

**IMPLEMENTASI EVOLUTIONARY STRATEGY DALAM
OPTIMASI BERKENDALA**



Ida Bagus Nyoman Pascima (15/388483/PPA/04922)

**PROGRAM STUDI S2 ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA**

2016

1. KASUS

Permasalahan yang diangkat adalah permasalahan optimasi dari sebuah keterbatasan (kendala). Permasalahan ini sering ditemui di kehidupan sehari-hari misalnya seperti pencampuran pakan ternak untuk mendapat hasil optimum, pemberian jumlah pembuatan sesuatu untuk keuntungan maksimum. Kasus Ini diambil dari Modul Kuliah Semester Ganjil 2013-2014 **ALGORITMA EVOLUSI** Karangan Wayan Firdaus Mahmudy Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

Contoh Masalah.

Sebuah perusahaan yang akan memproduksi dua jenis

lemari, 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

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

Persediaan bahan baku kayu, aluminium, dan kaca di gudang adalah 350, 200, dan 300. Jika keuntungan penjualan sebuah lemari A sebesar 400 dan B sebesar 500, berapakah banyaknya lemari A dan B harus diproduksi agar didapatkan keuntungan maksimum?

Pada implementasi, aplikasi ini dapat berjalan dengan data yang dinamis dan dengan kasus yang serupa dengan kasus diatas. Jika banyaknya lemari yang harus diproduksi dilambangkan x_1 dan x_2

maka fungsi tujuannya adalah sebagai berikut:

$$f(x_1, x_2) = 400x_1 + 500x_2 \quad (1)$$

Kendala ketersediaan bahan baku:

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

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

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

2. EVOLUTIONARY STRATEGY

a. Inisialisasi

Kromosom dipresentasikan dengan 6 gen dimana gen tersebut adalah $x_1, x_2, \sigma_1, \sigma_2, f(x_1, x_2), Fitness$. Nilai x_1, x_2 merupakan representasi dari lemari A dan lemari B. x_1, x_2 dibangkitkan dengan nilai random[0,50]. σ_1, σ_2 adalah nilai random[0,1] untuk mendapatkan nilai x' (*offspring*). $f(x_1, x_2)$ adalah maksimal keuntungan yang mungkin didapat dan fitness merupakan nilai kekuatan individu (individu paling fit). Untuk menentukan nilai fitness dapat menggunakan persamaan:

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

$f(x_1, x_2)$: Diambil dari persamaan (1)

M : Nilai konstanta yang nilainya bulat untuk membuat jarak yg jauh pada indifidu yang melanggar.

$$c_1 = \begin{cases} 0, & \text{jika } 10x_1 + 20x_2 \leq 350 \\ (10x_1 + 20x_2) - 350, & \text{selainnya} \end{cases} \quad (6)$$

$$c_2 = \begin{cases} 0, & \text{jika } 9x_1 + 8x_2 \leq 200 \\ (9x_1 + 8x_2) - 200, & \text{selainnya} \end{cases} \quad (7)$$

$$c_3 = \begin{cases} 0, & \text{jika } 12x_1 + 18x_2 \leq 300 \\ (12x_1 + 18x_2) - 300, & \text{selainnya} \end{cases} \quad (8)$$

Banyaknya individu dalam populasi awal μ adalah sebanyak dimana nilai μ dapat ditentukan oleh pengguna aplikasi. Namun dalam kasus ini nilai μ diinisialisasi dengan 4.

b. Reproduksi

Siklus ES yang digunakan adalah siklus $(\mu+\lambda)$ dimana reproduksi yang digunakan hanyalah proses mutasi saja dengan generasi baru yang dihasilkan adalah hasil *elitism* dari indifidu *offspring* dan induk. *Offspring* yang digunakan ialah sejumlah $\mu * \lambda$ dimana seperti halnya μ nilai λ dapat juga ditentukan oleh pengguna. Untuk penentuan nilai x' dapat menggunakan persamaan berikut:

$$x' = x + \sigma N(0,1) \quad (9)$$

Dimana:

$x = x$ dari induk.

σ = nilai σ dari induk

$N(0,1)$ = nilai acak yang mengikuti sebaran normal dengan rata-rata sebesar 0 dan standard deviasi sebesar 1.

Pada aplikasi ini nilai $N(0,1)$ didapatkan dengan membangkitkan dua bilangan random r_1 dan r_2 pada interval $[0,1]$. Adapun persamaan yang digunakan ialah:

$$N(0,1) = \sqrt{-2 \cdot \ln r_1} \sin 2\pi r_2 \quad (10)$$

c. Seleksi

Pada bagian seleksi digunakan *elitism* dimana pengurutan dilakukan pada *offspring* dan induk. Untuk mendapatkan individu yang paling fit maka pengurutan dilakukan dengan mengurutkan dari individu yang memiliki nilai fitness tertinggi (*descending*). Setelah melakukan pengurutan maka akan dilakukan pemilihan individu terbaik sebanyak μ

Aplikasi ini memberikan keleluasaan pengguna dalam menentukan jumlah perulangan yang akan digunakan. Pengguna dapat dengan leluasa melakukan pengaturan penggunaan perulangannya. Terdapat pula pengaturan μ dan λ yang dapat ditentukan oleh pengguna namun disini memiliki keterbatasan dimana array penampung hanya disediakan sebanyak 10000 elemen. Jadi jumlah $\mu + (\mu * \lambda) \leq 10000$.

3. LAMPIRAN

Tampilan aplikasi.

