PENERAPAN ALGORITMA GENETIKA DAN IMPLEMENTASI DALAM MATLAB

Oleh

I MADE SUWIJA PUTRA, ST., MT

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS UDAYANA
DENPASAR – BALI
2018

KATA PENGHANTAR

Puji syukur kehadapan Tuhan Yang Maha Kuasa atas karuniaNya, pada kesempatan ini penulis dapat menyelesaikan makalah mengenai PENERAPAN ALGORITMA GENETIKA DAN IMPLEMENTASI DALAM MATLAB. Untuk memudahkan informasi ini diperoleh secara cepat bagi yang memerlukan, maka informasi ini lebih awal didokumentasikan di Perpustakaan Pusat Universitas Udayana Kampus Bukit Jimbaran.

Pada kesempatan ini penyusun menyampaikan ucapan terimakasih kepada para rekan-rekan dosen di Program Studi Teknologi Informasi, Fakultas Teknik Universitas Udayana karena atas dukungannya sehingga tulisan ini bisa selesai. Terimakasih pula kepada Validator Fakultas maupun Validator Universitas atas review tulisan ini di penyusuan Beban Kerja Dosen. Semoga tulisan bermanfaat bagi para pembaca terkait dengan bidang Soft Computing dan Data Mining

Denpasar, 21 Mei 2018

Penulis

I Made Suwija Putra

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

1 Pendahuluan

Algoritma genetik sebenarnya terinspirasi dari prinsip genetika dan seleksi alam(teori Darwin) yang ditemukan di Universitas Michigan, Amerika Serikat oleh John Holland melalui sebuah penelitian dan dipopulerkan oleh salah satu muridnya, David Goldberg menghasilkan buku berjudul "Adaption in Natural and Artificial Systems" pada tahun 1975. Konsep dasar algoritma genetika sebenarnya dirancang untuk mensimulasikan proses-proses dalam sistem alam yang diperlukan untuk evolusi, khusunya teori evolusi alam yang dicetuskan oleh Charles Darwin, yaitu survival of the fittest. Menurut teori ini, di alam terjadi persaingan antara individu-individu untuk memperebutkan sumber daya lam yang langka sehingga makhluk yang kuat mendominasi makhluk yang lemah (Hermawanto, 2007).

Algoritma genetika adalah suatu algoritma pencarian yang meniru mekanisme dari genetika alam. Algoritma Genetika banyak dipakai pada aplikasi bisnis, teknik maupun pada bidang keilmuan lainnya. Algoritma ini dimulai dengan kumpulan solusi yang disebut dengan populasi. Solusi-solusi dari sebuah populasi diambil dan digunakan untuk membentuk populasi yang baru. Hal ini dimotivasi dengan harapan bahwa populasi yang baru dibentuk tersebut akan lebih baik daripada yang lama. Solusi-solusi yang dipilih untuk membentuk solusisolusi yang baru dipilih sesuai dengan fitness mereka masing-masing (Maulik and Bandyopadhyay, 2000).

Pertahanan yang tinggi dari individu memberikan kesempatan untuk melakukan reproduksi melalui perkawinan silang dengan individu yang lain dalam populasi tersebut. Individu baru yang dihasilkan dalam hal ini dinamakan keturunan, yang membawa beberapa sifat induknya.Sedangkan individu dalam populasi yang tidak terseleksi dalam reproduksi akan mati dengan sendirinya. Dengan jalan ini, beberapa generasi dengan karakteristik yang bagus akan bermunculan dalam populasi tersebut, untuk kemudian dicampur dan ditukar dengan karakter lain. Dengan mengawinkan semakin banyak individu, maka akan semakin banyak kemungkinan terbaik yang dapat diperoleh.

Sebelum Algoritma Gentika dapat dijalankan, maka sebuah kode yang sesuai(representatif) untuk persoalan harus dirancang. Untuk ini maka titik solusi

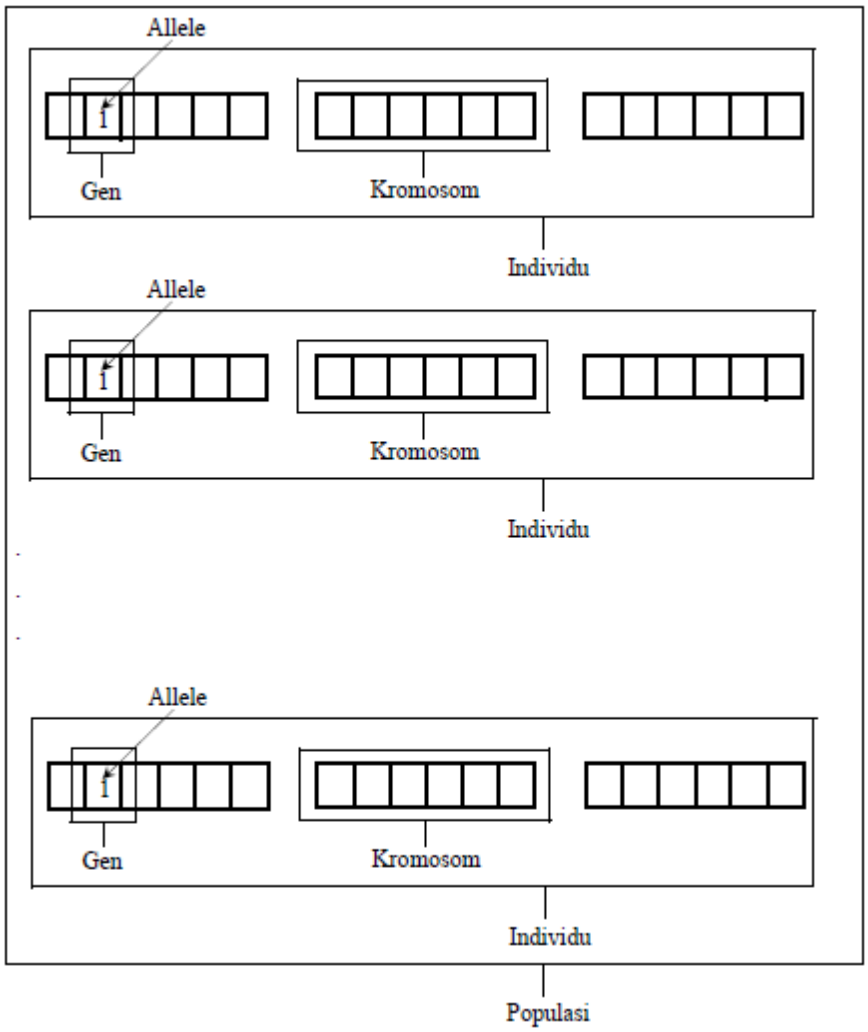
dalam ruang permasalahan dikodekan dalam bentuk kromosom/string yang terdiri atas komponen genetik terkecil yaitu gen. Dengan teori evolusi dan teori genetika, di dalam penerapan Algoritma Genetika akan melibatkan beberapa operator, yaitu

1. Operasi Evolusi yang melibatkan proses seleksi(selection) di dalamnya.
2. Operasi Genetika yang melibatkan operator pindah silang(crossover) dan muatsi(mutation).

Untuk memeriksa hasil optimasi, kita membutuhkan fungsi fitness, yang menandakan gambaran hasil(solusi) yang sudah dikodekan. Selama berjalan, induk harus digunakan untuk reproduksi, pindah silang dan mutasi untuk mencipatkan keturunan. Jika Algoritma Genetika didesain secara naik, populasi akan mengalami konvergensi dan akan didapatkan sebuah solusi yang optimum.

Beberapa pengertian dasar yang perlu diketahui (Whitley, 1994)

1. Gen(Genotype) adalah variabel dasar yang membentuk suatu kromosom. Dalam algoritma genetika, gen ini bisa bernilai biner, float, integer maupun karakter.
2. Alelle adalah nilai suatu gen, bisa berupa biner, float, integer maupun karakter.
3. Kromosom adalah gabungan dari gen-gen yang membentuk arti tertentu.
4. Individu adalah kumpulan gen, bisa dikatakan sama dengan kromosom. Individu menyatakan salah satu kemungkinan solusi dari suatu permasalahan.
5. Populasi adalah sekumpulan individu yang akan diproses secara bersama-sama dalam satu siklus proses evolusi.
6. Generasi menyatakan satu satuan siklus proses evolusi.
7. Nilai Fitness menyatakan seberapa baik nilai dari suatu individu atau solusi yang didapatkan. Nilai inilah yang dijadikan acuan untuk mencapai nilai optimal.



Gambar 1 Visualisasi gen, allele, kromosom, individu, dan populasi pada algoritma genetika

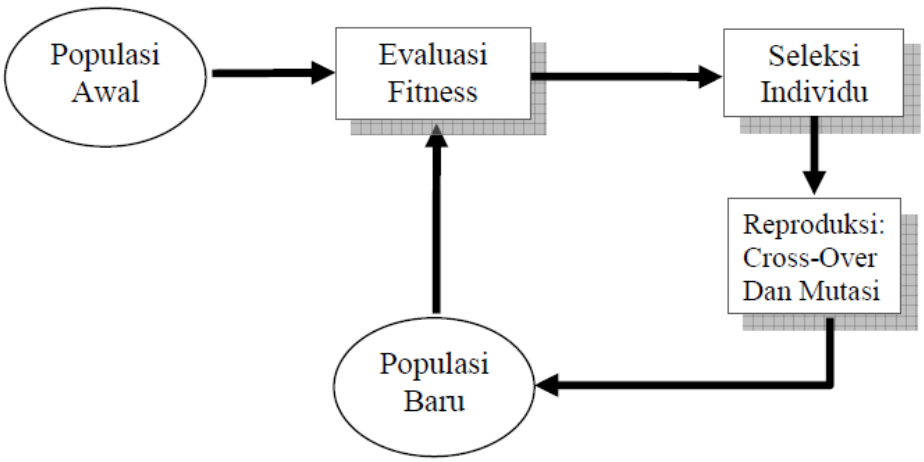
1.1 Hal-hal yang Harus Dilakukan Dalam Menggunakan Algoritma Genetika

1. Mendefinisikan individu, dimana individu menyatakan salah satu solusi (penyelesaian) yang mungkin dari permasalahan yang diangkat.
2. Mendefinisikan nilai fitness, yang merupakan ukuran baik-tidaknya sebuah individu atau baik-tidaknya solusi yang didapatkan.
3. Menentukan proses pembangkitan populasi awal. Hal ini biasanya dilakukan dengan menggunakan pembangkitan acak seperti random-walk.
4. Menentukan proses seleksi yang akan digunakan.

5. Menentukan proses perkawinan silang (cross-over) dan mutasi gen yang akan digunakan.

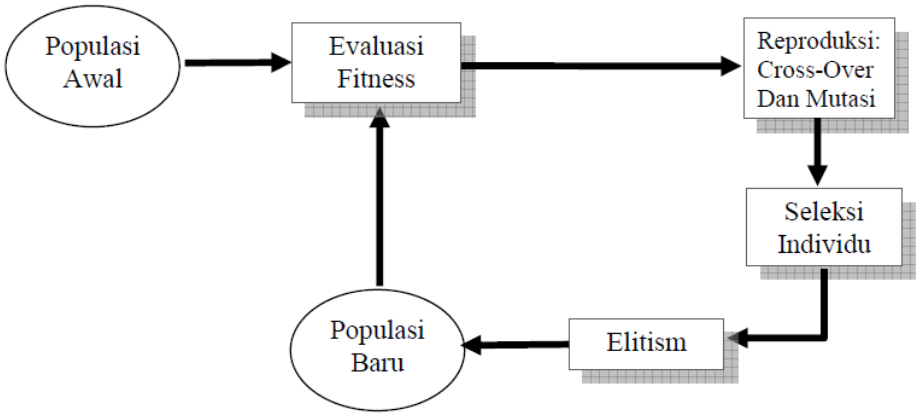
1.2 Siklus Algoritma Genetika

David Goldberg orang pertama kali memperkenalkan siklus algoritma genetika yang di gambarkan seperti gambar 2. Siklus dimulai dari membuat populasi awal secara acak, kemudian setiap individu dihitung nilai fitness nya. Proses berikutnya adalah menyelesi individu terbaik, kemudian dilakukan cross-over dan dilanjutkan oleh proses mutasi sehingga terbentuk populasi baru. Selanjutnya populasi baru ini mengalami siklus yang sama dengan populasi sebelumnya. Proses ini berlangsung terus hingga generasi ke –n.



Gambar 2 Siklus algoritma genetika yang diperkenalkan oleh David Goldberg

Siklus ini kemudian diperbaiki oleh Zbigniew Michalewicz dengan menambahkan satu proses elitisme dan membalik proses reproduksi terlebih dahulu, kemudian proses seleksi seperti tampak pada gambar 2.



Gambar 3 Siklus Algoritma Genetika Zbigniew Michalewics hasil perbaikan dari siklus algoritma genetika yang diperkenalkan oleh David Goldberg

2 Komponen-komponen Algoritma Genetika

Pada dasarnya Algoritma Genetika memilki 8 komponen, tetapi banyak metode yang bervariasi yang diusulkan pada masing-masing komponen tersebut. Masing-masing metode mempunyai kelebihan dan kekurangan. Suatu metode yang bagus untuk menyelesaikan masalah A belum tentu bagus untuk masalah B, atau bahkan bisa digunakan untuk masalah C. Untuk memudahkan cara kerja AG, pada subbab ini akan dibahas AG standar yang diapakai.

