan dijadikan parameter dinamis adalah peluang pindah silang dan peluang mutasi.

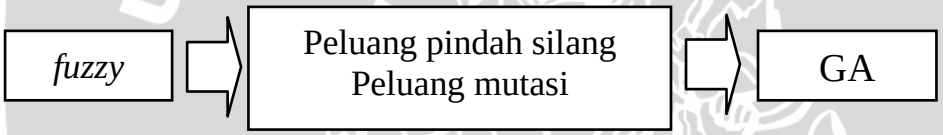
1. Peluang pindah silang

   Peluang pindah silang mengontrol frekuensi terjadinya proses pindah silang pada tiap kromosom. Peluang pindah silang yang tinggi akan menghasilkan lebih banyak kromosom baru dalam populasi. Akan tetapi jika peluang pindah silang terlalu tinggi kromosom dengan kinerja yang bagus akan lebih cepat menghilang. Peluang pindah silang yang rendah akan menghasilkan lebih sedikit kromosom baru dalam populasi. Jika peluang pindah silang terlalu rendah pencarian akan stagnan pada ruang pencarian yang kecil. Oleh karena itu, penentuan peluang pindah silang yang tepat sangat mempengaruhi kinerja algoritma genetika. Peluang pindah silang dibuat dinamis menggunakan Logika *fuzzy*. Peluang pindah silang akan berubah-

ubah nilainya pada tiap generasi dengan kisaran 0.7 sampai dengan 0.9.

2. Peluang mutasi
   Mutasi adalah operator pencarian yang dapat meningkatkan keanekaragaman populasi. Setelah proses seleksi masing-masing gen dari tiap kromosom dalam populasi mengalami perubahan secara acak dengan peluang mutasi yang telah ditentukan. Peluang mutasi yang terlalu rendah dapat mengakibatkan gen-gen yang berpotensi tidak dicoba. Sebaliknya jika terlalu tinggi, dapat menghilangkan sifat kemiripan dengan induknya, sehingga proses pencarian akan menjauh dari daerah solusi yang diinginkan. Dengan menggunakan logika *fuzzy* peluang mutasi dibuat dinamis yang akan berubah-ubah di tiap generasi, dengan kisaran 0 sampai dengan 0.02.

Secara garis besar penggunaan logika *fuzzy* pada algoritma genetika dapat dilihat pada Gambar 3.1



Gambar 3.1 Algoritma genetika dengan parameter dinamis

3.2 Perancangan *Fuzzy Logic Controller*

Pada algoritma genetika dengan parameter dinamis, *fuzzy logic controller* yang digunakan adalah sistem *fuzzy* Mamdani. Sistem yang digunakan untuk mengontrol parameter dinamis algoritma genetika ini menggunakan 3 masukan, yaitu $BF = \frac{Best\ Fitness}{Average\ Fitness}$, $AF = \frac{Average\ Fitness}{Worst\ Fitness}$, *Change of Best Fitness* (*CBF*), dan mempunyai 2 keluaran, yaitu peluang pindah silang (P_c) dan peluang mutasi (P_m). *Change of Best Fitness* merupakan perubahan nilai *Fitness* terbaik dari satu generasi ke generasi selanjutnya. Masukan untuk *BF* dan *AF* dibatasi pada selang [0,1], sedangkan untuk *CBF* dibatasi pada selang [0,20].

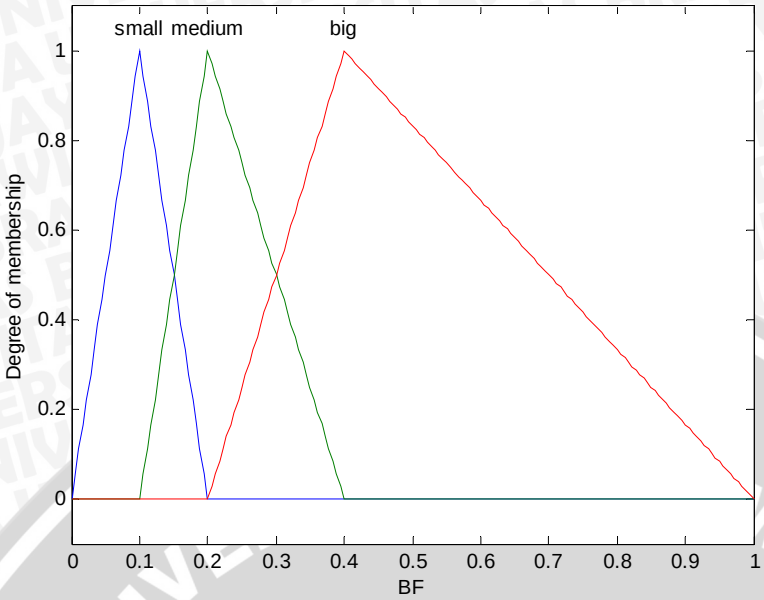
Jika nilai *BF* dan *AF* tinggi artinya populasi konvergen, yaitu memiliki nilai yang hampir sama atau sama, sehingga P_m perlu ditingkatkan. Sedangkan jika *BF* dan *AF* rendah artinya populasi masih beragam. Jika nilai *CBF* tinggi artinya *fitness* terbaik masih

terus mangalami perbaikan, sedangkan jika nilainya rendah artinya *fitness* terbaik tidak mengalami perubahan atau mengalami perubahan, tapi perubahannya sangat kecil.

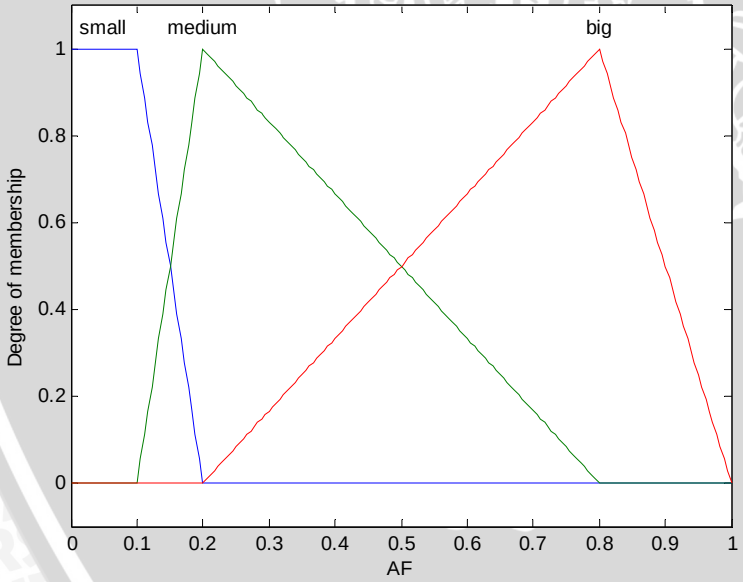
Semesta pembicaraan untuk masing-masing variabel masukan dan keluaran dibagi menjadi tiga himpunan *fuzzy*, yaitu *small*, *medium*, dan *big*. Fungsi keanggotaan yang digunakan adalah fungsi keanggotaan segitiga. Untuk lebih jelasnya dapat dilihat pada Tabel 3.1 sedangkan untuk grafik fungsi keanggotaan dapat dilihat pada Gambar 3.2.

Tabel 3.1 Keanggotaan himpunan *fuzzy*

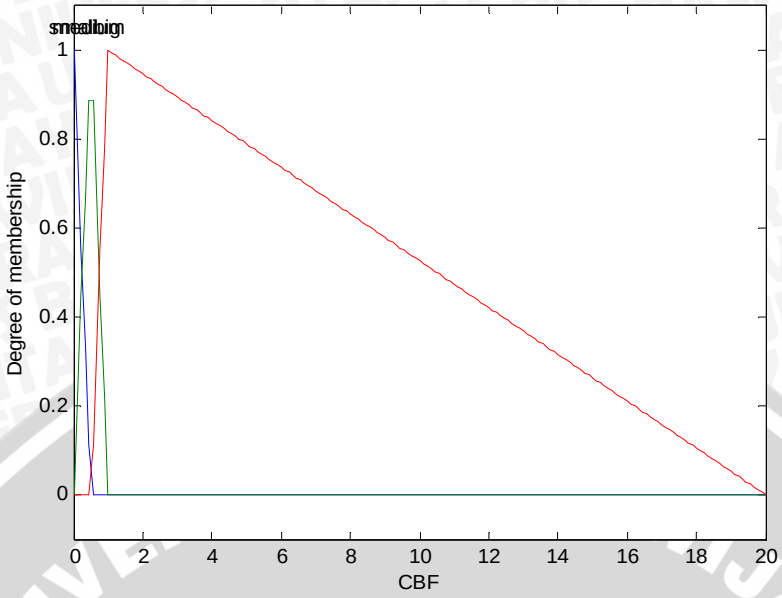
Jenis	Variabel	Himpunan *Fuzzy*	Fungsi keanggotaan	Parameter
Masukan	*BF*	Small	Segitiga	(0 0.1 0.2)
		Medium		(0.1 0.2 0.4)
		Big		(0.2 0.4 1)
	AF	Small	Trapesium	(0 0 0.1 0.2)
		Medium	Segitiga	(0.1 0.2 0.8)
		Big		(0.2 0.8 1)
	CBF	Small	Segitiga	(0 0 0.5)
		Medium		(0 0.5 1)
		Big		(0.5 1 20)
Keluaran	P_c	Small	Segitiga	(0.7 0.7 0.8)
		Medium		(0.7 0.8 0.9)
		Big		(0.8 0.9 0.9)
	P_m	Small	Trapesium	(0 0 0.0013 0.008)
		Medium	Segitiga	(0.0013 0.008 0.013)
		Big		(0.008 0.013 0.02)



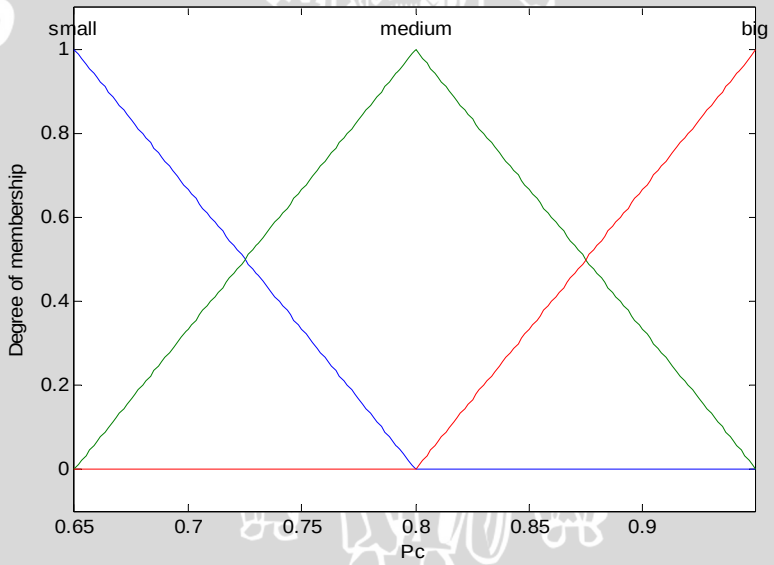
Gambar 3.2 (a) Fungsi keanggotaan *BF*



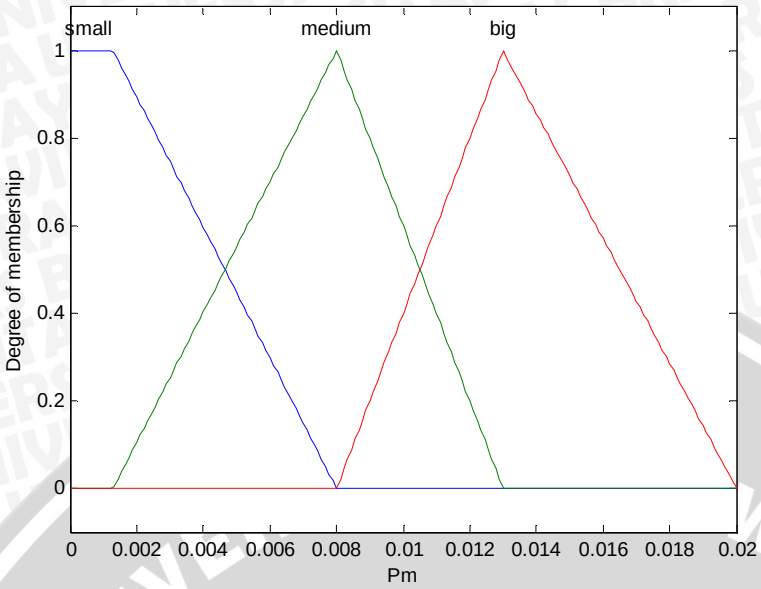
Gambar 3.2 (b) Fungsi keanggotaan *AF*



Gambar 3.2 (c) Fungsi keanggotaan *CBF*



Gambar 3.2 (d) Fungsi keanggotaan P_c



Gambar 3.2 (e) Fungsi keanggotaan P_m

Pada *Fuzzy Logic Controller*, terdapat aturan-aturan yang disebut basis aturan *fuzzy*. Basis aturan yang digunakan dapat dilihat pada Tabel 3.2 dan 3.3.

