

TUGAS KECERDASAN KOMPUTASIONAL A

Implementasi *Evolutionary Strategies*

“Optimasi Fungsi Berkendala”



Disusun oleh:

Fathin Ulfah Karimah

(15/388471/PPA/04910)

PROGRAM STUDI S2 ILMU KOMPUTER

JURUSAN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GAJAH MADA

YOGYAKARTA

2016

OPTIMASI FUNGSI BERKENDALA DENGAN *EVOLUTIONARY STRATEGIES*

1. Permasalahan

Permasalahan yang diangkat dalam implementasi *evolutionary strategies* ini adalah contoh permasalahan yang pernah dibahas di kelas sebelumnya, yaitu sebuah perusahaan yang akan memproduksi dua jenis lemari, yaitu lemari A dan lemari B. Untuk memproduksi kedua lemari tersebut dibutuhkan tiga macam bahan baku, yaitu: kayu, aluminium, dan kaca. Kebutuhan detail tiga bahan baku tersebut (dalam unit tertentu) untuk tiap buah lemari ditampilkan pada tabel berikut:

Tabel 1. Contoh kebutuhan tiga bahan baku untuk tiap lemari

Lemari	Kayu	Alumunium	Kaca
A	10	9	12
B	20	8	18

Persediaan bahan baku kayu, aluminium, dan kaca di gudang berturut-turut adalah 350, 200, dan 300. Jika keuntungan penjualan sebuah lemari A sebesar 400 dan B sebesar 500, maka akan dihitung banyaknya lemari A dan B harus diproduksi agar didapatkan keuntungan maksimum pada perusahaan tersebut.

Pada pengimplementasiannya, program dibuat untuk data yang dinamis (Gambar 1). User dapat menginputkan banyaknya kebutuhan dari tiga bahan baku untuk membuat lemari A dan B, persediaan bahan baku, dan keuntungan penjualan tiap lemari. Jika banyaknya lemari yang harus diproduksi dilambangkan x_1 dan x_2 maka fungsi tujuannya adalah sebagai berikut:

$$\text{Maksimumkan } f(x_1, x_2) = 400x_1 + 500x_2 \quad (1)$$

Kendala ketersediaan bahan baku bias dinyatakan sebagai berikut:

$$\text{Kendala 1 : } 10x_1 + 20x_2 \leq 350 \quad (2)$$

$$\text{Kendala 2 : } 9x_1 + 8x_2 \leq 200 \quad (3)$$

$$\text{Kendala 3 : } 12x_1 + 18x_2 \leq 300 \quad (4)$$

Gambar 1. Potongan program untuk input data secara dinamis

2. Struktur *Evolutionary Strategies* (ES)

a. Representasi Kromosom

Pada kasus ini, kromosom dipresentasikan dengan gen sebanyak 6 gen. Gen-gen tersebut akan mempresentasikan x_1 , x_2 , σ_1 , σ_2 , $f(x,y)$, fitness. Jika P merupakan satu kromosom, maka $P = \{x_1, x_2, \sigma_1, \sigma_2, f(x,y), \text{fitness}\}$. Nilai x_1 , x_2 merupakan representasi lemari A dan lemari B, sigma merupakan nilai random yang akan dibangkitkan untuk mendapatkan nilai x pada *offspring*, $f(x,y)$ merupakan nilai maksimal dari keuntungan penjualan, dan fitness merupakan nilai yang merepresentasikan individu yang paling fit untuk digunakan sebagai solusi. Berikut ini nilai fitness yang digunakan dalam program:

$$\text{fitness}(x_1, x_2) = f(x_1, x_2) - M(c_1 + c_2 + c_3) \quad (5)$$

$$c_1 = \begin{cases} 0, & \text{jika } 10x_1 + 20x_2 \leq 350 \\ (10x_1 + 20x_2) - 350, & \text{selainnya} \end{cases}$$

$$c_2 = \begin{cases} 0, & \text{jika } 9x_1 + 8x_2 \leq 200 \\ (9x_1 + 8x_2) - 200, & \text{selainnya} \end{cases}$$

$$c_3 = \begin{cases} 0, & \text{jika } 12x_1 + 18x_2 \leq 300 \\ (12x_1 + 18x_2) - 300, & \text{selainnya} \end{cases}$$

b. Inisialisasi

Nilai masing-masing gen yaitu x_1 , x_2 , σ_1 (sigma), σ_2 dibangkitkan secara random dalam rentang [0,50] untuk x dan rentang [0,1] untuk nilai sigma. Sedangkan nilai $f(x,y)$ dan fitness dihitung menggunakan rumus (1) dan (5). Banyaknya individu dalam populasi awal μ adalah