2.1 Teknik Encoding/Decoding Gen dan Individu

Encoding (pengkodean) berguna untuk mengkodekan nilai gen-gen pembentuk individu. Nilai-nilai gen ini diperoleh secara acak. Ada 3 pengkodean yang paling umum digunakan, yaitu :

1. pengkodean bilangan real : nilai gen berada dalam interval [0 1]

   contoh

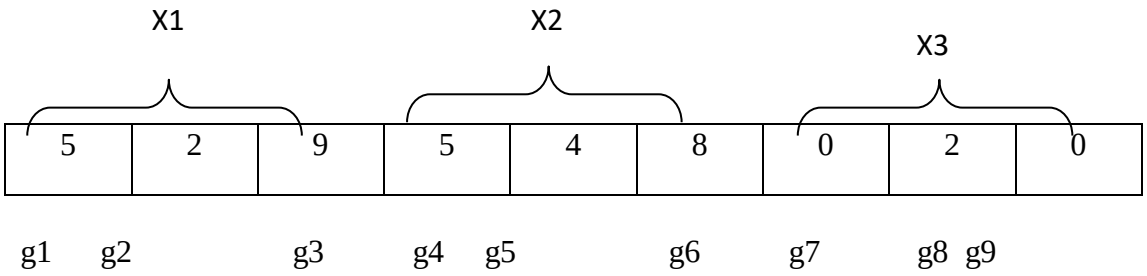
   3 variabel (x1,x2,x3) dikodekan ke dalam individu yang terdiri dari 3 gen

x1	x2	x3
0,2431	0,9846	0,5642
g1	g2	g3

2. Pengkodean diskrit desimal : nilai gen berupa bilangan bulat dalam interval [0 9]

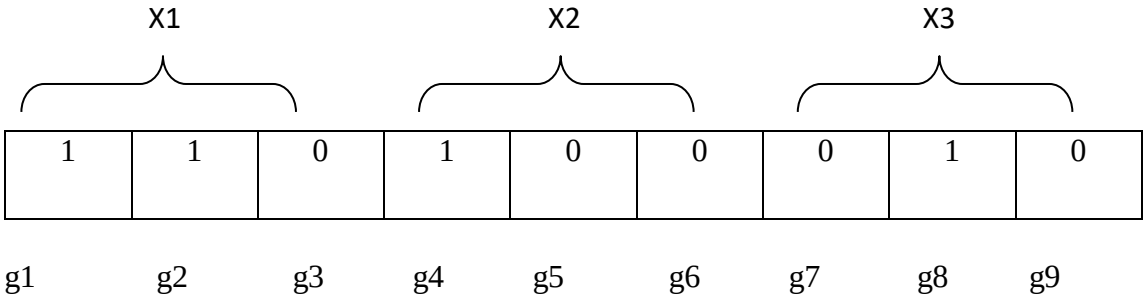
   Contoh

   3 Variabel (x1, x2, x3) dikodekan ke dalam individu yang terdiri dari 9 gen, tiap-tiap variabel dikodekan ke dalam 3 gen.



3. Pengkodean Biner : nilai gen berupa bilangan biner 0 atau 1

   3 Variabel (x1, x2, x3) dikodekan ke dalam individu yang terdiri dari 9 gen, tiap-tiap variabel dikodekan ke dalam 3 gen.



Decoding (pengkodean) berguna untuk mengkodekan gen-gen pembentuk individu agar nilainya tidak melebihi range yang telah ditentukan dan sekaligus menjadi nilai variabel yang akan dicari sebagai solusi permasalahan. Jika nilai variabel x yang telah dikodekan tersebut range-nya dirubah menjadi [ra rb], yaitu rb=batas bawah, ra = batas atas, maka cara untuk mengubah nilai-nilai variabel di atas hingga berada dalam range yang baru [rb ra], disebut Decoding (pengkodean)

1. Pendekodean Bilangan real :

   $X = rb + (r_a - r_b)\, g$

2. Pendekodean Diskrit Desimal :

   $X = rb + (r_a - r_b)\ (g_1 \times 10^{-1} + g_2 \times 10^{-2} + ... + g_n \times 10^{-n})$

3. Pendekodean Biner :

   $X = rb + (r_a - r_b)\ (g_1 \times 2^{-1} + g_2 \times 2^{-2} + ... + g_n \times 2^{-n})$

Dengan catatan bahwa N adalah jumlah gen dalam individu.

Contoh Teknik Encoding/Decoding

Misalkan diketahui fungsi $f(x1, x2, x3) = 2x_1 + x_2 + 4x_3$, nilai variabel x dibatasi dalam range [-4 6]. Tentukan besar nilai f[x1, x2, x3] jika :

Variabel [x1, x2, x3] diperoleh secara acak menggunakan pengkodean bilangan real berikut :

x1	x2	x3
0,2431	0,9846	0,5642
g1	g2	g3

Jawab :

Sebelum menghitung fungsi f(x1, x2, x3), tersebut dahulu range variabel harus diubah menjadi [-4 6] sehingga nilai-nilai variabel menjadi :

$X_1 = rb + (r_a - r_b)g_{1=} -4 + (6-(-4)) * 0,2431 = -1,569$

$X_2 = rb + (r_a - r_b)g_{2=} -4 + (6-(-4)) * 0,9846 = 5,846$

$X_3 = rb + (r_a - r_b)g_{3=} -4 + (6-(-4)) * 0,5642 = 1,642$

$f(x1, x2, x3) = 2x_1 + x_2 + 4x_3 = 2.(-5,569) + 5, 846 – 4.(1,642) = -3,86$

2.2 Membangkitkan Populasi Awal

Sebelum membangkitkan populasi awal, terlebih dahulu kita harus menentukan jumlah individu dalam populasi tersebut. Misalnya jumlah individu tersebut N. Setelah itu, baru kita membangkitkan populasi awal yang mempunyai N individu secara random.

2.3 Nilai Fitness

Nilai fitness menyatakan nilai dari fungsi tujuan. Tujuan dari algoritma genetika adalah memaksimalkan nilai fitness. Jika yang dicari nilai maksimal, maka nilai fitness adalah nilai dari fungsi itu sendiri. Tetapi jika yang dibutuhkan adalah nilai minimal, maka nilai fitness merupakan invers dari nilai fungsi itu sendiri. Proses invers dapat dilakukan dengan cara berikut.

Fitness = C — *f(x)* atau Fitness = $\frac{c}{f(x)\in}$.....(1)

C adalah konstanta dan ∈ adalah bilangan kecil yang ditentukan untuk menghindari agar tidak terjadi pembagian oleh nol dan x adalah individu.

Contoh Nilai Fitness

Diketahui fungsi f(x1,x2)=$2x^2$+$3x^2$. nilai x dibatasi pada [-2 4]

1. Buatlah populasi awal secara acak, yang terdiri dari 5 individu yang dikodekan secara biner.
2. Bila diinginkan mencari nilai maksimal fungsi f*(xi, xi),* tentukan individu mana yang mempunyai nilai terbaik
3. Bila diinginkan mencari nilai minimal fungsi *i(xi, xi),* tentukan individu mana yang mempunyai nilai terbaik

Jawab:

Karena fungsi tersebut mengandung 2 variabel, maka satu individu terdiri dari 2 variabel. Jumlah gen dalam satu variabel sembarang, untuk contoh ini sama dengan 3.

1. Membangkitkan populasi awa1 secara acak:

   Kode biner

   x1 x2

   1 0 0, 0 1 1 individu ke-1

   0 0 0, 0 0 0 individu ke-2

   1 0 1, 0 0 0 individu ke-3

   1 1 1 ,1 1 1 individu ke-4

   1 0 0, 1 0 0 individu ke-5

2. x1= -2+(4+2)(1x2^{-1} + 0x2^{-2}+.0x2^{-3})=1

x2 = -2+(4+2)(0x2^{-1} +0x2^{-2}+.lx2^{-3}) = -1,25

x1 = -2+(4+2)(0x2^{-1} +0x2^{-2} +.0x2^{-3})= - 2

x2 = -2+(4+2)(0x2^{-1}+0x2^{-2}+.0x2^{-3}) = -2

x1 = -2+(4+2)(1x2^{-1}+0x2-^{2}÷.1x2^{-3})=1,75

x2 = -2+(4+2)(0x2^{-1} +0x2^{-2}+.0x2^{-3})= -2

x1 = -2+(4+2)(1x2^{-1} +1x2^{-2} +.1x2^{-3})=3.25

x2 = -2+(4+2)(1x2^{-1} +1x2^{-2} +.1x2^{-3})=3.25

x1 = -2+(4+2)(1x2^{-1} +0x2^{-2} +.0x2^{-3})=1

x2 = -2+(4+2)(1x2^{-1} +0x2^{-2} +.0x2^{-3})=1

agar f(*x*1**,** *x2)* maksimal, maka nilai fitness f(x1,x2)=2x^{2} +3x^{2}

individu ke-1 nilai fitness = 2(1)^{2}+ 3(-1,25)^{2} = 6,6875

individu ke-2 nilai fitness = 2(-2)^{2}+ 3(-2)^{2} = 20

indivjdu ke-3 nilai fitness = 2(1,75)^{2}+ 3(-2)^{2} = 18,125

indivjdu ke-4 nilai fitness 2(3,25)^{2}+ 3(3,25)^{2} = 52,8125

individu ke-5 nilai fitness = 2(1)^{2}+ 3(1)^{2} = 5

nilai fitness terbesar dicapai oleh individu ke-4. Jadi individu ke-4 adalah yang terbaik.

3. Agar (x1,x2) minimal, maka nilai fitness = $\frac{1}{f(x1,x2)+\in}=\frac{1}{2x1^2+3x2^2+\in}$

dimisalkan nilai ∈= 0,1 maka:

individu ke-1 nilat fitness = $\frac{1}{2(1)^2+3(-1.25)^2+0,1}$ = 0,147

individu ke-2 nilai fitness = $\frac{1}{2(-2)^2+3(-2)^2+0,1}$ = 0,049

individu ke-3 nilai fitness = $\frac{1}{2(1,75)^2+3(-2)^2+0,1}$ = 0,054

individu ke-4 nilai fitness = $\frac{1}{2(3,25)^2+3(3,25)^2+0,1}$ = 0,019

individu ke-5 nilai fitness = $\frac{1}{2(1)^2+3(1)^2+0,1}$= 0, 196

Nilai fitness terbesar dicapai oleh individu ke-5. Jadi individu ke-5 adalah yang terbaik.

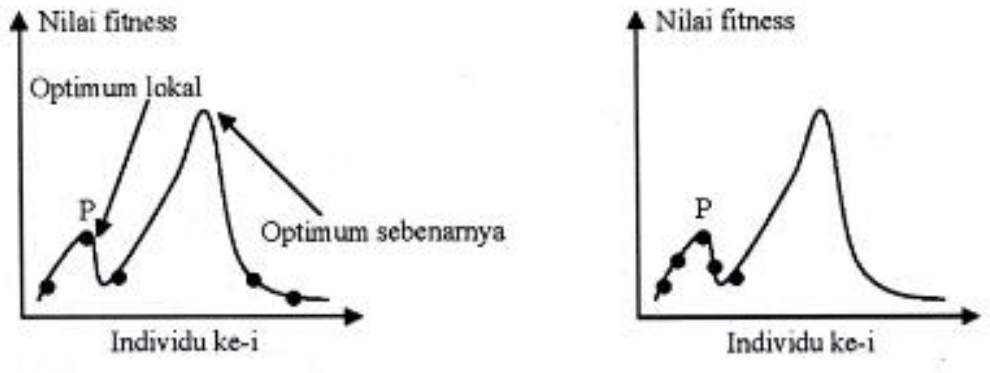
2.4 Elitisme

Elitisme adalah prosedur untuk meng-copy individu yang mempunyai nilai fitness tertinggi sebanyak satu (bila jumlah individu dalam suatu populasi adalah ganjil) atau dua (bila jumlah individu dalam suatu populasi adalah genap). Hal ini dilakukan agar individu ini tidak mengalami kerusakan (nilai fitness-nya menurun) selama proses pindah silang maupun mutasi. Misalnya pada contoh sebelumnya, maka individu ke-4 dibuat copy-nya sebanyak satu, kemudian disimpan dalam variabel temporer_individu.

temporer_individu =111, 111;

2.5 Seleksi

Kadang kala suatu fungsi tertentu dapat menyebabkan setiap individu mempunyai nilai fitness hampir sama. Hal ini bisa berakibat buruk pada saat dilakukan proses seleksi untuk memilih orangtua karena dapat menyebabkan optimum lokal. Gambar 4 menggambarkan bagaimana terjadinya optimum lokal.



Gambar 4 Pada gambar sebelah kiri, individu P mempunyai nilai fitness tertinggi dibanding individu lainnya. Pada gambar sebelah kanan, populasi konvergen pada suatu optimum lokal dekat individu P. Tidak ada individu yang mencapai nilal optimal sebenarnya

Pada gambar sebelah kiri, semua individu berada sangat jauh dari titik optimal sesungguhnya. Kebetulan ada satu individu P yang nilai fitness-nya tertinggi dibanding individu lainnya. Akibatnya individu P akan memproduksi

banyak anak. Pada generasi *tertentu* individu P dan anak-anaknya, melalui proses pindah silang dan mutasi, akan menghasilkan individu-individu yang mendekati lokal optimum (gambar sebelah kanan), sehingga terjadilah konvergensi prematur, di mana populasi konvergen pada suatu solusi optimal lokal. Permasalahan inilah yang sering muncul pada algoritma genetika.

Untuk menghindari hal ini, dibuatlah suatu mekanisme yang disebut dengan *Linier Fitness Rangking* (LFR). Tujuan dan mekanisme mi sebenarnya adalah untuk melakukan penskalaan nilai-nilai fitness dengan menggunakan persamaan berikut.

$$LFR(i) = f_{max} - (f_{max} - f_{min})\left(\frac{R(i)-1}{N-1}\right) \quad \text{.....(2)}$$

Dengan catatan bahwa:

LFR(i) = nilai LFR individu ke-i

N = jumlah individu dalam populasi

R(i) = rangking individu ke-i setelah diurutkan dan nilai fitness terbesar hingga terkecil

fmax = nilai fitness tertinggi

fmin = nilai fitness terendah

Contoh Seleksi

Pada contoh sebelumnya, untuk mendapatkan nilai $f(x_1, x_2)$ maksimal, telah dihitung nilai fltness tiap-tiap individu, yaitu:

individu ke-1 nilai fitness = 6,6875

individu ke-2 nilai fitness = 20

individu ke-3 nilai fitness = 18,125

individu ke-4 nilai fitness = 52,8125

individu ke-5 nilai fitness = 5

buatlah Linier Fitness Rangking-nya.