Tabel 3.2 Aturan *fuzzy* untuk peluang pindah silang

Aturan	*BF*	*AF*	*CBF*	Maka
1	*Small*	*Small*	*Small*	*Small*
2	*Small*	*Big*	*Medium*	*Small*
3	*Small*	*Big*	*Big*	*Small*
4	*Medium*	*Small*	*Small*	*Medium*
5	*Medium*	*Medium*	*Medium*	*Big*
6	*Big*	*Big*	*Small*	*Medium*
7	*Big*	*Big*	*Big*	*Medium*

Tabel 3.3 Aturan *fuzzy* untuk peluang mutasi

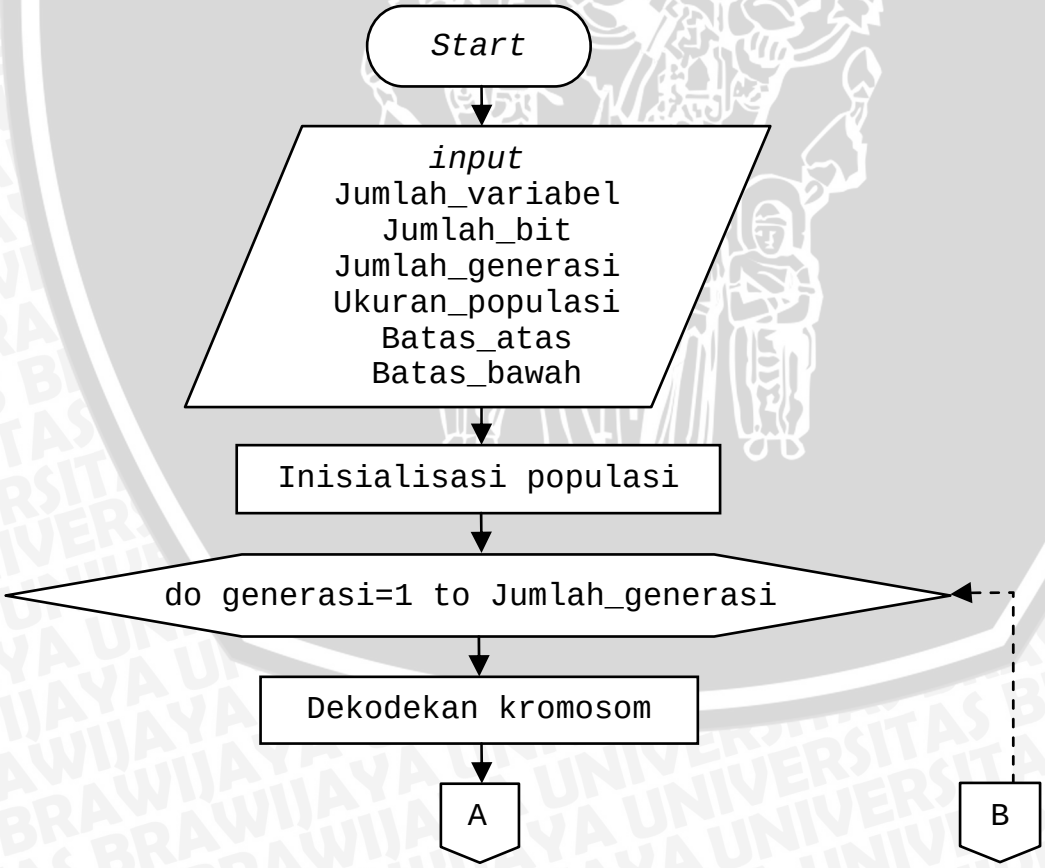
Aturan	*BF*	*AF*	*CBF*	Maka
1	*Small*	*Small*	*Small*	*Small*
2	*Small*	*Small*	*Big*	*Medium*
3	*Medium*	*Small*	*Small*	*Medium*
4	*Medium*	*Small*	*Medium*	*Big*
5	*Medium*	*Big*	*Small*	*Big*
6	*Big*	*Small*	*Small*	*Small*
7	*Big*	*Small*	*Big*	*Big*

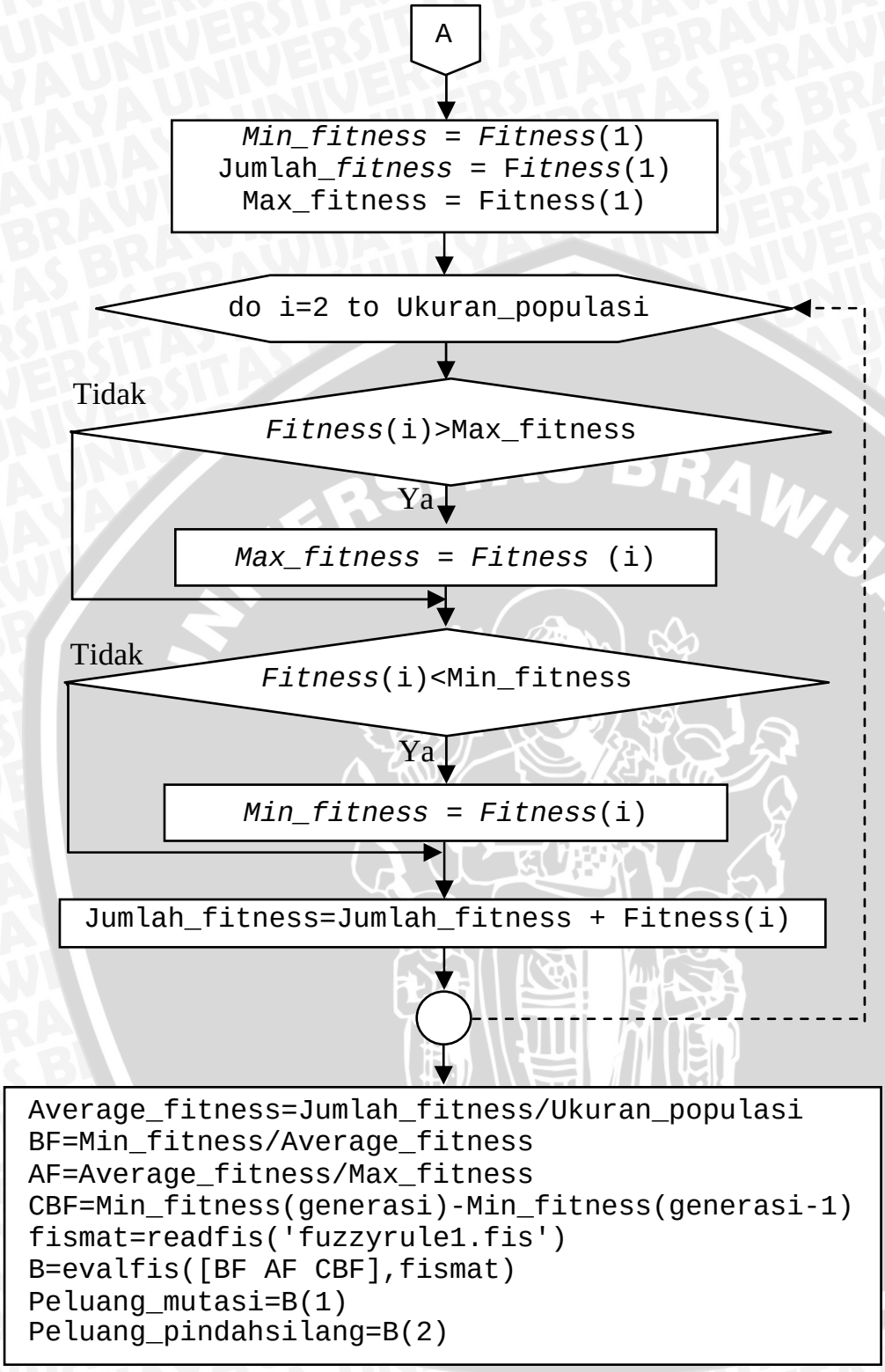
Keluaran dari *Fuzzy Logic Controller* terdiri dari beberapa besaran linguistik, masing-masing dengan derajat keanggotaan tertentu. Agar bisa digunakan dalam penghitungan, maka besaran linguistik ini harus dikonversikan terlebih dahulu ke besaran numerik yang disebut sebagai proses defuzzifikasi. Metode defuzzifikasi yang digunakan adalah metoda titik berat (COA).

Algoritma genetika dengan parameter dinamis menggunakan perintah readfis dan evalfis. Readfis adalah perintah yang digunakan untuk membaca file *.fis, yaitu file yang menyimpan sistem *fuzzy* yang digunakan sebagai pengontrol sedangkan evalfis adalah perintah yang digunakan untuk memproses hasil pembacaan sistem *fuzzy*. Perintah readfis mengolah variabel masukan, yaitu BF, AF dan CBF, menjadi variabel keluaran P_m dan P_c.

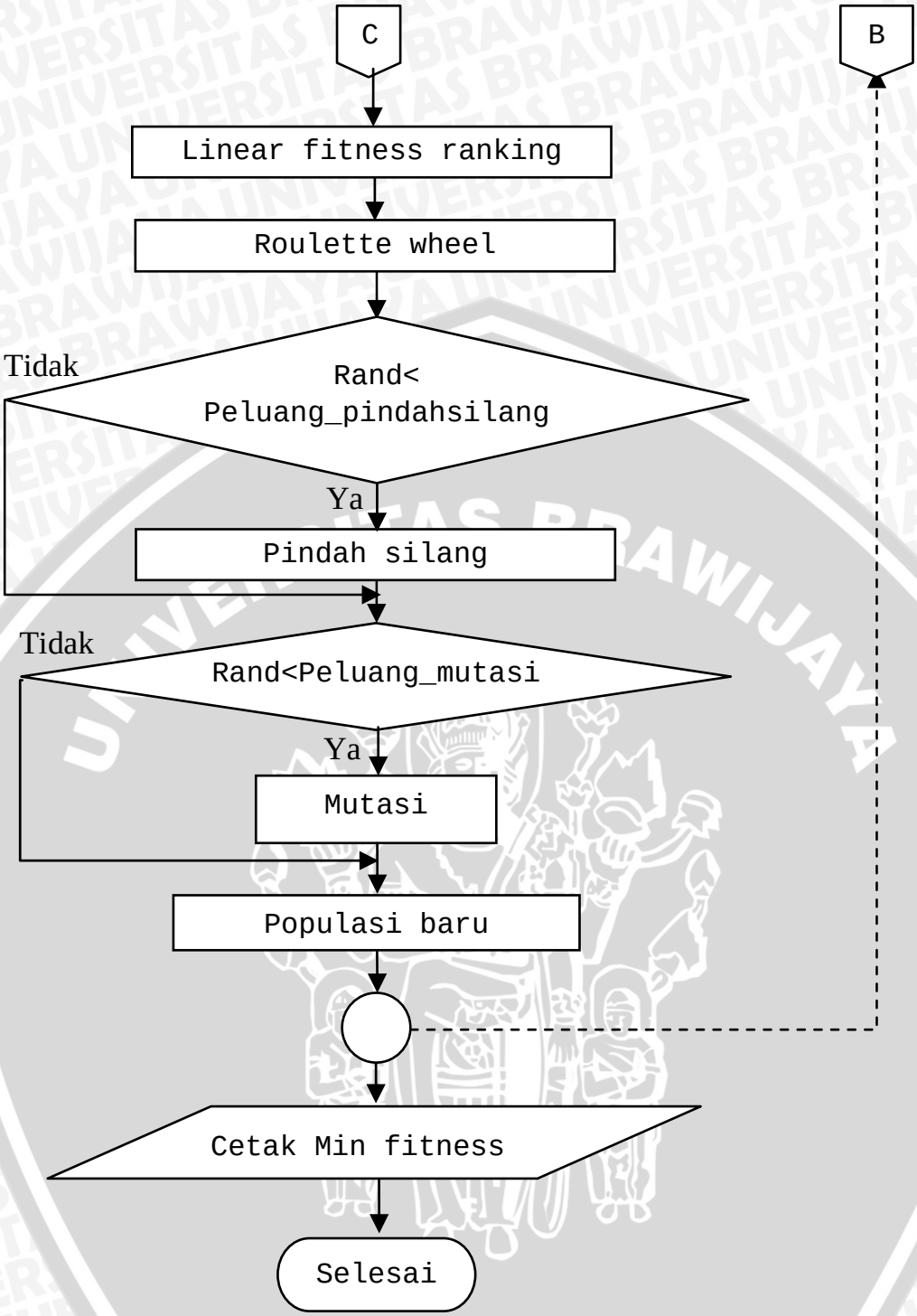
3.3 *Flowchart* Algoritma Genetika dengan Parameter Dinamis

Gambaran umum algoritma genetika dengan parameter dinamis dapat dilihat pada Flowchart 3.1




A
Min_fitness = Fitness(1)
Jumlah_fitness = Fitness(1)
Max_fitness = Fitness(1)
do i=2 to Ukuran_populasi
Tidak
Fitness(i)>Max_fitness
Ya
Max_fitness = Fitness (i)
Tidak
Fitness(i)<Min_fitness
Ya
Min_fitness = Fitness(i)
Jumlah_fitness=Jumlah_fitness + Fitness(i)
Average_fitness=Jumlah_fitness/Ukuran_populasi
BF=Min_fitness/Average_fitness
AF=Average_fitness/Max_fitness
CBF=Min_fitness(generasi)-Min_fitness(generasi-1)
fismat=readfis('fuzzyrule1.fis')
B=evalfis([BF AF CBF],fismat)
Peluang_mutasi=B(1)
Peluang_pindahsilang=B(2)

C



Pseudocode algoritma genetika dengan parameter dinamis dapat dilihat pada lampiran 1 dan contoh perhitungan manual algoritma genetika dengan parameter dinamis dapat dilihat pada lampiran 2.