sebanyak 5. Berikut ini merupakan potongan program untuk inisialisasi populasi awal dan contoh output dari inisialisasi.

```
public void inisialisasi () {
    int pos=jTextHasil.getCaretPosition();
    jTextHasil.insert("x1\tx2\tsigma1\tsigma2\tf(x,y)\troundx1\troundx2"
        + "\tc1\tc2\tc3\tfitness\n", pos);

    for (int i=0;i<5;i++){
        ix1 = Math.random()*50;
        ix2 = Math.random()*50;
        sigma1 = Math.random()*1;
        sigma2 = Math.random()*1;

        fitness();

        chromosome[i][0]=Double.valueOf(f.format(ix1));
        chromosome[i][1]=Double.valueOf(f.format(ix2));
        chromosome[i][2]=Double.valueOf(f.format(sigma1));
        chromosome[i][3]=Double.valueOf(f.format(sigma2));
        chromosome[i][4]=ifx;
        chromosome[i][5]=fitness;

        int pos1=jTextHasil.getCaretPosition();
        jTextHasil.insert(chromosome[i][0]+"\\t"+chromosome[i][1]+"\\t"
            +chromosome[i][2]+"\\t"+chromosome[i][3]+"\\t"
            +chromosome[i][4]+"\\t"+x1+"\\t"+x2+"\\t"+c1+"\\t"+c2+"\\t"
            + ""+c3+"\\t"+chromosome[i][5]+"\\n", pos1);
    }
}
```

Gambar 2. Potongan kode program untuk inisialisasi

Kode *Math.random()* digunakan untuk membangkitkan nilai random untuk x dan sigma. Sedangkan kode setelah tanda * pada *Math.random()* merupakan rentang nilai yang akan dibangkitkan untuk variabel tersebut dari nilai 0.

===== EVOLUTIONARY STRATEGIES (ES) =====										
LAMDA = 5, MIU = 2*LAMDA, (MIU + LAMDA)										
INISIALISASI										
x1	x2	sigma1	sigma2	f(x,y)	roundx1	roundx2	c1	c2	c3	fitness
42.79	47.29	0.13	0.15	40700.0	43	47	1020	610	780	-2369300.0
46.86	1.21	0.32	0.34	19300.0	47	1	140	232	276	-628700.0
22.47	47.34	0.56	0.32	32300.0	22	47	810	421	528	-1726700.0
4.58	2.61	0.95	0.26	3500.0	5	3	0	0	0	3500.0
30.39	35.4	0.64	0.0	29500.0	30	35	650	385	480	-1485500.0

Gambar 3. contoh output inisialisasi

c. Reproduksi

Siklus *evolutionary strategies* yang digunakan dalam program ini adalah siklus $(\mu+\lambda)$. Dengan siklus ini, reproduksi yang digunakan hanyalah proses mutasi saja, sedangkan hasil generasi baru yang dihasilkan adalah hasil *elitism* dari banyaknya μ (populasi awal) dan λ (*offspring*). *Offspring* yang digunakan adalah 2 kali dari μ . Berikut ini Gambar 4 yang merupakan potongan program untuk melakukan reproduksi pada populasi.

```
public void reproduksi(){
    int pos=jTextHasil.getCaretPosition();
    jTextHasil.insert("HASIL REPRODUKSI (MUTASI)\n x1\tx2\tsigma1\tsigma2"
        + "\tf(x,y)\tfitness\n", pos);

    for (int i=0;i<10;i++){
        n1 = random.nextGaussian();
        n2 = random.nextGaussian();

        int a=i/2;
        cx1 = chromosome[a][0] + (chromosome[a][2] * n1);
        cx2 = chromosome[a][1] + (chromosome[a][3] * n2);

        ix1 = cx1;
        ix2 = cx2;

        fitness();

        chromosome[i+5][0]=Double.valueOf(f.format(ix1));
        chromosome[i+5][1]=Double.valueOf(f.format(ix2));
        chromosome[i+5][4]=ifx;
        chromosome[i+5][5]=fitness;

        if (i%2==1) {
            if (chromosome[i+4][4] > chromosome[a][4] ||
                chromosome[i+5][4] > chromosome[a][4]) {
                sigma1 = chromosome[a][2] * 1.1;
                sigma2 = chromosome[a][3] * 1.1;
            } else {
                sigma1 = chromosome[a][2] * 0.9;
                sigma2 = chromosome[a][3] * 0.9;
            }

            chromosome[i+4][2]=Double.valueOf(f.format(sigma1));
            chromosome[i+4][3]=Double.valueOf(f.format(sigma2));
            chromosome[i+5][2]=Double.valueOf(f.format(sigma1));
            chromosome[i+5][3]=Double.valueOf(f.format(sigma2));
        }
    }
}
```

Gambar 4. Potongan kode program untuk reproduksi (mutasi)

Kode `random.nextGaussian()` merupakan kode untuk membangkitkan nilai random yang mengikuti sebaran normal menggunakan Gaussian dengan rata-rata sebesar 0 dan standar deviasi sebesar 1, $N(0,1)$. Variabel `cx1` dan `cx2` merupakan variabel x baru untuk *offspring* yang didapat dengan rumus yang terdapat pada persamaan 6.

$$x' = x + (\sigma * N(0,1)) \quad (6)$$

Sedangkan Gambar 5 merupakan contoh output dari reproduksi $2*\mu$ yaitu 10.

