Laporan Observasi
Tugas Pemrograman 01 CII-2M3 Pengantar Kecerdasan Buatan
Genap 2020/2021

Disusun Oleh :
1. Irwan Reza Firmansyah (1301194018)
2. Ramadito Ferdian Assa (1301194005)
3. Artisa Bunga Syahputri (1301194007)

FAKULTAS INFORMATIKA
PROGRAM STUDI S1 INFORMATIKA
UNIVERSITAS TELKOM
2020/2021

Analisis, desain, dan implementasi algoritma Genetic algorithm (GA) ke dalam suatu program komputer untuk menemukan nilai minimum dari fungsi:

$$h(x,y)=(\cos x^2 * \sin y^2)+(x+y)$$

dengan Batasan $-1 \leq x \leq 2$ dan $-1 \leq y \leq 1$.

Hal yang di Observasi:

1. Desain kromosom dan metode pendekodean
   (generated individual)
   Evolusi biasanya dibentuk dari populasi individu(kromosom) yang dibuat secara random. Dimana desain kromosom yang digunakan pada algoritma yang kami bangun menggunakan import random pada library python yang merandom nilai antara 1 dan 0 sebagai bagian dari kromosom sepanjang n panjang kromosom. Setelah didapatkan nilai random antara 1 dan 0, maka terbentuklah nilai biner yang kemudian akan di decode. Pada proses pendekodean kami menggunakan metode individual representation binary encoding, yaitu metode yang membagi kromosom menjadi dua bagian x dan y dimana indeks pertama dari kromosom hingga indek tengah dari panjang kromosom menjadi bagian x dan sisa nya menjadi bagian y, yang kemudian bagian x akan diubah menjadi bilangan desimal dengan menggunakan rumus :

$$x = r_{min} + \frac{r_{max} - r_{min}}{\Sigma_{i=1}^{N} 2^{-i}} (g_1 * 2^{-1} + g_1 * 2^{-2} + \cdots + g_N * 2^{-N})$$

   Dan untuk bagian y juga akan diubah menjadi bilangan desimal dengan menggunakan rumus:

$$y = r_{min} + \frac{r_{max} - r_{min}}{\Sigma_{i=1}^{N} 2^{-i}} (g_1 * 2^{-1} + g_1 * 2^{-2} + \cdots + g_N * 2^{-N})$$

```python
def hitung_xy(kromosom,xr, yr):
  #potong kromosom menjadi 2 bagian untuk x dan y
  half_kromosom = len(kromosom)//2

  #rumus

  # menghitung bagian atas untuk x
  atas_x = 0
  kromosom_x = kromosom[:half_kromosom]
  for i in range(1, half_kromosom + 1):
    atas_x += kromosom_x[i - 1] * (2 ** -i)

  # menghitung bagian atas untuk y
  atas_y = 0
  kromosom_y = kromosom[half_kromosom:]
  for i in range(1, half_kromosom + 1):
    atas_y += kromosom_y[i - 1] * (2 ** -i)


  # menghitung bagian bawah untuk x dan y
  bawah = sum([2 ** -(i) for i in range (1, half_kromosom + 1)])

  # memasukkan bagian atas dan bawah dari rumus kedalam RUMUS untuk mencari x dan y
  x = xr["min"] + (atas_x * (xr["max"] - xr["min"]) / bawah)
  y = yr["min"] + (atas_y * (yr["max"] - yr["min"]) / bawah)


  return x,y
```

2. Ukuran populasi
   (20 dan 50)
   Untuk ukuran populasi yang kami gunakan dalam membangun algoritma ini adalah sebanyak 20 dan 50 pada setiap generasi. Alasan kami menggunakan 20 dan 50 populasi yaitu untuk membandingkan kecepatan pencarian generasi terbaik serta nilai maksimum dari fungsi yang kita observasi.

