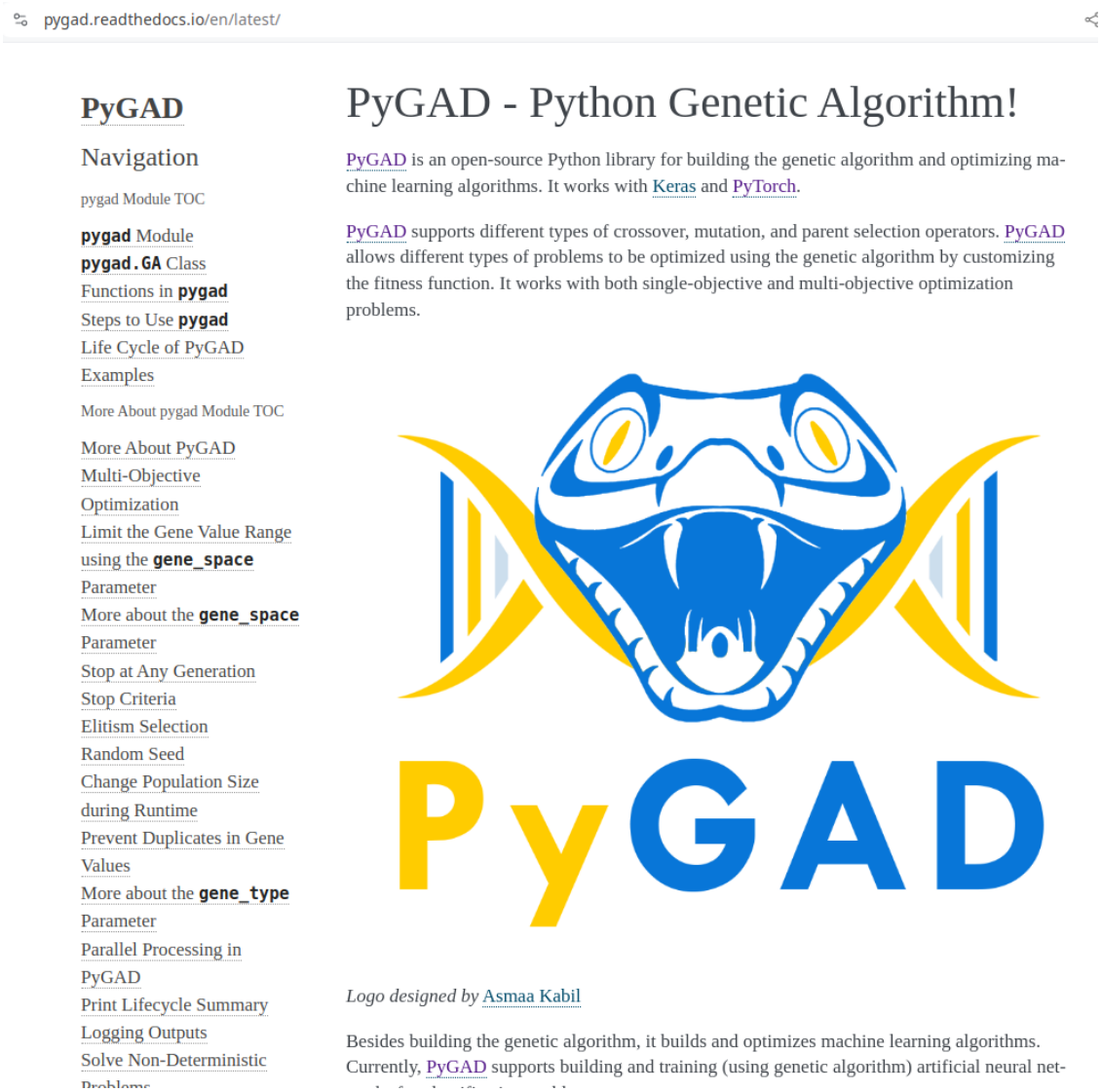
05.applications_ga

March 8, 2024

1 Beberapa Contoh Penerapan Algoritma Genetika

1. Menggunakan bahasa pemrograman python
2. Menggunakan library **PyGAD**

PyGAD: Python library untuk algoritma genetika `https://pygad.readthedocs.io/en/latest/`



- Cara install library **PyGAD**:

```
pip install pygad
```

1.1 Bagaimana cara menggunakan PyGAD?

1. Persiapkan parameter `fitness_func`
2. Persiapkan parameter lainnya
3. Membuat *instance* dari `pygad.GA` *class*
4. Jalankan GA
5. Plot hasilnya
6. Tampilkan solusi terbaik

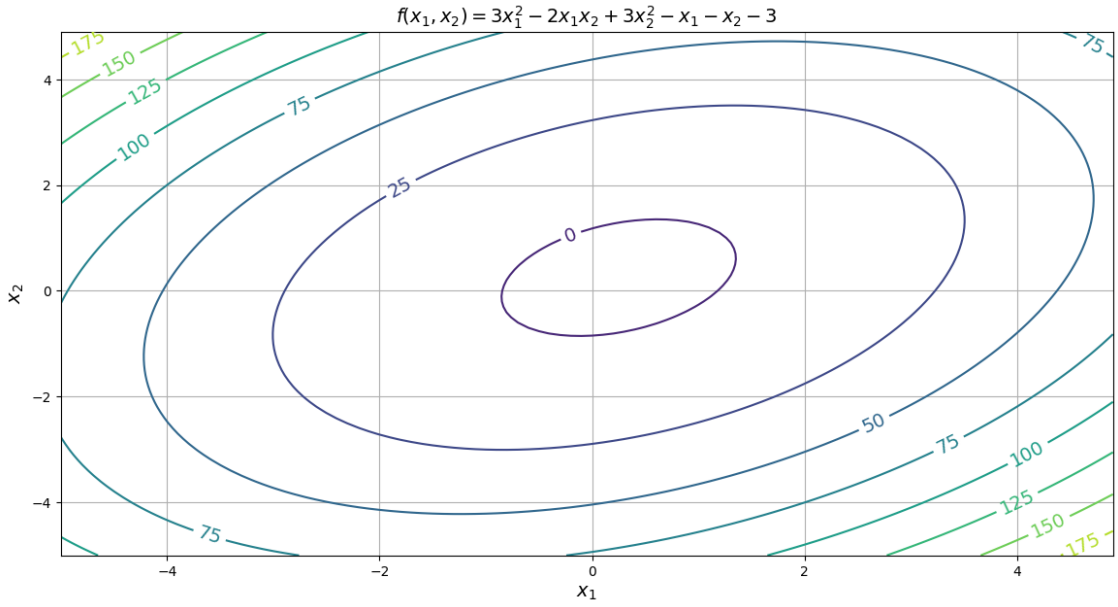
1.2 Contoh 1 - Fungsi Minimum

1. Contoh penggunaan PyGAD dalam optimalisasi.
2. Mengoptimalkan persamaan

$$f(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 3x_2^2 - x_1 - x_2 - 3$$