Iterasi ke-1					
HASIL REPRODUKSI (MUTASI)					
x1	x2	sigma1	sigma2	f(x,y)	fitness
42.79	47.29	0.13	0.15	40700.0	-2369300.0
46.86	1.21	0.32	0.34	19300.0	-628700.0
22.47	47.34	0.56	0.32	32300.0	-1726700.0
4.58	2.61	0.95	0.26	3500.0	3500.0
30.39	35.4	0.64	0.0	29500.0	-1485500.0
42.59	47.13	0.12	0.14	40700.0	-2369300.0
43.06	47.02	0.12	0.14	40700.0	-2369300.0
46.78	1.09	0.29	0.31	19300.0	-628700.0
47.0	0.69	0.29	0.31	19300.0	-628700.0
21.54	47.36	0.62	0.35	32300.0	-1726700.0
22.98	47.05	0.62	0.35	32700.0	-1757300.0
4.96	2.36	0.86	0.23	3000.0	3000.0
4.1	2.59	0.86	0.23	3100.0	3100.0
31.32	35.4	0.7	0.0	29900.0	-1516100.0
29.71	35.4	0.7	0.0	29500.0	-1485500.0

Gambar 5. Contoh output hasil proses reproduksi

d. Seleksi

Seleksi yang digunakan adalah *elitism* yaitu melakukan pengurutan pada hasil reproduksi dan populasi awal. Pengurutan dilakukan berdasarkan nilai fitness yang paling besar. Kemudian diambil individu terbaik sebanyak μ (dalam program ini populasi awal sebanyak 5). Berikut ini Gambar 6 yang merupakan potongan kode program untuk melakukan pengurutan. Contoh output dari kode program pada Gambar 6 ditunjukkan pada Gambar 7.

```

public static void sort(double[][] ar) {
    Arrays.sort(ar, new Comparator<double[]>() {
        @Override
        public int compare(double[] int1, double[] int2) {
            double numOfKeys1 = int1[5];
            double numOfKeys2 = int2[5];
            return Double.compare(numOfKeys2, numOfKeys1);
        }
    });
}

```

Gambar 6. Potongan kode program untuk pengurutan populasi

GENERASI BARU					
x1	x2	sigma1	sigma2	f(x,y)	fitness
4.58	2.61	0.95	0.26	3500.0	3500.0
4.1	2.59	0.86	0.23	3100.0	3100.0
4.96	2.36	0.86	0.23	3000.0	3000.0
46.86	1.21	0.32	0.34	19300.0	-628700.0
46.78	1.09	0.29	0.31	19300.0	-628700.0
populasi terbaik: x1=4.58, x2=2.61, f(x,y)=3500.0 dengan fitness=3500.0					

Gambar 7. Output dari hasil pengurutan populasi

e. Iterasi

Generasi yang diproses pada program ini dibuat dinamis. Itu berarti user dapat menginputkan sendiri iterasi yang diinginkan. Namun, banyak iterasi awal yang telah ditetapkan pada program adalah sebanyak 3 iterasi. Berikut ini Gambar 8 yang merupakan output dari program ES secara keseluruhan, dengan iterasi sebanyak 3. Kemudian, menghasilkan individu terbaik pada iterasi ketiga yaitu dengan nilai $x_1 = 16$, $x_2 = 5$, dan maksimal keuntungan dari perusahaan yang akan didapat adalah $f(x,y) = 8900$.

Evolutionary Strategies

OPTIMASI FUNGSI BERKENDALA EVOLUTIONARY STRATEGIES

Maksimal fungsi $x_1 +$ x_2 Jumlah Iterasi

Kendala

Kendala 1 $x_1 +$ $x_2 \leq$

Kendala 2 $x_1 +$ $x_2 \leq$

Kendala 3 $x_1 +$ $x_2 \leq$

15.38	3.4	0.58	0.74	7500.0	7500.0
16.12	2.19	0.58	0.74	7400.0	7400.0
15.87	4.59	0.7	0.89	8900.0	8900.0
16.79	3.88	0.7	0.89	8800.0	8800.0
15.64	5.01	0.7	0.9	8900.0	8900.0
17.56	2.36	0.7	0.9	8200.0	8200.0
16.79	2.97	0.64	0.81	8300.0	8300.0
15.82	2.21	0.64	0.81	7400.0	7400.0
15.37	3.13	0.64	0.81	7500.0	7500.0
15.2	3.51	0.64	0.81	8000.0	8000.0
16.43	1.93	0.64	0.81	7400.0	7400.0
15.84	2.71	0.64	0.81	7900.0	7900.0

GENERASI BARU

x1	x2	sigma1	sigma2	f(x,y)	fitness
15.87	4.59	0.7	0.89	8900.0	8900.0
15.64	5.01	0.7	0.9	8900.0	8900.0
16.79	3.88	0.7	0.89	8800.0	8800.0
16.39	3.81	0.64	0.81	8400.0	8400.0
16.79	2.97	0.64	0.81	8300.0	8300.0

pupulasi terbaik: x1=15.87, x2=4.59, f(x,y)=8900.0 dengan fitness=8900.0

Gambar 8. Contoh program yang dijalankan