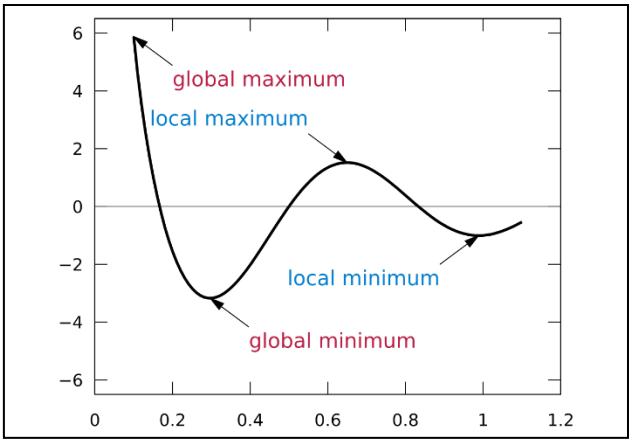
# Laporan Pendekatan Algoritma Genetika Untuk Mencari Nilai Minimum dari Suatu Fungsi

**Muhammad Shabri Arrahim Mardi**

Alamat : Jl. Perintis No. 46A, Baubau, Sulawesi Tenggara
e-mail: shabri@student.telkomuniversity.ac.id

## 1. PENDAHULUAN

Formula dalam model persamaan matematika biasanya dibangun dari permasalahan optimasi tanpa kendala berupa pemaksimalan atau peminimalan dari fungsi. Tetapi akan tidak mudah jika harus menetukan maximasi dan minimisasi dari suatu fungsi *f* yang merupakan fungsi multidimensional yang terdiri dari beberapa minimum local dan terdapat satu minimum global, karena ada kemungkinan solusi yang dihasilkan bukan merupakan nilai minimum global. Untuk permasalahan seperti ini, diperlukan suatu teknik pencarian yang dapat menjangkau keseluruhan ruang solusi untuk menghindari *pencarian* terjebak pada titik minimum lokal.




**Gambar 1 Minimum local dan Minimum Global**

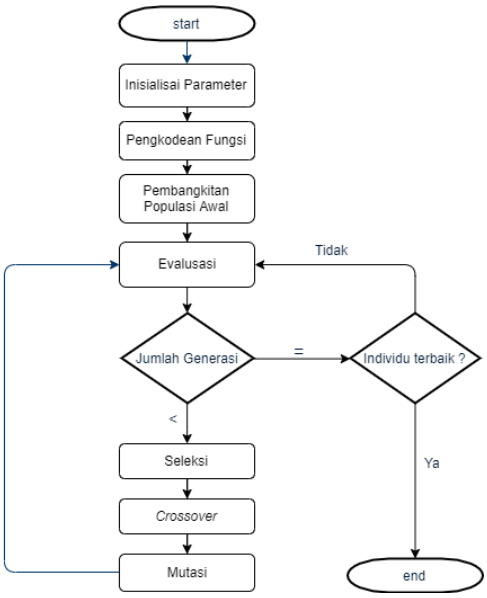
## 2. METODE

Sebagai algoritma pencarian evolusioner, Algoritma Genetika melakukan pencarian dengan meniru evolusi yang terjadi di dunia nyata. Proses diawali dengan membangkitkan populasi yang terdiri dari sekumpulan individu yang selanjutnya disebut kromosom. Kromosom tersusun atas satuan terkecil yang disebut gen, sedangkan posisi gen dalam kromosom disebut lokus. Nilai dari gen disebut allele. Setiap kromosim memiliki nilai ukuran yang disebut nilai *fitness.*

Proses evolusi dilakukan pertama kali dengan menyeleksi kromosom. Seleksi dilakukan berdasarkan nilai fitness, kromosom dengan nilai fitness yang tinggi memiliki peluang lebih besar terseleksi untuk dilakukan *crossover*. Setelah proses seleksi, akan dipilih 2 individu untuk dilakukan proses penyilangan berdasarkan titik potong yang dipilih secara acak (*crossover*). Berikutnya dilakukan proses *mutasi* yaiutu dengan mengganti allele dari suatu lokus (*gen*). Proses ini akan terus berulang sampai mendapatkan populasi dengan kromosom yang dianggap '*sempurna*'.

## 3. PEMBAHASAN DAN HASIL

Proses pencarian solusi dalam Algoritma Genetika dengan alur sebagai berikut :




**Gambar 2 Proses Pencarian Solusi GA**

### 1. Inisialisasi Parameter

Ini inisialisasi parameter dilakukan dengan cara merubah string biner menjadi panjang n dimaksudkan untuk melakukan pemetaan nilai pada $-3 \leq x_1 \leq 3$ dan $-2 \leq x_2 \leq 2$ dimana setiap bit bernilai 1 dan 0.