3.4 Implementasi Program Algoritma Genetika dengan Parameter Dinamis

Pengimplementasian algoritma genetika dengan parameter dinamis untuk masalah optimasi dengan menggunakan fungsi percobaan dibuat dengan MATLAB 7. Hasil dan waktu komputasi algoritma genetika dengan parameter dinamis dibandingkan dengan hasil dan waktu komputasi algoritma genetika yang biasa.

3.4.1 Penentuan Parameter

Parameter algoritma genetika sangat menentukan kinerja dari algoritma dalam menyelesaikan masalah optimasi. Sehingga penentuan nilai parameter harus dilakukan dengan hati-hati. Parameter yang digunakan pada algoritma genetika biasa adalah sebagai berikut :

- ukuran populasi : 40
- jumlah bit : 25
- P_c : 0.8
- P_m : 0.013
- jumlah generasi : 5000.

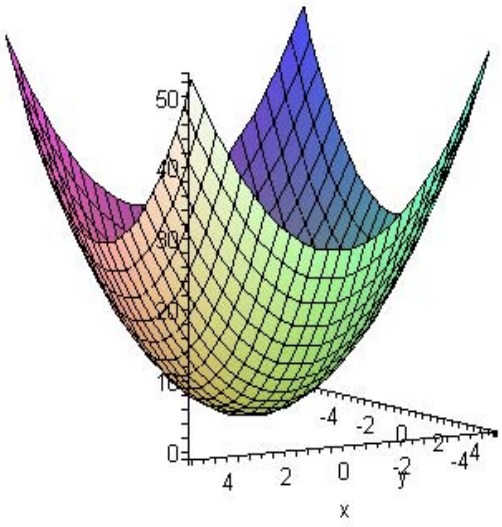
Pada algoritma genetika dengan parameter dinamis peluang pindah silang dan peluang mutasi dibuat dinamis menggunakan *fuzzy*, sehingga parameter yang ditetapkan diawal adalah ukuran populasi, jumlah bit dan jumlah generasi. Penetapan nilai parameter ukuran populasi didasarkan pada percobaan, yang mana percobaan dilakukan pada jumlah populasi 10-160 individu.

3.4.2 Keluaran Program

Karena algoritma genetika merupakan proses stokastik, maka dilakukan percobaan minimal lima kali pada setiap fungsi uji. Pada skripsi ini dilakukan dua puluh kali percobaan. Masing-masing fungsi uji dibatasi pada suatu sub ruang dari $\mathbb{R}^n$, yang mana pada masing-masing sub ruang tersebut terdapat titik minimum global dari fungsi. Fungsi-fungsi yang dipakai sebagai fungsi uji untuk menguji algoritma genetika adalah sebagai berikut.

1. $F_1(\boldsymbol{x}) = \sum_{i=1}^{3} x_i^2$

$F_1(\boldsymbol{x})$ merupakan fungsi yang kontinu, *convex*, unimodal dan merupakan fungsi kuadrat yang memiliki nilai minimum global $F_1(\boldsymbol{x}) = 0$ pada saat $(x_1, x_2, x_3) = (0,0,0)$. Fungsi $F_1(\boldsymbol{x})$ merupakan fungsi yang sederhana dan simetris, sehingga mudah di analisis, karena itu $F_1(\boldsymbol{x})$ dipilih sebagai fungsi uji yang pertama. Fungsi ini dibatasi pada ruang A yang digambarkan sebagai $-5.12 \leq x_i \leq 5.12, i = 1,2,3$. Fungsi $F_1(\boldsymbol{x})$ untuk dimensi tiga dapat dilihat pada Gambar 3.3.

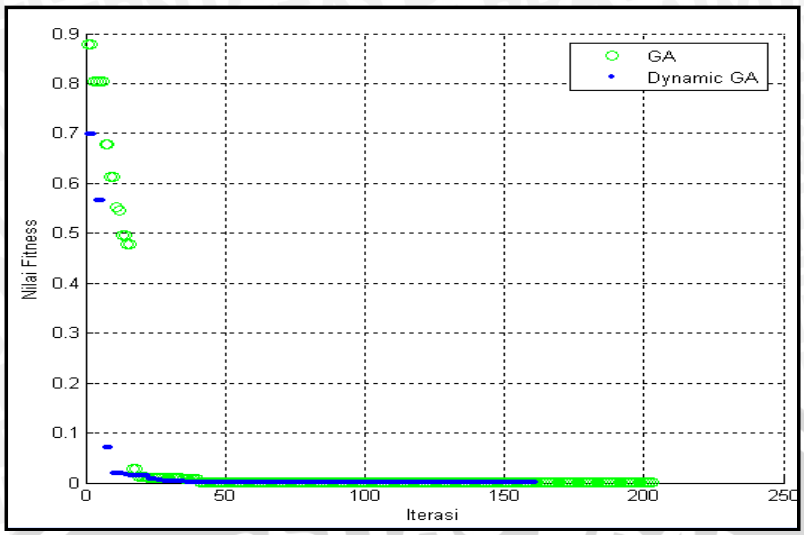


Gambar 3.3 Fungsi $F_1(\boldsymbol{x})$

Hasil rata-rata dan hasil terbaik untuk fungsi $F_1(\boldsymbol{x})$ dapat dilihat pada Tabel 3.4.

Tabel 3.4 Hasil uji algoritma menggunakan fungsi $F_1(\boldsymbol{x})$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
Waktu komputasi	5.273	9.758	3.141	7.704
Nilai fungsi	0	0	0	0



Gambar 3.4 Hasil minimasi fungsi $F_1(\boldsymbol{x})$

Pada fungsi uji $F_1(\boldsymbol{x})$ algoritma genetika biasa dan algoritma genetika dengan parameter dinamis sama-sama memperoleh solusi minimal sama dengan 0 baik untuk hasil rata-rata maupun untuk hasil terbaik. Akan tetapi algoritma genetika dengan parameter dinamis memerlukan waktu yang lebih lama untuk mendapatkan solusi minimal. Walaupun untuk mendapatkan solusi minimal tersebut algoritma genetika dengan parameter dinamis mencapai generasi yang lebih singkat atau sama dengan algoritma genetika biasa.

Proses iterasi algoritma genetika biasa dan algoritma genetika dengan parameter dinamis berhenti sebelum generasi mencapai 5000 karena solusi telah ditemukan. Nilai optimum fungsi yang dicapai algoritma genetika biasa dan algoritma genetika dengan parameter dinamis sama dengan nilai minimum global fungsi $F_1(\boldsymbol{x})$, demikian juga nilai variabel untuk masing-masing algoritma sama dengan titik minimum fungsi $F_1(\boldsymbol{x})$.

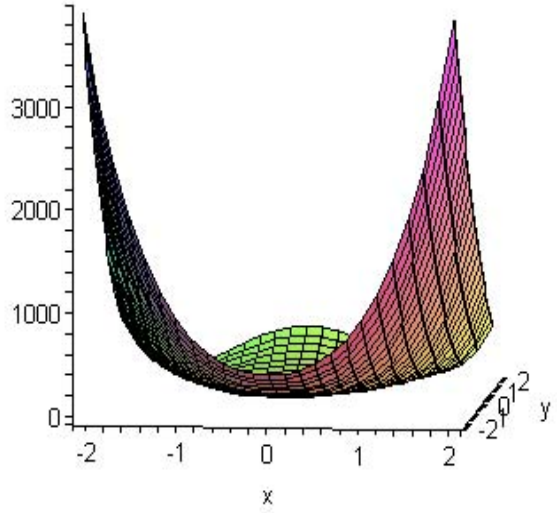
Nilai masing-masing variabel $\boldsymbol{x}_i$, untuk hasil terbaik algoritma genetika biasa dan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

Tabel 3.5 Nilai variabel $\boldsymbol{x_i}$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
x_1	0	0	0	0
x_2	0	0	0	0
x_3	0	0	0	0

2. $F_2(\boldsymbol{x}) = 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2$

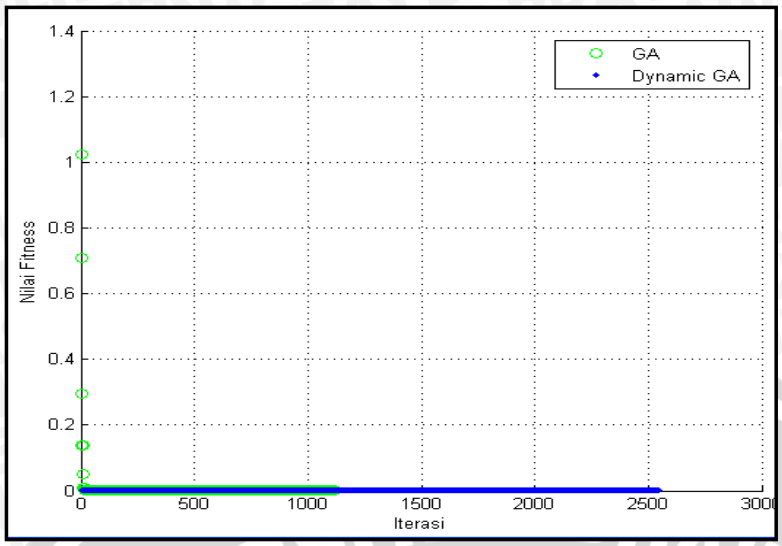
$F_2(\boldsymbol{x})$ merupakan fungsi kontinu, *nonconvex*, unimodal dan merupakan fungsi yang memiliki nilai minimum global $F_2(\boldsymbol{x}) = 0$ pada saat $(x_1, x_2) = (1,1)$. Fungsi $F_2(\boldsymbol{x})$ merupakan salah satu masalah minimasi fungsi yang sulit, karena memiliki titik sadel. Fungsi ini dibatasi pada ruang A yang digambarkan sebagai $-2.048 \leq x_i \leq 2.048,\ i = 1,2$. Fungsi $F_2(\boldsymbol{x})$ untuk dimensi tiga dapat dilihat pada Gambar 3.5.



Gambar 3.5 Fungsi $F_2(\boldsymbol{x})$

Tabel 3.6 Hasil uji algoritma menggunakan fungsi $F_2(\boldsymbol{x})$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
Waktu komputasi	50.086	121.979	16.031	43.938
Nilai fungsi	0.03387	0.00357	0	0



Gambar 3.6 Hasil minimasi fungsi $F_2(\boldsymbol{x})$

Pada fungsi uji $F_2(\boldsymbol{x})$ algoritma genetika biasa memperoleh waktu komputasi yang lebih singkat untuk hasil rata-rata maupun hasil terbaik. Untuk hasil terbaik, algoritma genetika biasa maupun algoritma genetika dengan parameter dinamis mencapai nilai optimum fungsi yang sama, yaitu sama dengan 0. Akan tetapi untuk hasil rata-rata algoritma genetika dengan parameter dinamis memperoleh nilai optimum fungsi yang lebih baik.