3. Seleksi orang tua: roulette wheel selection
   Roulette wheel selection adalah metode seleksi yang memilih orang tua berdasarkan nilai kecocokannya (fitness). Kromosom yang lebih baik memiliki persentase dipilih yang lebih besar.
   Proses seleksi bertujuan untuk memilih kromosom yang akan dijadikan sebagai parent kromosom induk pada proses crossover pindah silang.
   Pada roulette wheel selection, kromosom akan dipilih secara acak ditentukan dengan memperhitungkan nilai kelayakan masing-masing kromosom. Semakin besar nilai kelayakan suatu kromosom, semakin besar pula peluang kromosom tersebut untuk terpilih sebagai parent kromosom induk. Pengkodean roulette wheel dapat dianalogikan seperti permainan roda putar

```
def select_2parent(population, fitness_population):

  # Menggunakan Roulette Wheel Selection

  temp = 0

  for krom in range(len(population)):
    temp += fitness_population[krom]

  rng = random.random()

  krom = 0
  while rng > 0 :
    rng -= fitness_population[krom] / temp
    krom += 1

  return krom - 1
```

```
def parent_selection(population):

  n = len(population)


  fitness_population = []
  for krom in range(n):
    x, y = hitung_xy(population[krom], xr, yr)
    fitness_population.append(fitness(x, y))



  # Normalisasi fitness value
  min_ = min(fitness_population)
  max_ = max(fitness_population)

  for i in range(n):
    fitness_population[i] = (fitness_population[i] - min_) / (max_ - min_)


  # Menghitung total dari semua Fitness Population
  total_fitness = sum(fitness_population)


  # Melakukan Roulette Wheel untuk mendapatkan 2 parent

  parent = []

  while len(parent) != 2:
    krom = select_2parent(population, fitness_population)
    parent.append(population[krom])

  return parent
```

4. Pemilihan teknik crossover dan mutasi
   (teknik skema umum yaitu single point)
   Crossover adalah operasi genetik yang digunakan untuk menciptakan variasi dari kromosom dari satu generasi ke generasi berikutnya dengan dua orang tua yang diambil bagian nya untuk disilangkan dengan tujuan untuk menghasilkan kromosom yang unggul. Metode yang kami gunakan untuk crossover ini adalah dengan menggunakan salah satu teknik dari skema umum yaitu single point. Teknik single point adalah teknik persilangan dari dua orang tua dengan memilih titik silang dari kromosom, kemudian semua data di luar titik silang itu pada kromosom dipertukarkan antar dua orang tua.

```
def crossover1point(parent):

  len_krom = len(parent[0])

  titik_potong = random.randint(0, len_krom)


  child = [0, 0]
  child[0] = parent[0][:titik_potong] + parent[1][titik_potong:]
  child[1] = parent[1][:titik_potong] + parent[0][titik_potong:]

  return child
```

   Mutasi adalah operator genetika yang digunakan untuk memelihara keragaman genetik dari satu generasi ke generasi berikutnya. Teknik mutasi yang kami gunakan dalam membangun algoritma ini adalah dengan mengguna tipe mutasi bit string mutation, yaitu

mutasi terjadi pada bit-bit di kromosom dimana bit-bit yang bermutasi akan di flips secara acak.

```
def mutasi_krom(child, prob):

  len_krom = len(anak[0])

  for i in range(2):
    for j in range(len_krom):
      if random.random() <= prob:
        child[i][j] = [0, 1][not child[i][j]]

  return child
```

5. Probabilitas operasi genetik(pc dan Pm):
   probabilitas crossover pada algoritma ini yang akan kami observasi adalah sebesar 60% dan 80%.
   Probabilitas mutasi pada algoritma ini yang akan kami observasi adalah sebesar 5% dan 10%.
   Hal ini bertujuan untuk melihat pengaruh perubahan perubahan parameter terhadap proses kecepatan dalam pencarian generasi yang optimum

