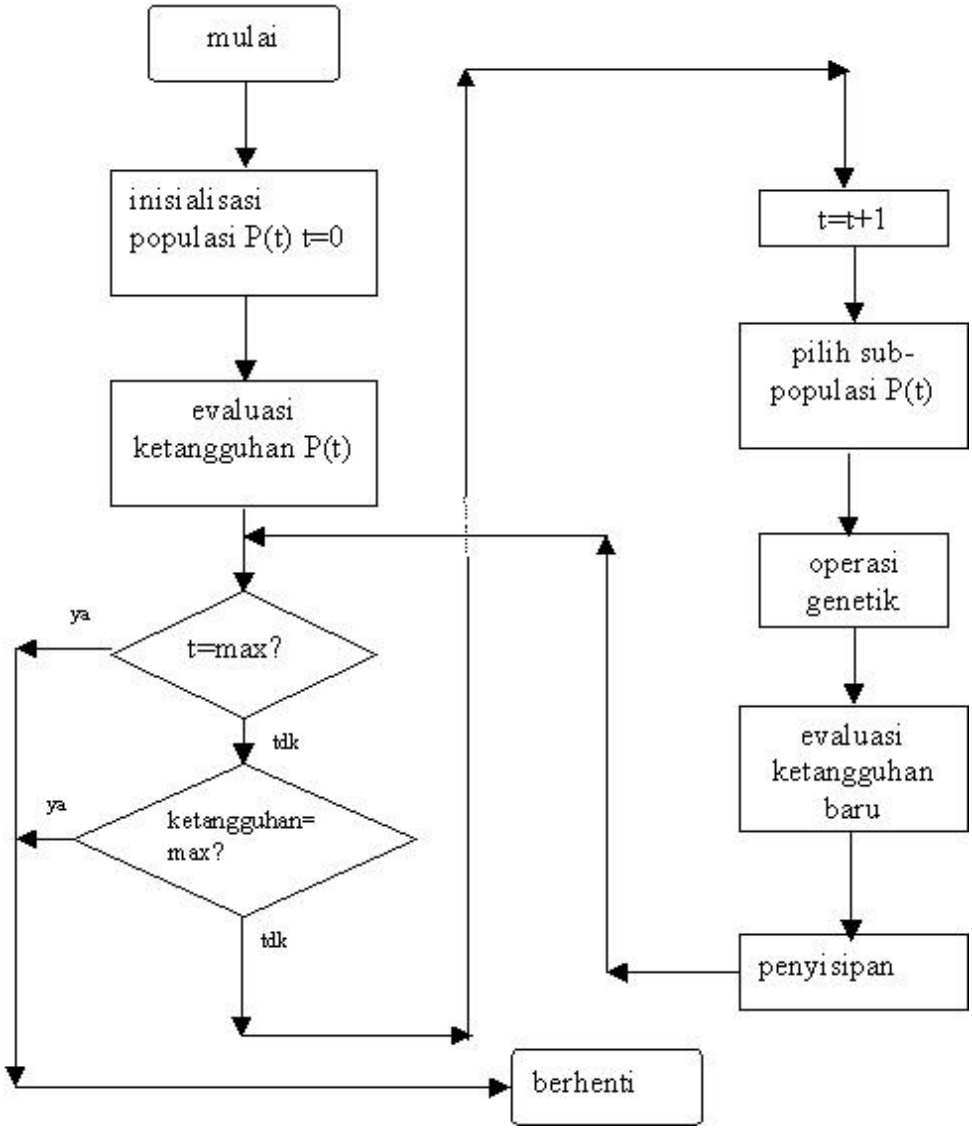
OPTIMASI MENGGUNAKAN ALGORITMA GENETIK

I. Konsep Dasar

Prinsip dasar algoritma genetik pertama kali disampaikan oleh Holland pada tahun 1975. Algoritma genetik diilhami oleh mekanisme seleksi alam, proses biologis yang sebagai pemenang dalam kompetisi lingkungan adalah individu yang lebih kuat. Di sini, algoritma genetik menggunakan analogi langsung evolusi alami. Algoritma ini menganggap bahwa solusi yang tepat untuk suatu masalah adalah individual dan dapat direpresentasikan oleh himpunan parameter-parameter. Parameter-parameter tersebut dipandang sebagai gen-gen dari kromosom dan disusun oleh sebuah *string* nilai dalam bentuk biner. Suatu nilai positif yang biasanya dikenal sebagai nilai fitness (*fitness value*) digunakan untuk merefleksikan derajad "kebaikan" kromosom untuk memecahkan masalah, dan nilai ini sangat berkaitan erat dengan fungsi objektif.

Melalui evolusi genetik, kromosom yang lebih *fit* memiliki kecenderungan untuk menghasilkan keturunan yang bermutu, ini berarti merupakan solusi masalah yang lebih baik. Pada aplikasi praktis, sekumpulan populasi kromosom harus diinstalasikan dan diinisialisasikan secara acak. Ukuran populasi tersebut bervariasi dari satu masalah ke masalah yang lain. Pada masing-masing siklus operasi genetik, disebut proses pengembangan, generasi berikutnya dihasilkan dari kromosom-kromosom yang sedang berada dalam populasi sekarang. Proses ini akan berhasil apabila sekelompok kromosom tersebut, biasa disebut induk atau *matepool*, dipilih melalui suatu rutin seleksi. Gen-gen induk dicampur dan dikombinasi ulang untuk produksi keturunan pada generasi berikutnya. Diharapkan bahwa dari proses evolusi ini (manipulasi gen-gen), kromosom yang lebih baik akan menghasilkan keturunan yang lebih baik, dengan demikian memiliki kesempatan lebih besar untuk bertahan hidup pada generasi berikutnya.

Ada dua operasi dasar dalam siklus evolusi algoritma genetik, yakni persilangan dan mutasi. Operasi yang lain adalah rutin seleksi. Untuk lebih jelasnya, jalannya algoritma genetik ada di gambar 1. Proses evolusi akan berlangsung sampai mendapatkan ketangguhan maksimal atau sampai dengan jumlah generasi yang ditentukan.



Gambar 1. Algoritma genetik (sumber : Man *et al.*, 1996, dengan modifikasi bahasa).