3. Berapa nilai x_1 dan x_2 sehingga y atau $f(x_1, x_2)$ paling minimum?
4. Kita coba tampilkan `loss surface`-nya

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)

def pers1(X,Y):
    return 3*X**2 - 2*X*Y + 3*Y**2 - X - Y - 3

x = np.vstack((np.array([np.arange(-5,5,0.1)]),
               np.array([np.arange(-5,5,0.1)])))

X,Y = np.meshgrid(x[0], x[1])
Z = pers1(X,Y)
fig, ax = plt.subplots(figsize=(14, 7))
CS = ax.contour(X, Y, Z)
ax.clabel(CS, inline=True, fontsize=14)
ax.set_title('$ f(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 3x_2^2 - x_1 - x_2 - 3$',␣
 ↪fontsize=14)
ax.set_xlabel('$x_1$', fontsize=14)
ax.set_ylabel('$x_2$', fontsize=14)
ax.grid()
```



1.2.1 Persiapkan parameter `fitness_func`

- Persiapkan parameter `fitness_func`
- Gunakan persamaan

$$f(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 3x_2^2 - x_1 - x_2 - 3$$

- Berapa nilai x_1 dan x_2 sehingga y atau $f(x_1, x_2)$ paling minimum?
- **PyGad** akan mencari nilai fitness tertinggi.
- Karena permasalahan kita berupa mencari nilai minimum, maka luaran fitness function perlu dikali dengan -1

[2]:
```
def pers1(X,Y):
    return 3*X**2 - 2*X*Y + 3*Y**2 - X - Y - 3

def calculate_fitness(ga_instance, solution, solution_idx):
    output = pers1(solution[0],solution[1])
    fitness = -output # perlu dikalikan dengan -1

    return fitness
```

1.2.2 Persiapkan parameter lainnya

- `num_genes` 2 karena variabel yang dicari ada 2 yaitu x_1 dan x_2
- Parameter lainnya dapat dilihat di website **PyGad**: https://pygad.readthedocs.io/en/latest/pygad.html#pygad-ga-class

[3]:
```
num_genes = 2
num_generations = 20
```

```
num_parents_mating = 3
sol_per_pop = 10
parent_selection_type="sss"
keep_parents=1
crossover_type="uniform"
mutation_type="random"
mutation_probability = .5

def on_generation(ga):
    print("\nGenerasi ke:", ga.generations_completed)
    print(ga.population)
```

1.2.3 Membuat *instance* dari `pygad.GA` *class*

- Argumen lainnya dapat dilihat di website **PyGad**: https://pygad.readthedocs.io/en/latest/pygad.html#pygad-ga-class

[4]:
```
import pygad

ga_instance1 = pygad.GA(num_generations = num_generations,
                        num_parents_mating = num_parents_mating,
                        fitness_func = calculate_fitness,
                        sol_per_pop = sol_per_pop,
                        num_genes = num_genes,
                        parent_selection_type = parent_selection_type,
                        keep_parents = keep_parents,
                        crossover_type = crossover_type,
                        mutation_type = mutation_type,
                        mutation_probability = mutation_probability,
                        on_generation = on_generation
                        )
```

```
/home/fafa/miniconda3/envs/py311/lib/python3.11/site-
packages/pygad/pygad.py:1139: UserWarning: The 'delay_after_gen' parameter is
deprecated starting from PyGAD 3.3.0. To delay or pause the evolution after each
generation, assign a callback function/method to the 'on_generation' parameter
to adds some time delay.
  warnings.warn("The 'delay_after_gen' parameter is deprecated starting from
PyGAD 3.3.0. To delay or pause the evolution after each generation, assign a
callback function/method to the 'on_generation' parameter to adds some time
delay.")
```

1.2.4 Jalankan GA

- Jalankan GA dengan menggunakan method **run** dari instance yang sudah dibuat ga_instance1

[5]:
```
ga_instance1.run()
```

```
Generasi ke: 1
[[ 0.80892009  1.66458062]
 [ 0.14996834  1.66458062]
 [-0.63479799  0.19805145]
 [ 0.80892009  0.85992485]
 [ 1.85595153  0.90865709]
 [-0.74742125 -0.28438858]
 [-1.56606206  1.70471666]
 [ 0.80892009  1.72843713]
 [ 1.85595153  0.19805145]
 [ 0.80892009  0.19805145]]

Generasi ke: 2
[[ 0.80892009  0.19805145]
 [ 0.09076854  0.19805145]
 [-0.63479799  1.17182533]
 [-0.63479799 -0.79090431]
 [ 0.80892009  0.85992485]
 [-0.63479799  0.85992485]
 [ 0.04065821  0.9242583 ]
 [ 0.80892009 -0.67483185]
 [ 1.26813245  0.4731664 ]
 [ 0.80892009 -0.56276006]]

Generasi ke: 3
[[ 0.09076854  0.19805145]
 [ 0.43763206  0.19805145]
 [ 0.80892009  0.19805145]
 [ 0.60187082  0.31752118]
 [-0.58678888  1.05744676]
 [ 0.80892009  0.19805145]
 [ 0.80892009  0.85992485]
 [ 0.16945303  0.19805145]
 [ 0.80892009  0.85992485]
 [ 0.26479042  0.71414043]]

Generasi ke: 4
[[ 0.16945303  0.19805145]
 [ 0.49680646  0.19805145]
 [ 1.01566313  0.19805145]
 [ 0.03938868 -0.23226756]
 [ 0.09612659  0.19805145]
 [ 0.90730032 -0.32282477]
 [ 1.40893296 -0.317838  ]
 [ 0.16945303  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.43763206  0.19805145]]
```

```
Generasi ke: 5
[[ 0.17319832  0.19805145]
 [ 0.17319832  0.48839703]
 [-0.05707628  0.19805145]
 [ 0.16945303 -0.11981584]
 [ 0.92413173  0.19805145]
 [ 0.80389743  0.19805145]
 [ 0.16945303  0.19805145]
 [ 0.96388454  0.99888757]
 [ 0.16945303 -0.1035294 ]
 [ 0.16945303  0.19805145]]

Generasi ke: 6
[[ 0.17319832  0.19805145]
 [ 0.16945303 -0.48033244]
 [ 0.16945303  0.19805145]
 [ 0.17319832  0.6224099 ]
 [ 0.66618113  0.49731725]
 [ 0.16945303  0.19805145]
 [ 0.17319832 -0.06651694]
 [ 1.11547414 -0.0157531 ]
 [ 0.16945303  0.19805145]
 [ 0.17319832  0.19805145]]

Generasi ke: 7
[[ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 1.02609015  0.05441975]
 [ 0.16945303  0.19805145]
 [ 0.17319832 -0.03175309]
 [ 0.17319832 -0.46296305]
 [ 0.16945303  0.19805145]
 [ 0.17319832  0.19805145]
 [ 1.15330602  0.19805145]
 [ 0.92794447  0.19805145]]

Generasi ke: 8
[[ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.17319832 -0.05078264]
 [-0.75491713  0.19805145]
 [-0.25371917  0.19805145]
 [ 0.17319832 -0.72725217]
 [ 0.17319832 -0.54782752]
 [ 0.17319832  0.19805145]]
```