6. Metode penggantian populasi:
   Metode untuk pergantian populasi pada algoritma ini kami menggunakan metode generational replacement dimana generational replacement akan menghasilkan n off-springs , dimana n adalah ukuran populasi dan seluruh populasi akan diganti pada akhir iterasi. Pada metode generational replacement kita harus menambahkan mekanisme untuk memastikan individu terbaik tetap bertahan, mekanisme ini disebut elitisme
   Nilai elitisme yang kami gunakan dalam observasi ini adalah 2 dan 4 sehingga dengan itu kami mempertahankan 2 dan 4 individu terbaik pada kromosom di setiap generasi

7. Ukuran kromosom
   Untuk observasi ini kami menggunakan dua nilai kromosom yaitu 10 dan 15, ini dikarenakan untuk nilai kromosom yang berbeda nilai maksimum yang dihasilkan juga berbeda setelah dilakukan analisis dari hasil running pada algoritma yang dibangun. Nilai maksimum fungsi untuk kromosom yang berjumlah 10 adalah 2.4804 dan untuk nilai maksimum fungsi dari kromosom yang berjumlah 15 adalah 2.4817

8. Evolusi akan berhenti pada generasi ke 50
   Kami membatasi proses regenerasi hanya sampai generasi ke 50 saja yang kita observasi. Namun jika evolusi tidak dibatasi kemungkinan akan muncul nilai maksimum yang lebih tinggi di generasi-generasi berikutnya.

Fungsi Utama

```
generation = 50
generasi_terbaik = []
populasi = generate_populasi(len_krom, n_pop)

print(f"Prob. Crossover   : {prob_crossover}")
print(f"Prob. Mutasi      : {prob_mutasi}")
print(f"Banyak Populasi   : {n_pop}")
print(f"Panjang Kromosom  : {len_krom}")
print(f"Banyak Elitisme   : {n_elitisme}")
print(f"Banyak Generasi   : {generation}")

print()

for i in range(generation):

  fitness_populasi = []
  for kromosom in populasi:
    x,y = hitung_xy(kromosom, xr, yr)
    fitness_populasi.append(fitness(x, y))


  generasi_terbaik.append(max(fitness_populasi))


  sorted_populasi = [m for _, m in sorted(zip(fitness_populasi, populasi), reverse=True)]


  populasi_baru = sorted_populasi[:n_elitisme]


  if (i + 1) % 5 == 0:
    print(f"Generasi ke-{i + 1}, Best : {generasi_terbaik[i]}")


  while len(populasi_baru) != n_pop:

    orang_tua = parent_selection(populasi)

    if random.random() < prob_crossover:
      anak = mutasi_krom(crossover1point(orang_tua), prob_mutasi)
    else:
      anak = orang_tua + []

    populasi_baru = populasi_baru + anak

  populasi = populasi_baru + []
```

N_Populasi = 20
N_Kromosom= 10

No	CrossOver	Mutasi	Elistisme	Gen 5	Gen 10	Gen 15	Gen 20	Gen 25	Gen 30	Gen 35	Gen 40	Gen 45	Gen 50
1	0.6	0.1	2	2.4	2.4	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.4804
2	0.6	0.1	4	2.3731	2.4	2.4	2.4	2.45	2.45	2.45	2.45	2.45	2.45
3	0.6	0.05	2	2.3879	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.45
4	0.6	0.05	4	2.4107	2.4107	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.45
5	0.8	0.1	2	2.4213	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
6	0.8	0.1	4	2.4	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.4748	2.4748
7	0.8	0.05	2	2.3879	2.4337	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.45
8	0.8	0.05	4	2.3879	2.4337	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.45

N_Populasi = 50
N_Kromosom= 10

No	CrossOver	Mutasi	Elistisme	Gen 5	Gen 10	Gen 15	Gen 20	Gen 25	Gen 30	Gen 35	Gen 40	Gen 45	Gen 50
1	0.6	0.1	2	2.4	2.4748	2.4748	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
2	0.6	0.1	4	2.4748	2.4748	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
3	0.6	0.05	2	2.4213	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
4	0.6	0.05	4	2.4213	2.4591	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
5	0.8	0.1	2	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
6	0.8	0.1	4	2.4748	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
7	0.8	0.05	2	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804
8	0.8	0.05	4	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804	2.4804