II. Operasi-operasi genetik

Untuk memenuhi nilai objektif atau nilai fitness yang optimal, dilakukan tiga operasi genetik, yaitu :

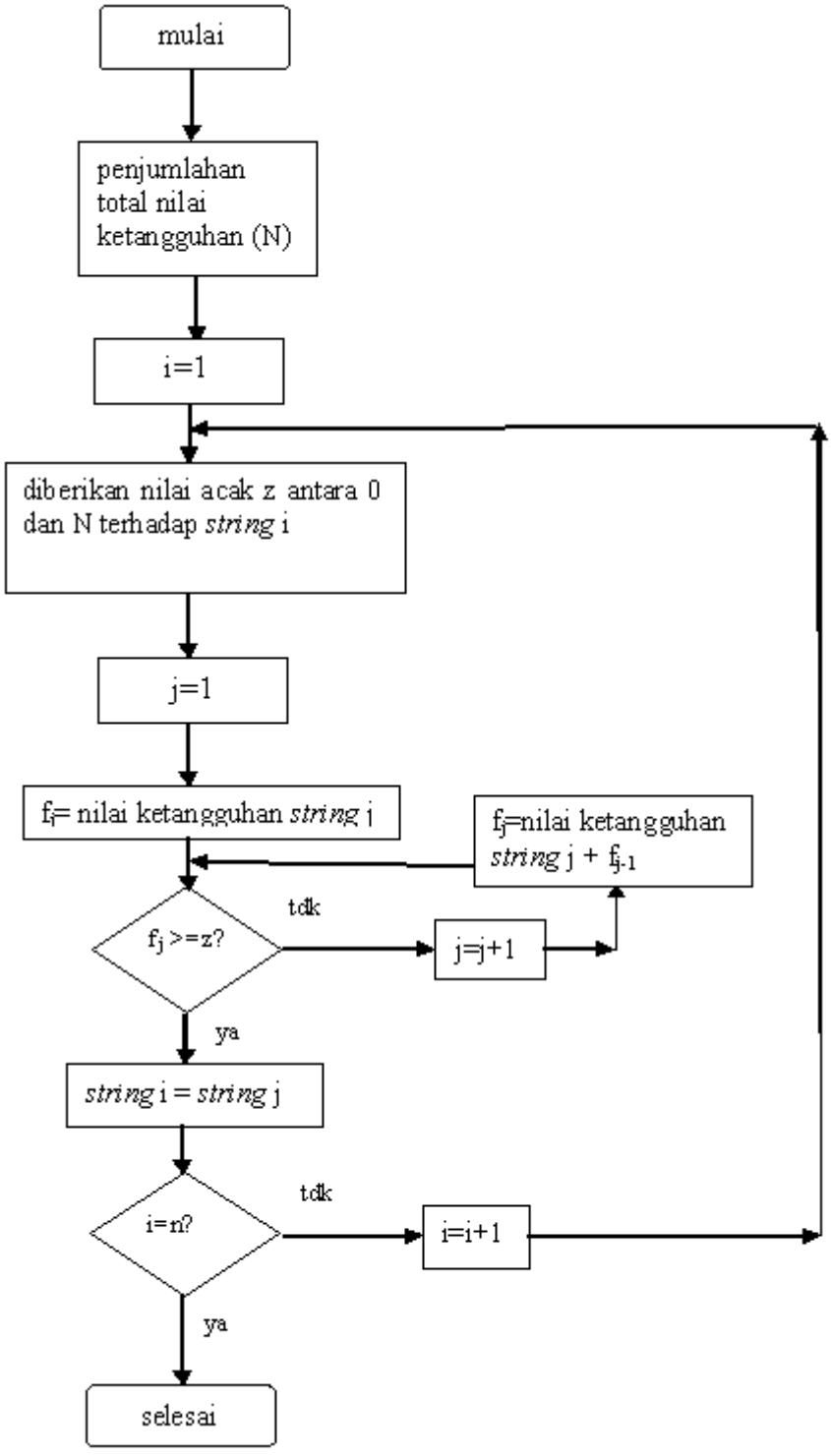
(1) reproduksi berdasarkan kepantasan,

(2) persilangan, dan

(3) mutasi.

Seleksi adalah proses pembentukan dan penyalinan nilai fitness untuk masing-masing *string* sebagai anggota populasi awal secara acak berdasar ruang solusi dan banyaknya populasi. Selanjutnya setiap *string* dalam populasi akan mengalami seleksi berdasar nilai fitness untuk menentukan kesempatan menjadi induk.

Metoda seleksi roda rolet adalah jenis fungsi seleksi tradisional dengan persamaan probabilitas ketangguhan aktif ke i dibagi dengan jumlah ketangguhan seluruh individual. Jalannya seleksi roda rolet ada di gambar 2.

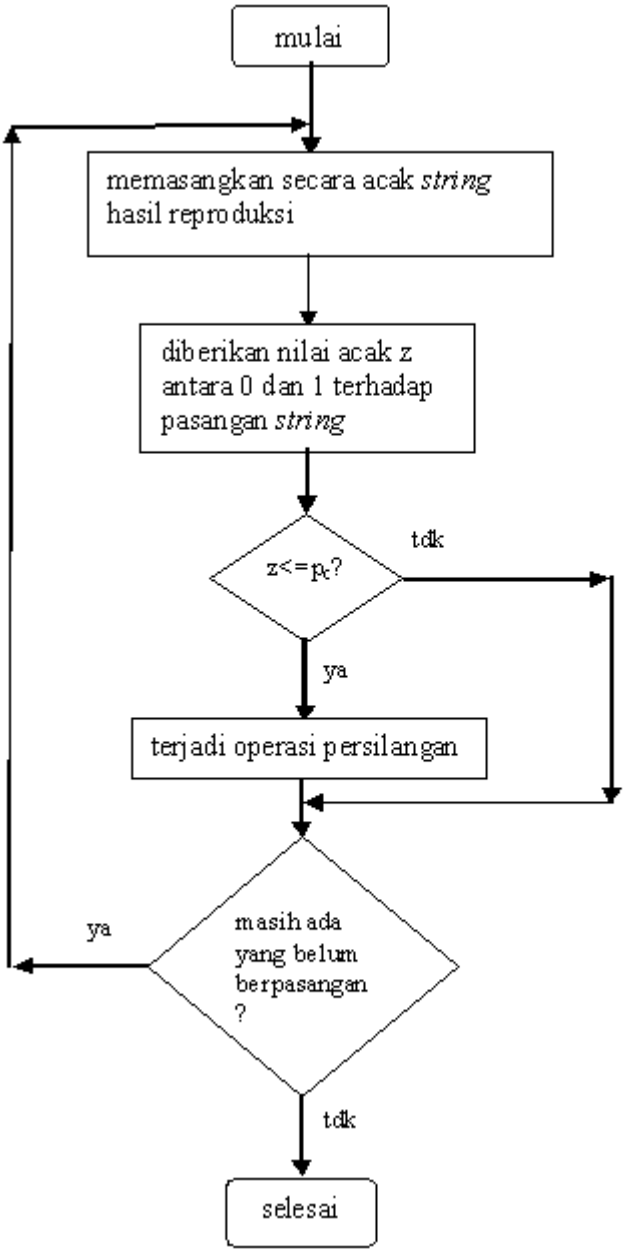


Gambar 2. Gaftar alir jalannya seleksi roda rolet untuk populasi n *string*.