```
Generasi ke: 9
[[ 0.17319832  0.19805145]
 [ 0.17319832  0.78842384]
 [-0.66988892 -0.75124706]
 [ 0.17319832  0.19805145]
 [ 0.17319832 -0.45535991]
 [ 0.27165165  0.6272433 ]
 [ 0.17319832  1.10778201]
 [ 0.17319832  0.19805145]
 [ 0.17319832 -0.30648657]
 [-0.7980147   0.19805145]]

Generasi ke: 10
[[ 0.17319832  0.19805145]
 [-0.02979221  0.42975165]
 [ 0.17319832 -0.05272332]
 [ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.45803688 -0.74892593]
 [ 0.17319832  0.19805145]
 [ 0.17319832  0.48462789]
 [ 1.05612794  0.19805145]
 [ 0.98389961  0.19805145]]

Generasi ke: 11
[[ 0.17319832  0.19805145]
 [-0.26309213  0.19805145]
 [ 0.43108402  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.17319832 -0.44706946]
 [ 0.17319832  0.19805145]
 [ 0.17319832 -0.05791238]
 [ 0.17319832  1.0595661 ]
 [ 0.17319832  0.69979359]
 [ 0.17319832  1.00315727]]

Generasi ke: 12
[[ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [-0.16947259  0.54308837]
 [ 0.17319832  0.19805145]
 [ 0.17319832  0.18689206]
 [ 0.05625932  0.19805145]
 [ 0.17319832 -0.56781451]
 [ 0.40963445  0.19805145]
 [ 0.57513658 -0.65642253]]
```

```
Generasi ke: 13
[[ 0.17319832  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.72749215  0.31485995]
 [-0.60440671  0.19805145]
 [ 0.11051961 -0.689342  ]
 [ 0.47161893  0.69014121]
 [ 0.17319832  0.19805145]
 [ 0.91039658 -0.35475687]
 [ 0.17319832  1.13780911]
 [ 0.22860054  0.19805145]]

Generasi ke: 14
[[ 0.22860054  0.19805145]
 [ 0.22860054  0.19805145]
 [ 1.07402129  0.97857902]
 [-0.27203931  0.19805145]
 [-0.12009722  0.36536368]
 [ 1.14561981  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.54266067  0.19805145]
 [ 0.81827281  0.19805145]
 [ 0.17319832  0.19805145]]

Generasi ke: 15
[[ 0.22860054  0.19805145]
 [-0.00961775  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.22860054  0.02769046]
 [ 0.95804529  0.82385347]
 [ 0.22860054  0.19805145]
 [ 0.17319832  0.19805145]
 [ 0.22860054  0.19805145]
 [-0.51308063  1.10615351]
 [ 0.22860054  0.54145282]]

Generasi ke: 16
[[ 0.22860054  0.19805145]
 [ 0.22860054  0.19805145]
 [-0.71383411  0.70832596]
 [ 0.22860054  0.19805145]
 [-0.74231013 -0.10077343]
 [ 0.22860054  0.0730013 ]
 [ 0.22860054  0.22603043]
 [ 0.22860054  0.44222485]
 [ 0.22860054  0.19805145]
 [ 0.21283313  0.19805145]]
```

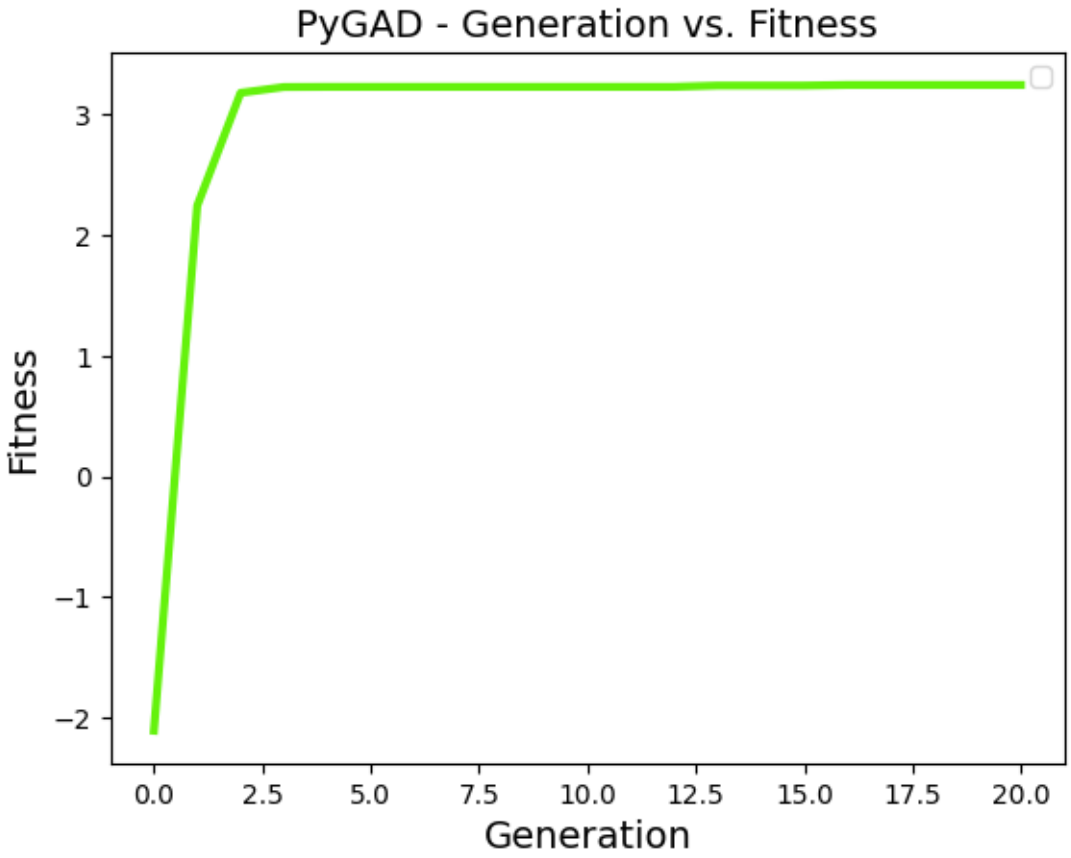
```
Generasi ke: 17
[[ 0.22860054  0.22603043]
 [-0.49374511  0.47980094]
 [ 1.02217736  0.14597473]
 [ 0.22860054 -0.38939153]
 [-0.21421878 -0.44792758]
 [ 0.15015808 -0.38928111]
 [ 0.6093902   0.19805145]
 [ 0.48440132  0.22603043]
 [ 1.07034534  0.19805145]
 [ 0.8410031   0.69457083]]

Generasi ke: 18
[[ 0.22860054  0.22603043]
 [ 1.01901422  0.22603043]
 [ 0.48440132  0.22603043]
 [ 0.67462473 -0.28902856]
 [-0.47545629 -0.12981124]
 [-0.15108554  0.97910601]
 [ 0.6093902   0.22603043]
 [ 0.48440132 -0.60012899]
 [ 0.48440132  0.19805145]
 [ 0.22860054 -0.54678794]]