N_Populasi = 20
N_Kromosom= 15

No	CrossOver	Mutasi	Elistisme	Gen 5	Gen 10	Gen 15	Gen 20	Gen 25	Gen 30	Gen 35	Gen 40	Gen 45	Gen 50
1	0.6	0.1	2	2.3672	2.4219	2.4455	2.4455	2.45	2.45	2.45	2.474	2.474	2.4817
2	0.6	0.1	4	2.3368	2.45	2.45	2.45	2.45	2.45	2.45	2.45	2.4807	2.4807
3	0.6	0.05	2	2.3368	2.3368	2.358	2.4341	2.45	2.45	2.45	2.4807	2.4807	2.4807
4	0.6	0.05	4	2.3368	2.3368	2.3754	2.45	2.45	2.45	2.45	2.45	2.4807	2.4807
5	0.8	0.1	2	2.4011	2.4307	2.4307	2.4455	2.4455	2.4533	2.4533	2.4782	2.4817	2.4817
6	0.8	0.1	4	2.4745	2.4745	2.4745	2.481	2.481	2.481	2.481	2.481	2.481	2.481
7	0.8	0.05	2	2.4455	2.45	2.4533	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817
8	0.8	0.05	4	2.4455	2.4786	2.4786	2.4786	2.481	2.481	2.481	2.481	2.481	2.481

N_Populasi = 50
N_Kromosom= 15

No	CrossOver	Mutasi	Elistisme	Gen 5	Gen 10	Gen 15	Gen 20	Gen 25	Gen 30	Gen 35	Gen 40	Gen 45	Gen 50
1	0.6	0.1	2	2.3665	2.4489	2.4489	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817
2	0.6	0.1	4	2.4612	2.481	2.481	2.481	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817
3	0.6	0.05	2	2.3485	2.414	2.4399	2.4807	2.4807	2.4817	2.4817	2.4817	2.4817	2.4817
4	0.6	0.05	4	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817
5	0.8	0.1	2	2.481	2.481	2.481	2.481	2.481	2.4817	2.4817	2.4817	2.4817	2.4817
6	0.8	0.1	4	2.3999	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817
7	0.8	0.05	2	2.4685	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817
8	0.8	0.05	4	2.3571	2.45	2.4786	2.481	2.4817	2.4817	2.4817	2.4817	2.4817	2.4817

KESIMPULAN

Dari hasil observasi yang kami dapatkan, ternyata setiap parameter dalam membangun algoritma ini memberikan pengaruh terhadap kecepatan program dalam menemukan nilai maksimum dari fungsi yang sudah ditentukan, parameter yang mempengaruhi diantaranya:

- Parameter Crossover
  Dari hasil observasi didapatkan pengaruh parameter crossover terhadap kecepatan pencarian nilai maksimum dari fungsi adalah untuk nilai crossover yang lebih besar maka pencarian cenderung lebih cepat dari pada yang nilai crossovernya lebih kecil
- Parameter Mutasi
  Dari hasil observasi yang dilakukan, untuk nilai mutasi yang lebih kecil maka kebanyakan akan mendapatkan nilai maksimum lebih cepat ketimbang nilai mutasi yang lebih besar dimana nilai mutasi berada di rentang 1% - 10 %

- Elitisme
  Setelah melakukan observasi kecenderungan dari hasil observasi di atas adalah elitisme yang kecil lebih cepat dalam menemukan nilai maksimum dari fungsi yang telah ditentukan dengan memastikan elitisme yang digunakan bernilai genap.

