Manualisasi Perhitungan dan Analisis Algoritma Genetik

agungsetiabudi@ub.ac.id

Manualisasi Algoritma Genetik dengan Representasi Real untuk Meminimalkan $f(x) = x^2$ dalam Rentang $[-10, 10]$

- Dalam pendekatan ini, kita akan menggunakan **representasi real** untuk kromosom, bukan representasi biner seperti sebelumnya.
- Representasi real lebih efisien karena tidak memerlukan konversi biner ke desimal dan dapat menangani nilai kontinu langsung.

Tahap 1: Inisialisasi Populasi

- Misalkan kita memiliki **populasi awal** dengan **4 individu**, diambil secara acak dalam rentang $[-10, 10]$:

Individu	Nilai x	$f(x) = x^2$
1	-8.71	75.91
2	-1.61	2.59
3	4.68	21.89
4	6.13	37.59

- Karena kita ingin **meminimalkan** $f(x) = x^2$, maka nilai fitness dapat dihitung dengan rumus:
  - $\text{fitness} = \frac{1}{f(x)+1}$
  - (Penentuan fungsi fitness akan dibahas lebih lanjut di bagian akhir)

- Sehingga hasil fitness:

Individu	$f(x)$	Fitness
1	75.91	0.013
2	2.59	0.278
3	21.89	0.043
4	37.59	0.026

Tahap 2: Seleksi

- Seleksi dilakukan menggunakan **Roulette Wheel Selection**, di mana probabilitas seleksi dihitung sebagai:
  - $P_i = \frac{\text{fitness}_i}{\sum \text{fitness}}$
  - Total fitness:
    $0.013 + 0.278 + 0.043 + 0.026 = 0.36$

- Probabilitas seleksi:

Individu	Fitness	Probabilitas Seleksi
1	0.013	3.6%
2	0.278	77.2%
3	0.043	11.9%
4	0.026	7.2%

- Individu **2** memiliki peluang seleksi terbesar, sehingga kemungkinan besar akan terpilih lebih sering.

Tahap 3: Crossover

- Crossover dalam representasi real biasanya menggunakan metode **aritmetika crossover**, misalnya:
  - $\text{offspring} = \alpha \times \text{parent}_1 + (1 - \alpha) \times \text{parent}_2$
    -dengan α adalah faktor acak dalam interval $[0, 1]$.
- Misalkan kita memilih **Individu 2 (-1.61)** dan **Individu 3 (4.68)** untuk crossover, dengan $\alpha = 0.7$:
  - $\text{offspring}_1 = 0.7(-1.61) + 0.3(4.68) = -0.28$
  - $\text{offspring}_2 = 0.3(-1.61) + 0.7(4.68) = 3.35$

- Hasil crossover:

Offspring	Nilai x
1	-0.28
2	3.35

Tahap 4: Mutasi

- Mutasi dalam representasi real bisa dilakukan dengan menambahkan nilai acak dalam rentang tertentu. Misalnya, kita menggunakan **mutasi Gaussian**:
  - $x' = x + \text{mutasi} \times \sigma$
- Misalkan untuk **offspring 1 (-0.28)**, kita menambahkan mutasi acak **+0.5**:
  - $x' = -0.28 + 0.5 = 0.22$
- Dan untuk **offspring 2 (3.35)**, kita menambahkan mutasi acak **-0.7**:
  - $x' = 3.35 - 0.7 = 2.65$

- Hasil mutasi:

Offspring	Sebelum Mutasi	Setelah Mutasi
1	-0.28	0.22
2	3.35	2.65

Tahap 5: Evaluasi Ulang

- Kita menghitung kembali **fitness** dari individu baru:

Individu	Nilai x	$f(x) = x^2$	Fitness
1	0.22	0.0484	0.954
2	2.65	7.02	0.124

- Individu **1 (x = 0.22)** sekarang memiliki nilai fitness tertinggi, menunjukkan bahwa GA telah menemukan solusi lebih optimal.

Tahap 6: Penggantian (Populasi Generasi Berikutnya)

- Individu **terbaik dari generasi sebelumnya** tetap ada ($x = -1.61$).
- **Dua individu baru** hasil crossover dan mutasi ($x = 0.22, x = 2.65$).
- **Tambahkan satu individu baru secara acak** untuk menjaga keberagaman ($x = -5.7$).

- Populasi di generasi berikutnya:

Individu	x	$f(x) = x^2$
1	-1.61	2.59
2	0.22	0.0484
3	2.65	7.02
4	-5.7	32.49

Tahap 7: Pengulangan dan Konvergensi

- Proses di atas (seleksi, crossover, mutasi, evaluasi) terus diulang selama beberapa generasi.
- Setelah beberapa iterasi, nilai x akan mendekati **0**, yang merupakan nilai optimal karena x^2 minimum pada $x = 0$.
- Misalnya, setelah **10 generasi**, kita mendapatkan solusi optimal:
  - $x = 0.001, \quad f(x) = (0.001)^2 = 0.000001$

Kesimpulan

1. **Representasi real lebih efisien** karena tidak perlu konversi biner-desimal.
2. **Crossover real** menggunakan metode aritmetika menghasilkan nilai yang lebih halus.
3. **Mutasi real** mempertahankan keragaman tanpa mengubah skala pencarian secara drastis.
4. **GA berhasil menemukan minimum** dari $f(x) = x^2$ dalam rentang $[-10, 10]$, yaitu $x \approx 0$.