Operasi persilangan (*crossover*) dalam proses evolusi algoritma genetik berfungsi untuk menambah keanekaragaman *string* dalam populasi melalui penyilangan antar *string* yang diperoleh dari proses reproduksi sebelumnya. Proses dimulai dengan memisahkan suatu *string* menjadi dua bagian. Selanjutnya salah satu bagian disilangkan dengan salah satu bagian dari *string* yang lain yang telah dipisahkan dengan cara yang sama. Proses ini dikenal dengan nama operasi persilangan satu titik.

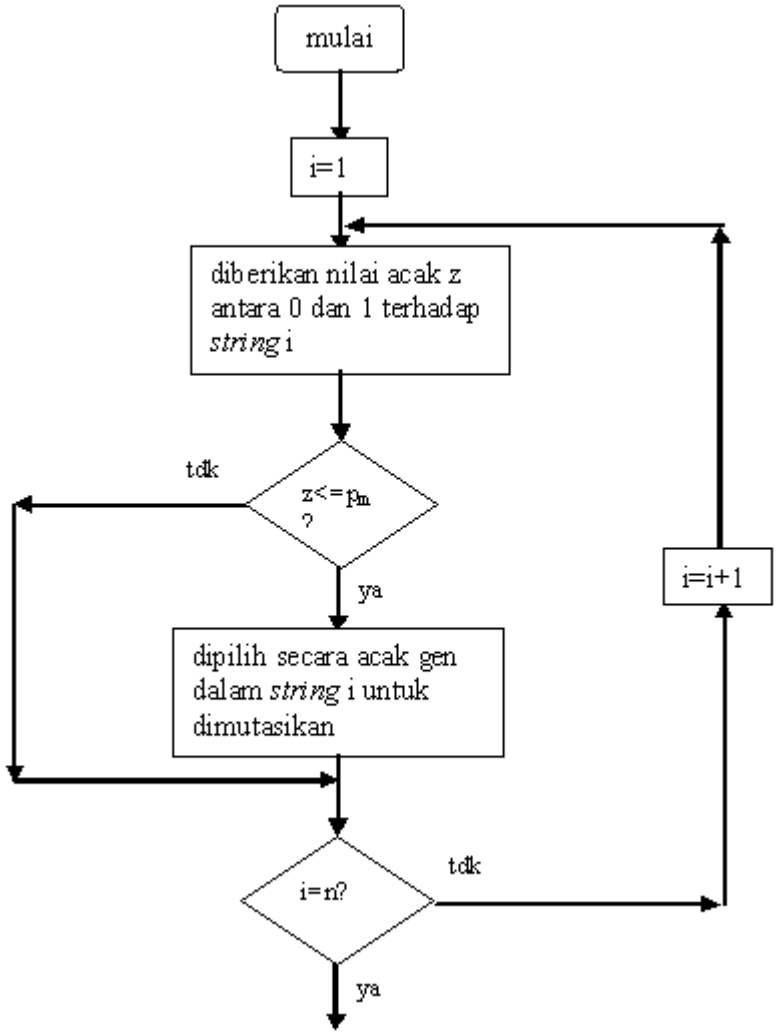
Operasi mutasi merupakan operasi pelengkap yang mengubah nilai salah satu atau beberapa bit. Biasanya operasi seleksi dan persilangan bekerja dengan cukup efektif dan melakukan rekombinasi terhadap *string* untuk mendapatkan *string* yang diinginkan.

Jalannya operasi persilangan ada di gambar 3.



Gambar 3. Gaftar alir operasi persilangan satu titik dengan peluang p_c

Proses mutasi ada di gambar 4.



Gambar 4. Gaftar alir proses mutasi n *string* dengan peluang p_m

Pemilihan p_m maupun p_c sebagai parameter-parameter kendali merupakan masalah optimisasi yang dapat kompleks maupun nonlinier. Penetapannya sangat bergantung pada fungsi objektifnya. Pemilihan ini masih terbuka bagi yang ingin berpendapat walaupun beberapa petunjuk dapat dilihat sebagai berikut:

1. Untuk ukuran populasi besar (100) : $p_c = 0{,}6$; $p_m = 0{,}001$

2. Untuk populasi kecil (30) : $p_c = 0{,}9$; $p_m = 0{,}01$

Operasi genetik yang berlaku untuk aturan adalah mutasi bit. Masing-masing bit gen kendali diguling ("1" atau "0") jika memenuhi hasil tes probabilitas (angka acak yang diberikan lebih kecil atau sama dengan probabilitas mutasi

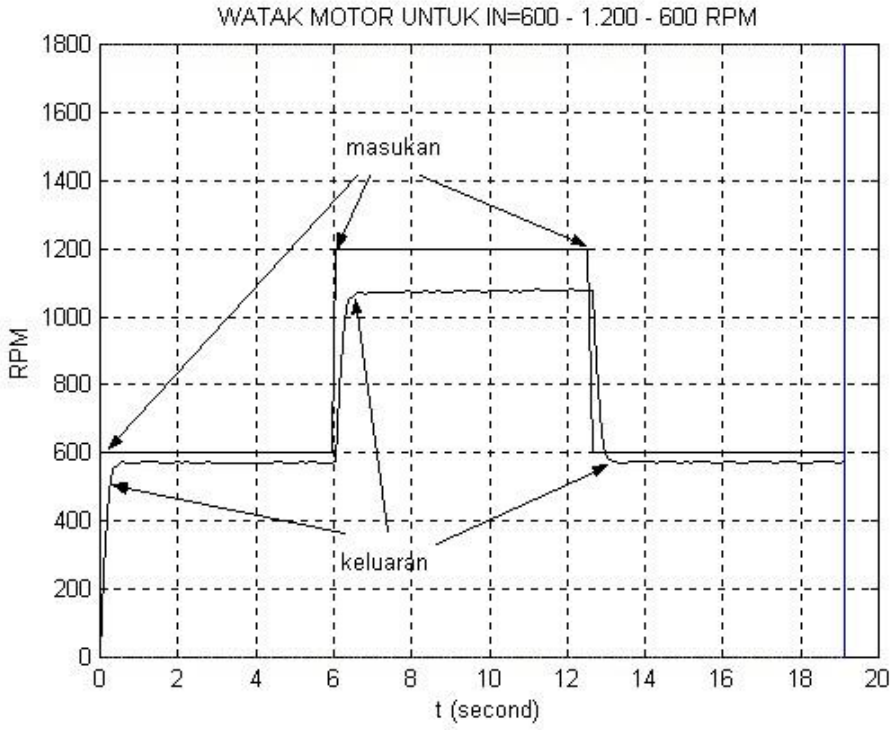
aturan, r_c). Operasi mutasi yang diberikan berupa pergeseran delta terhadap kromosom aturan fuzi sebagai berikut:

$$h_{i,j} = h_{i+\Delta i, j+\Delta j} \quad \text{.....(1)}$$

dengan $\Delta i, \Delta j$ memiliki dapat 1 atau -1 dengan probabilitas 0,01.

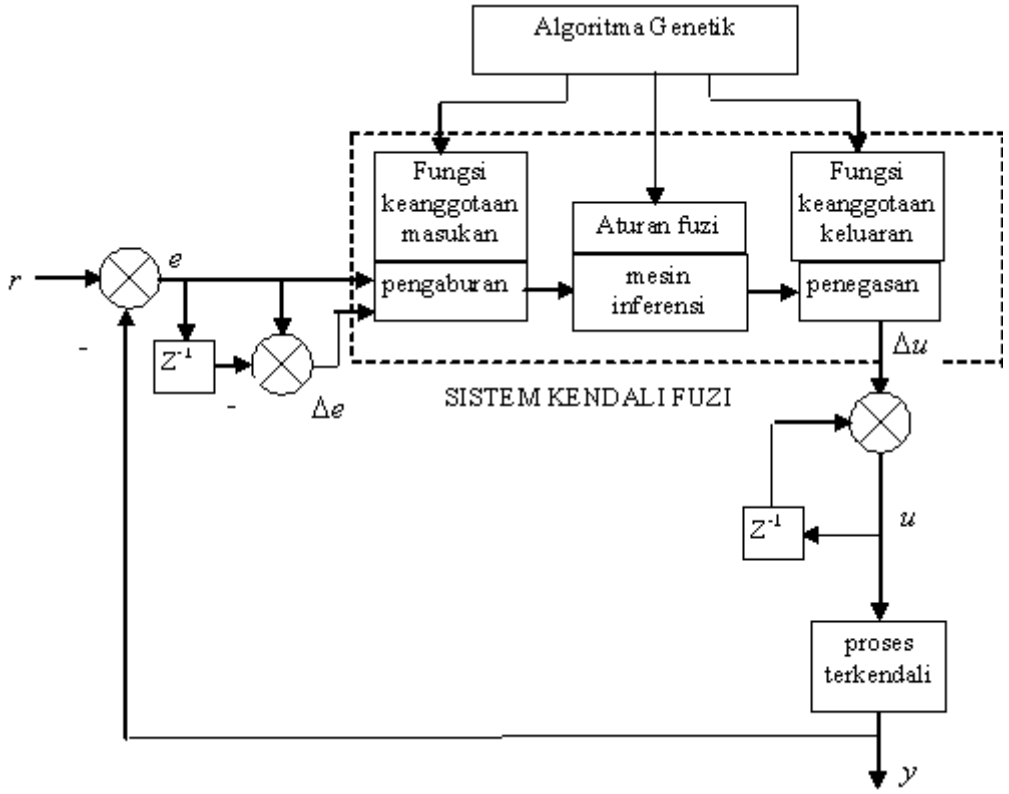
III. Aplikasi pada Optimasi Pengendali Logika Fuzi

Watak motor dc yang dikendalikan terlihat di gambar 5.



Gambar 5. Watak kalang terbuka dengan masukan 600;1.200;600 rpm.

Struktur pengendali logika fuzi yang dioptimasikan menggunakan algoritma genetik terlihat di gambar 6.



Ganbar 6. Pengendali logika fuzi untuk kecepatan motor dc yang teroptimasi menggunakan algoritma genetik

Proses optimasinya adalah sebagai berikut.

Di sini dicoba optimasi pengendali fuzi menggunakan algoritma genetik dengan ukuran populasi 3 *string*, lebar 21 bit, peluang persilangan (p_c) sebesar satu, peluang mutasi (p_m) sebesar 0,02 , peluang mutasi aturan (r_c) sebesar 0,02 , dan dibatasi maksimal 20 generasi. Jalannya proses optimasi adalah sebagai berikut.

Generasi 1

Populasi inisial :

Populasi inisial dihasilkan dari nilai acak yang dibangkitkan menggunakan program MATLAB. Hasilnya adalah sebagai berikut.

Tabel 1a. Operasi genetik 21 bit, 3 *string*, pada generasi inisial.

Kromosom	Ketangguhan
101011001011110011010	125,6
010000100010100110110	88,6
101110000101010111011	267,1

Terlihat di tabel 1a bahwa nilai fitness yang terbesar ada di *string* 3. Nilai fitness yang ditampilkan di sini adalah nilai fitness dikalikan seribu.

Populasi hasil seleksi roda rolet :

Tabel 1b. Operasi genetik 21 bit, 3 *string*, pada generasi pertama : seleksi

Kromosom	Ketangguhan
101110000101010111011	
101110000101010111011	
101110000101010111011	

Terlihat di tabel 1b bahwa *string* tiga dari tabel 1a mendominasi kedudukan setiap *string*. Hal ini disebabkan karena *string* ketiga memiliki nilai fitness terbesar sehingga peluang untuk diseleksi adalah terbesar.

Populasi hasil operasi persilangan :

Tabel 1c. Operasi genetik 21 bit, 3 *string*, pada generasi pertama : persilangan.

Kromosom	Ketangguhan
1011100 00101010111011 1011100 00101010111011 1011100 00101010111011	

Terlihat di tabel 1c bahwa operasi persilangan yang memiliki peluang satu tidak kelihatan karena semua *string* adalah sama.

Populasi hasil operasi mutasi :

Tabel 1d. Operasi genetik 21 bit, 3 *string*, pada generasi pertama : mutasi.

Kromosom	Ketangguhan
101110000101010111010 101110000101010111011 101110000111011111011	262,5 267,1 164,9

Terlihat di tabel 1d bahwa mutasi terjadi di *string* 1, bit 21, dan *string* 3, bit 12. Nilai fitness terbesar ada di *string* 2 sehingga hasil optimasi generasi pertama adalah *string* 2 dari hasil operasi mutasi.