Generasi ke: 19
[[ 0.22860054  0.22603043]
 [-0.45003671 -0.40083552]
 [ 1.27793217 -0.64148106]
 [ 0.22860054  1.19078767]
 [ 1.42334219  0.95704468]
 [ 0.48440132 -0.43219439]
 [ 0.48440132  0.22603043]
 [ 0.22860054  0.22603043]
 [-0.14151118  0.22603043]
 [ 0.2438213  -0.31712908]]

Generasi ke: 20
[[ 0.22860054  0.22603043]
 [ 1.208521   -0.12926188]
 [ 0.48440132  0.58903588]
 [ 0.22860054  0.22603043]
 [-0.07366293  1.08508872]
 [ 0.22860054  0.22603043]
 [ 1.10528145  0.22603043]
 [ 0.71084184  0.37497666]
 [ 0.22860054  0.81656506]
 [-0.44286786  0.85517987]]
```

1.2.5 Plot hasilnya

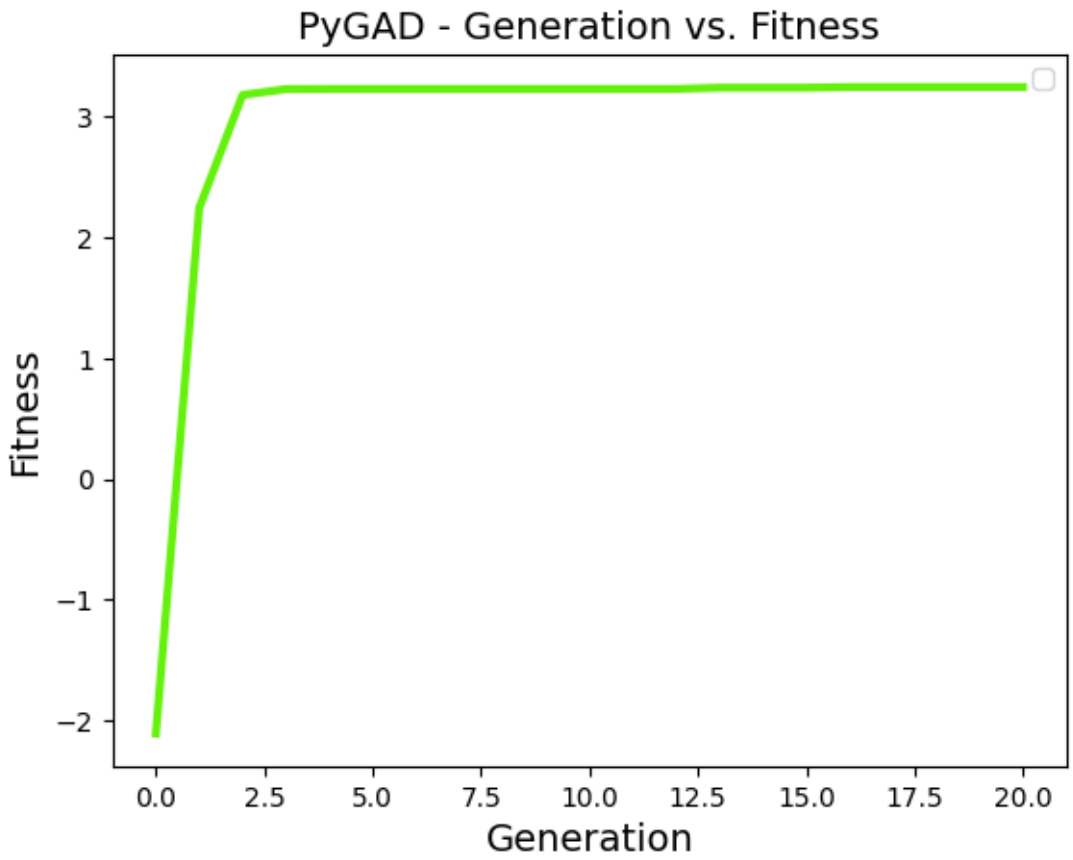
- Menampilkan plot hasil dengan menggunakan method `plot_fitness`

[6]:
```
ga_instance1.plot_fitness()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



[6]:



1.2.6 Tampilkan solusi terbaik

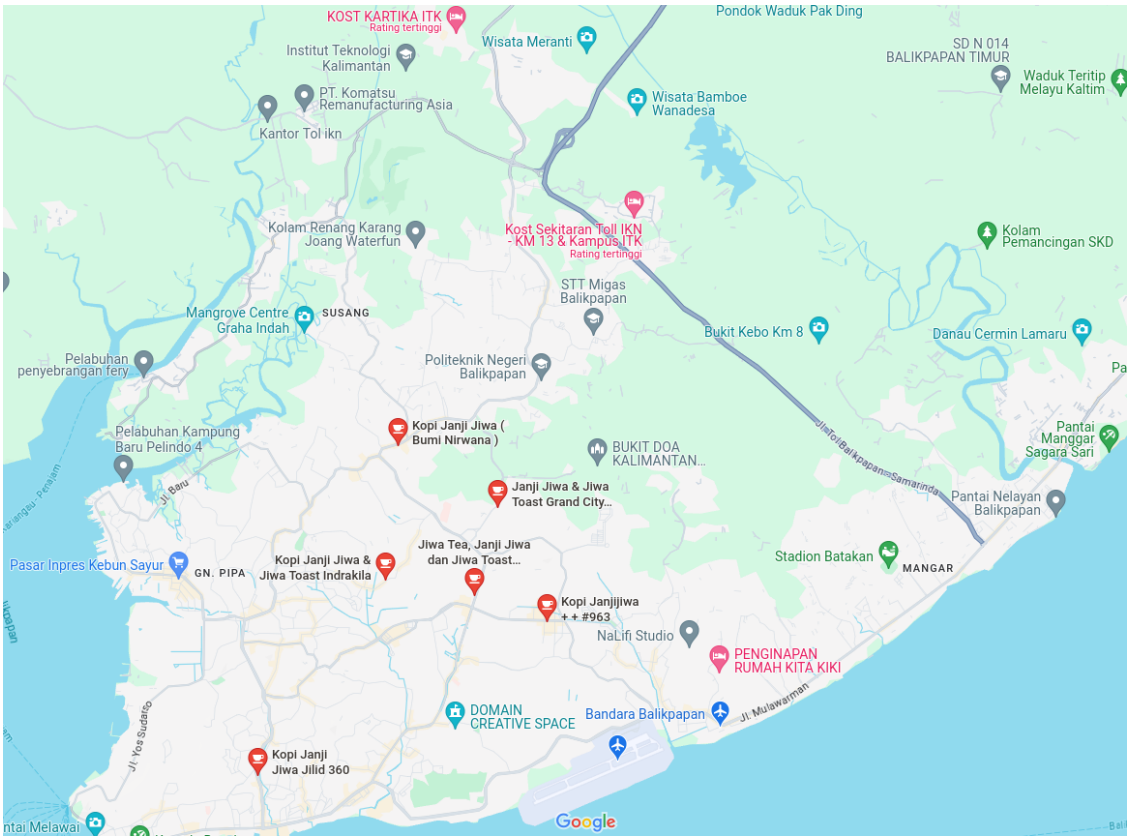
- Menampilkan solusi terbaik dengan menggunakan method `best_solution`