Selanjutnya transformasi dari string biner ke bilangan *decimal* $x_1$ dan $x_2$ dilakukan dengan rumus :

$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^{N} 2^{-i}} (g_1.2^{-1} + g_2.2^{-2} + ... + g_N.2^{-N})$$

Pada kasus ini Panjang bit yang digunakan untuk variable $x_1$ dan $x_2$ adalah 5. Maka Panjang string unuk setiap kromosom didalam populasi adalah n = $n_1 + n_2 = 10$. Contoh: 1 0 0 1 0 0 1 1 1 1 merupakan representasi dari pasangan nilai $(x_1, x_2) = (-2.225806451612903, 0.0)$ dengan nilai $f(x_1, x_2) = 8.806428895138282$.

**2. Pengkodean fungsi**

Pada tahap ini fungsi – fungsi yang dibutuhakan didalam program dibangun. Berikut

```
def fitness(value): return abs(1/(value + np.min(value)))
```

merupakan contoh fungsi *fitness* pada program :

**3. Pembangkitan populasi awal**

Populasi awal dibangun dengan masing – masing panjang himpunan string biner 10. Berikut merupakan 3 individu pertama dari populasi berukuran 2000 yang dibangkitkan pada program menggunakan software Python

**Tabel 1. Populasi Awal**

| No. | Populasi Awal |
|---|---|
| 1 | 1 0 1 0 0 0 1 1 1 0 |
| 2 | 1 1 1 0 1 1 0 0 0 1 |
| 3 | 1 1 1 1 1 1 0 0 1 1 |

**4. Evaluasi**

Evaluasi dilakukan dengan cara menghitung nilai *fitness* setiap kromosom/individu. Berikut nilai *fitness* dari setiap individu pada Tabel 1 :

**Tabel 2. Nilai Fitness**

| Kromosom | Nilai Fitness |
|---|---|
| 1 | 5.48635772e-01 |
| 2 | 2.78001194e-02 |
| 3 | 9.11384833e-03 |

**5. Proses Seleksi**

Proses seleksi dilakukan dengan menghitung nilai fitness, total fitness, peluang fitness dan peluang kumulatif fitness untuk setiap kromosom/individu. Pemilihan individu yang akan terseleksi dilakukan dengan membangkitkan bilangan random rand(0,1). Jika (kumulatiffitnes[i]/fitness.sum()) > rand(0,1) maka individu ke-i yang terpilih.

**6. *Crossover***

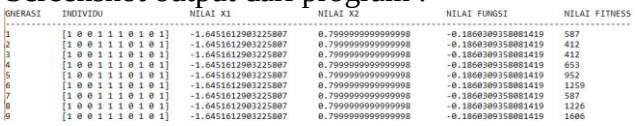
*Crossover* dilakukan dengan mengambil beberapa titik (*point*) sebagai titik penyilangan untuk menentukan pemotongan kromosom. *Crossover* terjadi jika nilai random yang dibangkitkan tidak lebih dari cross_rate yang telah di inisialisai, yaitu sebesar 0.8 (80%). np.random.rand() < CROSS_RATE.

**7. Mutasi**

Dalam kasusu ini mutation_rate di inisialisasi sebesar 0.1, dengan artian sebesar 1% pada gen dalam populasi akan mengalami mutasi. Mutasi terjadi jika nilai random yang dibangkitkan tidak lebih dari mutation_rate. np.random.rand() < MUTATION_RATE. Jika gen bernilai 1 akan diganti menjadi 0 dan begitu juga sebaliknya.

Berikut ini merupakan solusi yang dianggap optimal pada fungsi $f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$ dilakukan dengan bantuan software Python. Proses iterasi dilakukan sebanyak 100 kali dan dari running program diperoleh nilai minimum sebesar $f(x_1, x_2) = -0.1860309358081419$. Sesuai dengan kromosom [1 0 0 1 1 1 0 1 0 1]. Hasil representasi $(x_1, x_2) =$ $(-1.6451612903225807 \quad 0.7999999999999998)$

Screenshot output dari program :



| GNERASI | INDIVIDU | NILAI X1 | NILAI X2 | NILAI FUNGSI | NILAI FITNESS |
|---|---|---|---|---|---|
| 1 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 587 |
| 2 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 412 |
| 3 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 412 |
| 4 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 653 |
| 5 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 952 |
| 6 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 1259 |
| 7 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 587 |
| 8 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 1226 |
| 9 | [1 0 0 1 1 1 0 1 0 1] | -1.6451612903225807 | 0.7999999999999998 | -0.1860309358081419 | 1606 |

**4. KESIMPULAN**

Panjang dari kromosom dan jumlah populasi mempengarusi hasil optimasi dari nilai minimasi suatu fungsi. Semakin besar nilai Panjang kromosom maka ruang pencarian nilai optimasi akan semakin besar pula, dengan memperbesar banyak populasi maka ruang pencarian yang besar akan bisa teratasi.

Ukuran peluang mutasi sebesar 1% dimasksudkan untuk meminimalkan kromosom – kromosom yang mengalami mutase. Nilai peluang mutasi yang lebih besar akan menyebabkan terjadinya kerusakan pada kromosom sehingga mempersulit pencarian nilai minimasi dari fungsi.