Proses iterasi algoritma genetika biasa dan algoritma genetika dengan parameter dinamis berhenti setelah solusi ditemukan untuk hasil terbaik. Sedangkan untuk hasil rata-rata proses iterasi berhenti karena tidak terjadi perubahan selama 1000 generasi. Nilai optimum fungsi yang dicapai algoritma genetika biasa dan algoritma genetika dengan parameter dinamis untuk hasil terbaik sama dengan nilai minimum global fungsi $F_2(\boldsymbol{x})$, demikian juga nilai variabel untuk masing-masing algoritma sama dengan titik minimum fungsi $F_2(\boldsymbol{x})$. Akan tetapi nilai optimum fungsi yang dicapai untuk hasil rata-rata masih jauh dari nilai minimum global fungsi $F_2(\boldsymbol{x})$. *Error* untuk nilai variabel yang dicapai algoritma genetika biasa adalah 0.00219 dan *error* untuk nilai variabel yang dicapai algoritma genetika dengan parameter dinamis adalah 0.00045.

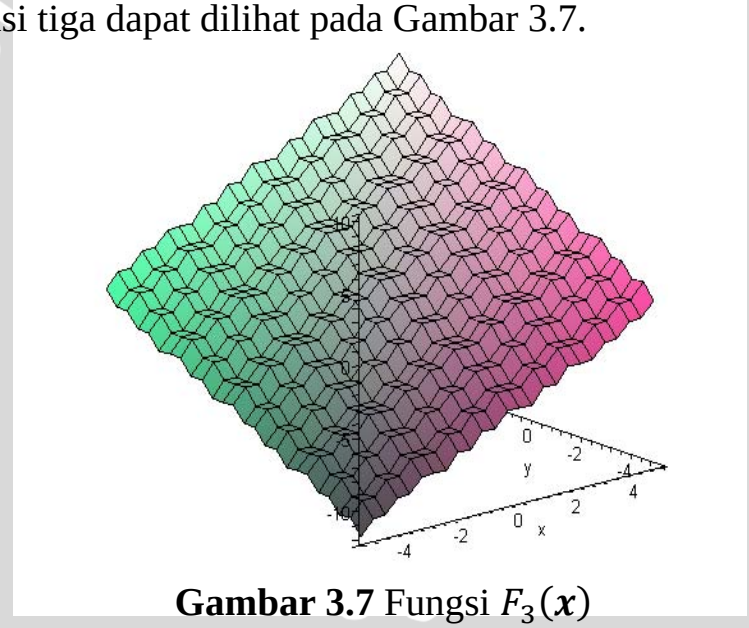
Nilai masing-masing variabel $\boldsymbol{x}_i$, untuk hasil terbaik dan hasil rata-rata algoritma genetika biasa dan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

Tabel 3.7 Nilai variabel x_i

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
x_1	0.95763	1.01187	1	1
x_2	0.9492	1.02738	1	1

3. $F_3(\boldsymbol{x}) = \sum_{i=1}^{5}[\boldsymbol{x}_i]$

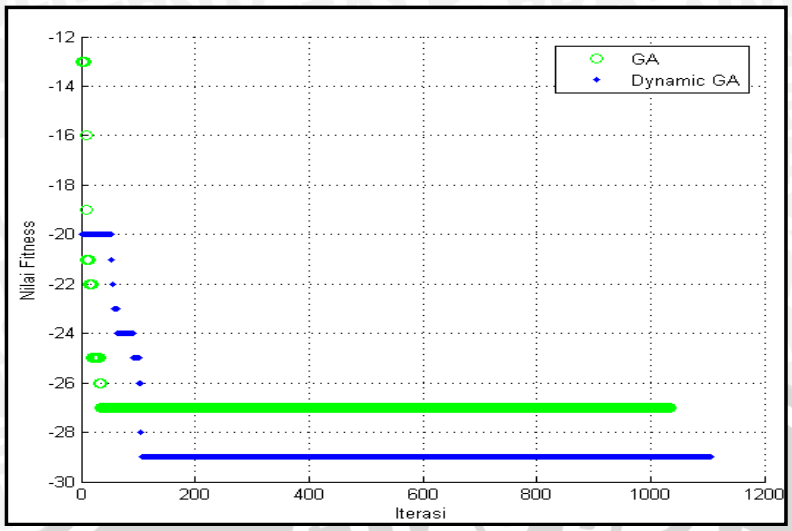
$F_3(\boldsymbol{x})$ merupakan fungsi yang tidak kontinu, yang mana $[\boldsymbol{x}_i]$ mewakili integer terbesar yang kurang dari sama dengan $\boldsymbol{x}_i$. Fungsi $F_3(\boldsymbol{x})$ dipilih sebagai fungsi uji yang ketiga untuk melihat kinerja algoritma genetika pada fungsi yang tidak kontinu. Fungsi ini dibatasi pada ruang A yang digambarkan sebagai $-5.12 \leq \boldsymbol{x}_i \leq 5.12, i = 1,2,\ldots,5$. $F_3(\boldsymbol{x})$ memiliki nilai minimum global $F_3(\boldsymbol{x}) = -30$ pada saat $(x_1, x_2, \ldots, x_5) = (-5.12, -5.12, \ldots, -5{,}12)$, $i = 1,2,\ldots,5$. Fungsi $F_3(\boldsymbol{x})$ untuk dimensi tiga dapat dilihat pada Gambar 3.7.



Gambar 3.7 Fungsi $F_3(\boldsymbol{x})$

Tabel 3.8 Hasil uji algoritma menggunakan fungsi $F_3(\boldsymbol{x})$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
Waktu komputasi	38.259	70.877	13.437	14.016
Nilai fungsi	-27.7	-28.35	-30	-30



Gambar 3.8 Hasil minimasi fungsi $F_3(\boldsymbol{x})$

Pada fungsi $F_3(\boldsymbol{x})$ algoritma genetika biasa memperoleh waktu komputasi yang lebih singkat jika dibandingkan dengan algoritma genetika dengan parameter dinamis baik untuk hasil rata-rata maupun hasil terbaik. Nilai optimum fungsi yang didapat adalah sama untuk hasil terbaik. Akan tetapi algoritma genetika dengan parameter dinamis mendapat nilai optimum fungsi yang lebih baik untuk hasil rata-rata.

Untuk hasil terbaik, proses iterasi algoritma genetika biasa dan algoritma genetika dengan parameter dinamis berhenti setelah solusi ditemukan. Sedangkan untuk hasil rata-rata proses iterasi berhenti karena tidak terjadi perubahan selama 1000 generasi. Nilai optimum fungsi yang dicapai algoritma genetika biasa dan algoritma genetika dengan parameter dinamis untuk hasil terbaik sama dengan nilai minimum global fungsi $F_3(\boldsymbol{x})$, meskipun nilai variabel untuk masing-masing algoritma tidak sama dengan titik minimum fungsi $F_3(\boldsymbol{x})$. Untuk hasil rata-rata, *error* nilai variabel yang dicapai algoritma genetika biasa adalah 0.14021 dan *error* nilai variabel yang dicapai algoritma genetika dengan parameter dinamis adalah 0.08087.

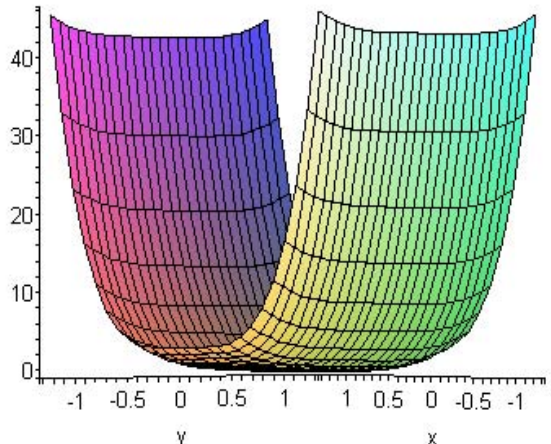
Nilai masing-masing variabel $\boldsymbol{x}_i$, untuk hasil terbaik dan hasil rata-rata algoritma genetika biasa dan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

Tabel 3.9 Nilai variabel $\boldsymbol{x_i}$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
x_1	-4.85078	-4.77621	-5.1031	-5.04836
x_2	-4.62154	-4.81468	-5.0117	-5.0238
x_3	-4.78801	-4.85536	-5.03291	-5.06284
x_4	-4.72541	-4.87611	-5.04643	-5.01163
x_5	-4.78211	-4.86821	-5.06686	-5.00327

4. $F_4(\boldsymbol{x}) = \sum_{i=1}^{30} i x_i^4$

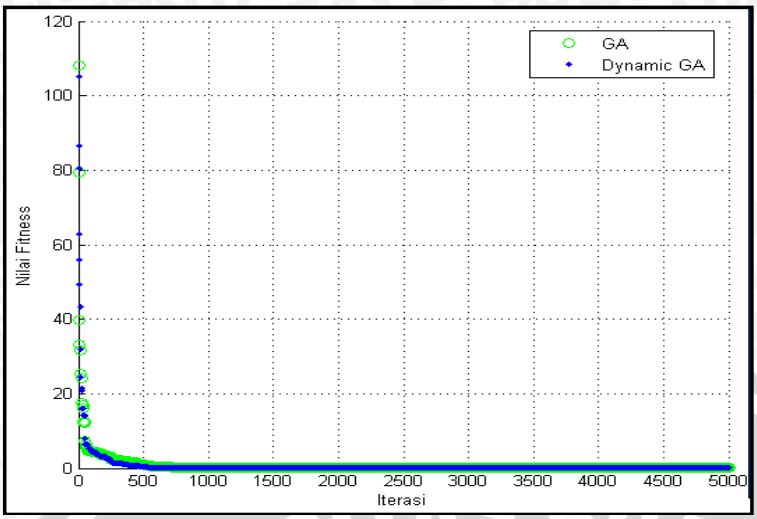
$F_4(\boldsymbol{x})$ merupakan fungsi kontinu, *convex*, unimodal dan merupakan fungsi berdimensi tinggi yang memiliki nilai minimum global $F_4(\boldsymbol{x}) = 0$ pada saat $(x_1, \dots, x_{30}) = (0, \dots, 0)$. Fungsi $F_4(\boldsymbol{x})$ dipilih sebagai fungsi uji yang keempat untuk melihat kinerja algoritma genetika pada fungsi berdimansi tinggi. Fungsi ini dibatasi pada ruang A yang digambarkan sebagai $-1.28 \leq x_i \leq 1.28, i = 1,2, \dots,30$. Fungsi $F_4(\boldsymbol{x})$ untuk dimensi tiga dapat dilihat pada Gambar 3.9.



Gambar 3.9 Fungsi $F_4(\boldsymbol{x})$

Tabel 3.10 Hasil uji algoritma menggunakan fungsi $F_4(\boldsymbol{x})$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
Waktu komputasi	452.783	566.441	455.562	564.39
Nilai fungsi	9.05×10^{-9}	2.08×10^{-10}	1.2×10^{-10}	6.07×10^{-12}



Gambar 3.10 Hasil minimasi fungsi $F_4(\boldsymbol{x})$

Pada fungsi $F_4(\boldsymbol{x})$ algoritma genetika biasa memperoleh waktu komputasi yang lebih singkat untuk hasil rata-rata maupun hasil terbaik. Sedangkan algoritma genetika dengan parameter dinamis memperoleh nilai optimum fungsi yang lebih kecil untuk hasil rata-rata maupun untuk hasil terbaik.

Proses iterasi algoritma genetika biasa dan algoritma genetika dengan parameter dinamis berhenti setelah generasi mencapai 5000. Nilai optimum fungsi yang dicapai algoritma genetika biasa dan algoritma genetika dengan parameter dinamis belum mencapai nilai minimum global fungsi $F_4(\boldsymbol{x})$ akan tetapi perbaikan nilai optimum fungsi terus dilakukan hingga akhir generasi. Untuk hasil rata-rata, *error* nilai variabel yang dicapai algoritma genetika biasa adalah 9.8×10^{-8} dan *error* nilai variabel yang dicapai algoritma genetika dengan parameter dinamis adalah 9.7×10^{-9}. Untuk hasil terbaik *error* nilai variabel yang dicapai algoritma genetika biasa adalah 3.2×10^{-7} dan *error* nilai variabel yang dicapai algoritma genetika dengan parameter dinamis adalah 6.2×10^{-8}.