[7]:
```
solution, solution_fitness, solution_idx = ga_instance1.best_solution()
print(f"Parameters dari solusi terbaik [x1, x2] = {solution}")
print(f"Nilai fitness dari solusi terbaik = {-solution_fitness}")
print(f'Solusi terbaik berada di indeks ke-{solution_idx}')
```

```
Parameters dari solusi terbaik [x1, x2] = [0.22860054 0.22603043]
Nilai fitness dari solusi terbaik = -3.247928440301812
Solusi terbaik berada di indeks ke-0
```

1.3 Contoh 2 - Travelling Salesman Problem

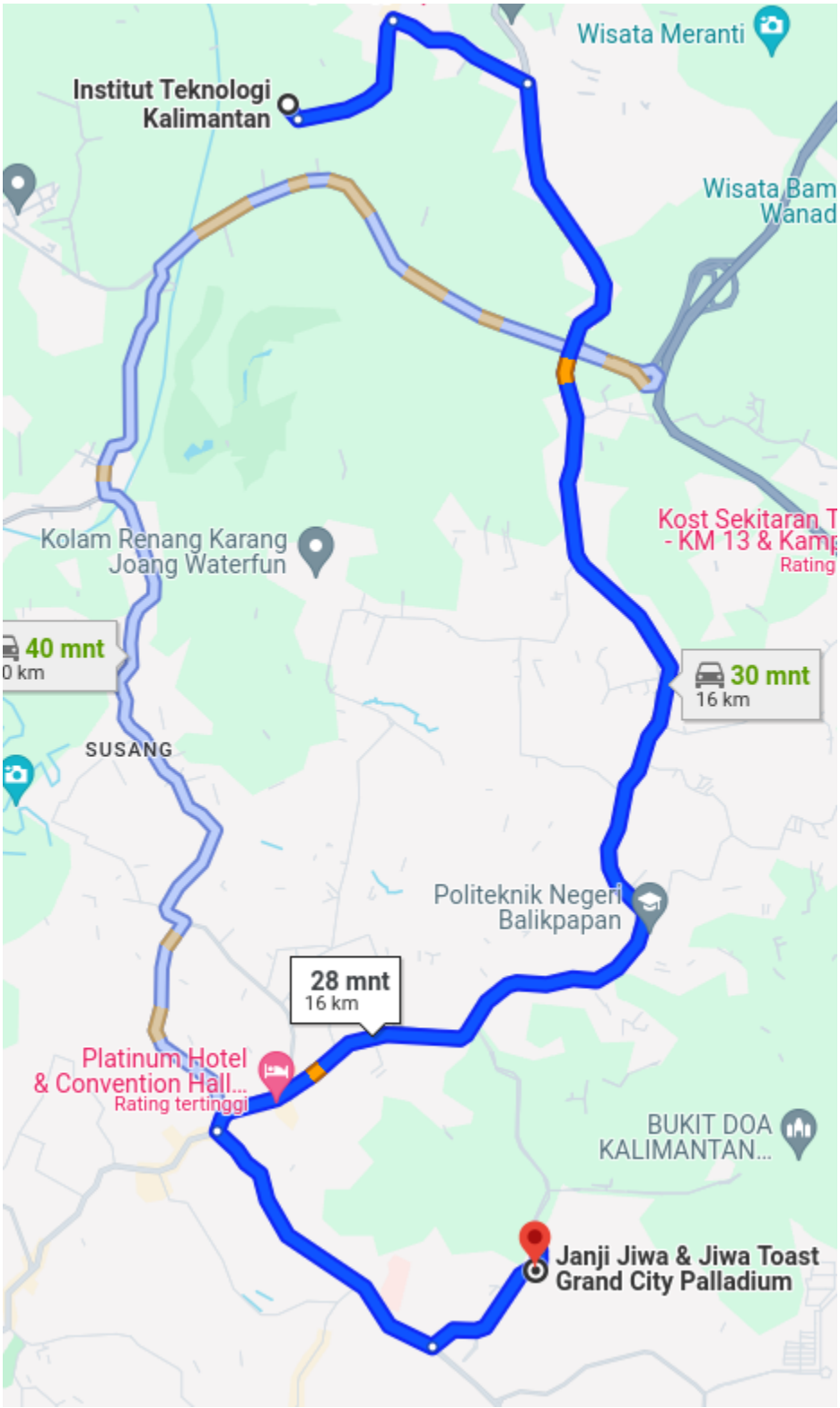
1. Mencari rute terbaik
2. Mengunjungi semua kedai kopi janji jiwa
3. Mulai dari ITK, kembali ke ITK



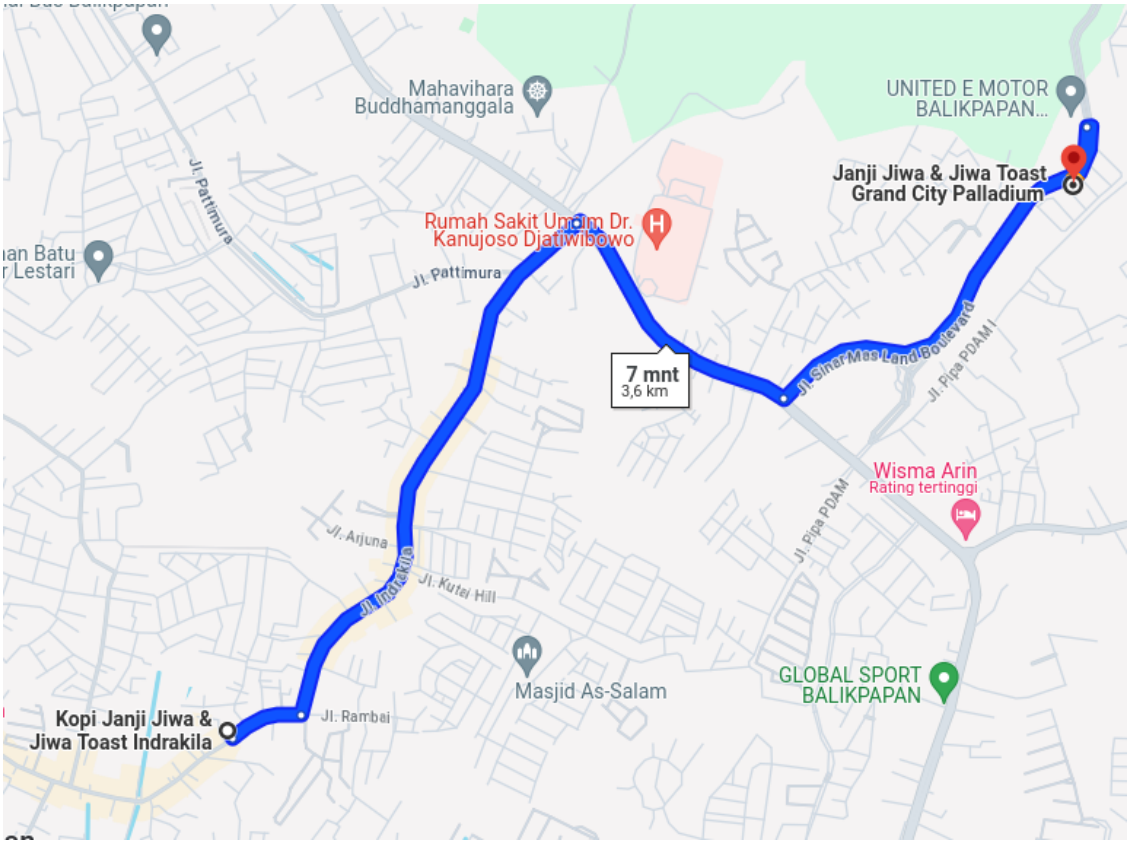
1.3.1 Persiapkan parameter `fitness_func`