Jawab:

Pertama kali harus dilakukan sorting terhadap nilai fitness tersebut dan diperoleh:

Rangking ke-1	individu ke-4	nilai fitness = 52,8125
Rangking ke-2	individu ke-2	nilai fitness = 20
Rangking ke-3	individu ke-3	nilai fitness = 18,125
Rangking ke-4	individu ke-1	nilai fitness = 6,6875
Rangking ke-5	individu ke-5	nilai fitness = 5

f_{max} = 52,8 125 f_{min} =5

LFR(individu ke-1) = 52,8125 —(52,8125 — 5) $\left(\frac{4-1}{5-1}\right)$=16, 9531

LFR(individu ke-2) = 52,8125 —(52,8125 — 5)$\left(\frac{2-1}{5-1}\right)$ = 40,8593

LFR(individu ke-3) = 52,8125 —(52,8125 — 5)$\left(\frac{3-1}{5-1}\right)$= 28,9063

LFR(individu ke-4) = 52,8125 — (52,8125 — 5) $\left(\frac{1-1}{5-1}\right)$= 52,8125

LFR(individu ke-5) = 52,8125 —(52,8125 — 5)$\left(\frac{5-1}{5-1}\right)$ =5

Seleksi digunakan untuk memilih dua buah individu yang akan dijadikan orang tua, kemudian dilakukan pindah silang *(Cross-Over)* untuk mendapatkan keturunan yang baru. Metode seleksi yang sering digunakan adalah *Roulette- Wheel* (Roda Roulette). Dengan menggunakan Roda Roulette, masing-masing individu menempati potongan lingkaran roda secara proporsional sesuai dengan nilai fitnessnya.

Contoh Seleksi

Pada contoh sebelumnya telah dibuat LFR untuk masing-masing individu. Gunakan metode seleksi *Roulette- Wheel* untuk memilih dua orangtua yang akan digunakan untuk proses pindah silang.

Jawab :

Tabel 1 Seleksi Individu

Individu ke-	Nilai LFR	Jumlah kumulatif	Probabilitas
1	16,9531	16,9531	0,11
2	40,8593	57,8124	0,39
3	28,9063	86,7187	0,59
4	52,8125	139,5312	0,96
5	5	144,5312	1,00
Jumlah	144,5312		

Pemilihan dilakukan secara acak dengan membangkitkan bilangan random. Jika probabilitas individu ke-i < bilangan random, maka individu ke-i terpilih sebagai orangtua. Misalkan bilangan random pertama yang muncul adalah 0,6 dan yang kedua adalah 0,2. Maka individu yang terpilih adalah individu ke-3 dan ke-1 karena 0,59 < 0,6 dan 0,11 <0,2. Jadi,

Orangtua ke-1 = individu ke-3 = 1 0 1, 0 0 0

Orangtua ke-2 = individu ke-1 = 1 0 0, 0 1 1

2.6 Cross-Over (Pindah Silang)

Sebuah individu yang mengarah pada solusi optimal bisa diperoleh melalui proses pindah silang, dengan catatan bahwa pindah silang hanya bisa dilakukan jika sebuah bilangan random r dalam interval [0 1] yang dibangkitkan nilainya lebih kecil dari probabilitas tertentu prob, dengan kata lain : r < prob. Biasanya nilai prob diset mendekati 1. Cara yang paling sederhana untuk melakukan pindah silang adalah pindah silang satu titik potong. Posisi titik potong dilakukan secara random.

Contoh Cross-Over

Lakukan proses pindah silang untuk mendapatkan populasi yang baru, jika nilai prob = 0,8.

Jawab:

Pada saat proses elitisme, individu ke-4 telah di-copy satu kali, hasilnya disimpan dalam variabel temporari_individu,

temporer_individu = 111, 111;

sehingga kita tinggal menentukan 4 anak keturunan lagi.

Misalkan bilangan random yang dibangkitkan adalah 0,6 dan 0,2 maka:

Bilangan random	probabilitas	individu terpilih	kode biner
0,6	0,59 < 0,6	individuke-3	101, 000
0,2	0,11 < 0,2	individu ke-1	100, 011

Bangkitkan bilangan random r, misalnya r = 0,4. Karena r < prob, lakukan proses pindah silang. Bangkitkan bilangan random antara 1 s/d 6 (jumlah gen) untuk menentukan posisi titik pindah silang. Misalnya bilangan random itu 3, artinya gen-gen ke 1 s/d 3 dan orangtua dipindah silang, maka

individu ke-3 ,	101, 000
individu ke-1	100, 011

hasil pindah silang:

anakke-1,	100, 000
anak ke-2,	101,011

simpan hasilnya ke dalam variabel temporer_individu, temporer_individu= 111, 111; 100,000; 101,011; pilih 2 orangtua lagi untuk dilakukan pindah silang hingga didapatkan 2 anak lagi.Misalkan bilangan random yang dibangkitkan adalah 0,40 dan 1,00 maka:

Bilangan random	probabilitas	individu terpilih	kode biner
0,40	0,39 < 0,40	individu ke-2	0 0 0, 0 0 0
1,00	1,00 <1,00	individu ke-5	1 0 0, 1 0 0

Bangkitkan bilangan random r, misalnya r = 0,9. Karena r > prob, maka proses pindah silang tidak perlu dilakukan. Jadi,

anak ke-1 = individu ke-2 0 0 0, 0 0 0
anak ke-2 = individu ke-5 1 0 0, 1 0 0

simpan hasilnya ke dalam variabel temporer_individu,

temporer_individu = 111,111; 100,000; 101,011; 000,000; 100,100

2.7 Mutasi

Mutasi dilakukan untuk semua gen yang terdapat pada individu, jika bilangan random yang dibangkitkan lebih kecil dan probabilitas mutasi p yang ditentukan. Biasanya p diset = 1/N, di mana N adalah jumlah gen dalam individu. Untuk kode biner, mutasi dilakukan dengan cara membalik nilai bit 0 menjadi bit 1, sebaliknya bit 1 diubah menjadi bit 0.

Contoh Mutasi

Pada contoh sebelumnya, lakukan proses mutasi untuk mendapatkan populasi yang baru, jika nilai probabilitas mutasi p = 1/6.

Jawab:

untuk menjaga proses elitisme, temporer_individu ke-1 tidak ikut dimutasi. Jadi kita tinggal memproses 4 temporer_individu sisanya untuk dilakukan proses mutasi. Misalkan bilangan random yang dibangkitkan untuk setiap gen dan setiap individu seperti berikut, maka:

Bilangan random	: 1/5	1/8	1/4	1/3	1/3	1/2
temporer_individu(2)	: 1	0	0,	0	0	0;
hasil mutasi	: 1	1	0,	0	0	0;

Bilangan random	: 1/4	1/4	1/5	1/3	1/7	1/3
temporer_indivjdu(3)	: 1	0	1,	0	1	1;
hasil mutasi	: 1	0	1,	0	0	1;

Bilangan random	: 1/4	1/7	1/2	1/8	1/5	1/3
temporer_individu(4)	: 0	0	0,	0	0	0;
hasil mutasi	: 0	1	0,	1	0	0;

Bilangan random	: 1/9	1/4	1/5	1/3	1/4	1/2
temporer_individu(5)	: 1	0	0,	1	0	0;
has ii mutasi	: 0	0	0,	1	0	0;

Gen yang mengalami mutasi hanyalah gen yang bilangan randomnya < p. Simpan hasil mutasi ke dalam variabel temporer_individu,
temporer_individu = 111,111; 110,000; 101,001; 010,100; 000,100

2.8 Penggantian Populasi

Penggantian populasi *(generational replacement)* dimaksudkan bahwa semua individu awal dan satu generasi diganti oleh temporer_indivjdu hasil proses pindah silang dan mutasi.

Contoh Penggantian Populasi

Penggantian populasi dilakukan dengan cara mengisi variabel individu pada populasi awal dengan nilai-nilai yang ada pada temperatur_individu, yaitu: individu = temporerr_individu sehingga nilai-nilai individu dalam populasi yang baru menjadi:

Kode biner

x1	x2	
1 1 1,	1 1 1	individuke-1
1 1 0,	0 0 0	individu ke-2
1 0 1,	0 0 1	individu ke-3
0 1 0,	1 0 0	individu ke-4
0 0 0,	1 0 0	individuke-5

Setelah dilakukan penggantian populasi, maka populasi baru mi disebut sebagai generasi kedua dan siap diproses lagi hingga generasi ke-n.

3 Teknik Cross-Over (Pindah Silang)

Berdasarkan tipe data dan nilai gen, ada beberapa cara untuk melakukan pindah silang. yaitu:

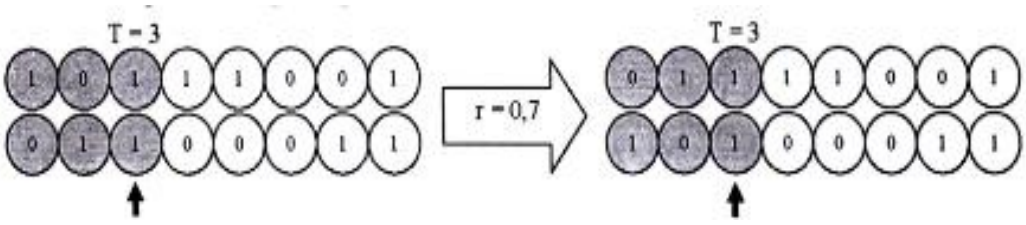
3.1 Gen bertipe data Biner

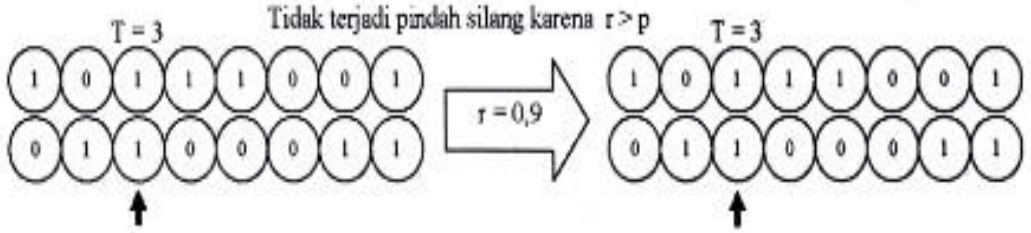
Pindah silang melibatkan dua buah individu sebagai orangtua dan akan menghasilkan dua anak sebagai individu baru. Prinsip dan pindah silang ini adalah menentukan posisi titik potong T (posisi gen) secara random, yang dipakai sebagai acuan untuk menukar gen-gen orangtua sehingga dihasilkan gen-gen anak. Jika bilangan random yang dibangkitkan r lebih kecil daripada probabilitas pindah silang p (r < p), maka proses pindah silang dikerjakan.

Contoh

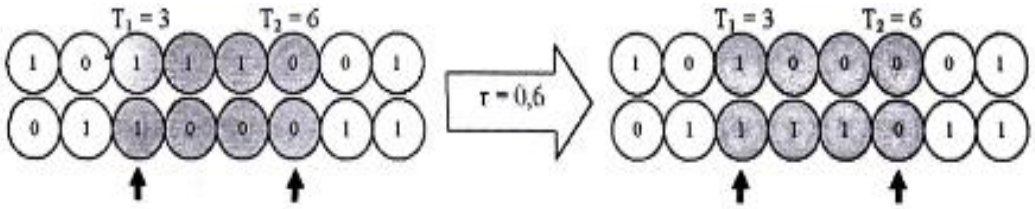
Misalkan ditentukan p = 0,8

Pindah silang satu titik potong:

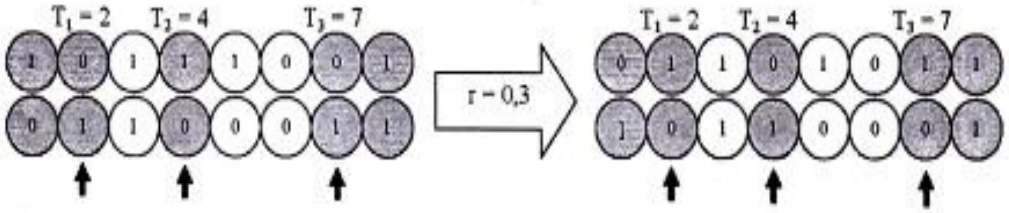




Pindah silang dua titik potong:



Pindah silang tiga titik potong:



3.2 Gen bertipe data Real

Untuk gen bertipe real cara pindah silangnya mengikuti persamaan berikut.

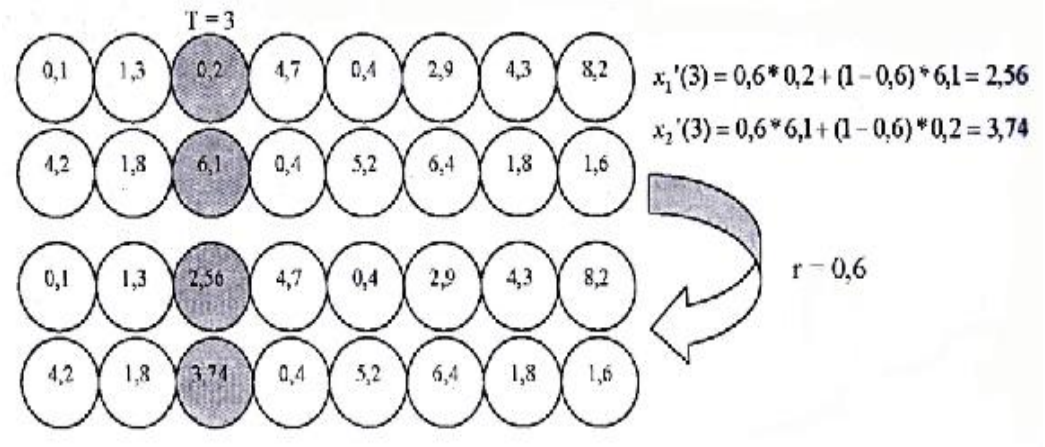
$x_1'(T) = r * x_1(T) + (r - 1).x_2(T)$

$x_2'(T) = r * x_2(T) + (r - 1).x_1(T)$

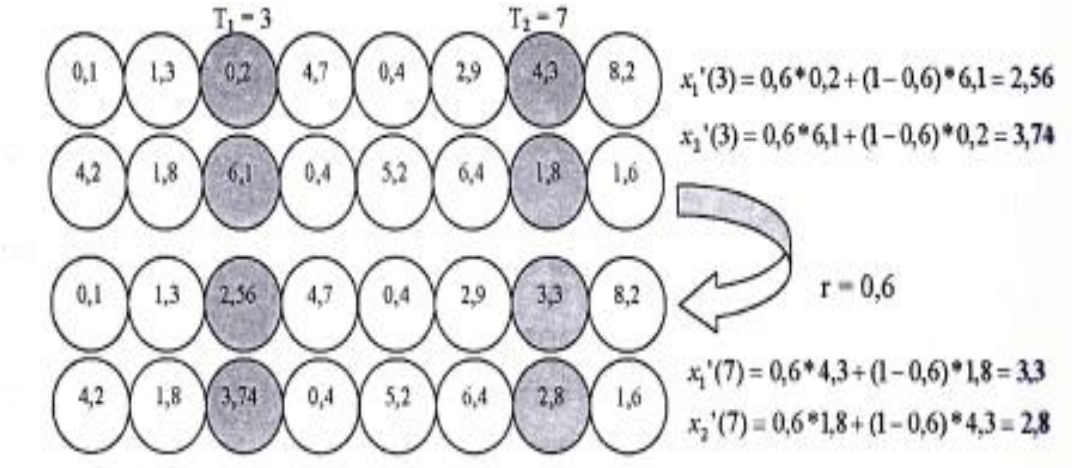
T adalah posisi gen yang mengalami mutasi r adalah bilangan random [0 1].