Nilai masing-masing variabel $\boldsymbol{x}_i$, untuk hasil terbaik dan hasil rata-rata algoritma genetika biasa dan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

Tabel 3.11 Nilai variabel x_i

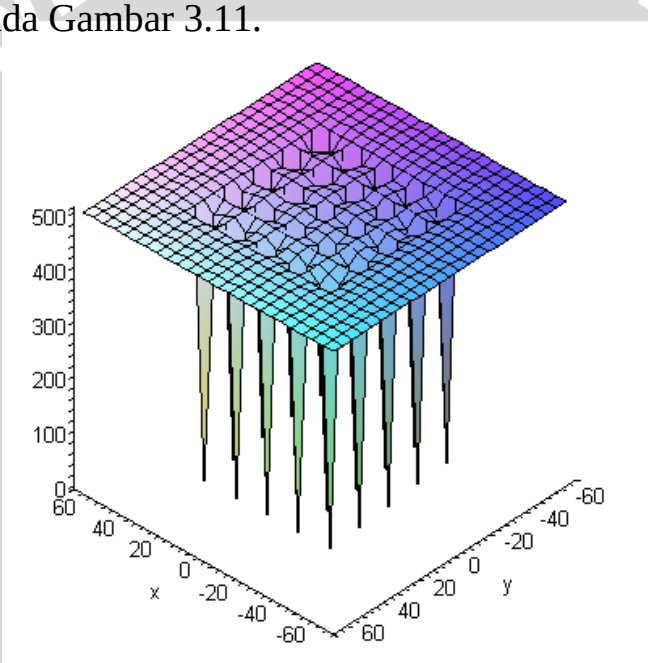
	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
x_1	-0.00013	-0.00002	0.00138	-0.00021
x_2	-0.00024	0.00013	-0.00003	-0.00002
x_3	-0.00009	0.00024	-0.00008	0.00047
x_4	0.00005	-0.00004	0.00097	-0.00032
x_5	0.00027	0.00008	0.00039	0.00011
x_6	-0.00019	-0.00006	-0.00063	0.0004
x_7	0.00064	0.00006	-0.00004	0
x_8	-0.00047	0.0001	0.00067	0.00002
x_9	0.00015	-0.00013	-0.0018	-0.00083
x_{10}	-0.00047	0.00006	-0.00053	0.00003
x_{11}	0.00024	-0.00011	-0.00071	-0.0002
x_{12}	-0.00013	-0.0001	-0.00035	0.00005
x_{13}	0.00056	0.00006	-0.00007	0
x_{14}	0.00031	-0.00004	-0.00073	0.00049
x_{15}	0.0003	-0.00019	-0.00005	0.00006
x_{16}	-0.00016	-0.00006	-0.00081	0.00016
x_{17}	-0.00002	-0.00001	-0.00018	0.00006
x_{18}	0.00065	-0.00015	0.00002	-0.00007
x_{19}	0.00053	0.00005	0.00002	-0.00002
x_{20}	0.00019	-0.00013	-0.00003	-0.0002
x_{21}	-0.00012	0.00014	-0.00033	0.0001
x_{22}	-0.00007	-0.00004	0.00001	-0.00008
x_{23}	0.00006	0.00009	0.00017	0.00023
x_{24}	0.00005	0.00007	0.00009	0.00008
x_{25}	-0.00008	-0.00003	0.00017	0.00004
x_{26}	0.00031	0.00009	-0.00005	0.00023
x_{27}	0.00006	-0.00003	-0.00003	-0.00011
x_{28}	-0.00014	-0.00003	0.00006	-0.00004
x_{29}	0.00057	0.00013	-0.00033	0.00035
x_{30}	0.00032	-0.00003	0.00022	0.00008

5. $\frac{1}{F_5(\boldsymbol{x})} = \frac{1}{K} + \sum_{j=1}^{25} \frac{1}{f_j(\boldsymbol{x})}$ dengan $f_j(\boldsymbol{x}) = c_j + \sum_{i=1}^{2} (\boldsymbol{x}_i + \boldsymbol{a}_{ij})^6$

$F_5(\boldsymbol{x})$ merupakan fungsi kontinu, *nonconvex*, multimodal dan merupakan fungsi yang memiliki 25 minimum lokal. Fungsi $F_5(\boldsymbol{x})$ dipilih sebagai fungsi uji yang kelima untuk melihat kinerja algoritma genetika pada fungsi yang memiliki banyak minimum local. Fungsi ini dibatasi pada ruang A yang digambarkan sebagai $-65.536 \leq x_i \leq 65.536, i = 1,2, \ldots ,30$. Dalam skripsi ini nilai $\boldsymbol{a}_{ij}$ yang dipergunakan adalah

$$[\boldsymbol{a}_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \ldots & 32 & 32 & 32 \end{bmatrix}$$

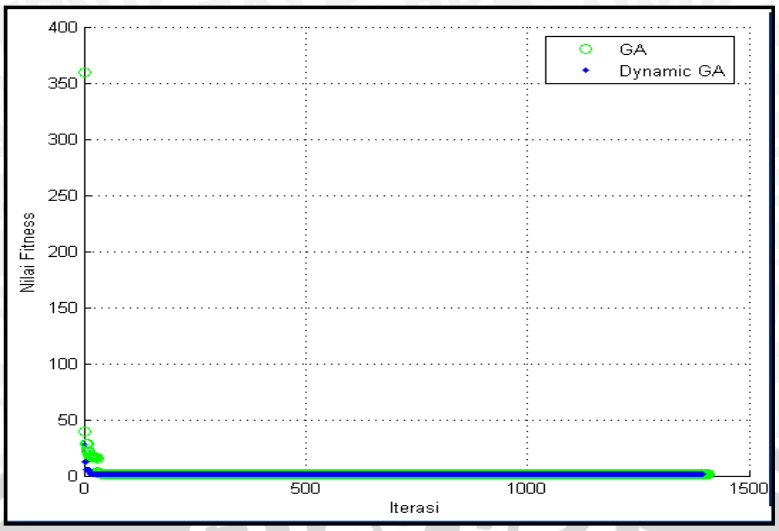
dengan nilai $c_j = j$ dan $K = 500$. Fungsi ini memiliki nilai minimum global $F_5(\boldsymbol{x}) \cong 1$. Fungsi $F_5(\boldsymbol{x})$ untuk dimensi tiga dapat dilihat pada Gambar 3.11.



Gambar 3.11 Fungsi $F_5(\boldsymbol{x})$

Tabel 3.12 Hasil uji algoritma menggunakan fungsi $F_5(\boldsymbol{x})$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
Waktu komputasi	108.712	211.825	78.14	122.14
Nilai fungsi	0.998	0.998	0.998	0.998



Gambar 3.12 Hasil minimasi fungsi $F_5(\boldsymbol{x})$

Pada fungsi $F_5(\boldsymbol{x})$ algoritma genetika biasa dan algoritma genetika dengan parameter dinamis mendapatkan hasil yang sama untuk nilai optimum fungsi, baik untuk hasil rata-rata maupun untuk hasil terbaik. Akan tetapi algoritma genetika biasa memperoleh waktu komputasi yang lebih singkat.

Untuk hasil terbaik maupun untuk hasil rata-rata proses iterasi algoritma genetika biasa dan algoritma genetika dengan parameter dinamis berhenti karena tidak terjadi perubahan selama 1000 generasi.

Nilai masing-masing variabel $\boldsymbol{x}_i$, untuk hasil rata-rata dan hasil terbaik algoritma genetika biasa dan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

Tabel 3.13 Nilai variabel $\boldsymbol{x}_i$

	Hasil rata-rata		Hasil terbaik	
	GA	GA dinamis	GA	GA dinamis
x_1	-31.984	-31.984	-31.984	-31.9857
x_2	-31.9783	-31.9783	-31.9783	-31.9783

repository.ub.ac

UNIVERSITAS BRAWIJAYA

BAB IV
KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan hasil dan pembahasan yang diperoleh, terdapat beberapa kesimpulan yang dapat diambil, yaitu

1. parameter algoritma genetika dengan parameter dinamis dikontrol menggunakan *Fuzzy Logic Controller* (*FLC*). *FLC* digunakan untuk mengontrol parameter peluang pindah silang dan peluang mutasi. Variabel masukan *FLC* adalah *BF*, *AF*, dan *CBF*, dengan variabel keluaran P_c dan P_m.
2. kinerja algoritma genetika dengan parameter dinamis lebih baik untuk masalah optimasi dimensi tinggi, karena algoritma genetika dengan parameter dinamis mendapatkan nilai fungsi yang lebih mendekati nilai eksak jika dibandingkan dengan algoritma genetika biasa.

4.2 Saran

Saran yang dapat diberikan untuk pengembangan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

1. Algoritma genetika dengan parameter dinamis diselidiki lebih lanjut untuk meningkatkan efisiensi algoritma, misalnya dengan menyelidiki bentuk fungsi keanggotaan dan basis aturan yang digunakan pada *Fuzzy Logic Controller* atau dengan menyelidiki masukan.
2. Parameter dari algoritma genetika yang dijadikan parameter dinamis tidak hanya peluang mutasi dan peluang pindah silang, bisa ditambah dengan ukuran populasi dan jumlah generasi.

repository.ub.ac

UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Bergh, V. D. 2006. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria. South Africa.

Bronson, R. 1996. *Teori dan Soal-Soal Operation Research, Alih Bahasa : Hans J. Wospakrik*. Erlangga. Jakarta.

Engelbrecht, A. P. 2007. *Computational Intelligence : An Introduction*. John Wiley & Sons, Ltd. Prentice-Hall, Inc. England.

Gen, M. dan R. Cheng. 2000. *Genetic Algorithms & Engineering Optimization*. John Wiley & Sons, Inc. Canada.

Jang, R., C. Sun dan E. Mizutani. 1997. *Neuro-Fuzzy and Soft Computing: A Computional Approach to Learning and Machine Intelligence*. Prentice-Hall, Inc. USA.

Luknanto, D. 2000. *Pengantar Optimasi Nonlinier*. Jurusan Teknik Sipil Fakultas Teknik Universitas Gajah Mada.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag Berlin Heidelberg New York. USA.

Mitchell, M. 1999. *An Introduction to Genetic Algorithms*. Massachusetts Institute of Technology. England.

Nocedal, J dan S. J. Wright. 2006. *Numerical Optimization*. Springer Science+Business Media, LLC. New York.

Obitko, M. 1998. *Introduction to Genetic Algorithm*, http://cs.felk.cvut.cz/~xobitko/ga/. Tanggal akses: 23 Februari 2010.

Pedrycz, W dan F. Gomide. 2007. *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley & Sons, Inc. Hoboken, New Jersey.

Suyanto. 2005. *Algoritma Genetika dalam Matlab*. Andi.Yogyakarta.

Wang, L. 1997. *A Course in Fuzzy Systems and Control*. Prentice-Hall, Inc. USA.

Weise, T. 2007. *Global Optimization Algorithms : Theory and Application.* http://www.it-weise.de/projects/book.pdf. Tanggal akses: 30 Oktober 2010.