Mengapa Nilai Fitness Dihitung dengan Rumus $\text{fitness} = \frac{1}{f(x)+1}$?

- Dalam algoritma genetik (GA), **fitness** adalah ukuran kualitas suatu individu dalam menyelesaikan masalah yang diberikan. Untuk masalah **minimisasi**, kita sering menggunakan rumus:

  $$\text{fitness} = \frac{1}{f(x)+1}$$

- **Alasan utama penggunaan rumus ini adalah:**

1. Mengubah Masalah Minimisasi menjadi Maksimisasi

- **GA secara alami bekerja lebih baik dengan masalah maksimisasi**, karena seleksi biasanya memilih individu dengan fitness yang lebih tinggi.
- Fungsi fitness standar biasanya mengasumsikan semakin besar nilainya, semakin baik individu tersebut.
- Namun, dalam masalah minimisasi, semakin kecil $f(x)$, semakin baik. Oleh karena itu, kita perlu **membalikkan** logika ini.
- Dengan rumus $\frac{1}{f(x)+1}$, individu dengan nilai $f(x)$ lebih kecil akan memiliki **fitness lebih tinggi**.

Contoh

- Misalkan kita ingin meminimalkan: $f(x) = x^2$

x	$f(x) = x^2$	**Fitness** $\frac{1}{f(x)+1}$
0	0	1.000
2	4	0.200
5	25	0.038
10	100	0.0099

- 👉 **Semakin kecil** $f(x)$**, semakin besar fitnessnya**, sehingga seleksi akan lebih sering memilih nilai yang lebih kecil.

2. Mencegah Fitness Menjadi Negatif

- Beberapa fungsi minimisasi dapat menghasilkan **nilai negatif**, misalnya:

  $f(x) = x^3 - 2x^2$

  Jika x negatif, maka $f(x)$ bisa negatif juga.

- Jika kita hanya membalikkan fungsi fitness menjadi $1/f(x)$, maka kita bisa mendapatkan **nilai fitness negatif atau tak terdefinisi (pembagian oleh nol)**.

- Dengan menambahkan **+1** di penyebut ($f(x) + 1$), kita memastikan bahwa fitness tetap positif dan terdefinisi.

3. Mencegah Fitness Mendekati Tak Hingga

- Jika kita hanya menggunakan $\frac{1}{f(x)}$, ada risiko **terjadi tak hingga** jika $f(x)$ mendekati **nol**.
- Dengan $\frac{1}{f(x)+1}$, kita memastikan bahwa fitness tetap terbatas dan tidak mencapai **tak hingga**.

- **Contoh Perbandingan**

x	$f(x)$	$\frac{1}{f(x)}$	$\frac{1}{f(x)+1}$
0	0	**∞ (tak hingga)**	1.000
1	1	1.000	0.500
2	4	0.250	0.200
5	25	0.040	0.038

- 👉 **Metode tanpa "+1" bisa membuat fitness tidak stabil ketika $f(x)$ mendekati nol.**

Mengapa Kita Harus Menghindari Nilai Fitness Tak Hingga atau Negatif?

- Dalam algoritma genetik (GA), **fitness** adalah ukuran seberapa baik suatu individu dalam populasi menyelesaikan masalah yang diberikan.
- Nilai fitness digunakan dalam **seleksi**, yang menentukan individu mana yang akan bertahan dan bereproduksi untuk generasi berikutnya.
- Oleh karena itu, nilai fitness yang **tak hingga** atau **negatif** bisa menyebabkan berbagai masalah dalam algoritma. Berikut adalah alasan mengapa kita harus menghindarinya:

1. Menghindari Kesalahan Komputasi (Runtime Error)

- Jika kita menggunakan rumus fitness seperti:
  - $\text{fitness} = \frac{1}{f(x)}$
- dan mendapatkan $f(x) = 0$, maka:
  - $\text{fitness} = \frac{1}{0}$
- ini menyebabkan **pembagian oleh nol**, yang menghasilkan error dalam perhitungan komputasi.
- Jika $f(x) < -0$, maka: $\frac{1}{f(x)}$ akan menghasilkan **nilai negatif**, yang dapat menyebabkan error atau hasil yang tidak diinginkan dalam tahap seleksi.

2. Seleksi dalam GA Mengasumsikan Fitness yang Lebih Tinggi Lebih Baik

- Sebagian besar metode seleksi dalam GA **menggunakan nilai fitness sebagai dasar pemilihan**, seperti:
  - **Roulette Wheel Selection**
  - **Tournament Selection**
  - **Rank Selection**

- Pada **Roulette Wheel Selection**, probabilitas seleksi dihitung berdasarkan:
  - $P_i = \frac{\text{fitness}_i}{\sum \text{fitness}}$
- Jika salah satu individu memiliki fitness negatif, maka:
  - Total fitness ($\sum \text{fitness}$) bisa menjadi **negatif atau nol**, menyebabkan error dalam perhitungan probabilitas.
  - Individu dengan fitness negatif mungkin **tidak bisa dipilih**, atau malah memiliki probabilitas seleksi yang salah.