Contoh

Pindah silang satu titik :



Pindah silang dua titik potong :



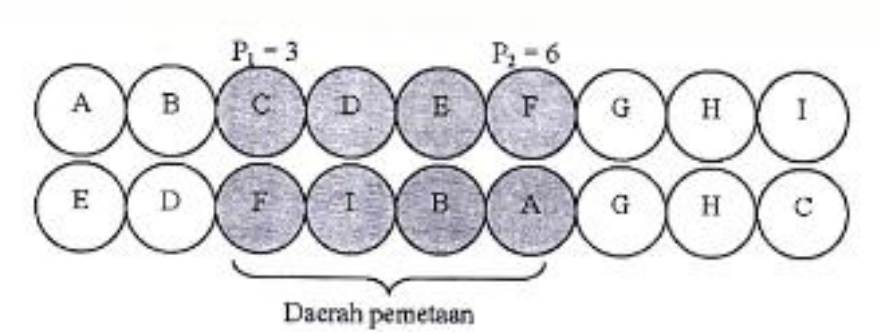
3.3 Gen bertipe data Permutasi

Bila tipe data dan gen adalah permutasi (contohnya pada Travels Salesman Problem) kita bisa gunakan metode *Partial-Mapped CrossOver* (PMX). Metode ini diciptakan ole Goldberg dan Lingle dengan langkah-langkah berikut.

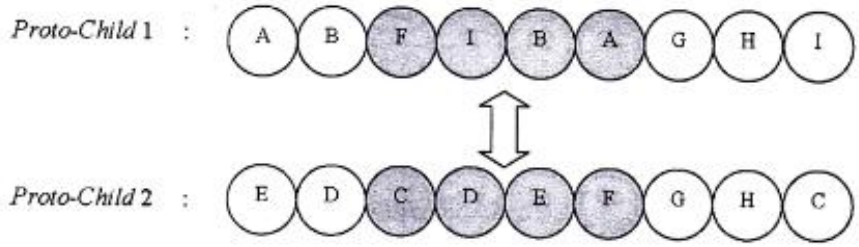
a. Tentukan dua posisi (P1 dan P2) pada individu secara random. Substring yang berada dalam dua posisi ini disebut daerah pemetaan.
b. Tukar dua substring antar dua induk untuk menghasilkan proto-child
c. Di antara daerah dua pemetaan, tentukan hubungannya.
d. Mengacu pada hubungan pemetaan tadi, tentukan individu keturunannya.

Contoh

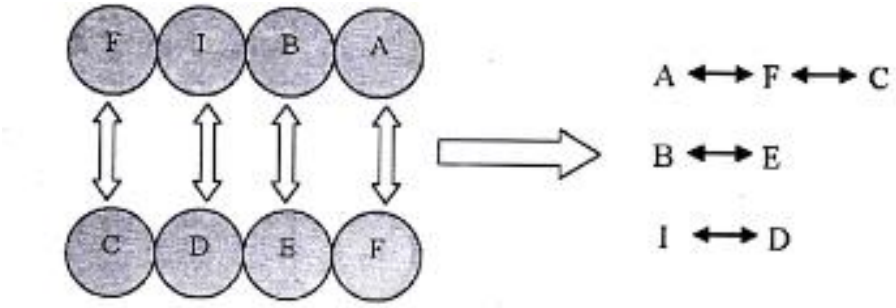
a. Tentukan dua posisi (P1 dan P2) pada individu secara random



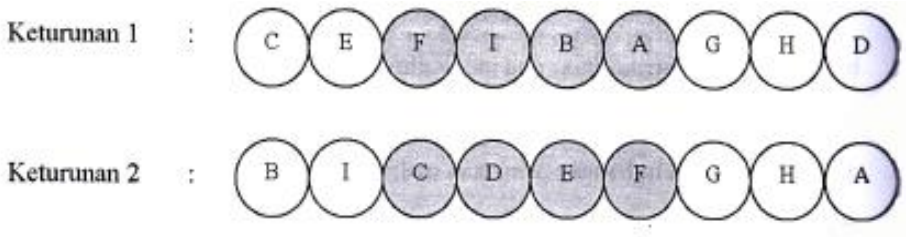
b. Tukar dua substring antar dua induk untuk menghasilkan *photo-child*



c. Di antara daerah dua pemetaan, tentukan hubungannya.



d. Mengacu pada hubungan pemetaan tadi, tentukan individu keturunannya.



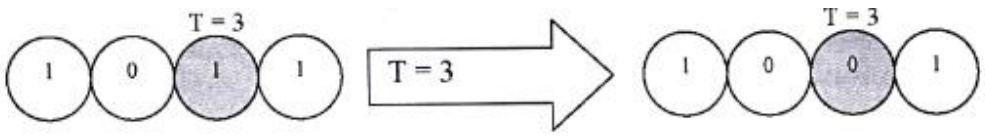
4 Teknik Mutasi

Pada gen yang mempunyai tipe data biner, mutasi dilakukan dengan menukar bit 0 menjadi 1, dan bit 1 menjadi 0. Untuk gen yang mempunyai tipe data real, mutasi dilakukan dengan cara menggeser nilai gen termutasi (pada posisi T,

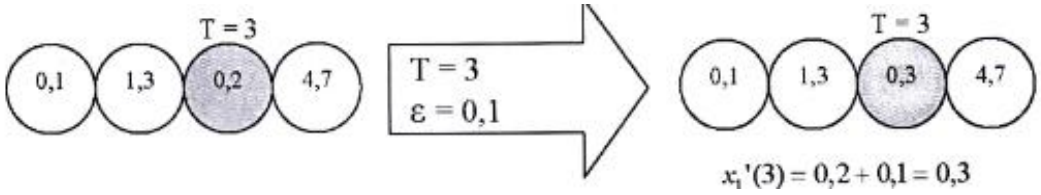
dimana T = random) sebesar bilangan kecil E yang ditentukan dalam interval [0 1]. Nilai gen yang barn adalah: x'(r) = x(r) + E

Contoh Teknik Mutasi

Gen bertipe data biner



Gen betipe data real



5 Algoritma Genetik Lanjutan

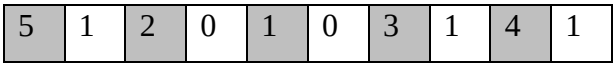
5.1 Skema Pengkodean

Pada masalah tertentu, skema pengkodean *binary encoding* menghadapi masalah serius. Pada masalah lain, seperti traveling salesman problem, binary encoding tidak dapat digunakan. Pada subbab ini akan dibahas beberapa skema pengkodean lain.

5.1.1 Pengkodean Rumit (Messy Encoding)

Pada schema theorem dijelaskan bahwa suatu schema dengan defining length yang panjang akan memiliki probabilitas tinggi untuk dirusak. Sebagai contoh, schema S_1 = lxxx0l akan lebih mudah rusak, selama proses pindah silang, dibandingkan dengan schema S_2 = 101xxx. Tetapi, jika schema S_1 memiliki nilai fitness tinggi, sangat disayangkan jika schema ini rusak. Padahal schema bernilai fitness tinggi seharusnya dipertahankan. Bagaimana agar schema S_1 tidak mudah rusak? Suatu meode pengkodean rumit atau **messy encoding** bisa digunakan untuk masalah ini. Idenya adalah dengan membuat representasi kromosom yang tidak bergantung pada posisi. Jika schema S_1 diubah menjadi xxx01 1, maka schema

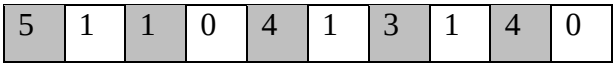
tersebut tidak akan mudah rusak karena defining length menjadi lebih pendek. Perhatikan gambar di bawah ini.



Gambar 5 Representasi kromosom menggunakan messy encoding

Angka pertama dalam setiap pasangan (warna abu-abu) diterjemahkan sebagai posisi gen, sedangkan angka kedua (warna putih) menunjukkan nilai gen. Pada gambar 4 di atas, posisi 5 berisi 1, posisi 2 berisi 0, posisi 1 berisi 0, posisi 3 berisi 1, dan posisi 4 berisi 1. Jadi, kromosom pada gambar tersebut merepresentasikan 00111. Dengan demikian, jika schema S = 1xxx0 mcmiliki nilai fitness tinggi, maka schema tersebut tidak akan mudah rusak jika direpresentasikan dalam messy encoding, dimana gen 1 dan gen 5 bertetangga. Hal mi tidak bisa dilakukan dalam binary encoding biasa.

Namun ada dua masalah dalam *messy encoding*. Pertama, adanya kemungkinan kromosom akan bcnsi beberapa posisi gen yang sama tetapi nilai gen-nya berbeda. Kedua, posisi gen tidak direpresentasikan sama sekali.



Gambar 6 Sebuah masalah dalam messy encoding. Terdapat dua posisi gen 4 dengan nilaigen berbeda. Sebaliknya posisi gen 2 tidak direpresentaskan sama sekali

Penyelesaian masalah pertama adalah dengan mengambil posisi gen yang pertama kali. Pada Gambar 6 di atas diputuskan bahwa posisi gen 4 adalah bernilai 1, dan posisi 4 yang bemiiai 0 diabaikan. Masalah kedua dapat diselesaikan dengan membuat kromosom berukuran beberapa kali lebih panjang, misalkan 5 kali sehingga panjang kromosom pada Gambar 6 di atas berubah dari 10 menjadi 50. Jika dengan panjang 5 kali temyata masih ada posisi gen yang tidak terepresentasikan maka nilai posisi gen terscbut dibangkitkan secara random.

5.1.2 Pengkodean dengan Panjang Bervariasi

AG standar menggunakan kromosom dengan binary encoding yang panjangnya tetap. Hal ini sangat bengantung pada masalah yang akan

diselesaikan.Selanjutnya dijelaskan bahwa binary encoding dengan jumlah bit yang lebih banyak akan menghasilkan solusi yang lebih akurat. Untuk masalah yang sederhana dcngan jumlah variabel yang akan dioptimasi dan tingkat akurasinya sudah dikctahui, sangatlah mudah untuk menentukan panjang kromosom. Tetapi tidak semua masalah memiliki karakteristik seperti ini. Untuk masalah-masalah di mana akurasi yang diinginkan tidak diketahui, pendekatan yang aman adalah dengan menggunakan jumlah bit yang panjang, misalkan 50 untuk setiap variabel. Tetapi hal ini tentu saja akan mempengaruhi kecepatan proses dari AG secara signifikan karena kromosom yang panjang memperlama proses AG.

Dalam situasi tertentu terkadang diperlukan penggunaan kromosom dengan panjang yang bervariasi. Pada subbab biasanya dipergunakan untuk masalah Jaringan Syaraf Tiruan dan Traveling Salesman Problem.

5.1.3 Grammatical Encoding

Otak manusia merupakan suatu komputer sangat kompleks yang terdiri dari sekitar 10^{11} elemen komputasi(neurons). Terdapat sekitar 10^{14} sampai 10^{15} koneksi antar neuron, atau sekitar 1000 sampai 10000 koneksi per neuron. Jika setiap koneksi dikodekan ke dalam kromosom, maka informasi yang mengisi kromosom akan sekitar 10^5 GB, di mana bobot-bobot sinaptik dikodekan menggunakan hanya 1 byte. Tetapi pada kenyataannya ukuran genome manusia hanya sekitar 3 GB. Oleh karena itu para peneliti percaya bahwa pengkodean otak manusia bukanlah menggunakan pengkodean langsung, melainkan pengkodean prosedur di mana otak dibentuk.

Di dalam AG, skema pengkodean yang mengkodekan suatu prosedur untuk membangun JST misalnya, dikenal sebagai *gramatical encoding schemes*. Pada skema ini kromosom dipandang sebagai kalimat yang diekspresikan menggunakan grammar (tata bahasa). Ketika sebuah kalimat dibaca (atau kromosom didekodekan), maka individu dibangkitkan menggunakan tata bahasa tersebut.

Dalam skema Kitano, setiap kromosom dikodekan ke dalam suatu string seperti pada Gambar 7 di bawah ini. Skema ini diaplikasikan untuk kasus khusus, yaitu FFNN yang berisi maksimum 8 neuron.