Lampiran 1
Pseudocode

Input : Jumlah_bit, Jumlah_variabel, Ukuran_populasi, Jumlah_generasi, Batas_atas, Batas_bawah

1. Inisialisasi populasi
   for $i \leftarrow 1$ to Ukuran_populasi do
     for $j \leftarrow 1$ to Jumlah_variabel do
       for k $\leftarrow 1$ to Jumlah_bit do
         $\text{populasi}_{i,j,k} \leftarrow$ random
2. while generasi < Jumlah_generasi
3. generasi $\leftarrow$ generasi+1
4. dekodekan kromosom
   $i \leftarrow 1$
   for $j \leftarrow 1$ to Jumlah_variabel do
     $x_{i,j} \leftarrow 0$
     for k $\leftarrow 1$ to Jumlah_bit do
       $x_{i,j} \leftarrow x_{i,j} + \text{populasi}_{i,j,k} * 2^{-k}$
     $x_{i,j} \leftarrow$ Batas_bawah+(Batas_atas−Batas_bawah)$* x_{i,j}$
5. Bestx $\leftarrow x_{i,j}$
6. $\text{Fitness}_1 \leftarrow f(x_{i,j})$
7. Max_fitness $\leftarrow \text{Fitness}_1$
8. Min_fitness $\leftarrow \text{Fitness}_1$
9. Indeks_individu_terbaik $\leftarrow 1$
10. Jumlah_fitness $\leftarrow \text{Fitness}_1$
11. Dekodekan kromosom
    for $i \leftarrow 2$ to Ukuran_populasi do
      for $j \leftarrow 1$ to Jumlah_variabel do
        $x_{i,j} \leftarrow 0$
        for k $\leftarrow 1$ to Jumlah_bit do
          $x_{i,j} \leftarrow x_{i,j} + \text{populasi}_{i,j,k} * 2^{-k}$
        $x_{i,j} \leftarrow \text{Batas\_bawah} + (\text{Batas\_atas} - \text{Batas\_bawah}) * x_{i,j}$
12. $\text{Fitness}_i \leftarrow f(x_{i,j})$
13. if $\text{Fitness}_i <$ Min_fitness
14. Min_fitness $\leftarrow \text{Fitness}_i$
15. Indeks_individu_terbaik $\leftarrow i$
16. Bestx $\leftarrow x_{i,j}$
17. if $\text{Fitness}_i >$ Max_fitness

Max_fitness ← $Fitness_i$
Jumlah_fitness ← Jumlah_fitness+ $Fitness_i$

18. Average_fitness ←Jumlah_fitness/Ukuran_populasi
19. BF ← Min_fitness / Average_fitness
20. AF ← Average_fitness / Max_fitness
21. CBF← Min_fitness (generasi) − Min_fitness (generasi−1)
22. Fismat ← readfis('fuzzyrule1.fis')
23. B ← evalfis([BF AF CBF],fismat)
24. Peluang_mutasi ← B_1
25. Peluang_pindahsilang ← B_2
26. Temporary_populasi ← populasi
27. if Ukuran_populasi mod 2 ← 0
    Iterasi_mulai← 3
    $Temporary\_populasi_1 = populasi_{indeks\_individu\_terbaik}$
    $Temporary\_populasi_2 = populasi_{indeks\_individu\_terbaik}$
    else
    Iterasi_mulai← 2
    $Temporary\_populasi_1 = populasi_{indeks\_individu\_terbaik}$
28. linear fitness ranking
    for $ii \leftarrow 1$ to Ukuran_populasi
    $FitnessR_{ii} \leftarrow Max\_fitness - (Max\_fitness - Min\_fitness) * \left(\frac{ii-1}{Ukuran\_populasi-1}\right)$
29. Jumlah_fitnessR← 0
30. for $ii \leftarrow 1$ to Ukuran_populasi
    Jumlah_fitnessR←Jumlah_fitnessR+$FitnessR_{ii}$
31. Roulette wheel
    for $ii \leftarrow 1$ to Ukuran_populasi
    $Probabilitas\_seleksi_{ii} \leftarrow \frac{FitnessR_{ii}}{Jumlah\_fitnessR}$
    $Probabilitas\_kumulatif_1 \leftarrow Probabilitas\_seleksi_1$
    for $jj \leftarrow 2$ to ii
    $Probabilitas\_kumulatif_{ii} \leftarrow Probabilitas\_kumulatif_{ii-1} + Probabilitas\_seleksi_{jj}$
    $r \leftarrow rand$
    if $r <$ $Probabilitas\_kumulatif_{ii}$
    induk ← $populasi_{ii}$
32. Pindah silang
    for $kk \leftarrow$ Iterasi_mulai to Ukuran_populasi

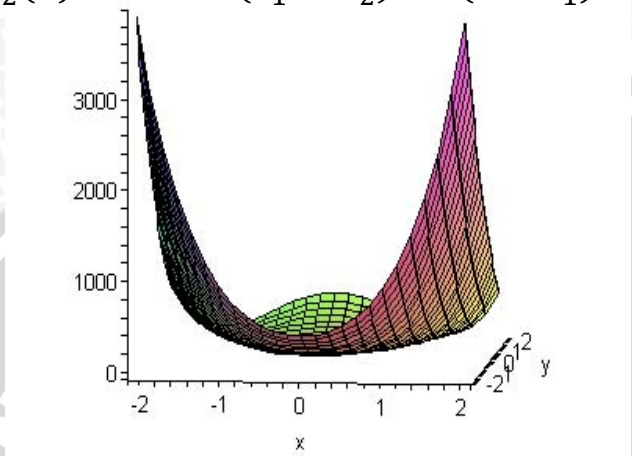
```
    r ← rand
    if r < Peluang_pindahsilang
      Titik_potong ← random [1,Jumlah_bit * Jumlah_variabel−1]
      Keturunan1←induk1(1:Titik_potong)  induk2(Titik_potong+1:
                  Jumlah_bit*Jumlah_variabel)
      Keturunan2←induk2(1:Titik_potong)  induk1(Titik_potong+1:
                  Jumlah_bit*Jumlah_variabel)
      Temporary_populasikk←Keturunan1
      Temporary_populasikk←Keturunan2
    else
      Temporary_populasikk←induk1
      Temporary_populasikk←induk2
33. Mutasi
    r ← rand
    for i ← 1 to Ukuran_populasi
      for j ← 1 to Jumlah_variabel
        for k ← 1 to Jumlah_bit
          if r < Peluang_mutasi
            if Temporary_populasi_{i,j,k} ← 0
              populasi_mutasi_{i,j,k} ← 1
            else
              populasi_mutasi_{i,j,k} ← 0
34. populasi ← populasi_mutasi
```

Lampiran 2

Contoh masalah optimasi yang diselesaikan menggunakan algoritma genetika dengan parameter dinamis secara manual

Misalkan masalah optimasi yang diselesaikan adalah meminimumkan fungsi uji $F_2(\boldsymbol{x})$, yaitu:

$$F_2(\boldsymbol{x}) = 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2$$



Fungsi ini memiliki nilai minimum global di titik $(1,1)$ dengan nilai fungsi $f = 0$.

Langkah-langkah yang dilakukan untuk mengimplementasikan algoritma genetika dengan parameter dinamis adalah sebagai berikut.

1. Menentukan nilai parameter

   Parameter algoritma genetika dengan parameter dinamis yang digunakan yaitu:

   - Ukuran populasi : 20
   - Jumlah bit : 10
   - Jumlah generasi : 100

2. Inisialisasi populasi

   Dibentuk suatu populasi yang terdiri dari 20 individu, yang mana masing-masing individu terdiri dari 2 variabel, yaitu x_1 dan x_2, sehingga dalam populasi terdapat 40 kromosom dan 400 gen. Yang mana masing-masing gen bernilai biner.

i	Individu$_i$																			
	Kromosom$_1$										Kromosom$_2$									
1	0	0	1	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0	1	0	1	1	0	1	0	1	1	1
3	1	1	0	1	1	1	0	1	1	0	1	0	1	1	1	1	1	0	1	0
4	0	0	1	0	1	1	1	1	1	1	1	0	0	0	1	0	0	0	1	0

i	Individu$_i$																			
	Kromosom$_1$										Kromosom$_2$									
5	0	1	0	0	0	1	1	0	1	0	0	1	0	1	0	1	1	1	1	1
6	1	0	1	0	0	0	1	0	1	1	0	1	1	1	1	1	1	0	1	1
7	1	0	1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0	0
8	1	0	1	1	0	1	0	1	0	0	0	1	1	1	1	1	0	0	1	0
9	1	0	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0	1	1	0
10	0	1	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	0	0	1
11	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	0
12	0	1	1	0	1	1	0	1	1	0	0	1	0	1	0	0	1	1	0	1
13	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0
14	0	1	1	0	1	1	0	0	1	0	0	0	1	0	0	0	1	1	0	1
15	1	0	1	0	1	1	1	0	1	0	1	0	1	0	1	0	1	1	1	0
16	1	1	0	0	0	0	0	1	0	1	1	1	1	1	1	1	0	0	0	1
17	0	0	1	1	1	0	0	0	0	0	1	0	0	1	1	0	1	0	1	0
18	0	0	0	0	1	1	0	0	1	1	0	1	0	1	1	1	1	1	1	1
19	1	1	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1
20	0	1	0	1	0	1	1	0	1	1	0	1	0	0	1	0	1	1	1	1

3. Dekode kromosom

   Nilai kromosom yang dibuat pada saat inisialisasi populasi adalah biner, sehingga perlu dilakukan dekode kromosom, yaitu mengubah nilai kromosom menjadi real. Contoh penghitungan dekode kromosom untuk individu pertama.

$$x_{i,j} = r_b + (r_a - r_b)\left(g_{i,j,1}.2^{-1} + g_{i,j,2}.2^{-2} + \cdots + g_{i,j,n}.2^{-n}\right)$$

$$x_{1,1} = -2.048 + \left(2.048 - 2.048\right)\left(g_{1,1,1}.2^{-1} + g_{1,1,2}.2^{-2} + \cdots + g_{1,1,10}.2^{-10}\right)$$

$$x_{1,1} = -2.048 + \left(2.048 - 2.048\right)(0.2^{-1} + 0.2^{-2} + 1.2^{-3} + 1.2^{-4} + 1.2^{-5} + 0.2^{-6} + 1.2^{-7} + 0.2^{-8} + 1.2^{-9} + 0.2^{-10})$$

$$x_{1,1} = -2.048 + \left(2.048 - 2.048\right)(0.228516)$$

$$x_{1,1} = -1.112$$

$$x_{1,2} = -2.048 + \left(2.048 - 2.048\right)\left(g_{1,2,1}.2^{-1} + g_{1,2,2}.2^{-2} + \cdots + g_{1,2,10}.2^{-10}\right)$$

$$x_{1,2} = -2.048 + (2.048 - 2.048)(0.2^{-1} + 1.2^{-2} + 0.2^{-3} + 0.2^{-4} + 0.2^{-5} + 0.2^{-6} + 0.2^{-7} + 0.2^{-8} + 0.2^{-9} + 0.2^{-10})$$

$$x_{1,2} = -2.048 + (2.048 - 2.048)(0.25)$$

$$x_{1,2} = -1.024$$

Nilai x_1 dan x_2 setelah mengalami proses dekode kromosom adalah sebagai berikut.

	x_1	x_2
1	-1.112	-1.024
2	0.256	0.86
3	1.496	1
4	-1.284	0.136
5	-0.92	-0.644
6	0.556	-0.02
7	0.544	1.328
8	0.848	-0.056
9	1.016	1.304
10	-0.62	-0.7
11	-0.296	1.456
12	-0.296	-0.716
13	-1.496	-1.92
14	-0.312	-1.484
15	0.744	0.696
16	1.044	1.988
17	-1.152	0.424
18	-1.844	-0.516
19	1.9	1.532
20	-0.66	-0.836

4. Menghitung nilai *fitness* dari masing-masing individu, yaitu dengan memasukkan nilai x_1 dan x_2 yang didapat dari proses dekode kromosom ke dalam fungsi F_2.

$$F_2(x) = 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2$$

$$F_2(x) = 100 \times \left(-1.112^2 - (-1.024)\right)^2 + \left(1 - (-1.112)\right)^2$$

$$F_2(x) = 515.4665$$

Individu	Nilai *fitness*
1	515.4665
2	63.6708
3	153.5144
4	234.0295
5	225.8156
6	11.0302
7	106.7235
8	60.1017
9	7.3847
10	120.2167
11	188.9271
12	66.2595
13	1735.1
14	251.8
15	2.0951
16	80.7
17	86.2
18	1541.9
19	432.6
20	164.5

5. Menentukan nilai *fitness* minimum dan maksimum dari masing-masing individu.

Fitness minimum	2.0951
Fitness maksimum	1735.1

6. Menentukan peluang pindah silang dan peluang mutasi menggunakan logika *fuzzy*. Langkah-langkahnya adalah sebagai berikut.
   a) Menentukan masukan
      Masukan yang digunakan adalah fitness minimum/rata-rata fitness (*BF*), rata-rata fitness/fitness maksimum (*AF*), dan perubahan nilai fitness(*CBF*).

$$\text{Rata-rata fitness} = \frac{\text{jumlah fitness}}{\text{ukuran populasi}}$$

$$\text{Rata-rata fitness} = \frac{6048.0353}{20}$$

$$\text{Rata-rata fitness} = 302.40177$$

$$BF = 0.0069$$

$$AF = 0.1743$$

b) Melakukan fuzzifikasi masukan
Perhitungan fuzzifikasi masukan dilakukan pada semua variabel masukan *fuzzy*.