3. Fitness Tak Hingga Bisa Mendominasi Populasi

- Jika satu individu memiliki **fitness yang sangat besar (tak hingga)**, maka individu tersebut akan memiliki **probabilitas seleksi yang mendominasi**. Akibatnya:
  - Semua individu lain mungkin tidak pernah dipilih untuk reproduksi.
  - Algoritma **akan cepat terkunci ke satu solusi**, menyebabkan **konvergensi prematur** (terjebak dalam solusi lokal yang tidak optimal).

4. Menghindari Perhitungan yang Tidak Stabil

Jika nilai fitness sangat kecil atau sangat besar, operasi matematis bisa menjadi **tidak stabil**. Misalnya:

- Fitness yang **sangat besar** bisa menyebabkan **overflow** dalam komputasi floating-point.
- Fitness yang **sangat kecil** bisa menyebabkan **underflow**, membuat perhitungan tidak akurat.

5. Mencegah Efek Negatif dalam Mutasi dan Crossover

Jika individu dengan fitness negatif atau tak hingga dipilih untuk **crossover dan mutasi**, hasilnya mungkin tidak akan memiliki makna yang jelas. Misalnya:

- Jika individu dengan fitness negatif lebih sering dipilih, hasil crossover bisa memiliki **nilai yang tidak sesuai dengan ruang pencarian**.
- Jika individu dengan fitness sangat besar selalu dipilih, **keberagaman populasi akan menurun**, menyebabkan konvergensi ke solusi yang tidak optimal.

Problem pada fungsi fitness sebelumnya

$$\text{fitness} = \frac{1}{f(x)+1}$$

Ada potensi **nilai fitness menjadi negatif atau tak hingga** jika $f(x)$ **sama dengan -1 atau kurang dari -1**. Ini adalah **masalah yang harus diperhatikan**.

Bagaimana Menghindari Fitness Negatif atau Tak Hingga?

Agar fitness selalu **terdefinisi positif**, kita perlu memastikan bahwa:

1. $f(x) + 1 \neq 0$ (tidak boleh ada pembagian oleh nol).
2. $f(x) + 1 > 0$ (agar fitness tidak negatif).

Solusi 1: Penyesuaian dengan Offset Positif

- Alih-alih menggunakan $f(x) + 1$, kita bisa menambahkan **nilai offset positif** C yang cukup besar sehingga nilai dalam penyebut tidak pernah nol atau negatif:
  - $\text{fitness} = \frac{1}{f(x) - f_{\min} + 1}$
  - di mana:
    - $f_{\min}$ adalah nilai minimum $f(x)$ dalam domain pencarian.
    - Dengan mengurangkan $f_{\min}$, kita memastikan bahwa nilai terkecil dari $f(x) - f_{\min}$ adalah nol atau lebih besar.

Contoh Perhitungan

- Misalkan kita ingin meminimalkan $f(x) = x^3 - 2x^2$, dan dalam domain pencarian $x \in [-2, 2]$, kita mengetahui bahwa nilai minimum dari $f(x)$ adalah **-4**.
- Kita atur:
  - $\text{fitness} = \frac{1}{f(x)+4+1} = \frac{1}{f(x)+5}$
- Sehingga fitness selalu positif dan tidak ada pembagian oleh nol.

Perbandingan Fitness Sebelum dan Sesudah Offset

x	$f(x) = x^3 - 2x^2$	Tanpa Offset $\frac{1}{f(x)+1}$	Dengan Offset $\frac{1}{f(x)+5}$
-2	-4	**Tak hingga**	**0.2**
-1	-3	**-1** (negatif)	**0.25**
0	0	1	**0.2**
1	-1	**Tak hingga**	**0.33**
2	0	1	**0.2**

Hasilnya:

- Tanpa offset, fitness bisa menjadi **tak hingga atau negatif**.
- Dengan offset, fitness selalu berada dalam rentang yang **terdefinisi positif**.

Solusi 2: Menggunakan Normalisasi Fitness

- Cara lain adalah dengan **menormalisasi fitness** agar selalu berada dalam rentang **0 hingga 1**:
  - $\text{fitness} = \frac{f_{\max} - f(x)}{f_{\max} - f_{\min}}$
  - di mana:
    - $f_{\max}$ adalah nilai maksimum dari $f(x)$ dalam domain pencarian.
    - $f_{\min}$ adalah nilai minimum dari $f(x)$.
- Ini memastikan fitness **selalu positif dan berada dalam rentang 0 hingga 1**.

Contoh Perhitungan

- Jika $f(x)$ dalam rentang ([-4, 8]), maka:
  - $\text{fitness} = \frac{8 - f(x)}{8 - (-4)}$
- Sehingga, ketika $f(x) = 8$, $\text{fitness} = 0$, dan ketika $f(x) = -4$, $\text{fitness} = 1$.