S	ACBA	Aadfb	Bbefd	Dfanp	Bahjm	Ckhgf	⋱

Gambar 7 Kromosom menggunakan gramatical encoding

S dalam kromosom adalah start symbol atau simbol awal. Ketika dibaca, S akan membangkitkan sebuah matriks yang bensi 4 simbol (persamaan 3).

$$S \rightarrow \begin{bmatrix} A & C \\ B & A \end{bmatrix} \quad \text{..... (3)}$$

Dalam kasus ini berisi A, C, B, dan A. Tetapi urutan keempat simbol tidak selalu tetap dalam matriks tersebut. S dan keempat simbol tersebut disebut sebagai non-terminals yaitu simbol-simbo1 yang ketika dibaca, akan membangkitkan struktur lain yang berisi non-terminals maupun terminals, Sedangkan terminals adalah simbol-simbol yang tidak akan diproses lebih lanjut. Kitano menggunakan 26 huruf kapital, A sampai Z,untuk non-terminals sehingga kombinasi simbol non-terminals berjumla 26^4. Setiap huruf kapital ini mengkodekan suatu matriks berukuran 2x2 yang berisi 4 huruf kecil. Keempat huruf kecil ini diambil dari 16 huruf yang disediakan yaitu a sampai p sehingga jumlah kombinasi untuk simbol terminals adalah 16^4. Aturan pendekodean huruf kapital diambil dari empat huruf kecil di sebelah kanannya. Dan Gambar 6 diperoleh bahwa pendekodean untuk A dan B, adalah sebagai berikut:

$$A \rightarrow \begin{bmatrix} a & d \\ f & b \end{bmatrix} \text{.....} \quad \text{(4)}$$

$$B \rightarrow \begin{bmatrix} b & e \\ f & d \end{bmatrix} \quad \text{..... (5)}$$

Selanjutnya, setiap huruf kecil mengkodekan suatu matriks khusus berukuran 2x2 yang berisi bilangan biner dengan aturan sebagai berikut. Perlu diperhatikan bahwa setiap huruf kecil mengkodekan matriks yang tetap.

$$a \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, b \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, c \rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \dots p \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{..... (6)}$$

Pada skema pengkodean ini dimungkinkan adanya huruf besar yang muncul lebih dari satu kali. Pada Gambar 6 di atas, huruf besar B muncul dua kali. Yang pertama adalah ‘Bbefd’dan yang ke dua ‘Bahjm’.Dalam kasus ini yang digunakan adalah yang pertama kali muncul (paling kiri), yaitu ‘Bbefd’.

Jadi prosedur pada *gramatical encoding schemes* selengkapnya adalah: Pertama, bangkitkan sebuah matriks 2x2 yang berisi empat huruf besar. Langkah kedua, bangkitkan matriks 4x4 yang berisi 16 huruf kecil. Dan langkah ketiga, bangkitkan matriks 8x8 yang berisi bilangan biner, 0 dan 1.

Berdasarkan kromosom di atas, setelah langkah pertama maka akan diperoleh matriks 2x2 sebagai berikut :

$$\begin{bmatrix} A & C \\ B & A \end{bmatrix}$$

Setelah langkah kedua diperoleh matriks 4x4 sebagai berikut:

$$\begin{bmatrix} a & d & k & h \\ f & b & g & f \\ b & e & a & d \\ f & d & f & b \end{bmatrix}$$

Dan akhirnya, setelah langkah ketiga diperoleh matriks 4x4 sebagai berikut:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Matriks 8x8 inilah yang digunakan untuk merepresentasikan JST. Elemen-elemen diagonal digunakan untuk menunjukkan keberadaan neuron (jika bernilai 1) dan ketidakberadaan neuron (jika bernilai 0). Elemen-elemen yang tidak berada pada posisi diagonal digunakan untuk menunjukkan koneksi antarneuron. Nilai 1 menunjukkan adanya koneksi, sedangkan 0 menunjukkan tidak adanya koneksi.

5.2 Seleksi Orang Tua

Selain *roulette-wheel* dan tournament, seleksi orang tua dapat dilakukan dengan berbagai macam cara. Pada subbab ini hanya dibahas dua skema seleksi, yaitu seleksi Boltzmann dan seleksi kompetitif.

5.2.1 Tournament Selection

Pada seleksi alami yang terjadi di dunia nyata, beberapa individu (biasanya individu jantan) berkompetisi dalam sebuah kelompok kecil sampai tersisa hanya satu individu pemenang. Individu pemenang inilah yang bisa kawin (pindah silang). Metode *roulette-wheel* selection tidak mengakomodasi masalah ini. Sebuah metode lain yang disebut *tournament selection* mencoba mengadopsi karakteristik alami ini. Dalam bentuk paling sederhana metode mi mengarnbil dua kromosom secara random dan kemudian menyeleksi salah satu yang bernilai fitness paling tinggi untuk menjadi orang tua pertama. Cara yang sama dilakukan lagi untuk mendapatkan orang tua yang kedua. Metode *tournament selection* yang lebih rumit adalah dengan mengambil m kromosom secara random. Kemudian kromosom bernilai fitness tertinggi dipilih sebagai orang tua pertama jika bilangan random yang dibangkitkan kurang dari suatu nilai batas yang ditentukan p dalam interval [0,1]. Pemilihan orang tua akan dilakukan secara random dan m – 1 kromosom yang ada jika bilangan random yang dibangkitkan lebih dai atau sama dengan p. Pada tournament selection, variable m adalah *tournament size* dan p adalah *tournament probability*.Biasanya m diset sebagai suata nilai yang sangat kecil, misal 4 atau 5. Sedangkan p biasanya diset sekitar 0,75.

5.2.2 Seleksi Boltzmann

Skema seleksi ini didasari oleh konsep fisika tentang pemanasan dan pendinginan atau annealing. Pada skema seleksi ini digunakan parameter T (*temperature*) untuk mengatur tingkatan nilai fitness individu. Terdapat banyak versi yang dapat digunakan untuk mengimplementasikan skema ini. Salah satunya adalah dengan memilih individu dan sepasang individu yang dipilih secara random, berdasarkan fungsi b berikut ini:

$$b(f_{j1},f_{j2}) = \frac{1}{1+e^{\frac{1}{T}(\frac{1}{f_{j1}}+\frac{1}{f_{j1}})}} \qquad \text{.....(7)}$$

di mana f_{j1}, dan f_{j2} adalah nilai-nilai fitness dan sepasang individu yang dipilih secara random tersebut. Selama seleksi, sebuah bilangan random *r* dibangkitkan dan individu *j* dipilih berdasarkan :

$$j = \begin{cases} j_1 & \text{jika b } (f_{i1}, f_{i2}) > r \\ j_2 & \text{yang lainnya} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(8)}$$

Jika *T* bernilai kecil mendekati 0, dan $f_{j1} > f_{j2}$, maka individu j_1 akan terpilih dengan probabilitas mendekati 1. Sebaliknya, jika *T* bernilai besar, maka individu j_1 dan j_2 akan memiliki probabilitas yang hampir sama untuk terpilih, tanpa terpengaruh oleh perbedaan nilai fitness dan kedua individu tersebut. Biasanya, pada skema seleksi ini, *T* awal diberi nilai yang besar. Tujuannya adalah agar AG dapat menjelajahi ruang pencarian sebanyak mungkin tanpa terpengaruh nilai-nilai fitness. Pada generasi-generasi berikutnya, nilai *T* diperkecil secara bertahap, dengan harapan AG dapat menemukan solusi yang 'bagus' dengan lebih cepat.

Satu alternatif lain dan skema seleksi Boltzmann adalah memilih individu dengan probabilitas p_j berikut ini:

$$p_j(f_j) = \frac{e^{\frac{f_j}{T}}}{\sum_{k=1}^{N} e^{\frac{f_k}{T}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(9)}$$

di mana f_k menyatakan nilai fitness dan individu *k* dan *N* adalah jumlah individu dalam populasi. Dengan skema alternatif ini, individu-individu akan dipilih dengan probabilitas yang hampir sama jika *T* bernilai besar. Sedangkan jika *T* benilai kecil, maka individu bernilai fitness tinggi memiliki peluang Lebih besar untuk terpilih sesuai dengan persamaan 9 di atas.

5.2.3 Seleksi Kompetitif dan Co-evolution

Sejauh ini proses seleksi yang digunakan selalu berdasarkan fungsi fitness yang sudah didefinisikan sebelum AG dijalankan. Artinya, fungsi fitness yang digunakan adalah mutlak dan selalu tetap selama evolusi. Hal ini berbeda dengan fungsi fitness pada sistem biologi yang sebenamya. Di dunia nyata tidak ada nilai

fitness yang mutlak. Nilai fitness dan suatu individu bergantung pada individu-individu lain yang berada dalam spesies yang sama maupun dalam spesies berbeda. Hal inilah yang mendasari suatu skema seleksi yang disebut *competitive selection* (seleksi kompetitif). Skema seleksi ini selalu dihubungkan dengan istilah *co-evolution* (evolusi bersamaan), yaitu evolusi dua atau lebih spesies dalam waktu yang bersamaan. Di dunia nyata, *co-evolution* merupakan fenomena yang senang terjadi. Misalnya, gigi taring pemangsa (*predator*) berevolusi menjadi lebih tajam sebagai akibat dari kulit mangsa (*prey*) yang berevolusi menjadi lebih tebal. Sebaliknya, kulit mangsa berevolusi menjadi lebih tebal untuk bertahan hidup dan berlindung dan gigi taring pemangsa yang semakin tajam. Demikianlah evolusi ini berulang terus sehingga nilai fitness selalu berubah dan satu generasi ke generasi berikutnya. Dalam AG, *co-evolution* seringkali diimplementasikan untuk dua populasi (spesies), di mana nilai fitness dan individu-individu pada populasi pertama diperoleh dan interaksi dengan individu-individu pada populasi kedua, dan sebaliknya.

Suatu masalah pada *co-evolution* adalah berapa nilai fitness yang sesungguhnya jika nilai fitness yang digunakan adalah relatif. Suatu prosedur yang sederhana adalah mengukur individu terbaik berdasarkan suatu individu bernilai fitness tetap. Tetapi pendefinisian individu bernilai fitness tetap ini tidak mudah dilakukan pada semua aplikasi. Oleh karena itu *co-evolution* jarang digunakan dalam AG.

5.3 NiIai Fitness

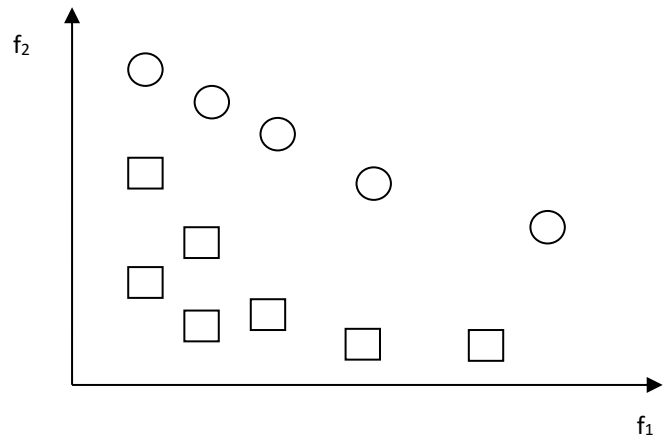
Pada kebanyakan masalah optimasi, menentukan fungsi fitness adalah hal yang mudah. Tetapi untuk masalah-masalah yang lain, penentuan fungsi fitness bukanlah hal yang mudah. Misalkan, suatu masalah dimana fungsi fitness harus memperhatikan beberapa kriteria (*multiobjective Optimization*). Masalah ini akan lebih rumit jika terdapat beberapa kriteria yang bertentangan. Masalah lain yang fungsi fitness-nya sulit ditentukan adalah *constrained optimization* di mana tidak semua solusi yang mungkin adalah valid.

5.3.1 Optimasi untuk Banyak Tujuan

Misalkan AG digunakan untuk optimasi perancangan handphone dengan tiga kriteria: penampilan harga, berat. Terdapat konflik pada ketiga kriteria tersebut. Penampilan bisa ditingkatkan dengan memilih bahan-bahan yang berkualitas tinggi, tetapi tentu saja akan membuat harga dan berat menjadi tidak optimal. Untuk menyelesaikan masalah seperti ini dapat digunakan satu cara yang menggunakan ide dari *pareto-optimality*.

Pareto-optimality

Misalkan vektor *x* adalah sebuah solusi yang potensial untuk masalah optimasi. Untuk vektor *x* tersebut, kriteria fitness f_i adalah $f_i(x)$, $i = 1, 2, \ldots K$, di mana K adalah jumlah kriteria optimasi. Vektor *x* disebut sebagai pareto-optimal jika tidak ada individu y sedemikian hingga $f_j(y) \geq f_j(x)$ untuk semua *i*, dan $f_j(y) \geq f_j(x)$ untuk paling sedikit satu *j*, di mana $j = 1, 2, ..., K$. Himpunan solusi-solusi ini membentuk pareto-optimal set, dan vektor yang termasuk dalam himpunan ini disebut sebagai non-dominated. Secara umum pareto-optimal set berisi lebih dari satu elemen. Gambar 8 di bawah ini mengilustrasikan ide dari pareto-optimality. Titik-titik yang digambarkan dengan lingkaran membentuk *pareto-optimal set* di antara titik-titik yang tersedia.



Gambar 8 Pareto-optimality. Lingkaran menunjukkan himpunan titik-titik non-dominated, yaitu pareto-optimal set

5.3.2 Optimasi dengan Batasan

Pada bab-bab sebelumnya optimasi dilakukan tanpa ada batasan. Misalkan, masalah maksimasi untuk fungsi $h(x_1, x_2)$ di mana x1 dan x2 bernilai real bebas. Jika masalah maksimasi fungsi tersebut dibatasi dengan sejumlah fungsi kriteria maka permasalahan tersebut menjadi optimasi dengan batasan atau constrained optimization.Sejumlah fungsi batasan bisa berupa, misalnya $c_i(x_1,x_2) =0$, i = 1,2,..., N_k, di manaN_k adalah jumlahkriteria. Terdapat dua jenis batasan yang dapat digunakan yaitu *soft constraints* dan *hard constraints*. Suatu kromosom boleh melanggar *soft constraints*, tetapi nilai fitness kromosom tersebut akan menjadi lebih rendah. Sedangkan hard constraints tidak boleh dilanggar. Artinya, jika suatu kromosom melanggar hard constraints, maka kromosom tersebut dianggap tidak valid. Satu contoh permasalahan optimasi dengan batasan adalah optimasi jadwal kuliah. *Hard constraints* pada masalah ini adalah bahwa pada satu waktu tertentu, mahasiswa dan dosen hanya boleh mengadakan perkuliahan pada satu ruangan. Pada satu waktu tertentu, satu ruangan hanya dapat digunakan untuk satu perkuliahan. Sedangkan *soft constraints* bisa berupa interval waktu perkuliahan untuk mata kuliah yang sama. Jika interval waktunya terlalu dekat, maka mahasiswa kurang mendapatkan kesempatan untuk mengerjakan tugas-tugas kuliah tersebut.