Hasil optimasi aturan :

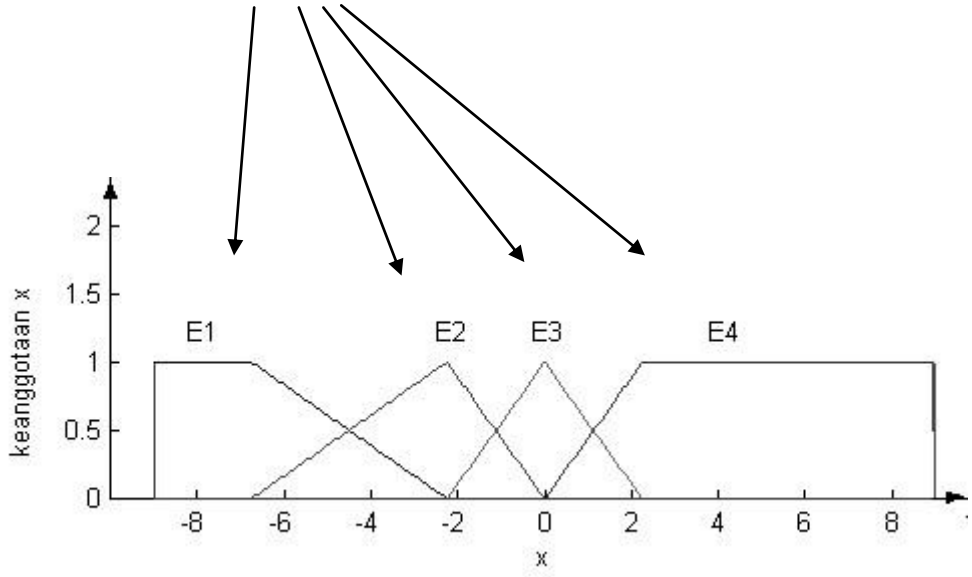
Tabel 1e. Operasi genetik 21 bit, 3 *string*, pada generasi pertama : mutasi aturan

D / E	D1	D2	D3
E1	DU1	DU2	DU3
E2	DU2	DU3	DU4
E3	DU3	DU4	DU5
E4	DU4	DU5	DU5

Fungsi keanggotaan *error*, *error rate*, dan *output rate* hasil optimasi genetik generasi 1 ada di gambar 7, dan grafik keluaran pengendali logika fuzi ada di gambar 4.7.

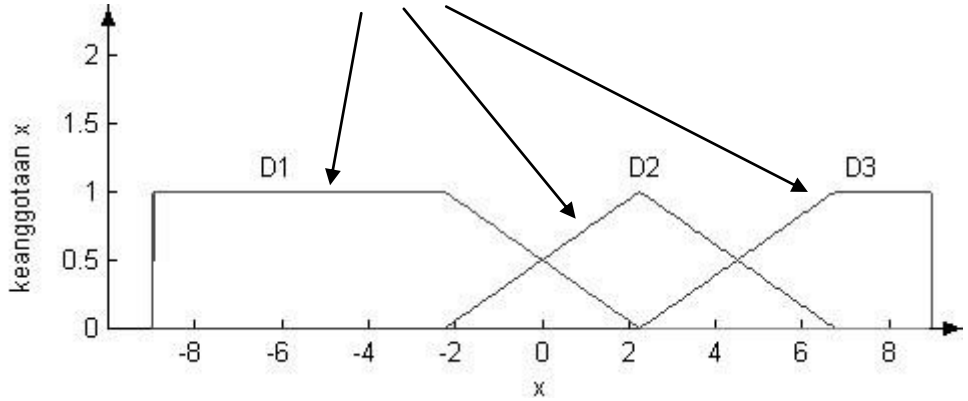
String : 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 1 1 0 1 1

Sub-string untuk *error* : 1 0 1 1 1 0 0



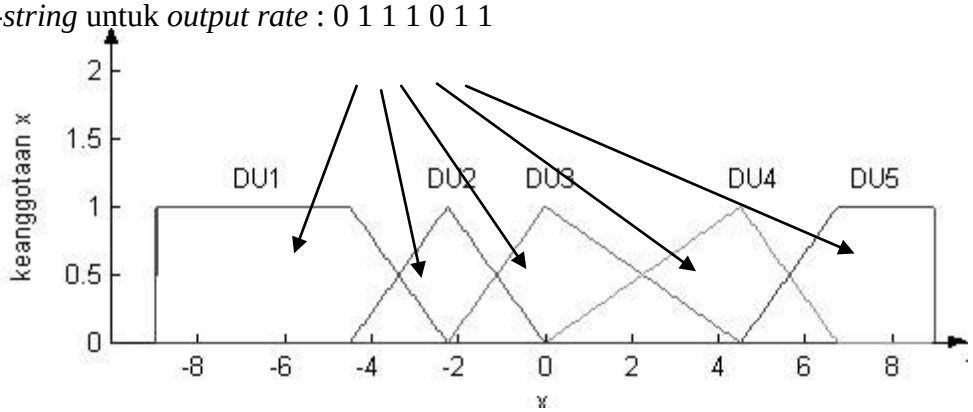
Gambar 7a. Fungsi keanggotaan *error* hasil optimasi genetik generasi pertama.