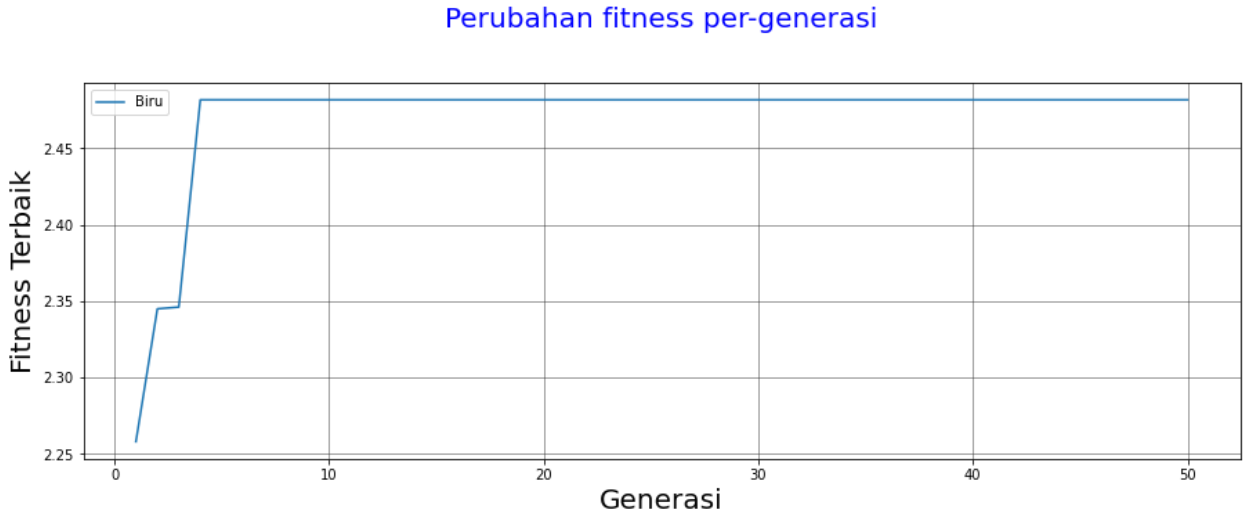
- Jumlah populasi
  Dari hasil observasi didapatkan bahwa untuk jumlah populasi yang lebih besar ternyata lebih cepat dalam menemukan nilai maksimum dari pada untuk jumlah populasi yang lebih sedikit
- Panjang kromosom
  setelah melakukan observasi terkait panjang kromosom ternyata untuk panjang kromosom tertentu dapat memberikan nilai maksimum yang berbeda dengan batasan gen evolusi yang sudah ditentukan.

Setiap parameter yang digunakan untuk observasi saling mempengaruhi satu sama lain sehingga kecepatan pencarian dapat berubah-ubah untuk parameter tertentu.

Nilai GA terbaik

```
Prob. Crossover    : 0.6
Prob. Mutasi       : 0.05
Banyak Populasi    : 50
Panjang Kromosom   : 15
Banyak Elitisme    : 4
Banyak Generasi    : 50

Generasi ke-5, Best : 2.4817210964013796
Generasi ke-10, Best : 2.4817210964013796
Generasi ke-15, Best : 2.4817210964013796
Generasi ke-20, Best : 2.4817210964013796
Generasi ke-25, Best : 2.4817210964013796
Generasi ke-30, Best : 2.4817210964013796
Generasi ke-35, Best : 2.4817210964013796
Generasi ke-40, Best : 2.4817210964013796
Generasi ke-45, Best : 2.4817210964013796
Generasi ke-50, Best : 2.4817210964013796
```



LINK VIDEO PRESENTASI

- Irwan Reza Firmasnyah

  https://drive.google.com/file/d/1CBmvdfuxRkCk-3G386E6F74H3x-L8Ysz/view?usp=sharing

- Artisa Bunga Syahputri

  https://drive.google.com/file/d/1uh14G01dkQrVEVwXnZzUBjQxNRN7o_M7/view?usp=sharing

- Ramadito Ferdian Assa

  https://drive.google.com/file/d/1E89yeHmpU8xGnHC2MonZevWmfZzAqjXg/view?usp=sharing