- Membuat tabel berisikan semua jarak antar titik
- Jarak dari ITK ke Janji Jiwa Grand City

- Jarak dari ITK ke Janji Jiwa Bumi Nirwana



- Jarak dari Janji Jiwa Grand City ke Janji Jiwa Bumi Nirwana



- Sehingga didapatkan keseluruhan jarak antara semua titik
- Simpan dalam format file **csv**

	A	B	C	D	E	F	G
1	**JARAK (KM)**	**ITK**	**INDRAKILA**	**GRAND CITY**	**MT. HARYONO**	**RUHUI RAHAYU**	**AHMAD YANI**
2	**ITK**	0	12.1	12.4	14.7	16.4	20.5
3	**INDRAKILA**	12.1	0	3.3	3.7	5.4	6.2
4	**GRAND CITY**	12.4	3.3	0	2.6	4.3	9
5	**MT. HARYONO**	14.7	3.7	2.6	0	1.7	6.8
6	**RUHUI RAHAYU**	16.4	5.4	4.3	1.8	0	7.9
7	**AHMAD YANI**	20.5	6.2	9	6.8	7.9	0

- Bagaimana `fitness_func`-nya?
- Optimalkan rute terpendek
- Contoh:
  - ITK → 12.1 km → Indrakila → 3.3 km → Grand City → 2.6 km → MT. Haryono → 1.8 km → Ruhui Rahayu → 7.9 km → Ahmad Yani → 20.5 km → ITK
  - TOTAL = 48.2 KM

[8]:
```
import pandas as pd

data = pd.read_csv('data_janji_jiwa.csv')
data
```

[8]:
```
      JARAK (KM)   ITK  INDRAKILA  GRAND CITY  MT. HARYONO  RUHUI RAHAYU  \
0            ITK   0.0       12.1        12.4         14.7          16.4
1     INDRAKILA   12.1        0.0         3.3          3.7           5.4
2    GRAND CITY   12.4        3.3         0.0          2.6           4.3
3   MT. HARYONO   14.7        3.7         2.6          0.0           1.7
4  RUHUI RAHAYU   16.4        5.4         4.3          1.8           0.0
5    AHMAD YANI   20.5        6.2         9.0          6.8           7.9

   AHMAD YANI
0        20.5
1         6.2
2         9.0
3         6.8
4         7.9
5         0.0
```

- Bagaimana cara mengambil nilai berdasarkan indeks-indeksnya

[9]:
```
data.iloc[0]
```

[9]:
```
JARAK (KM)       ITK
ITK               0.0
INDRAKILA        12.1
GRAND CITY       12.4
MT. HARYONO      14.7
RUHUI RAHAYU     16.4
AHMAD YANI       20.5
Name: 0, dtype: object
```

- Dari ITK ke ITK
- Indeksnya tidak sama

[10]:
```
data.iloc[0,1]
```

[10]:
```
0.0
```

- Sehingga dari ITK ke Indrakila perlu ditambahkan nilai 1

[11]:
```
data.iloc[0,1+1]
```

[11]:
```
12.1
```

[12]:
```
data.iloc[1]
```

[12]:
```
JARAK (KM)      INDRAKILA
ITK                  12.1
INDRAKILA             0.0
GRAND CITY            3.3
MT. HARYONO           3.7
```

```
RUHUI RAHAYU          5.4
AHMAD YANI            6.2
Name: 1, dtype: object
```

- Sehingga *fitness function* adalah:

[13]:
```python
def cal_fitness(ga_instance, solution, solution_idx):
    distance = 0

    for i in range(len(solution)):
        if i == 0:
            distance += data.iloc[0, solution[i]+1]
        elif i == len(solution)-1:
            distance += data.iloc[solution[i-1], solution[i]+1] + data.
 ↪iloc[solution[i], 1]
        else:
            distance += data.iloc[solution[i-1], solution[i]+1]

    return -distance
```

1.3.2 Persiapkan parameter lainnya

- Parameter berbeda dengan Contoh 1
- Jumlah gen `num_genes` adalah 5. Mewakili rute. Contoh:

(Gen 1)	(Gen 2)	(Gen 3)	(Gen 4)	(Gen 5)
Indrakila →	Grand City →	MT. Haryono →	Ruhui Rahayu →	Ahmad Yani

- Parameter jenis gen (`gene_type`) yang digunakan adalah `int`
- Karena nilai pada gen mewakili rute yang dipilih maka nilai pada gen harus bersifat unik
- Sehingga perlu ditambahkan parameter `allow_duplicate_genes = False`

[14]:
```python
num_generations = 20
num_parents_mating = 2
sol_per_pop = 10
num_genes = 5
gene_type = int
gene_space = np.arange(1,len(data))
allow_duplicate_genes = False
mutation_type = 'random'
mutation_percent_genes = 50

def on_generation(ga):
    print("Generation", ga.generations_completed)
    print(ga.population)
```

1.3.3 Membuat *instance* dari `pygad.GA` *class*

[15]:
```
import pygad

ga_instance2 = pygad.GA(num_generations = num_generations,
                        num_parents_mating = num_parents_mating,
                        fitness_func = cal_fitness,
                        sol_per_pop = sol_per_pop,
                        num_genes = num_genes,
                        gene_type = gene_type,
                        gene_space = np.arange(1,len(data)),
                        allow_duplicate_genes = allow_duplicate_genes,
                        on_generation = on_generation,
                        mutation_type = mutation_type,
                        mutation_percent_genes = mutation_percent_genes
                        )
```

```
/home/fafa/miniconda3/envs/py311/lib/python3.11/site-
packages/pygad/pygad.py:1139: UserWarning: The 'delay_after_gen' parameter is
deprecated starting from PyGAD 3.3.0. To delay or pause the evolution after each
generation, assign a callback function/method to the 'on_generation' parameter
to adds some time delay.
  warnings.warn("The 'delay_after_gen' parameter is deprecated starting from
PyGAD 3.3.0. To delay or pause the evolution after each generation, assign a
callback function/method to the 'on_generation' parameter to adds some time
delay.")
```