Sub-string untuk *error rate* : 0 0 1 0 1 0 1

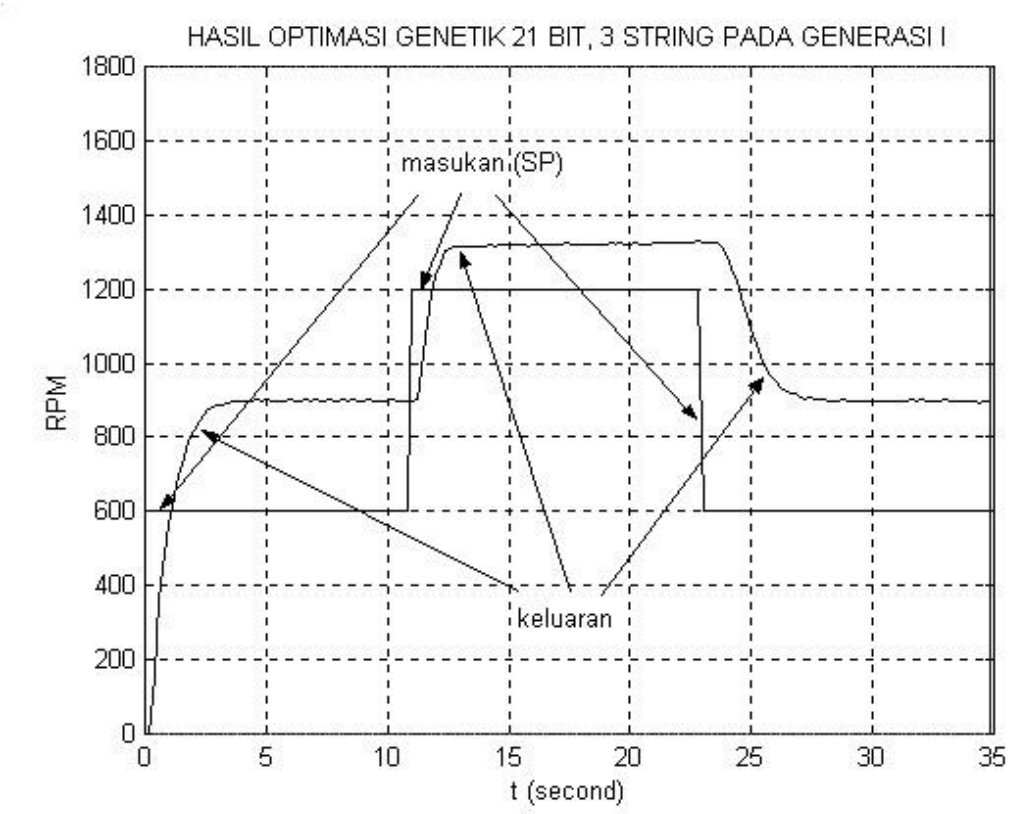


Gambar 7b. Fungsi keanggotaan *error rate* hasil optimasi genetik generasi pertama.

Sub-string untuk *output rate* : 0 1 1 1 0 1 1



Gambar 7c. Fungsi keanggotaan *output rate* hasil optimasi genetik generasi pertama.



Gambar 8. Hasil optimasi pengendali logika fuzi-genetik 21 bit, 3 *string* generasi I

Generasi 6

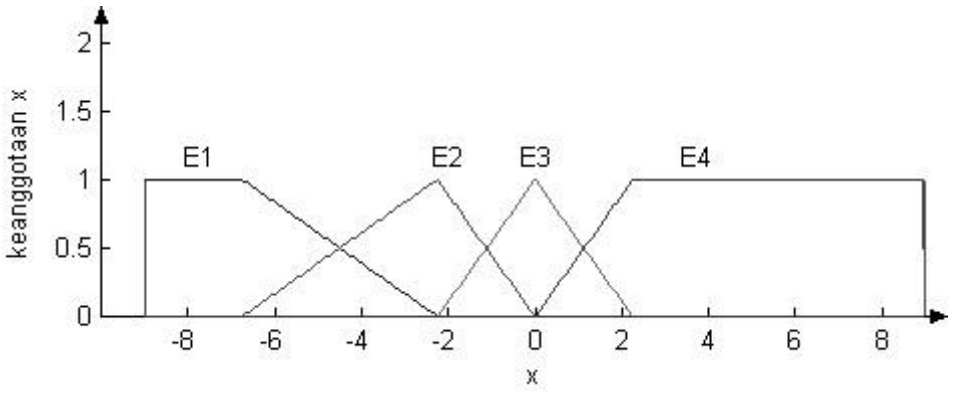
Dari proses optimasi selama enam generasi dihasilkan sebagai berikut.

Tabel 2 Operasi genetik 21 bit, 3 *string*, pada generasi keenam : mutasi aturan.

D / E	D1
E1	D1
E2	D2
E3	D3
E4	D4

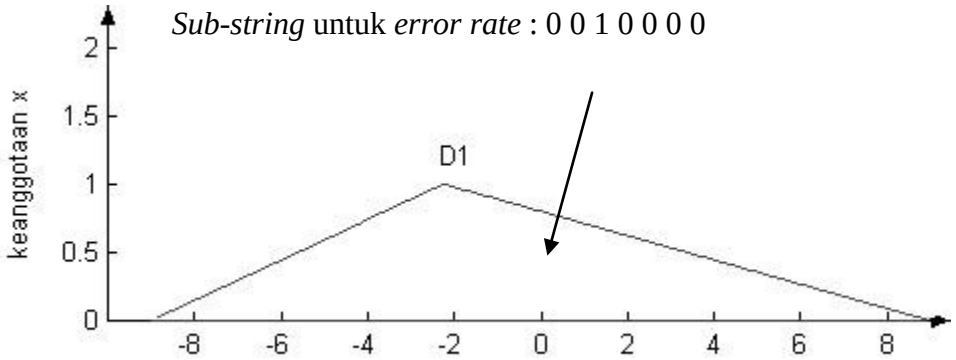
Fungsi keanggotaan *error*, *error rate*, dan *output rate* hasil optimasi genetik generasi 6 terlihat di gambar 4.16, dan grafik keluaran pengendali logika fuzi ada di gambar 4.17.

String : 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 0

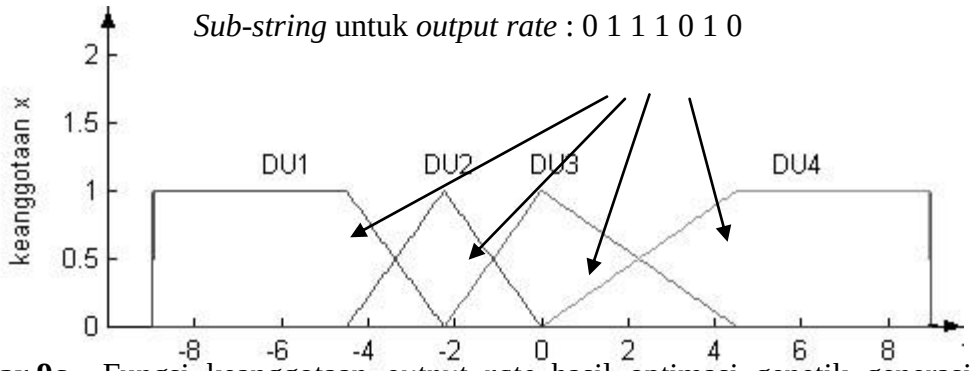


Gambar 9a. Fungsi keanggotaan *error* hasil optimasi genetik generasi keenam.