5.4 Pindah Silang dengan Batasan

Di dunia nyata terkadang terdapat batasan perkawinan di antara individu-individu. Misalkan karena masalah geografis maka tidak ada perkawinan antara individu-individu yang berada dalam pulau berbeda. Sebaliknya, pada satu populasi tertentu justru terdapat batasan perkawinan bagi individu-individu yang memiliki hubungan dekat. Kejadian ini diadopsi juga oleh AG untuk digunakan pada proses pindah silang. Pada bab ini hanya dibahas dua batasan untuk pindah silang, yaitu subpopulasi dan *grid-based crossover*.

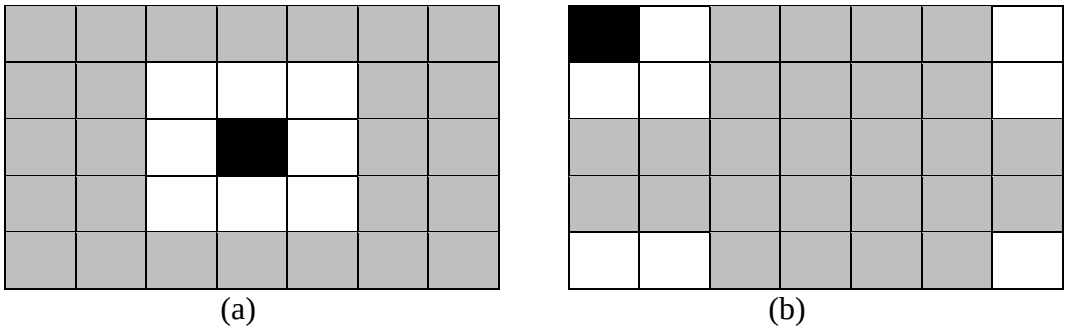
5.4.1 Subpopulasi

Pada metode ini *N* kromosom dalam satu populasi dibagi menjadi *N*k kelompok. Masing-masing kelompok berisi $v = N/N_k$ individu. Dalam metode ini pindah silang dan mutasi hanya terjadi di dalam subpopulasi. AG seperti ini disebut

sebagai island models.Tujuan dari subpopulasi adalah untuk menghindari konvergensi *premature*, di mana AG konvergen terlalu cepat ke suatu lokal optimum. Dengan subpopulasi dapat dikurangi adanya kemungkinan bahwa individu terbaik dalam semua subpopulasi berada dekat dengan lokal optimal. Hal inilah yang dapat menghindarkan AG dari konvergensi premature.Tetapi jika tidak ada interaksi antar individu dan subpopulasi yang berbeda, maka AG berbasis subpopulasi akan sama saja dengan mengerjakan sebanyak *N*k AG masing-masing dengan *v*individu. Oleh karena itu dengan tunneling probability P_t, suatu individu bisa dipindah ke subpopulasi lain. Agar jumlah individu dalam suatu subpopulasi tidak berubah, suatu individu dipilih secara random untuk dipindah ke subpopulasi lainnya. Keuntungan dari AG berbasis subpopulasi adalah dapat dikerjakan secara paralel dan terdistribusi ke beberapa komputer.

5.4.2 Grid-based Crossover

Pada metode ini individu-individu diletakkan dalam suatu pola yang teratur seperti pada Gambar 9 di bawah ini. Pada gambar tersebut hanya diperhatikan 8 kotak tetangga. Jumlah tetangga ini bisa diperbesar menjadi 24 atau lebih tergantung masalah yang akan diselesaikan. *Grid topology* yang paling umum digunakan adalah toroidal space, di mana ujung-ujung kotak tersebut disatukan dan membentuk ruang tiga dimensi seperti bola.



(a) (b)

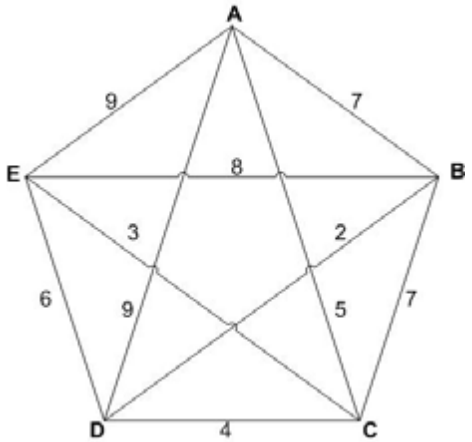
Gambar 9 Metode grid-based crossover. (a) individu yang dipilih secara random (kotak hitam di tengah) hanya dibolehkan pindah silang dengan salah satu dari 8 individu yang menjadi tetangganya (kotak putih). (b) Jika individu yang terpilih berada di kotak pojok kiri atas (kotak hitam), maka 8 individu yang menjadi tetangganya adalah kotak-kotak di ujung kanan dan bawah (kotak putih). Bayangkan jika

ujung-ujung kotak tersebut menyatu dan membentuk ruang tiga dimensi yang disebut toroidal

6 Studi Kasus

6.1 Traveling Salesman Problem (TSP)

Berikut contoh persoalan TSP yang diselesaikan dengan menggunakan algoritma genetika. Terdapat 5 buah kota yang akan dilalui oleh seorang pedangang keliling, misalnya Kota A,B,C,D,E. Perjalanan dimulai dari kota A dan berakhir di kota A. Jarak antar kota diperlihatkan pada graf di bawah ini:



Gambar 10 Jarak antar kota

Persoalan TSP tersebut akan diselesaikan dengan menggunakan algoritma genetika. Kriteria berhentiditentukan terlebih dahulu yaitu apabila setelah dalam beberapa generasi berturut-turut diperoleh nilai *fitness* yang terendah tidah berubah. Pemilihan nilai *fitness* yang terendah sebagai syarat karena nilai tersebut yang merepresentasikan jarak terdekat yang dicari pada persoalan TSP ini. Ada 4 kota yang akan menjadi gen dalam kromosom yaitu kota-kota selain kota asal.

a. Inisialisasi

Misalkan kita menggunakan 6 buah populasi dalam satu generasi, yaitu

Kromosom[1] = [B D E C]

Kromosom[2] = [D B E C]

Kromosom[3] = [C B D E]

Kromosom[4] = [E B C D]

Kromosom[5] = [E C B D]

Kromosom[6] = [C D E B]

b. Evaluasi kromosom

Kita akan menghitung nilai *fitness* dari tiap kromosom yang telah dibangkitkan:

Fitness[1] = AB+BD+DE+EC+CA
= 7 + 2 + 6 + 3 + 5 = 23

Fitness[2] = AD+DB+BE+EC+CA
= 9 + 2 + 8 + 3 + 5 = 27

Fitness[3] = AC+CB+BD+DE+EA
= 5 + 7 + 2 + 6 + 9 = 29

Fitness[4] = AE+EB+BC+CD+DA
= 9 + 8 + 7 + 4 + 9 = 37

Fitness[5] = AE+EC+CB+BD+DA
= 9 + 3 + 7 + 2 + 9 = 30

Fitness[6] = AC+CD+DE+EB+BA
= 5 + 4 + 6 + 8 + 7 = 30

c. Seleksi kromosom

Oleh karena pada persoalan TSP yang diinginkan yaitu kromosom dengan *fitness* yang lebih kecil akan mempunyai probabilitas untuk terpilih kembali lebih besar maka digunakan inverse.

Q[i] = 1/*Fitness (i)* ……..(10)

Q[1] = 1/23 = 0,043

Q[2] = 1/27 = 0,037

Q[3] = 1/29 = 0,034

Q[4] = 1/37 = 0,027

Q[5] = 1/30 = 0,033

Q[6] = 1/30 = 0,033

Total = 0,043+0,037+0,034+0,027+0,033+0,033
= 0,207

Untuk mencari probabilitas kita menggunakan rumus berikut :

P[i] = *Q*[*i*]/Total …….....(11)

P[1] = 0,043/0,207 = 0,208

P[2] = 0,037/0,207 = 0,179

P[3] = 0,034/0,207 = 0,164

P[4] = 0,027/0,207 = 0,130

P[5] = 0,033/0,207 = 0,159

P[6] = 0,033/0,207 = 0,159

Dari probabilitas di atas dapat terlihat bahwa kromosom ke-1 mempunyai *fitness* paling kecil mempunyai probabilitas untuk terpilih pada generasi selanjutnya lebih besar dari kromosom lainnya. Untuk proses seleksi kita menggunakan *rouletewheel*, untuk itu kita terlebih dahulu mencari nilai kumulatif dari probabilitasnya.

C[1] = 0,208

C[2] = 0,028+0,179 = 0,387

C[3] = 0,387+0,164 = 0,551

C[4] = 0,551+0,130 = 0,681

C[5] = 0,681+0,159 = 0,840

C[6] = 0,840+0,159 = 1

Proses *roulete-wheel* adalah membangkitkan nilai acak R antara 0-1. Jika **R[k]<C[k]** maka kromosom ke-k sebagai induk, selain itu pilih kromosom ke-k sebagai induk dengan syarat **C[k-1] < R[k] < C[k]**. Kita putar roulete-wheel sebanyak jumlah kromosom yaitu 6 kali (membangkitkan bilangan acak R).

R[1] = 0,314

R[2] = 0,111

R[3] = 0,342

R[4] = 0,743

R[5] = 0,521

R[6] = 0,911

Setelah itu, populasi baru akan terbentuk yaitu :

Kromosom[1] = [2] = [D B E C]

Kromosom[2] = [1] = [B D E C]

Kromosom[3] = [3] = [C B D E]

Kromosom[4] = [5] = [E C B D]

Kromosom[5] = [4] = [E B C D]

Kromosom[6] = [6] = [C D E B]

d. *Crossover* (pindah silang)

Pindah silang pada TSP dapat diimplementasikan dengan skema *order crossover*. Pada skema ini, satu bagian kromosom dipertukarkan dengan tetap menjaga urutan kota yang bukan bagian dari kromosom tersebut. Kromosom yang dijadikan induk dipilih secara acak dan jumlah kromosom yang dicrossover dipengaruhi oleh parameter *crossover probability* (ρc). Misal kita tentukan ρc = 35%, maka diharapkan dalam 1 generasi ada 50% (3 kromosom) dari populasi mengalami *crossover*. Pertama kita bangkitkan bilangan acak R sebanyak jumlah populasi yaitu 6 kali.

R[1] = 0,451

R[2] = 0,211

R[3] = 0,302

R[4] = 0,877

R[5] = 0,771

R[6] = 0,131

Kromosom ke-k yang dipilih sebagai induk jika R[k] < ρc. Maka yang akan dijadikan induk adalah kromosom[2], kromosom[3], dan kromosom[6]. Setelah melakukan pemilihan induk, proses selanjutnya adalah menentukan posisi *crossover*. Hal tersebut dilakukan dengan membangkitkan bilangan acak antara 1 sampai dengan panjang kromosom-1.

Dalam kasus TSP ini bilangan acaknya adalah antara 1-3. Misal diperoleh bilangan acaknya 1, maka gen yang ke-1 pada kromosom induk pertama diambil kemudian ditukar dengan gen pada kromosom induk kedua yang belum ada pada induk pertama dengan tetap memperhatikan urutannya.

Bilangan acak untuk 3 kromosom induk yang akan di-*crossover* :

C[2] = 2

C[3] = 1

C[6] = 2

Proses *crossover* :

Kromosom[2] = **Kromosom[2]** >< **Kromosom[3]**

= [B D E C] >< [C B D E]

= [B D C E]

Kromosom[3] = **Kromosom[3]** >< **Kromosom[6]**

= [C B D E] >< [C D E B]

= [C D E B]

Kromosom[6] = **Kromosom[6]** >< **Kromosom[2]**

= [C D E B] >< [B D E C]

= [C D B E]

Populasi setelah di-*crossover* :

Kromosom[1] = [D B E C]

Kromosom[2] = [B D C E]

Kromosom[3] = [C D E B]

Kromosom[4] = [E C B D]

Kromosom[5] = [E B C D]

Kromosom[6] = [C D B E]

e. Mutasi

Pada kasus TSP ini skema mutasi yang digunakan adalah *swapping mutation*. Jumlah kromosom yang mengalami mutasi dalam satu populasi ditentukan oleh parameter *mutation rate(*ρm*)*. Proses mutasi dilakukan dengan cara menukar gen yang dipilih secara acak dengan gen sesudahnya. Jika gen tersebut berada di akhir kromosom, maka ditukar dengan gen yang pertama. Pertama kita hitung dulu panjang total gen yang ada pada satu populasi:

Panjang total gen = jumlah gen dalam 1 kromosom * jumlah Kromosom ….(12)

= 4 * 6

= 24

Untuk memilih posisi gen yang mengalami mutasi dilakukan dengan membangkitkan bilangan acak antara 1 – Panjang total gen yaitu 1- 24. Misal kita tentukan ρm = 20 %.

Maka jumlah gen yang akan dimutasi adalah = 0,2*24 = 4,8 = 5

5 buah posisi gen yang akan dimutasi, setelah diacak adalah posisi 3,7,10,20,24.

Proses mutasi :

Kromosom[1] = [D B **C** E]

Kromosom[2] = [B D **E** C]

Kromosom[3] = [C **E** D B]

Kromosom[4] = [E C B D]

Kromosom[5] = [D B C **E**]

Kromosom[6] = [E D B **C**]

Proses algoritma genetik untuk 1 generasi telah selesai. Maka nilai *fitness* setelah 1 generasi adalah:

Fitness[1] = AD+DB+BC+CE+EA = 9 + 2 + 7 + 3 + 9 = 30

Fitness[2] = AB+BD+DE+EC+CA = 7 + 2 + 6 + 3 + 5 = 23

Fitness[3] = AC+CE+ED+DB+BA = 5 + 3 + 6 + 2 + 7 = 23

Fitness[4] = AE+EC+CB+BD+DA = 9 + 3 + 7 + 2 + 9 = 30

Fitness[5] = AD+DB+BC+CE+EA = 9 + 2 + 7 + 3 + 9 = 30

Fitness[6] = AE+ED+DB+BC+CA = 9 + 6 + 2 + 7 + 5 = 29

Sebelumnya telah ditentukan kriteria berhenti yaitu bila setelah dalam beberapa generasi berturut-turut diperoleh nilai *fitness* yang terendah tidah berubah. Pada 1 generasi telah terlihat bahwa terdapat nilai *fitness* terkecil yang tidak berubah. Apabila perhitungan dilanjutkan hingga ke generasi ke-N maka diyakinkan bahwa nilai *fitness* yang terendah tetap tidak akan berubah. Walaupun perhitungan cukup dijabarkan hingga generasi ke-1 saja namun solusi yang mendekati optimal telah didapatkan. Oleh karena itu, terbukti bahwa algoritma genetika dapat menyelesaikan persoalan TSP.

