Laporan Hasil Penelitian Terhadap Implementasi *Genetic Algorithm*

Disusun untuk memenuhi salah satu tugas mata kuliah Pengantar Kecerdasan Buatan

Oleh :
Herman Gemilang / 1301204014

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
2022

1. Pendahuluan

Algoritma Genetika adalah metode untuk menyelesaikan persoalan optimasi berbasis teori evolusi dalam biologi. Algoritma ini bekerja pada populasi calon penyelesaian yang disebut kromosom yang awalnya dibangkitkan secara random dari ruang penyelesaian fungsi tujuan. Dengan menggunakan mekanisme operator genetik yaitu persilangan dan mutasi populasi dievolusikan melalui fungsi fitness yang diarahkan pada kondisi konvergensi.Algoritma genetika diperkenalkan pertama kali oleh John Holland (1975) dari Universitas Michigan.John Holland menyatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan ke dalam terminologi genetika.

Sifat algoritma genetika adalah mencari kemungkinan dari calon solusi untuk mendapatkan solusi yang optimal dalam penyelesaian masalah. Ruang cakupan dari semua solusi yang layak, yaitu berbagai objek diantara solusi yang sesuai, yang dinamakan ruang pencarian. Tiap titik didalam ruang pencarian mempresentasikan satu solusi yang layak. Tiap solusi yang layak dapat ditandai dengan nilai fitnessnya. Solusi yang dicari dalam algoritma genetika adalah titik (satu atau lebih) diantara solusi yang layak dalam ruang pencarian.

2. Tujuan Penilitian

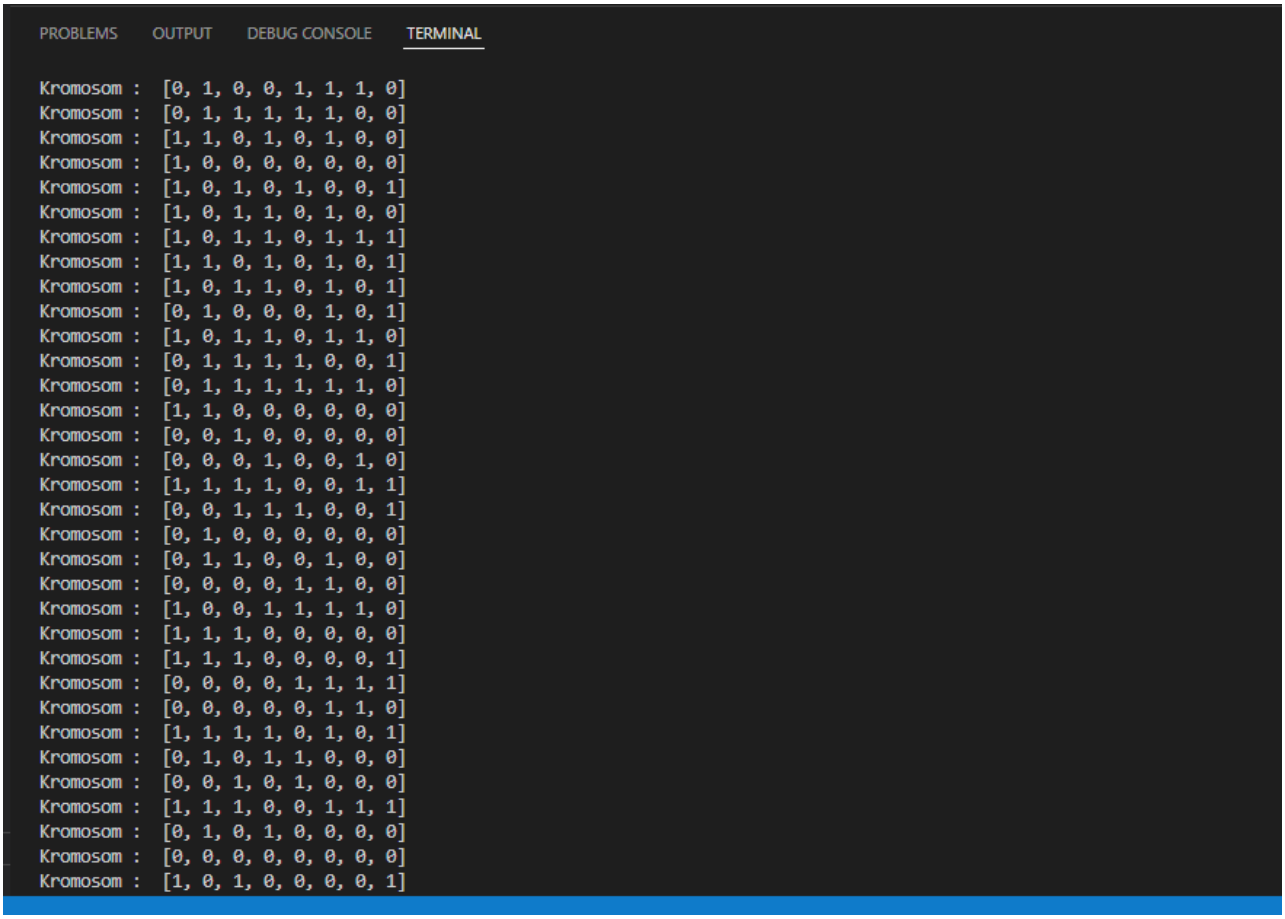
1. Desain Kromosom dan Metode Pendekodean
2. Ukuran Populasi
3. Metode Pemilihan Orang Tua
4. Pemilihan dan teknik operasi genetik (crossover dan mutasi)
5. Probabilitas operasi genetik (Pc dan Pm)
6. Metode Pergantian Generasi (seleksi survivor)
7. Kriteria Penghentian Evolusi