- $BF = 0.0069$

$\mu_{small}(BF, 0,0.1,0.2) = \max\left(\min\left(\frac{0.0069-0}{0.1-0}, \frac{0.2-0.0069}{0.2-0.1}\right), 0\right)$
$\mu_{small}(BF, 0,0.1,0.2) = \max(\min(0.069,1.931), 0)$
$\mu_{small}(BF, 0,0.1,0.2) = 0.069$
$\mu_{medium}(BF, 0.1,0.2,0.4) = \max\left(\min\left(\frac{0.0069-0.1}{0.2-0.1}, \frac{0.4-0.0069}{0.4-0.2}\right), 0\right)$
$\mu_{medium}(BF, 0.1,0.2,0.4) = \max(\min(-0.931,1.9655), 0)$
$\mu_{medium}(BF, 0.1,0.2,0.4) = 0$
$\mu_{big}(BF, 0.2,0.4,1) = \max\left(\min\left(\frac{0.0069-0.2}{0.4-0.2}, \frac{1-0.0069}{1-0.4}\right), 0\right)$
$\mu_{big}(BF, 0.2,0.4,1) = \max(\min(-0.9655,1.65517), 0)$
$\mu_{big}(BF, 0.2,0.4,1) = 0$

- $AF = 0.1743$

$\mu_{small}(AF, 0,0,0.1,0.2) = \max\left(\min\left(\frac{0.1743-0}{0-0}, 1, \frac{0.2-0.1743}{0.2-0.1}\right), 0\right)$
$\mu_{small}(AF, 0,0,0.1,0.2) = \max(\min(\infty, 1,0.257), 0)$
$\mu_{small}(AF, 0,0,0.1,0.2) = 0.257$
$\mu_{medium}(AF, 0.1,0.2,0.8) = \max\left(\min\left(\frac{0.1743-0.1}{0.2-0.1}, \frac{0.8-0.1743}{0.8-0.2}\right), 0\right)$
$\mu_{medium}(AF, 0.1,0.2,0.8) = \max(\min(0.743,1.04283), 0)$
$\mu_{medium}(AF, 0.1,0.2,0.8) = 0.743$
$\mu_{big}(AF, 0,0.0013,0.008,0.013) = \max\left(\min\left(\frac{0.1743-0.0013}{0.0013-0}, 1, \frac{0.013-0.1743}{0.013-0.008}\right), 0\right)$
$\mu_{big}(AF, 0,0.0013,0.008,0.013) = \max(\min(133.0769,1, -32.26), 0)$
$\mu_{big}(AF, 0.15,0.85,1) = 1$

- $CBF = 0$

$\mu_{small}(CBF, 0,0,0.5) = \max\left(\min\left(\frac{0-0}{0-0}, \frac{0.5-0}{0.5-0}\right), 0\right)$
$\mu_{small}(CBF, 0,0,0.5) = \max(\min(\infty, 1), 0)$
$\mu_{small}(CBF, 0,0,0.5) = 1$
$\mu_{medium}(CBF, 0,0.5,1) = \max\left(\min\left(\frac{0-0}{0.5-0}, \frac{1-0}{1-0.5}\right), 0\right)$
$\mu_{medium}(CBF, 0,0.5,1) = \max(\min(0,2), 0)$
$\mu_{medium}(CBF, 0,0.5,1) = 0$
$\mu_{big}(CBF, 0.5,1,20) = \max\left(\min\left(\frac{0-0.5}{1-0.5}, \frac{20-0}{20-1}\right), 0\right)$
$\mu_{big}(CBF, 0.5,1,20) = \max(\min(-1,1.0526), 0)$
$\mu_{big}(CBF, 0.5,1,20) = 0$

c) Menerapkan basis aturan

Aturan *fuzzy* untuk peluang pindah silang

- Jika *BF* adalah *small*, *AF* adalah *small* dan *CBF* adalah *small* maka P_c adalah *small*

  $\mu_{small}(z) = min(0.069,0.257,1) = 0.069$

  Kemudian dicari nilai z berdasarkan nilai himpunan *small* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.7}{0.7-0.7}, \frac{0.8-z}{0.8-0.7}\right), 0\right) = 0.069$$

  $$max\left(min\left(\infty, \frac{0.8-z}{0.1}\right), 0\right) = 0.069$$

  diperoleh $z = 0.7931$.

- Jika *BF* adalah *small*, *AF* adalah *big* dan *CBF* adalah *medium* maka P_c adalah *small*

  $\mu_{small}(z) = min(0.069,1,0) = 0$

  Kemudian dicari nilai z berdasarkan nilai himpunan *small* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.7}{0.7-0.7}, \frac{0.8-z}{0.8-0.7}\right), 0\right) = 0$$

  $$max\left(min\left(\infty, \frac{0.8-z}{0.1}\right), 0\right) = 0$$

  diperoleh $z = 0.8$.

- Jika *BF* adalah *small*, *AF* adalah *big* dan *CBF* adalah *big* maka P_c adalah *small*

  $\mu_{small}(z) = min(0.069,1,0) = 0$

  Kemudian dicari nilai z berdasarkan nilai himpunan *small* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.7}{0.7-0.7}, \frac{0.8-z}{0.8-0.7}\right), 0\right) = 0$$

  $$max\left(min\left(\infty, \frac{0.8-z}{0.1}\right), 0\right) = 0$$

  diperoleh $z = 0.8$.

- Jika *BF* adalah *medium*, *AF* adalah *small* dan *CBF* adalah *small* maka P_c adalah *medium*

  $\mu_{medium}(z) = min(0,0.257,1) = 0$

  Kemudian dicari nilai z berdasarkan nilai himpunan *medium* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.7}{0.8-0.7}, \frac{0.9-z}{0.9-0.8}\right), 0\right) = 0$$

$$max\left(min\left(\frac{z-0.7}{0.1},\frac{0.9-z}{0.1}\right),0\right)=0$$

Untuk $\frac{z-0.7}{0.1}=0$ diperoleh $z=0.7$ dan untuk $\frac{0.9-z}{0.1}=0$ diperoleh $z=0.9$.

- Jika *BF* adalah *medium*, *AF* adalah *medium* dan *CBF* adalah *medium* maka P_c adalah *big*

  $\mu_{big}(z)=min(0,0.0.743,0)=0$

  Kemudian dicari nilai *z* berdasarkan nilai himpunan *big* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.8}{0.9-0.8},\frac{0.9-z}{0.9-0.9}\right),0\right)=0$$

  $$max\left(min\left(\frac{z-0.8}{0.1},\infty\right),0\right)=0$$

  diperoleh $z=0.8$.
- Jika *BF* adalah *big*, *AF* adalah *big* dan *CBF* adalah *small* maka P_c adalah *medium*

  $\mu_{medium}(z)=min(0,1,1)=0$

  Kemudian dicari nilai *z* berdasarkan nilai himpunan *medium* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.7}{0.8-0.7},\frac{0.9-z}{0.9-0.8}\right),0\right)=0$$

  $$max\left(min\left(\frac{z-0.7}{0.1},\frac{0.9-z}{0.1}\right),0\right)=0$$

  Untuk $\frac{z-0.7}{0.1}=0$ diperoleh $z=0.7$ dan untuk $\frac{0.9-z}{0.1}=0$ diperoleh $z=0.9$.
- Jika *BF* adalah *big*, *AF* adalah *big* dan *CBF* adalah *big* maka P_c adalah *medium*

  $\mu_{medium}(z)=min(0,1,0)=0$

  Kemudian dicari nilai *z* berdasarkan nilai himpunan *medium* dari keluaran P_c

  $$max\left(min\left(\frac{z-0.7}{0.8-0.7},\frac{0.9-z}{0.9-0.8}\right),0\right)=0$$

  $$max\left(min\left(\frac{z-0.7}{0.1},\frac{0.9-z}{0.1}\right),0\right)=0$$

  Untuk $\frac{z-0.7}{0.1}=0$ diperoleh $z=0.7$ dan untuk $\frac{0.9-z}{0.1}=0$ diperoleh $z=0.9$.

Aturan *fuzzy* untuk untuk peluang mutasi

- Jika *BF* adalah *small*, *AF* adalah *small* dan *CBF* adalah *small* maka P_m adalah *small*
  $\mu_{small}(z) = min(0.069,0.257,1) = 0.069$
  Kemudian dicari nilai z berdasarkan nilai himpunan *small* dari keluaran P_m
  $$max\left(min\left(\frac{z-0}{0-0}, 1, \frac{0.008-z}{0.008-0.0013}\right), 0\right) = 0.069$$
  $$max\left(min\left(\infty, 1, \frac{0.008-z}{0.0067}\right), 0\right) = 0.069$$
  diperoleh $z = 0.008046$.
- Jika *BF* adalah *small*, *AF* adalah *small* dan *CBF* adalah *big* maka P_m adalah *medium*
  $\mu_{medium}(z) = min(0.069,0.257,0) = 0$
  Kemudian dicari nilai z berdasarkan nilai himpunan *medium* dari keluaran P_m
  $$max\left(min\left(\frac{z-0.0013}{0.008-0.0013}, \frac{0.013-z}{0.013-0.008}\right), 0\right) = 0$$
  $$max\left(min\left(\frac{z-0.0013}{0.0067}, \frac{0.013-z}{0.005}\right), 0\right) = 0$$
  Untuk $\frac{z-0.0013}{0.0067} = 0$ diperoleh $z = 0.0013$ dan untuk $\frac{0.013-z}{0.005} = 0$ diperoleh $z = 0.013$.
- Jika *BF* adalah *medium*, *AF* adalah *small* dan *CBF* adalah *small* maka P_m adalah *medium*
  $\mu_{medium}(z) = min(0,0.257,1) = 0$
  Kemudian dicari nilai z berdasarkan nilai himpunan *medium* dari keluaran P_m
  $$max\left(min\left(\frac{z-0.0013}{0.008-0.0013}, \frac{0.013-z}{0.013-0.008}\right), 0\right) = 0$$
  $$max\left(min\left(\frac{z-0.0013}{0.0067}, \frac{0.013-z}{0.005}\right), 0\right) = 0$$
  Untuk $\frac{z-0.0013}{0.0067} = 0$ diperoleh $z = 0.0013$ dan untuk $\frac{0.013-z}{0.005} = 0$ diperoleh $z = 0.013$.
- Jika *BF* adalah *medium*, *AF* adalah *small* dan *CBF* adalah *medium* maka P_m adalah *big*
  $\mu_{big}(z) = min(0,0.257,0) = 0$

Kemudian dicari nilai *z* berdasarkan nilai himpunan *big* dari keluaran P_m

$$max\left(min\left(\frac{z-0.008}{0.013-0.008},\frac{0.02-z}{0.02-0.013}\right),0\right)=0$$

$$max\left(min\left(\frac{z-0.008}{0.005},\frac{0.02-z}{0.007}\right),0\right)=0$$

Untuk $\frac{z-0.008}{0.005}=0$ diperoleh $z=0.008$ dan untuk $\frac{0.02-z}{0.007}=0$ diperoleh $z=0.02$.

- Jika *BF* adalah *medium*, *AF* adalah *big* dan *CBF* adalah *small* maka P_m adalah *big*

  $\mu_{big}(z)=min(0,1,1)=0$

  Kemudian dicari nilai *z* berdasarkan nilai himpunan *big* dari keluaran P_m

  $$max\left(min\left(\frac{z-0.008}{0.013-0.008},\frac{0.02-z}{0.02-0.013}\right),0\right)=0$$

  $$max\left(min\left(\frac{z-0.008}{0.005},\frac{0.02-z}{0.007}\right),0\right)=0$$

  Untuk $\frac{z-0.008}{0.005}=0$ diperoleh $z=0.008$ dan untuk $\frac{0.02-z}{0.007}=0$ diperoleh $z=0.02$.

- Jika *BF* adalah *big*, *AF* adalah *small* dan *CBF* adalah *small* maka P_m adalah *small*

  $\mu_{small}(z)=min(0,0.257,1)=0$

  Kemudian dicari nilai *z* berdasarkan nilai himpunan *small* dari keluaran P_m

  $$max\left(min\left(\frac{z-0}{0-0},1,\frac{0.008-z}{0.008-0.0013}\right),0\right)=0$$

  $$max\left(min\left(\infty,1,\frac{0.008-z}{0.0067}\right),0\right)=0$$

  diperoleh $z=0.008$

- Jika *BF* adalah *big*, *AF* adalah *small* dan *CBF* adalah *big* maka P_m adalah *big*

  $\mu_{big}(z)=min(0,0.257,0)=0$

  Kemudian dicari nilai *z* berdasarkan nilai himpunan *big* dari keluaran P_m

  $$max\left(min\left(\frac{z-0.008}{0.013-0.008},\frac{0.02-z}{0.02-0.013}\right),0\right)=0$$

$$max\left(min\left(\frac{z-0.008}{0.005}, \frac{0.02-z}{0.007}\right), 0\right) = 0$$

Untuk $\frac{z-0.008}{0.005} = 0$ diperoleh $z = 0.008$ dan untuk $\frac{0.02-z}{0.007} = 0$ diperoleh $z = 0.02$.

d) Melakukan defuzzifikasi keluaran

Metode defuzzifikasi yang digunakan adalah metode titik berat(COA). Dari hasil penerapan basis aturan *fuzzy* diperoleh beberapa nilai *z* untuk peluang pindah silang.