6.2 Traveling Salesman Problem (TSP)

Banyak aplikasi yang berhubungan dengan TSP, di antaranya adalah pembuatan PCB dalam bidang elektronika, jasa kurir, transportasi, dan sebagainya.Di sini hanya dibahas sebuah kasus kecil untuk memahami bagaimana AG dapat digunakan untuk menyelesaikan masalah TSP.

6.2.1 Permasalahan TSP

Pada TSP, jumlah jalur mungkin diperoleh dengan menggunakan rumus permutasi pada persamaan di bawah ini.

$${}_{n}P_{k} = \frac{n!}{(n-k)!} \qquad \text{..} \; (13)$$

Di mana n adalah jumlah seluruh kota, dan k adalah jumlah kota yang diseleksi. Terdapat dua jenis TSP, yaitu asimetris dan simetris. Pada TSP asimetris, biaya dari kota 1 ke kota 2 tidak sama dengan biaya dari kota 2 ke kota 1. Sedangkan pada TSP simetris, biaya dari kota 1 ke kota 2 adalah sama dengan biaya dari kota 2 ke kota 1.

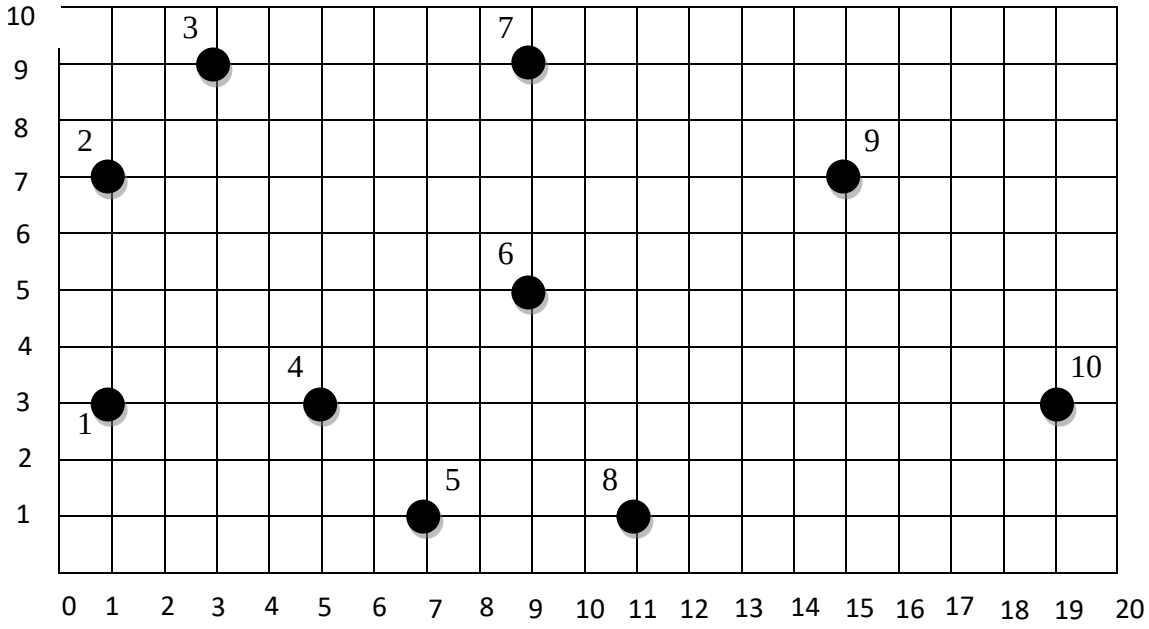
Untuk TSP asimetris, jumlah jalur yang mungkin adalah permutasi dari jumlah kota dibagi dengan jumlah kota. Hal ini dapat dipahami karena secara siklus, sebuah jalur dengan urutan 1-2-3 adalah sama dengan jalur 2-3-1 dan jalur 3-1-2. Tetapi jalur dengan urutan 1-2-3 tidaklah sama dengan jalur 3-2-1. Jadi jumlah jalur yang mungkin untuk TSP asimetris dengan 10 kota adalah :

$_{10}P_{10} = \frac{10!}{10\,x(10-10)!} = 362.880$ …… (14)

Sedangkan untuk TSP simetris, jumlah jalur yang mungkin adalah permutasi dari jumlah kota dibagi dengan dua kali jumlah kota. Hal ini dapat dipahami karena secara siklus sebuah jalur dengan urutan 1-2-3 adalah sama dengan jalur 2-3-1 dan jalur 3-1-2. Karena biaya dari kota 1 ke kota 2 adalah sama dengan biaya dari kota 2 ke kota 1, maka jalur dengan urutan 1-2-3 adalah sama dengan jalur 3-2-1. Jadi jumlah jalur yang mungkin untuk TSP simetris dengan 10 kota adalah :

$_{10}P_{10} = \frac{10!}{2\,x\,10\,x(10-10)!} = 181.440$ …… (15)

Misalkan terdapat 10 kota dengan koordinat-koordinat dua dimensinya seperti terlihat pada gambar di bawah ini. Jika diasumsikan bahwa masalahnya adalah TSP simetris, jalur manakah yang paling optimal berdasarkan biaya berupa jarak kartesian? Dan berapakah panjang jalur optimal tersebut?



Gambar 11 Sepuluh kota dengan koordinat dua dimensinya

6.2.2 Implementasikan TSP dalam MATLAB

Untuk memudahkan pemahaman, implementasi TSP dibahas secara berurutan berdasarkan urutan proses komponen-komponen AG dalam MATLAB (Chipperfield, 1995) (Houck and Kay, 2008). Perhatikan setiap langkah berikut ini dengan seksama.

6.2.2.1 Skema Pengkodean

Untuk TSP digunakan skema pengkodean permutation encoding. Setiap kromosom berisi gen-gen yang merepresentasikan nomor urut kota. Jumlah gen dalam setiap kromosom adalah sama dengan jumlah kota. Skema ini diimplementasikan secara sederhana menggunakan baris-baris perintah pada fungsi

```
%%Membangkitkan ukran populasi kromosom masing-masing kromosom
berisi
%%bilangan 1 sampai JumGen(jumlah kota)
%%
%%Input:
%%UkPop : Ukuran populasi atau jumlah kromosom dalm populasi
%%JumGen: jumlah gen(jumlah kota)
%%
%%Output:
%%Pupulasi : kumpulan kromosom matriks berukuran UkPop x JumGen
%%
function Populasi = TSPInisialisasiPopulasi(UkPop,JumGen)
for ii=1:UkPop,
    [Xval,Ind]= sort(rand(1, JumGen));
    Populasi(ii,:) =Ind;
end
```

Kode Program 1 TSPInisialisasiPopulasi

Masukan untuk fungsi ini adalah UkPop (ukuran populasi atau jumlah kromosom dalam populasi) dan JumGen (jumlah gen dalam kromosom yang juga menyatakan jumlah kota) Perintah rand(1,JumGen) menyatakan pembangkitan matriks berukuran 1 x JumGen yang berisi bilangan random dalam interval [0,1). Pada perintah [Xval, Ind] = sort(rand(1,JumGen)), Xval menyatakan bilangan-bilangan dalam interval [0,1) hasil pengurutan dari kecil ke besar (ascending). Sedangkan Ind menyatakan indeks dari bilangan-bilangan yang dibangkitkan secara random tetapi belum terurut. Jadi Ind inilah yang merupakan nomor urut kota yang dibangkitkan secara random secara tidak langsung melalui perintah

sort(rand(1,JumGen)). Populasi(ii,:) menyatakan kromosom ke-*ii* pada populasi yang jumlah kolomnya sama dengan jumlah gen.

6.2.2.2 Nilai Fitness

Untuk TSP, masalahnya adalah meminimalkan total biaya (masalah minimasi). Oleh karena itu nilai fitness yang bisa digunakan adalah 1 dibagi dengan total biaya. Dalam hal ini yang dimaksud total biaya adalah jumlah jarak kartesian antara satu kota dengan kota lainnya secara melingkar. Misalkan, untuk sejumlah *n* kota, total biayanya adalah jarak kartesian dari kota 1 ke kota 2 ditambah jarak kartesian dari kota 2 ke kota 3 dan seterusnya sampai jarak kertesian dari kota n ke kota 1. Jarak kartesian antara kota A dan kota B dihitung dengan rumus :

$$\|A - B\| = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (16)$$

(X_A,Y_A) menyatakan posisi koordinat kota A dan (X_B,Y_B) adalah posisi koordinat kota B. Penghitung jarak kartesian ini diimplementasikan menggunakan fungsi norm, yang sudah tersedia di dalam MATLAB.

Perhitungan nilai fitness diimplementasikan dalam fungsi TSPEvaluasiIndividu di bawah ini. Masukan untuk fungsi ini adalah tiga variabel: Kromosom, JumGen, XYkota. Variabel XYkota berisi koordinat-koordinat dari semua kota. Variabel TB menyatakan total biaya, dan nilai fitness suatu kromosom dinyatakan oleh fitness=1/TB.

```
%%Mengevaluasi sehingga mendapatkan nilai fitness nya
%%
%%Input:
%%Kromosom: kromosom dengan matrik ukuran 1XJumGen
%%XYKota : posisi-posisi koordinat kota, matrik berukuran UkPop
x 2
%%JumGen: jumlah gen
%%
%%Output:
%%fitnnes : nilai fitness
%%
function Fitness = TSPEvaluasiIndividu(Kromosom,JumGen,XYkota)
TB = 0;
for ii=1:JumGen-1,
    TB=TB+norm(XYkota(Kromosom(ii),:)-XYkota(Kromosom(ii+1),:));
end
% Jalur untuk kembali ke kota asal.
TB =TB +norm(XYkota(Kromosom(ii),:)-XYkota(Kromosom(1),:));
```

```
Fitness =1/TB;
```

Kode Program 2 TSPEvaluasiIndividu

6.2.2.3 Linear Fitness Ranking

Untuk menghindari kecenderungan konvergen pada optimum local, maka dilakukan proses penskalaan nilai fitness, sehingga diperoleh nilai fitness yang berada dalam interval [MaxF,MinF] sehingga diperoleh nilai fitness baru yang lebih baik, yaitu memiliki variansi tinggi.Fungsi ini disimpan pada LinearFitnessRanking.m.

```
%%Menskalakan nilai fitness ke dalam ranking sehingga diperoleh
nilai-nilai
%%fitness baru yang berada dalam rentang (MaxF,MinF)
%%
%%Input:
%%UkPop : ukuran populasi atau jumlah kromosom dalam populasi
%%Fitness: nilai fitness, matriks ukuran 1x UkPop
%%MaxF : Nilai Fitness Maximum
%%MinF : Nilai Fitness minimum
%%
%%Output :
%%LFR : Linear Fitness Ranking
%%
function LFR = LinearFitnessRanking(UkPop,Fitness,MaxF,MinF);

[SF,IndF]=sort(Fitness);

for rr=1; UkPop,
    LFR (IndF(UkPop-rr+1)) = MaxF-(MaxF-MinF)*((rr-1)/(UkPop-
1));
End
```

Kode Program 3 LinearFitnessRanking

6.2.2.4 Roulette-wheel

Fungsi *roulette-wheel* diimplementasikan untuk mendapatkan keluaran berupa Pindex, yaitu indeks dari individu terpilih sebagai orang tua.Fungsi ini disimpan pada roulettewheel.m.

```
%%Memilih orang tua menggunkan Linear Fitness yaitu nilai
fitness hasil
%%pen-skala-an. Pemilihan dilakukan secara proporsional sesuai
denhan nilai
%%fitness-nya
%%
%%Input:
%%UkPop : ukuran populasi atau jumlah kromosom dalam populasi
%%LinearFitness : nilai fitness yang sudah di skala-kan
```

```
%%
%%Output :
%%Pindex : indeks dari kromosom yang terpilih
%%

function Pindex = RouletteWheel(UkPop,LinearFitness);

JumFitness = sum(LinearFitness);
KumulatifFitness =0;
RN = rand;
ii = 1;

while ii<=UkPop,
    KumulatifFitness =KumulatifFitness + LinearFitness(ii);
if (KumulatifFitness/JumFitness)> RN,
        Pindex =ii;
break;
end
    ii =ii+1;
end
```

Kode Program 4 RouletteWheel

6.2.2.5 Pindah Silang

Pindah Silang untuk TSP dapat diimplementasikan dengan skema order crossover. Pada skema ini, satu bagian kromosom dipertukarkan dengan tetap menjaga urutan kota yang bukan bagian dari kromosom tersebut (Anies Hannawati, Thiang and Eleazar, 2002). Penjelasan lebih detail tentang skema order crossover dapat dilihat pada pembahasan sebelumnya.

```
%%Pindah Silang mengggunakan Order Crossover
%%
%%Input:
%%Bapak :kromosom, matriks berukuran 1x JumGen
%%Ibu :kromosom, matriks berukuran 1x JumGen
%%JumGen: jumlah gen
%%
%%Output:
%%Anak : kromosom hasil pindah silang, matriks berukuran 1 x
JumGen
%%

function Anak = TSPPindahSilang(Bapak,Ibu,JumGen)
cp1 = 1 + fix(rand*(JumGen-1));
cp2 = 1 + fix(rand*(JumGen-1));
while cp2==cp1,
    cp2=1+fix(rand*(JumGen-1));
end
if cp1<cp2,
    cps =cp1;
    cpd =cp2;
```

```
else
    cps =cp2;
    cpd =cp1;
end
Anak(1,cps+1:cpd)=Ibu(cps+1:cpd);
Anak(2,cps+1:cpd)=Bapak(cps+1:cpd);

SisaGenbapak =[];
SisaGenibu =[];
for ii=1:JumGen,
if ~ismember(Bapak(ii), Anak(1,:));
        SisaGenbapak = [SisaGenbapak Bapak(ii)];
end
if ~ismember(Ibu(ii), Anak(1,:));
        SisaGenibu = [SisaGenibu Ibu(ii)];
end
end

Anak(1,cpd+1:JumGen) = SisaGenbapak(1:JumGen-cpd);
Anak(1,1:cps) = SisaGenbapak(1+JumGen-cpd:length(SisaGenbapak));

Anak(2,cpd+1:JumGen) = SisaGenibu(1:JumGen-cpd);
Anak(2,1:cps) = SisaGenibu(1+JumGen-cpd:length(SisaGenibu));
```