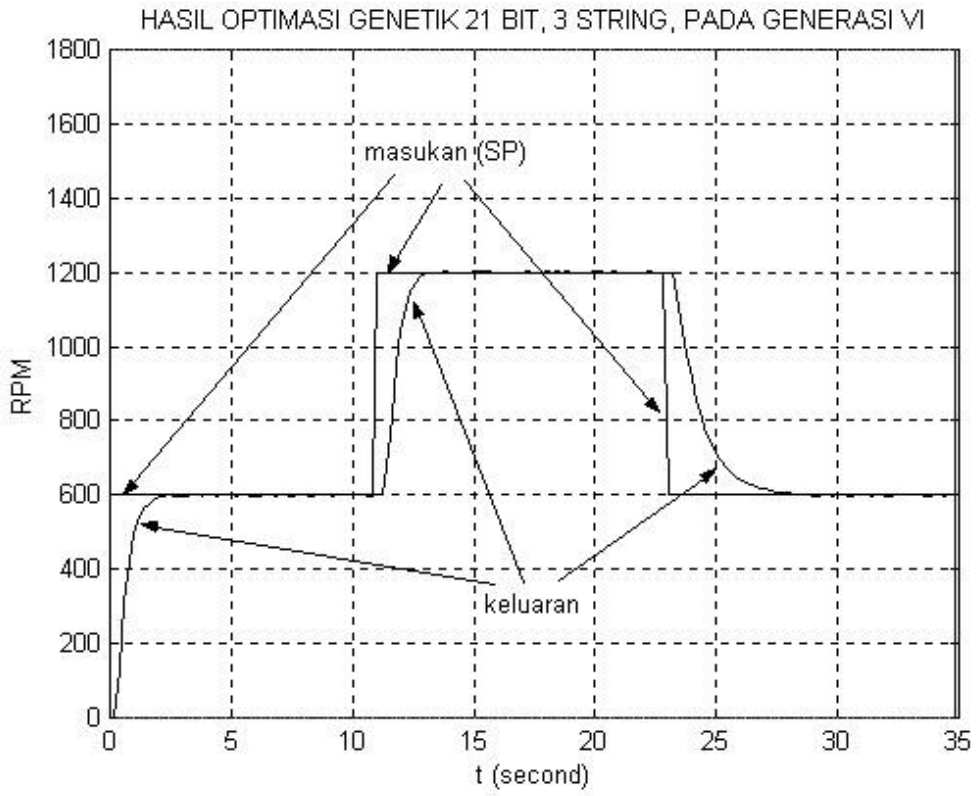
Sub-string untuk *error rate* : 0 0 1 0 0 0 0



Gambar 9b. Fungsi keanggotaan *error rate* hasil optimasi genetik generasi keenam.



Gambar 9c. Fungsi keanggotaan *output rate*$_x$ hasil optimasi genetik generasi keenam.



Gambar 10. Hasil optimasi pengendali logika fuzi-genetik 21 bit, 3 *string* generasi VI

Populasi inisial diberikan secara acak. Masing-masing *string* diaplikasikan ke logika fuzi, kemudian diuji kinerjanya menggunakan fungsi objektif pada persamaan sebagai berikut.

$$f = \frac{1}{1 + 0{,}05 * (os + us) + 8 * (ss1 + ss2) + 2 * (u + d) + qos + qus}$$

untuk :

$os <= 0{,}5; us <= 0{,}5; qos <= 0{,}125; qus <= 0{,}125$;

qos/os <=0,25; qus/us <=0,25,

= 0 lainnya.....(1)

dengan : f : nilai fitness,

os : *overshoot*,

us : *undershoot*,

$ss1$: MSE *steady state error* keadaan1,

$ss2$: MSE *steady state error* keadaan 2,

u : waktu naik,

d : waktu turun,

qos : *second overshoot*, dan

qus : *second undershoot*.

Nilai fitness yang dihasilkan seperti dalam tabel dibagi seribu. Nilai fitness terbaik ada di *string* 3.

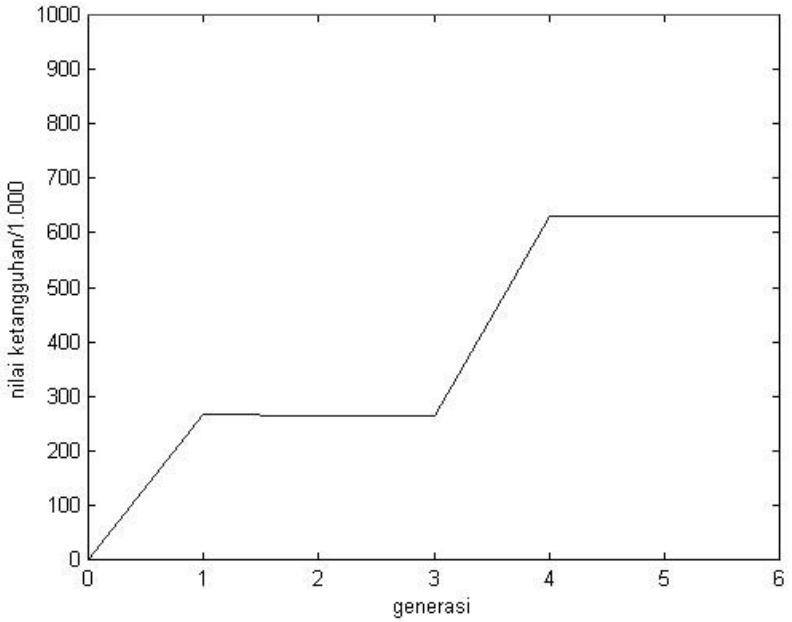
Operasi pertama adalah seleksi roda rolet. Karena *string* 3 memiliki nilai fitness terbaik, maka semua posisi didominasi oleh *string* 3 pada populasi inisial. Hal ini disebabkan karena banyaknya *string* yang sangat minim.

Operasi selanjutnya adalah persilangan satu titik dengan peluang satu. Karena semua *string* sama, maka praktis tidak terjadi persilangan.

Operasi terakhir adalah mutasi dengan peluang 0,02. Kebetulan salah satu bit dari *string* pertama dan ketiga mengalami mutasi gen. Hasil operasi ini dinyatakan sebagai hasil populasi generasi pertama. Terlihat bahwa di antara generasi pertama yang terbaik adalah *string* 2 dengan nilai fitness 0,2671. *String* tersebut kemudian diaplikasikan lagi dengan optimasi aturan (operasi mutasi dengan peluang 0,02).

Fungsi keanggotaan fuzi yang teroptimasi generasi pertama ada di gambar 7, aturan yang telah teroptimasi ada di tabel 1e, dan watak motor terkendali logika fuzi yang teroptimasi generasi pertama ada di gambar 8. Terlihat dari gambar 8 bahwa hasilnya masih meleset jauh dari *set point*-nya.