Aturan	Himpunan *fuzzy*	Derajat keanggotaan	Z	
1	*small*	0.257	0.7931	
2	*small*	0	0.8	
3	*small*	0	0.8	
4	*medium*	0	0.7	0.9
5	*big*	0	0.8	
6	*medium*	0	0.7	0.9
7	*medium*	0	0.7	0.9

Karena operator yang digunakan adalah *min-max* maka perlu dicari nilai minimum masing-masing himpunan *fuzzy* dan selanjutnya dihitung nilai *z*. Nilai minimum himpunan *small* adalah $z = 0.7743$ dengan derajat keanggotaan 0. Nilai minimum himpunan *medium* adalah $z = 0.7$ dengan derajat keanggotaan 0. Nilai minimum himpunan *big* adalah $z = 0.8$ dengan derajat keanggotaan 0. Karena nilai derajat keanggotaan semua himpunan *fuzzy* adalah 0 maka luas area adalah sama dengan 0, sehingga nilai dari keluaran tidak dapat ditentukan. Nilai tengah dari *range* variabel keluaran digunakan sebagai nilai keluaran, yaitu $P_c = 0.8$.

Dari hasil penerapan basis aturan *fuzzy* diperoleh beberapa nilai *z* untuk peluang mutasi.

Aturan	Himpunan *fuzzy*	Derajat keanggotaan	Z	
1	*small*	0.069	0.008046	
2	*medium*	0	0.0013	0.013
3	*medium*	0	0.0013	0.013
4	*big*	0	0.008	0.002
5	*big*	0	0.008	0.002

Aturan	Himpunan *fuzzy*	Derajat keanggotaan	Z	
6	*small*	0	0.008	
7	*big*	0	0.008	0.02

Karena operator yang digunakan adalah *min-max* maka perlu dicari nilai minimum masing-masing himpunan *fuzzy* dan selanjutnya dihitung nilai z. Nilai minimum untuk himpunan *small* adalah $z = 0.0013$ dengan derajat keanggotaan 0. Nilai minimum untuk himpunan *medium* adalah $z = 0.008$ dengan derajat keanggotaan 0. Nilai minimum untuk himpunan *big* adalah $z = 0.02$ dengan derajat keanggotaan 0. Karena nilai dari derajat keanggotaan semua himpunan *fuzzy* adalah 0 maka luas area adalah sama dengan 0, sehingga nilai dari keluaran tidak dapat ditentukan. Rata-rata dari *range* variabel keluaran digunakan sebagai nilai keluaran, yaitu $P_m = 0.015$.

7. Membuat populasi baru

Mengulang langkah-langkah sebagai berikut:

a) *Elitism*

Memasukkan individu dengan nilai *fitness* terbaik ke dalam populasi yang baru. Untuk kasus minimasi maka individu dengan nilai *fitness* terbaik adalah individu dengan nilai *fitness* yang paling minimum. Karena Ukuran populasi adalah 20(genap) maka, individu yang dimasukkan ke dalam populasi yang baru ada dua, yaitu individu ke-15 dengan nilai *fitness* 2.0951 dan satu individu baru dengan nilai yang sama yaitu 2.0951. Dengan demikian pada populasi baru sudah terdapat dua individu, 18 individu yang lain akan didapatkan dengan proses seleksi.

b) *Linear fitness ranking*

Melakukan penskalaan nilai *fitness*, nilai *fitness* tertinggi akan mendapat ranking yang paling besar.

Ranking	Individu	Nilai *fitness*
1	15	2.1
2	9	7.4
3	6	11.0
4	8	60.1
5	2	63.7
6	12	66.3
7	16	80.7
8	17	86.2
9	7	106.7
10	10	120.2
11	3	153.5
12	20	164.5
13	11	188.9
14	5	225.8
15	4	234.0
16	14	251.8
17	19	432.6
18	1	515.5
19	18	1541.9
20	13	1735.1

$$f(i) = f_{max} - (f_{max} - f_{min})\left(\frac{R(i)-1}{N-1}\right)$$

$$f(1) = 1735.1 - (1735.1 - 2.0951)\left(\frac{18-1}{20-1}\right) = 184.5$$

$$f(2) = 1735.1 - (1735.1 - 2.0951)\left(\frac{5-1}{20-1}\right) = 1370.3$$

$$f(3) = 1735.1 - (1735.1 - 2.0951)\left(\frac{11-1}{20-1}\right) = 822.99$$

$$f(4) = 1735.1 - (1735.1 - 2.0951)\left(\frac{15-1}{20-1}\right) = 458.2$$

$$f(5) = 1735.1 - (1735.1 - 2.0951)\left(\frac{14-1}{20-1}\right) = 549.4$$

$$f(6) = 1735.1 - (1735.1 - 2.0951)\left(\frac{3-1}{20-1}\right) = 1552.7$$

$$f(7) = 1735.1 - (1735.1 - 2.0951)\left(\frac{9-1}{20-1}\right) = 1005.4$$

$$f(8) = 1735.1 - (1735.1 - 2.0951)\left(\frac{4-1}{20-1}\right) = 1461.5$$

$$f(9) = 1735.1 - (1735.1 - 2.0951)\left(\frac{2-1}{20-1}\right) = 1643.9$$

$$f(10) = 1735.1 - (1735.1 - 2.0951)\left(\frac{10-1}{20-1}\right) = 899.99$$
$$f(11) = 1735.1 - (1735.1 - 2.0951)\left(\frac{13-1}{20-1}\right) = 640.6$$
$$f(12) = 1735.1 - (1735.1 - 2.0951)\left(\frac{6-1}{20-1}\right) = 1279.1$$
$$f(13) = 1735.1 - (1735.1 - 2.0951)\left(\frac{20-1}{20-1}\right) = 2.0951$$
$$f(14) = 1735.1 - (1735.1 - 2.0951)\left(\frac{16-1}{20-1}\right) = 366.9$$
$$f(15) = 1735.1 - (1735.1 - 2.0951)\left(\frac{1-1}{20-1}\right) = 1735.1$$
$$f(16) = 1735.1 - (1735.1 - 2.0951)\left(\frac{7-1}{20-1}\right) = 1187.8$$
$$f(17) = 1735.1 - (1735.1 - 2.0951)\left(\frac{8-1}{20-1}\right) = 1096.6$$
$$f(18) = 1735.1 - (1735.1 - 2.0951)\left(\frac{19-1}{20-1}\right) = 93.3$$
$$f(19) = 1735.1 - (1735.1 - 2.0951)\left(\frac{17-1}{20-1}\right) = 275.7$$
$$f(20) = 1735.1 - (1735.1 - 2.0951)\left(\frac{12-1}{20-1}\right) = 731.8$$

c) *Roulette wheel*

Memilih induk untuk proses seleksi dan mutasi berdasarkan nilai *fitness*-nya. Pertama menghitung nilai peluang seleksi (p_k) dari masing-masing individu, selanjutnya menghitung peluang kumulatifnya (q_k).

$$P_k = \frac{fitness(k)}{Jumlah\ fitness} \qquad q_k = \sum_{j=1}^{k} p_j$$

$$p_1 = \frac{184.5}{17357.78} = 0.01063 \qquad q_1 = 0.01063$$
$$p_2 = \frac{1370.3}{17357.78} = 0.07894 \qquad q_2 = 0.08957$$
$$p_3 = \frac{822.99}{17357.78} = 0.04741 \qquad q_3 = 0.13698$$
$$p_4 = \frac{458.2}{17357.78} = 0.02639 \qquad q_4 = 0.16337$$
$$p_5 = \frac{549.4}{17357.78} = 0.03165 \qquad q_5 = 0.19502$$
$$p_6 = \frac{1552.7}{17357.78} = 0.08945 \qquad q_6 = 0.28447$$
$$p_7 = \frac{1005.4}{17357.78} = 0.05792 \qquad q_7 = 0.34239$$
$$p_8 = \frac{1461.5}{17357.78} = 0.08419 \qquad q_8 = 0.42658$$

$p_9 = \frac{1643.9}{17357.78} = 0.09471$ $q_9 = 0.52129$

$p_{10} = \frac{899.99}{17357.78} = 0.05185$ $q_{10} = 0.57314$

$p_{11} = \frac{640.6}{17357.78} = 0.03691$ $q_{11} = 0.61005$

$p_{12} = \frac{1279.1}{17357.88} = 0.07369$ $q_{12} = 0.68374$

$p_{13} = \frac{2.0951}{17357.78} = 0.00012$ $q_{13} = 0.68386$

$p_{14} = \frac{366.9}{17357.88} = 0.02114$ $q_{14} = 0.705$

$p_{15} = \frac{1735.1}{17357.78} = 0.09996$ $q_{15} = 0.80496$

$p_{16} = \frac{1187.8}{17357.78} = 0.06843$ $q_{16} = 0.87339$

$p_{17} = \frac{1096.6}{17357.78} = 0.06318$ $q_{17} = 0.93657$

$p_{18} = \frac{93.3}{17357.78} = 0.00537$ $q_{18} = 0.94194$

$p_{19} = \frac{275.7}{17357.88} = 0.01588$ $q_{19} = 0.95782$

$p_{20} = \frac{731.8}{17357.88} = 0.04218$ $q_{20} = 1$

Selanjutnya akan dibangkitkan bilangan acak r, jika peluang kumulatif individu tersebut lebih dari r maka individu tersebut akan terpilih sebagai induk. Misal $r = 0.2311$, maka individu ke-6 dengan nilai peluang kumulatif sama dengan 0.28447 akan terpilih sebagai induk 1. Lalu dibangkitkan bilangan acak r lagi, misal $r = 0.6068$, maka individu ke-11 dengan nilai peluang kumulatif sama dengan 0.61005 akan terpilih sebagai induk 2.

d) Pindah silang

Dua induk yang telah dipilih dari proses seleksi akan mengalami proses pindah silang jika bilangan acak $r < P_c$. Misal $r = 0.486$, maka $r < P_c$, sehingga proses pindah silang dilakukan antara individu ke-6 dan individu ke-11. Sebelum melakukan proses pindah silang maka terlebih dahulu menentukan titik potong dari kedua individu, dari pengambilan nilai secara acak antara 1 sampai dengan 19

didapatkan titik potong yaitu 14. Proses pindah silang adalah sebagai berikut.

Induk 1	**1 0 1 0 0 0 1 0 1 1 0 1 1 1**	**1 1 1 0 1 1**
Induk 2	0 1 1 0 1 1 0 1 1 0 1 1 0 1	1 0 1 1 0 0
Anak 1	**1 0 1 0 0 0 1 0 1 1 0 1 1 1**	1 0 1 1 0 0
Anak 2	0 1 1 0 1 1 0 1 1 0 1 1 0 1	**1 1 1 0 1 1**

e) Mutasi

Untuk masing-masing gen dalam individu yang sudah mengalami proses pindah silang, akan dilakukan proses mutasi, jika bilangan acak $r < P_m$. Proses mutasi untuk anak 1 dan anak 2 adalah sebagai berikut:

Gen		r
1	1	0.2226
2	0	0.4966
3	1	0.007
4	0	0.3162
5	0	0.7384
6	0	0.7903
7	1	0.4003
8	0	0.3999
9	1	0.2517
10	1	0.0919
11	0	0.6022
12	1	0.8324
13	1	0.2935
14	1	0.4655
15	1	0.0455
16	0	0.0857
17	1	0.5131
18	1	0.7154
19	0	0.0354
20	0	0.9796

Gen		r
1	0	0.4974
2	1	0.7007
3	1	0.8031
4	0	0.5167
5	1	0.5027
6	1	0.7547
7	0	0.2381
8	1	0.9431
9	1	0.0886
10	0	0.543
11	1	0.9035
12	1	0.8613
13	0	0.6315
14	1	0.2624
15	1	0.379
16	1	0.3373
17	1	0.3146
18	0	0.9471
19	1	0.5352
20	1	0.702

Untuk anak 1, dari semua gen yang memenuhi $r < P_m$, adalah gen ke-3, sehingga nilai dari gen ke-3 dirubah dari 1 menjadi 0. Sedangkan untuk anak 2, dari semua gen tidak ada yang memenuhi $r < P_m$, sehingga tidak ada gen yang dimutasi.

8. Penggantian populasi yang lama dengan populasi yang baru
   Anak hasil proses mutasi dimasukkan ke dalam populasi yang baru, dilakukan terus hingga populasi yang baru berjumlah 20.
9. Mengulang langkah nomor 3 hingga nomor 8 untuk populasi yang baru, hingga jumlah generasi sama dengan 100.