Kode Program 5 TSPPindahSilang

6.2.2.6 Mutasi

Pada TSP biasanya digunakan skema mutasi yang disebut sebagai swapping mutation.Fungsi TSPMutasi di bawah ini merupakan implementasi swapping mutation.Masukan untuk fungsi ini adalah Kromosom, JumGen, dan Pmutasi. Untuk semua gen dalam kromosom, jika bilangan random[0,1) yang dibangkitkan kurang dari Pmutasi, maka nilai gen tersebut akan ditukarkan dengan nilai gen lain yang dipilih secara random. Keluaran dari fungsi ini adalah MutKrom, yaitu kromosom yang sudah termutasi.

```
%%Mutasi menggunakan swapping Mutasi
%%
%%Input:
%%Kromosom :kromosom, matriks berukuran 1x JumGen
%%JumGen: jumlah gen
%%Pmutasi: probabilitas mutasi
%%
%%Ouput:
%%MutKrom : kromosom hasil mutasi, matriks berukuran 1 x JumGen
%%

function MutKrom = TSPMutasi(Kromosom,JumGen,Pmutasi)
MutKrom =Kromosom;
for ii=1:JumGen,
if rand <Pmutasi,
```

```
        TM2=1+fix(rand*JumGen);
while TM2==ii,
            TM2=1+fix(rand*JumGen);
end
        temp = MutKrom(ii);
        MutKrom(ii)=MutKrom(TM2);
        MutKrom(TM2)=temp;
end
end
```

Kode Program 6 TSPMutasi

6.2.2.7 Program Utama

Sebagai program utama, file MainTSP.m, memanggil semua fungsi di atas. Posisi-posisi koordinat kota disimpan dalam variabel XYkota. Pada program di bawah ini juga ditambahkan perintah-perintah untuk menampilkan grafik.Pada akhir program, variabel JalurTerbaik, yang menyatakan rute terpendek, disimpan ke dalam sebuah file bernama JalurTerbaik.mat.

```
Clc
clear all
XYkota = [1 3; 1 7; 3 9; 5 3; 7 1; 9 5; 9 9; 11 1; 15 7; 19 3];
JumGen =length(XYkota(:,1));
%%Jumlah Gen
UkPop =100;
%%Jumlah Kromosom dalam Populasi
Psilang=0.8;
%%Probabilitas Pindah Silang
Pmutasi=0.005;
%%Probabilitas Mutasi
MaxG =100;
%%Jumlah Generasi

PanjJalHarp = 40;
Fthreshold = 1/PanjJalHarp;
Bgraf=Fthreshold;

%%Inisialisasi Grafis

hfig=figure;
hold on
set(hfig,'position',[50,50,600,400]);
set(hfig,'DoubleBuffer','on');
axis([1 MaxG 0 Bgraf]);
hbestplot1 = plot(1:MaxG,zeros(1,MaxG));
hbestplot2 = plot(1:MaxG,zeros(1,MaxG));
htext1 =text(0.6*MaxG,0.25*Bgraf, sprintf('Fitness terbaik:
%7.6f', 0.0));
htext2 =text(0.6*MaxG,0.20*Bgraf, sprintf('Fitness rata-rata:
%7.6f', 0.0));
htext3 =text(0.6*MaxG,0.15*Bgraf, sprintf('Panjang Jalur
Terbaik: %7.3f', 0.0));
```

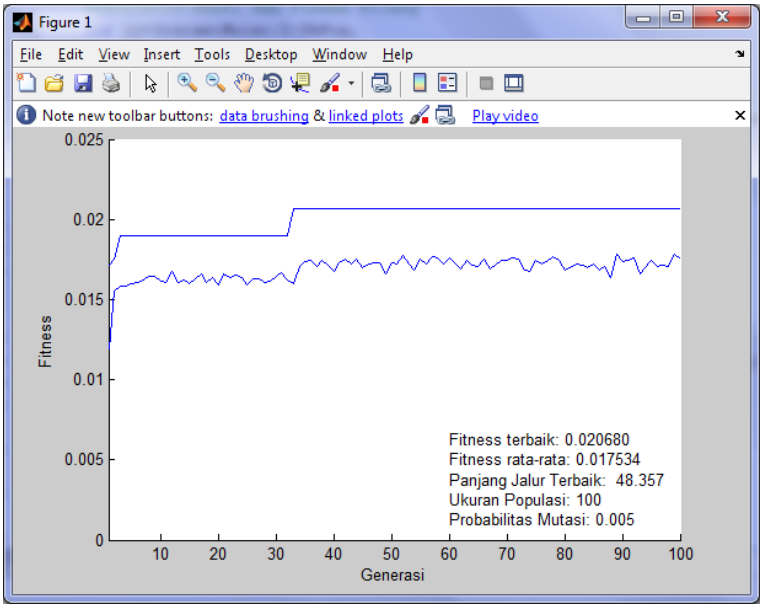
```
htext4 =text(0.6*MaxG,0.10*Bgraf, sprintf('Ukuran Populasi:
%3.0f', 0.0));
htext5 =text(0.6*MaxG,0.05*Bgraf, sprintf('Probabilitas Mutasi:
%4.3f', 0.0));
xlabel('Generasi');
ylabel('Fitness');
hold off
drawnow;

%%Inisialisasi Pupulasi
Populasi =TSPInisialisasiPopulasi(UkPop,JumGen);

for generasi=1:MaxG,
    MaxF= TSPEvaluasiIndividu(Populasi(1,:),JumGen,XYkota);
    MinF = MaxF;
    IndeksIndividuTerbaik=1;
for ii=1:UkPop,
        Fitness(ii) =
TSPEvaluasiIndividu(Populasi(ii,:),JumGen,XYkota);
if (Fitness(ii)> MaxF),
            MaxF = Fitness(ii);
            IndeksIndividuTerbaik=ii;
            JalurTerbaik = Populasi(ii,:);
end
if (Fitness(ii)<= MinF),
             MinF = Fitness(ii);
end
end

    FitnessRata = mean(Fitness);

    plotvector1 = get(hbestplot1,'YData');
    plotvector1(generasi)= MaxF;
    set(hbestplot1,'YData',plotvector1);
    plotvector2 = get(hbestplot2,'YData');
    plotvector2(generasi)= FitnessRata;
    set(hbestplot2,'YData',plotvector2);
    set(htext1,'String',sprintf('Fitness terbaik: %7.6f',MaxF));
    set(htext2,'String',sprintf('Fitness rata-rata:
%7.6f',FitnessRata));
    set(htext3,'String',sprintf('Panjang Jalur Terbaik:
%7.3f',1/MaxF));
    set(htext4,'String',sprintf('Ukuran Populasi:
%3.0f',UkPop));
    set(htext5,'String',sprintf('Probabilitas Mutasi:
%4.3f',Pmutasi));
    drawnow;

if MaxF >Fthreshold,
break;
end
    TemPopulasi = Populasi;

%%Elitisme
%%-Buat satu kopi kromosom terbaik jika ukuran populasi ganjil
```

```
%%-Buat 2 kopi kromosom terbaik jika ukuran populasi genap

if mod(UkPop,2)==0,
        IterasiMulai =3;
        TemPopulasi(1,:) = Populasi(IndeksIndividuTerbaik,:);
        TemPopulasi(2,:) = Populasi(IndeksIndividuTerbaik,:);
else
        IterasiMulai =2;
        TemPopulasi(1,:) = Populasi(IndeksIndividuTerbaik,:);
end

    LinearFitness
=LinearFitnessRanking(UkPop,Fitness,MaxF,MinF);

%%Roulette-wheel dan Pindah Silang
for jj=IterasiMulai:2:UkPop,
        IP1 =RouletteWheel(UkPop,LinearFitness);
        IP2 =RouletteWheel(UkPop,LinearFitness);
if (rand<Psilang),
            Anak =
TSPPindahSilang(Populasi(IP1,:),Populasi(IP2,:),JumGen);
            TemPopulasi(jj,:) = Anak(1,:);
            TemPopulasi(jj+1,:) = Anak(2,:);
else
            TemPopulasi(jj,:) = Populasi(IP1,:);
            TemPopulasi(jj+1,:) = Populasi(IP2,:);
end
end

%%Mutasi pada keseluruhan kromosom
for kk=IterasiMulai:UkPop,
        TemPopulasi(kk,:) = TSPMutasi(TemPopulasi(kk,:), JumGen,
Pmutasi);
end
     Populasi =TemPopulasi;

end
%%Tanpa tanda ';' berarti menampilkan nilai dari variabel
'JalurTerbaik'
 JalurTerbaik
%%Menyimpan nilai jalur terbaik dalam file JalurTerbaik.mat
 save JalurTerbaik.matJalurTerbaik
```

Kode Program 7 MainTSP

6.2.2.8 Running Program

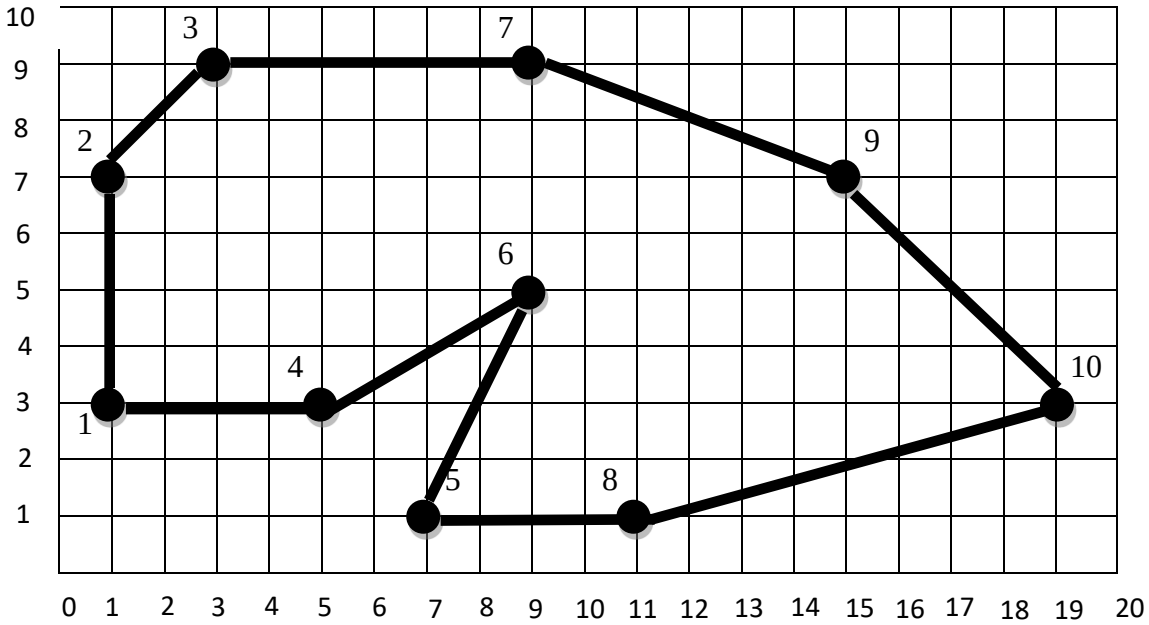
Gambar di bawah ini menunjukkan hasil evolusi dari generasi 1 sampai 100. Grafik atas menunjukkan nilai fitness yang terbaik dan grafik bawah menunjukkan nilai fitness rata-rata.



Gambar 12 Grafik evolusi dari generasi 1 sampai 100 untuk TSP

Pada gambar 12 di atas dilihat bahawa panjang jalur terbaik adalah 48,357 unit kartesian. Jalur optimal yang diperoleh adalah 4 - 1 - 2 - 3 - 7 - 9 - 10 - 8 5 - 6 terdapat pada variabel jalur terbaik pada *command window*.

Jadi solusi kasus TSP simetris yang didapat pada gambar 10 akan dibuat laur optimalnya pada gambar 13 di bawah ini dimana panjang jalurnya adalah 48,357 unit kartesian.



Gambar 13 Jalur terbaik yang dihasilkan oleh AG

Persoalan pedagang keliling (TSP) dapat diselesaikan dengan menggunakan algoritma genetika. Walaupun solusi TSP yang dihasilkan oleh algoritma ini belum tentu merupakan solusi paling optimal (misalnya apabila yang dilalui sangat banyak), namun algoritma genetika akan menghasilkan solusi yang lebih optimal pada setiap generasinya. Hal tersebut terlihat dari nilai fitness tiap generasi. Kelebihan algoritma genetika dibandingkan metode pencarian konvensiona pada TSP yaitu

1. Solusi dapat diperoleh kapanpun karena solusi dihasilkan pada generasi ke berapapun,
2. Algoritma genetika tidak harus membutuhkan waktu yang lama karena tidak semua kemungkinan dicoba, tergantung pada kriteria berakhirnya.

DAFTAR PUSTAKA

Anies Hannawati, Thiang and Eleazar (2002) 'Pencarian Rute Optimum Menggunakan Algoritma Genetika', *Jurnal Teknik Elektro*.

Chipperfield, A. J. (1995) 'The MATLAB Genetic Algorithm Toolbox', in *IEE Colloquium on Applied Control Techniques Using MATLAB*. doi: 10.1049/ic:19950061.

Hermawanto, D. (2007) 'Algoritma Genetika Dan Contoh Aplikasinya', *IlmuKomputer.com*.

Houck, C. R. and Kay, M. G. (2008) 'A Genetic Algorithm for Function Optimization : A Matlab Implementation', *Ncsuie Tr*. doi: 10.1109/ISDEA.2010.135.

Maulik, U. and Bandyopadhyay, S. (2000) 'Genetic algorithm-based clustering technique', *Pattern Recognition*. doi: 10.1016/S0031-3203(99)00137-5.

Whitley, D. (1994) 'A genetic algorithm tutorial', *Statistics and Computing*. doi: 10.1007/BF00175354.

Suyanto (2005). Algoritma Genetika dalam MATLAB, Andi,Yogyakarta

Sutujo, T (2011).Kecerdasan Buatan. Andi. Yogyakarta