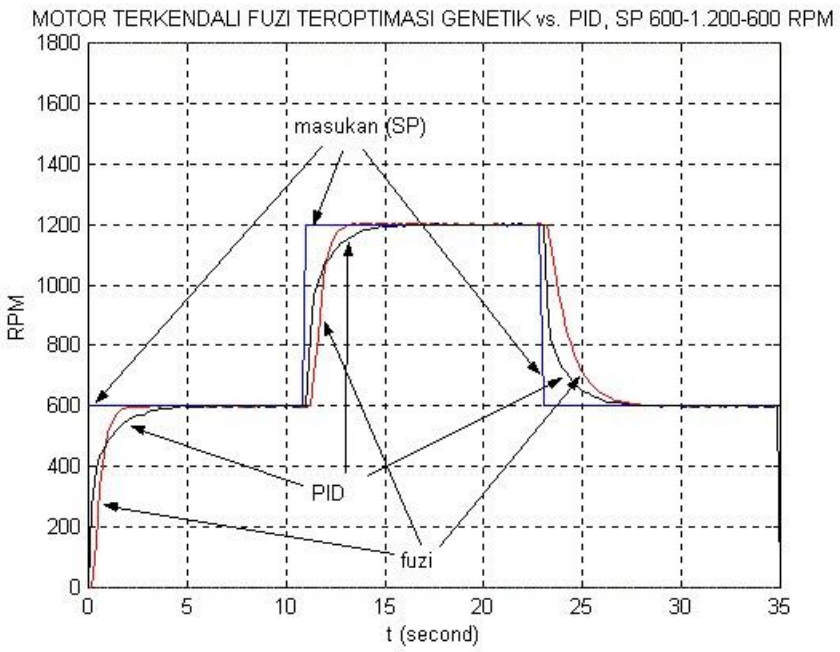
Proses optimasi dilanjutkan ke generasi kedua. Di sini yang diseleksi roda rolet adalah populasi hasil mutasi generasi pertama. Proses optimasi dilanjutkan terus sampai ke generasi keenam. Hasilnya di gambar 10 dengan nilai fitness 0,6908. Dari populasi generasi keenam terlihat bahwa semua *string* memiliki nilai fitness yang sama. Hal ini dapat dianggap sebagai keadaan sudah optimal sehingga proses optimasi dihentikan sampai dengan generasi keenam. Grafik nilai fitness fungsi generasi ada di gambar 11.



Gambar 11. Riwayat nilai fitness fungsi generasi.

Dari gambar 10 dan hasil pengukuran didapatkan bahwa motor tersebut tidak memiliki *overshoot* maupun *undershoot. Steady state error* saat *set point* 1.200 rpm adalah 0,09 %, dan saat *set point* turun ke 600 rpm adalah 0,13 %. *Error* sebesar itu termasuk sangat kecil sehingga dapat dianggap nol. Hal ini disebabkan oleh sifat integral pada keluaran pengendali fuzi yang mampu mengurangi *steady state error* sampai nol. Waktu naiknya adalah 1 detik, sedangkan waktu turunnya adalah 2,4 detik. Dibandingkan dengan watak kalang terbuka, waktu ini jauh lebih lama karena sifat integral pada keluaran pengendali fuzi, namun masih lebih cepat dari pengendali PID. Tetapi waktu turunnya justru lebih lama dari pengendali PID. Saat *start* juga tidak memiliki *overshoot*, dan waktu naik saat *start* adalah 0,89 detik. Hal ini merupakan kejanggalan karena untuk menggerakkan motor dari awal diperlukan torsi yang lebih besar. Rerata cuplikan untuk percobaan ini adalah 0,1 detik, hampir dua kali lipat dibandingkan dengan watak kalang terbuka, dan sedikit lebih cepat dari pengendali PID. Hal ini terjadi karena sistem inferensinya menggunakan *max-dot* yang berarti masih memerlukan operasi perkalian. Nilai fitness yang dihasilkan adalah 0,6908 yang

berarti masih sedikit lebih baik dari pengendali PID. Secara visual, perbandingan kinerja antara pengendali ini dan pengendali PID ada di gambar 12.



Gambar 12. Watak kalang tertutup kecepatan motor dc terkendali PID vs. fuzi teroptimasi genetik 21 bit, 3 *string* dengan *setpoint* 600; 1.200; 600 rpm.

IV Aplikasi Pada Optimasi Suatu Fungsi

Eksperimen diutamakan untuk mencoba jalannya operasi pada algoritma genetik dengan mengacu pada aplikasi logika kabur di sesi III.

Sebagai langkah pertama adalah inisialisasi populasi. Inisialisasi dilakukan dengan membentuk 10 populasi 32 bit secara acak menggunakan fungsi "genbin(32,10)" sehingga dihasilkan sebagai berikut.

$A_1 = 00111011001100001111101011100101$

$A_2 = 11001111011001000000110111010001$

$A_3 = 01001101011100111111100011101001$

$A_4 = 00001101011111100100000011000110$

$A_5 = 10000111110100100010001011100000$

$A_6 = 10010010000001000111011010111101$

$A_7 = 10111111011001101111110110010110$

$A_8 = 11110010001100100101100110011011$

$A_9 = 10011000011001011000001010010101$

$A_{10} = 00001111111010010111101000111100$

Sebagai langkah kedua adalah reproduksi berdasarkan nilai fitness yang proses seleksinya menggunakan roda roulette. Kemudian hasilnya dihitung nilai fitness dan ketangguhan totalnya. Pada simulasi ini, sebagai nilai fitness digunakan fungsi $f(x) = \frac{5}{x^2 + 2x + 0{,}1}$. Nilai fitness terbaik apabila $f(x)$ maksimum atau $y = \frac{1}{f(x)}$ minimum.

Sebagai langkah ketiga adalah operasi persilangan tunggal menggunakan peluang 0,9. Hasilnya juga dihitung nilai fitness dan ketangguhan totalnya.

Sebagai langkah keempat adalah operasi mutasi menggunakan peluang 0,01. Hasilnya juga dihitung nilai fitness dan ketangguhan totalnya. Setelah 10 generasi, kemudian semua nilai fitness total dibandingkan, dipilih yang terbaik. Di antara nilai fitness total yang terbaik, dipilih populasi yang nilai fitness terbaik. Hasilnya adalah sebagai berikut.

$A = \mathbf{00000000010110011011010100001101}$

Nilai fitnessnya adalah $39{,}1961^{-1}$=**0,0255**.