1.3.4 Jalankan GA

- Jalankan GA dengan menggunakan method `run`

[16]:
```
ga_instance2.run()
```

```
Generation 1
[[1 5 4 3 2]
 [1 3 4 5 2]
 [2 1 5 4 3]
 [1 2 5 4 3]
 [2 1 4 3 5]
 [1 5 2 4 3]
 [3 1 4 2 5]
 [5 1 4 2 3]
 [5 4 2 3 1]
 [1 2 5 4 3]]
Generation 2
[[1 5 4 3 2]
 [3 5 1 4 2]
 [1 2 4 3 5]
 [1 2 4 3 5]
```

```
 [2 4 3 5 1]
 [1 3 2 4 5]
 [3 5 4 2 1]
 [1 3 2 4 5]
 [2 1 4 3 5]
 [4 5 1 2 3]]
Generation 3
[[1 5 4 3 2]
 [1 4 3 5 2]
 [4 2 1 5 3]
 [4 1 2 5 3]
 [5 1 4 3 2]
 [2 4 1 5 3]
 [3 4 2 1 5]
 [5 1 3 2 4]
 [2 5 3 4 1]
 [4 5 1 2 3]]
Generation 4
[[1 5 4 3 2]
 [1 5 4 2 3]
 [1 5 3 2 4]
 [1 5 4 3 2]
 [5 4 3 2 1]
 [1 3 2 4 5]
 [4 5 1 3 2]
 [1 4 5 3 2]
 [2 4 5 3 1]
 [2 5 3 1 4]]
Generation 5
[[1 5 4 3 2]
 [5 1 4 3 2]
 [1 5 4 3 2]
 [1 5 4 3 2]
 [5 1 4 3 2]
 [1 5 4 3 2]
 [2 5 4 3 1]
 [4 1 5 3 2]
 [1 5 3 4 2]
 [3 5 4 1 2]]
Generation 6
[[1 5 4 3 2]
 [1 5 4 3 2]
 [2 5 4 3 1]
 [1 4 5 3 2]
 [4 5 1 3 2]
 [1 5 4 3 2]
 [3 2 4 1 5]
 [2 5 4 1 3]
```

```
 [1 5 2 3 4]
 [1 2 4 3 5]]
Generation 7
[[1 5 4 3 2]
 [1 5 3 4 2]
 [1 5 4 3 2]
 [1 2 4 3 5]
 [2 5 4 1 3]
 [1 5 4 3 2]
 [1 5 4 2 3]
 [1 4 5 3 2]
 [2 5 4 3 1]
 [1 3 4 5 2]]
Generation 8
[[1 5 4 3 2]
 [1 5 3 4 2]
 [4 5 1 3 2]
 [1 4 5 3 2]
 [5 1 4 2 3]
 [1 5 4 3 2]
 [1 5 4 3 2]
 [1 5 3 4 2]
 [2 5 4 3 1]
 [2 1 4 3 5]]
Generation 9
[[1 5 4 3 2]
 [1 5 4 3 2]
 [1 4 5 3 2]
 [3 5 4 1 2]
 [1 5 4 3 2]
 [2 5 3 4 1]
 [1 5 4 3 2]
 [2 5 4 3 1]
 [5 1 4 3 2]
 [2 5 4 3 1]]
Generation 10
[[1 5 4 3 2]
 [1 5 4 2 3]
 [1 5 4 3 2]
 [1 5 4 3 2]
 [5 1 4 3 2]
 [5 4 1 3 2]
 [1 4 5 3 2]
 [1 5 3 4 2]
 [2 4 5 3 1]
 [5 1 2 3 4]]
Generation 11
[[1 5 4 3 2]
```

```
 [5 1 4 3 2]
 [4 1 5 3 2]
 [1 5 4 2 3]
 [4 5 1 3 2]
 [4 5 2 3 1]
 [1 2 4 3 5]
 [1 5 4 3 2]
 [2 3 4 5 1]
 [5 1 2 3 4]]
Generation 12
[[2 3 4 5 1]
 [5 3 4 2 1]
 [2 3 4 5 1]
 [2 5 3 4 1]
 [3 4 5 1 2]
 [2 3 4 1 5]
 [1 2 4 5 3]
 [2 5 4 3 1]
 [1 2 3 5 4]
 [5 2 4 3 1]]
Generation 13
[[2 3 4 5 1]
 [2 3 5 4 1]
 [2 5 4 3 1]
 [2 1 4 5 3]
 [4 3 2 5 1]
 [5 3 1 2 4]
 [2 3 4 5 1]
 [4 3 2 5 1]
 [2 3 4 5 1]
 [3 2 4 5 1]]
Generation 14
[[2 3 4 5 1]
 [2 1 4 5 3]
 [2 3 4 5 1]
 [2 5 1 3 4]
 [2 1 3 5 4]
 [4 3 2 5 1]
 [2 5 4 3 1]
 [5 3 4 2 1]
 [2 4 5 3 1]
 [5 3 4 2 1]]
Generation 15
[[2 3 4 5 1]
 [2 3 4 5 1]
 [5 3 4 2 1]
 [2 3 4 5 1]
 [2 3 4 5 1]
```

```
 [2 3 4 5 1]
 [2 4 3 5 1]
 [2 1 5 4 3]
 [5 3 4 2 1]
 [2 1 5 4 3]]
Generation 16
[[2 3 4 5 1]
 [2 3 5 4 1]
 [2 3 1 5 4]
 [1 3 4 5 2]
 [2 3 4 5 1]
 [5 3 4 2 1]
 [1 3 4 5 2]
 [2 4 3 5 1]
 [2 3 4 5 1]
 [5 1 4 2 3]]
Generation 17
[[2 3 4 5 1]
 [2 3 4 5 1]
 [2 3 4 5 1]
 [4 3 5 2 1]
 [2 3 4 5 1]
 [2 3 4 5 1]
 [3 2 4 5 1]
 [1 3 4 5 2]
 [2 5 4 3 1]
 [2 3 4 5 1]]
Generation 18
[[2 3 4 5 1]
 [2 3 4 5 1]
 [5 2 4 3 1]
 [2 3 4 5 1]
 [2 4 3 5 1]
 [4 3 5 2 1]
 [2 4 5 3 1]
 [2 4 3 5 1]
 [2 1 4 5 3]
 [2 3 4 1 5]]
Generation 19
[[2 3 4 5 1]
 [2 1 4 5 3]
 [1 3 4 5 2]
 [1 3 4 5 2]
 [2 3 4 1 5]
 [5 3 4 1 2]
 [2 1 3 5 4]
 [3 1 4 5 2]
 [1 3 4 2 5]
```