3. Hasil Penilitian

3.1 Desain Kromosom dan Metode Pendekodean

Desain Kromosom yang digunakan dalam penilitian ini menggunakan kromosom yang memiliki panjang 8 dengan pembagian 4:4. Dengan mengambil nilai acak dan unik

dalam setiap pemilihan 1/0 di setiap barisnya sehingga tidak ada kromosom yang sama antara satu dengan yang lainnya kecuali antar generasi.



Dekode Kromosom merupakan suatu cara pengkodean isi kromosom menjadi suatu nilai tertentu yang mana hasil dekodenya mewakili tiap variabel dan terdiri dari beberapa jumlah bit yang ada. Hal ini dilakukan guna untuk merepresentasikan sifat genotip dan fenotip yang ada dari suatu populasi. Sifat genotip dari populasi ini dilambangkan sebagai suatu deret biner yang ada pada kromosom, sedangkan sifat fenotipnya merupakan nilai hasil dekode dari kromosom yang ada. Sifat genotip ini dipakai saat pada proses pindah silang(cross-over). Proses dekode tersebut menggunakan metode bernama individual representation binary encoding yang berfungsi untuk membagi kromosom menjadi x dan y. Indeks pertama dari kromosom sampai middle index kromosom menjadi x kromosom, sedangkan dari middle index kromosom hingga index terakhir kromosom menjadi y.

```
class Chromosome:
    def __init__(self):
        self.bit = random.choices([0, 1], k=8)
        self.x = self.decoding(batas_atasX, batas_bawahX, self.bit[:4])
        self.y = self.decoding(batas_atasY, batas_bawahY, self.bit[4:])
```

3.2 Ukuran Populasi

Dalam penelitian ini, ukuran populasi yang kami gunakan sebanyak 100 populasi dalam setiap generasi. Untuk ukuran gen yang kami gunakan yaitu 100 generasi .

```
84
85  def seleksi_survivor():
86      populasi.sort(key=lambda c: h(c.x, c.y))
87
88      while len(populasi) != 100:
89          populasi.pop()
90
91
92  populasi = []
93  generasi = 1
94  while len(populasi) != 100:
95      c = Chromosome()
96
97      if not exist(populasi, c):
98          populasi.append(c)
99  seleksi_survivor()
00  hasil(populasi)
01
02
03  while generasi < 100:
04      parent = seleksi_parent(2)
05      crossover(parent[0], parent[1])
06      seleksi_survivor()
07
08      generasi += 1
09      hasil(populasi)
10
```

3.3 Pemilihan Orang Tua

Pada pemilihan orang tua menggunakan metode Roulette Wheel. Dalam penelitian ini kita memanggil fitnes setiap kromosom pada setiap populasi terlebih dahulu dalam bentuk array. Setelah itu menjumlahkan semua fitnes. Jumlah itu dimasukkan ke array setiap populasi. Untuk pemilihannya menggunakan random dengan bobotnya berdasarkan fitnes setiap kromosom dibagi dengan jumlah kromosomnya.

```
def seleksi_parent(k):
    parent = []
    arr_fitness = list(map(lambda c: f(c.x, c.y), populasi))
    arr_weight = [arr_fitness[i] / sum(arr_fitness)
                  for i in range(len(populasi))]
    while len(parent) != k:
        kandidat = random.choices(populasi, weights=arr_weight)[0]
        if not exist(parent, kandidat):
            parent.append(kandidat)
    return parent
```

3.4 Pemilihan dan Operasi Genetik

Proses ini disebut juga dengan proses kawin silang (*Crossover)* dan mutasi. Proses operasi *crossover* dirancang untuk mencari kemungkinan yang lebih baik dari anggota populasi yang telah ada. Dari pasangan induk (*parent*) yang terpilih berdasarkan hasil seleksi fungsi fitness.

Mutasi adalah perubahan pada bit tunggal (bit 0 jadi 1 dan sebaliknya) anggota populasi yang terpilih. Banyaknya bit yang mengalami mutasi pada setiap generasi diatur oleh probabilitas mutasi (Pm).