IMPLEMENTASI ALGORITMA EVOLUTION STRATEGIES UNTUK OPTIMASI

Fungsi Maksimum: 400 X1 + 500 X2

Fungsi Kendala:

- 10 X1 + 20 X2 ≤ 350
- 9 X1 + 8 X2 ≤ 200
- 12 X1 + 18 X2 ≤ 300

Setting:

- Iterasi: 10
- Miu: 4
- Lamda: 2

HASIL

9 X1 + 14 X2

Run

Exit

	X1	X2	Tou 1	Tou 2	Fitness
▶	8.734088937008...	13.112909559975	0.2716355740032	1.2839025177495	-31900
	9.010807601874...	13.12432761340...	0.2222472878208	1.0504656963405	-31900
	8.825149967376...	13.46616081479...	0.246941430912	1.167184107045	-31900
	8.384879031651...	14.28232983199...	0.246941430912	1.167184107045	-47800

Tampilan elemen array yang disediakan.

```
Public populasi(10000) As anggota
Public rx1(10000), rx2(10000), c1(10000), c2(10000), c3(10000) As Double
Public n1(10000), n2(10000) As Double
```

Tampilan gen yang direpresentasikan pada aplikasi

```
2 references
Structure anggota
    Implements IComparable
    Dim x1 As Double
    Dim x2 As Double
    Dim tou1 As Double
    Dim tou2 As Double
    Dim fx As Double
    Dim fitness As Double

0 references
Public Function CompareTo(ByVal obj As Object) As Integer _
    Implements System.IComparable.CompareTo
    Return Me.fitness.CompareTo(CType(obj, anggota).fitness)
End Function

End Structure
```

Procedure untuk membentuk distribusi normal.

2 references

```
Public Function normal(min, max) As Double
    Dim nil, r1, r2 As Double
    r1 = aRand(min, max)
    r2 = aRand(min, max)
    nil = (Math.Sqrt(-2 * Math.Log(r1))) * (Math.Sin(2 * 3.14 * r2))
    Return nil
End Function
```

Procedure untuk membuat populasi awal dan populasi *offspring*

2 references

```
Public Sub create_pop(awal As Integer, offspring As Boolean)
    If offspring = False Then
        For i As Integer = awal To miu + (awal - 1) Step 1
            populasi(i).x1 = aRand(0, 50)
            populasi(i).x2 = aRand(0, 50)
            populasi(i).tou1 = aRand(0, 1)
            populasi(i).tou2 = aRand(0, 1)
        Next
    Else
        Dim j As Integer
        For i As Integer = awal To (miu * lamda) + (awal - 1) Step 1
            n1(i) = normal(0, 1)
            n2(i) = normal(0, 1)

            j = ((i - awal) \ lamda) 'index matrik dari 1

            populasi(i).x1 = populasi(j).x1 + (populasi(j).tou1 * n1(i))
            populasi(i).x2 = populasi(j).x2 + (populasi(j).tou2 * n2(i))
        Next
    End If
End Sub
```

Procedure untuk merubah σ_1 dan σ_2 pada generasi baru

1 reference

```
Public Sub create_tou(awal As Integer)
    Dim j As Integer
    For i As Integer = awal To (miu * lamda) + (awal - 1) Step 1

        j = ((i - awal) \ lamda)

        If populasi(i).fitnes > populasi(j).fitnes Then
            populasi(i).tou1 = populasi(j).tou1 * 1.1
            populasi(i).tou2 = populasi(j).tou2 * 1.1
        Else
            populasi(i).tou1 = populasi(j).tou1 * 0.9
            populasi(i).tou2 = populasi(j).tou2 * 0.9
        End If
    Next
End Sub
```

Procedure pengurutan array untuk seleksi

```
1 reference
Public Sub seleksi(index As Integer)
    Array.Sort(populasi, 0, index)
    Array.Reverse(populasi, 0, index)
    For i As Integer = miu To index - 1 Step 1
        populasi(i).x1 = 0
        populasi(i).x2 = 0
        populasi(i).tou1 = 0
        populasi(i).tou2 = 0
        populasi(i).fx = 0
        populasi(i).fitnes = 0
    Next
End Sub
```

Procedure untuk pemberian nilai $c_1, c_2, c_3, f(x_1, x_2), Fitness$

```
2 references
Public Sub inisialisasi(awal As Integer, offspring As Boolean)
    Dim k As Integer
    If offspring = False Then
        k = miu
    Else
        k = lamda * miu
    End If

    For i As Integer = awal To k + (awal - 1) Step 1
        rx1(i) = Math.Round(populasi(i).x1, 0)
        rx2(i) = Math.Round(populasi(i).x2, 0)

        'Perhatikan tanda
        populasi(i).fx = (x1_masalah.Text * rx1(i)) + (x2_masalah.Text * rx2(i))

        c1(i) = ((x1_kendala_1.Text * rx1(i)) + (x2_kendala_1.Text * rx2(i))) - h_kendala_1.Text
        If c1(i) < 0 Then
            c1(i) = 0
        End If

        c2(i) = ((x1_kendala_2.Text * rx1(i)) + (x2_kendala_2.Text * rx2(i))) - h_kendala_2.Text
        If c2(i) < 0 Then
            c2(i) = 0
        End If

        c3(i) = ((x1_kendala_3.Text * rx1(i)) + (x2_kendala_3.Text * rx2(i))) - h_kendala_3.Text
        If c3(i) < 0 Then
            c3(i) = 0
        End If

        'Fitness
        populasi(i).fitnes = populasi(i).fx - (m * (c1(i) + c2(i) + c3(i)))
    Next
End Sub
```