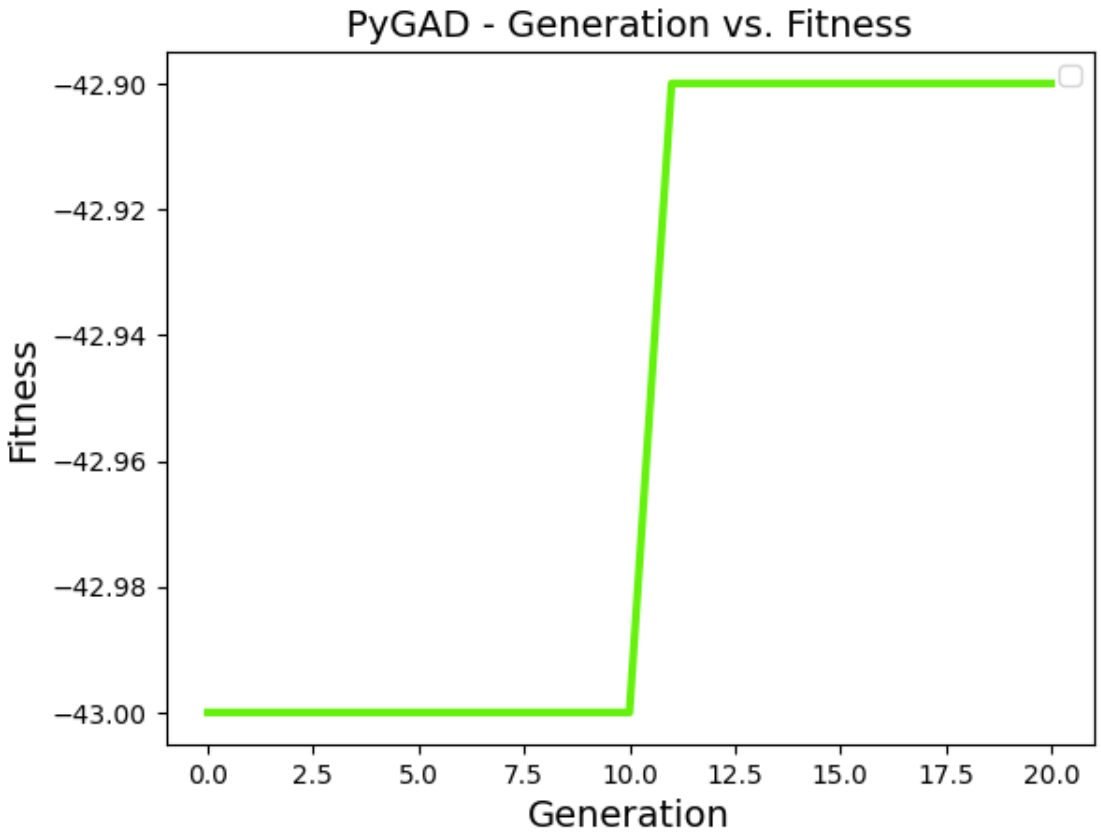
```
 [3 2 4 5 1]]
Generation 20
[[2 3 4 5 1]
 [3 2 4 5 1]
 [3 1 4 5 2]
 [2 3 4 5 1]
 [1 2 4 5 3]
 [1 3 4 5 2]
 [2 4 3 5 1]
 [2 1 4 5 3]
 [2 3 4 5 1]
 [2 3 4 5 1]]
```

1.3.5 Plot hasilnya

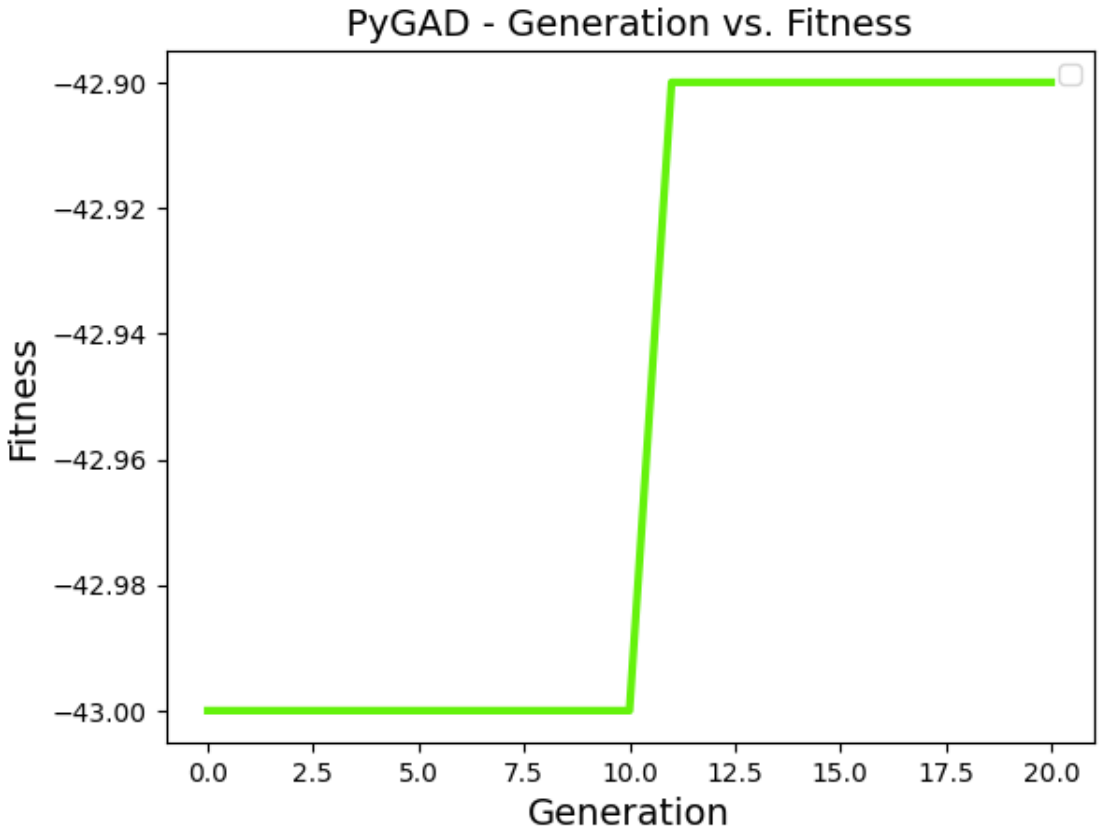
- Tampilkan plot hasilnya dengan menggunakan method `plot_fitness`

[17]:
```
ga_instance2.plot_fitness()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



[17]:



1.3.6 Tampilkan solusi terbaik

- Menampilkan solusi terbaik dengan menggunakan method `best_solution`

[18]:
```
solution, solution_fitness, solution_idx = ga_instance2.best_solution()
print(f"Parameters dari solusi terbaik : {solution}")
print(f"Nilai fitness dari solusi terbaik = {-solution_fitness}")
```

```
Parameters dari solusi terbaik : [2 3 4 5 1]
Nilai fitness dari solusi terbaik = 42.900000000000006
```

[19]:
```
rute_terbaik = data.iloc[0,0]
for i in range(len(solution)):
    rute_terbaik += ' -> ' + data.iloc[solution[i],0]

rute_terbaik += " -> " + data.iloc[0,0]

print("Rute ngopi terbaik adalah:\n'{}'\ndengan total jarak tempuh {:.2f} km".
  ↪format(rute_terbaik,-solution_fitness))
```

```
Rute ngopi terbaik adalah:
```

```
'ITK -> GRAND CITY -> MT. HARYONO -> RUHUI RAHAYU -> AHMAD YANI -> INDRAKILA ->
ITK'
dengan total jarak tempuh 42.90 km
```