Pada proses kawin silang ini kromosom anak terbuat dari kromosom ortu pertama untuk posisi 4 depan dan kromosom ortu kedua untuk posisi 4 belakang dan akan dilakukan mutasi jika kromosom anak sudah melebihi batas maksimumnya.

```python
def crossover(ortu1, ortu2):
    posisi = random.randint(1, len(ortu1.bit) - 2)

    bit_chro1 = ortu1.bit[:posisi] + ortu2.bit[posisi:]
    bit_chro2 = ortu2.bit[:posisi] + ortu1.bit[posisi:]

    rng_mutasi = random.uniform(0, 100)
    if rng_mutasi > (100 - 0.5):
        posisi_mutasi = random.randint(0, len(bit_chro1) - 1)
        if bit_chro1[posisi_mutasi] == 1:
            bit_chro1[posisi_mutasi] = 0
        else:
            bit_chro1[posisi_mutasi] = 1

    rng_mutasi = random.uniform(0, 100)
    if rng_mutasi > (100 - 0.5):
        posisi_mutasi = random.randint(0, len(bit_chro2) - 1)
        if bit_chro2[posisi_mutasi] == 1:
            bit_chro2[posisi_mutasi] = 0
        else:
            bit_chro2[posisi_mutasi] = 1
```

```
Kromosom :  [0, 1, 1, 1, 1, 0, 1, 0]
X :  -0.3333333333333339
Y:  1.666666666666666
fitness:  1.3032747557075801
====== generasi ke-100 ======
```

3.5 Probabilitas Crossover (Pc) Probabilitas Mutasi (Pm)

Peluang dari probabilitas crossover sebesar 100% sehingga anak hasil crossover tidak akan melebihi jumlah populasi awal.Probabilitas mutasi yang digunakan yaitu 0,5. Apabila nilai peluang mutasi lebih besar dari nilai random maka akan dilakukan mutasi terhadap nilai mutasi yang random tersebut.

```
rng_mutasi = random.uniform(0, 100)
if rng_mutasi > (100 - 0.5):
    posisi_mutasi = random.randint(0, len(bit_chro1) - 1)
    if bit_chro1[posisi_mutasi] == 1:
        bit_chro1[posisi_mutasi] = 0
    else:
        bit_chro1[posisi_mutasi] = 1

rng_mutasi = random.uniform(0, 100)
if rng_mutasi > (100 - 0.5):
    posisi_mutasi = random.randint(0, len(bit_chro2) - 1)
    if bit_chro2[posisi_mutasi] == 1:
        bit_chro2[posisi_mutasi] = 0
    else:
        bit_chro2[posisi_mutasi] = 1
```

3.6 Metode Pergantian Generasi (seleksi survivor)

Pada prosedur seleksi_survivor, program melakukan sorting terhadap individu-individu. Lalu, hasil yang menunjukan nilai paling buruk akan dieliminasi dengan syntax populasi.pop()

```
def seleksi_survivor():
    populasi.sort(key=lambda c: h(c.x, c.y))

    while len(populasi) != 50:
        populasi.pop()
```

3.7 Kriteria Penghentian Evolusi

Pemberhentian evolusi generasi di penelitian ini hingga generasi yang dibuat mencapai generasi ke 100.

```
while generasi < 100:
    parent = seleksi_parent(2)
    crossover(parent[0], parent[1])
    seleksi_survivor()

    generasi += 1
    hasil(populasi)
```

Kesimpulan

Genetic Algorithm ini bertujuan untuk mencari nilai minimum dari fungsi yang telah diberikan sebelumnya. Setelah diteliti lebih dalam ternyata hasil dari genetic algorithm itu sendiri tidak menentu dan akan berubah-ubah yang juga dipengaruhi beberapa faktor. Panjang bit kromosom akan berpengaruh terhadap nilai fitness. Semakin panjang kromosomnya, maka variasi dari nilai x,y akan semakin banyak yang akan menyebabkan nilai range fitness akan berbeda jauh antar generasi. Selain itu, semua nilainya bergantung pada parameter. Ketika parameter berubah maka hasilnya pun akan berubah.

DAFTAR PUSTAKA

Achmad Basuki. 2003. “Strategi Menggunakan Algoritma Genetika” dalam *slide presentasi arsip Politeknik Elektronika Negeri Surabaya PENS-ITS*

Agus Wahyu, Wayan firdaus. 2010. “Penerapan Algoritma Genetika Pada Sistem Rekomendasi Wisatau Kuliner” dalam *Jurnal Ilmiah KURSOR vol. 5 No. 4. Prodi Ilmu Komputer, Jurusan Matematika, FMIPA, Universitas Brawijaya*

Lusiana Paranduk , Aida Indriani , Muhammad Hafid , Suprianto. 2018. “Sistem Informasi Penjadwalan Mata Kuliah Menggunakan Algoritma Berbasis Web” dalam *Seminar nasional aplikasi teknologi informasi (SNATi*

Link Video: https://youtu.be/xec0X92Lclo