notebook2

November 7, 2022

[25]:
```
import pygad
import math
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
```

DEFINE VALORES DEL ENUNCIADO:

[26]:
```
arr = []
arr2 = []

matriz = [
    [5,2,4,8,9,0,3,3,8,7],
    [5,5,3,4,4,6,4,1,9,1],
    [4,1,2,1,3,8,7,8,9,1],
    [1,7,1,6,9,3,1,9,6,9],
    [4,7,4,9,9,8,6,5,4,2],
    [7,5,8,2,5,2,3,9,8,2],
    [1,4,0,6,8,4,0,1,2,1],
    [1,5,2,1,2,8,3,3,6,2],
    [4,5,9,6,3,9,7,6,5,10],
    [0,6,2,8,7,1,2,1,5,3]
]
```

DEFINE FUNCIÓN DE COSTO:

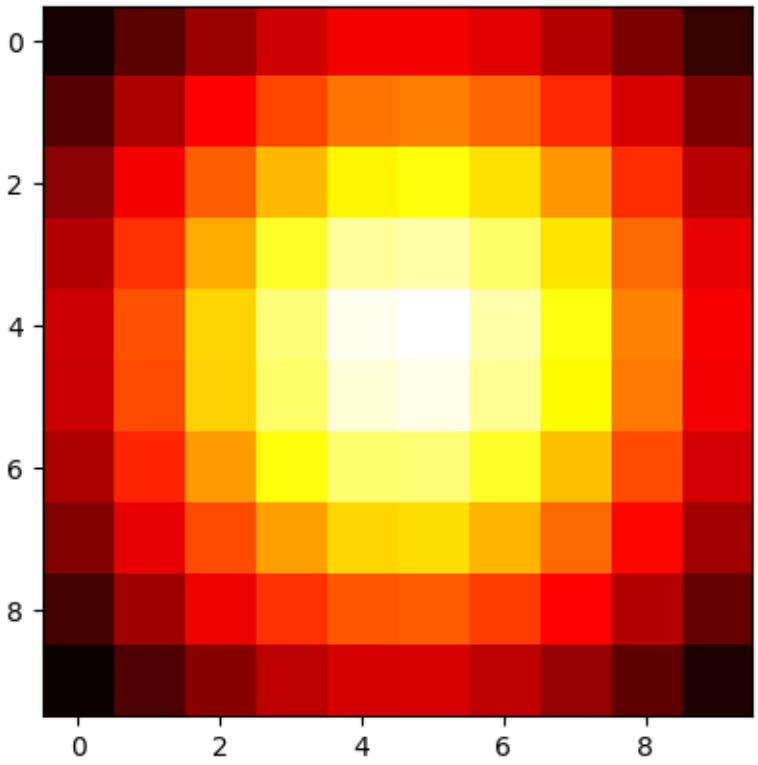
[27]:
```
def fitness_func(solution, solution_idx):
    x = solution[0]
    y = solution[1]
    z=0
    for i in range(0,10):
        for j in range(0,10):
            z+= math.sqrt((i-x)**2  + (j-y)**2)*matriz[i][j]
    return 1/z
```

MAPA DE CALOR SOBRE LOS VALORES DE LA FUNCIÓN DE COSTO:

[28]:
```
matriz2 = [[0 for _ in range(10)] for _ in range(10)]
for i in range(0,10):
```

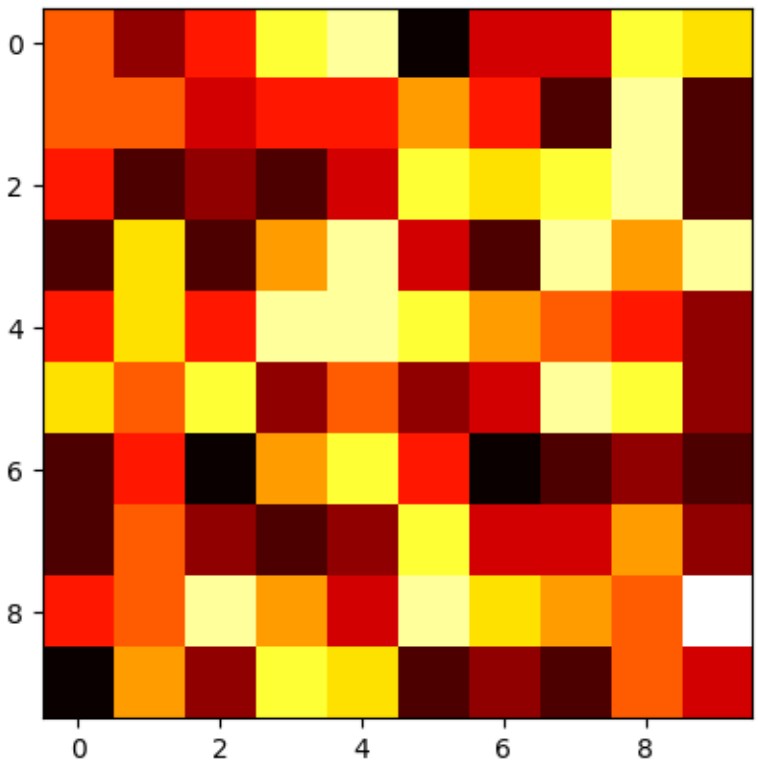
```
    for j in range(0,10):
        matriz2[i][j]=fitness_func([i,j],0)

fig,ax = plt.subplots()
ax.imshow(matriz2, cmap='hot', interpolation='nearest')
plt.show()
```



MAPA DE CALOR SOBRE LA PROBABILIDAD DE INCENDIO:

[29]:
```
fig,ax = plt.subplots()
ax.imshow(matriz, cmap='hot', interpolation='nearest')
plt.show()
```



DEFINE FUNCIÓN PARA LA RESOLUCIÓN DEL ALGORITMO:

[30]:
```
last_fitness = 0
def on_generation(ga_instance):
    global last_fitness
    print(f"-> Generation={ga_instance.generations_completed:3}  ",end=" ")
    print(f"Fitness={ga_instance.best_solution(pop_fitness=ga_instance.
 ↪last_generation_fitness)[1]:7.2}  ",end=" ")
    print(f"Change={ga_instance.best_solution(pop_fitness=ga_instance.
 ↪last_generation_fitness)[1] - last_fitness:7.2} ",end=" ")
    best = ga_instance.best_solution(pop_fitness=ga_instance.
 ↪last_generation_fitness)[0]
    best = [int(best[0]),int(best[1])]
    print(f"Best Solution:{best}",end=" ")
    print(f"Population:",end=" ")
    for x in ga_instance.population:
        print(f'[{int(x[0])}, {int(x[1])}]',end=" ")
    print("")

    global arr
    arr.append(ga_instance.best_solution(pop_fitness=ga_instance.
 ↪last_generation_fitness)[0])
```

```
    global arr2
    arr2.append(ga_instance.best_solution(pop_fitness=ga_instance.
 ↪last_generation_fitness)[1])

    last_fitness = ga_instance.best_solution(pop_fitness=ga_instance.
 ↪last_generation_fitness)[1]
```

EJECUTA EL ALGORITMO GENÉTICO CON LOS PARÁMETROS DE CROSSOVER, MUTATION Y SELECTION:

[31]:
```
ga_instance = pygad.GA(num_generations=20,
                       num_parents_mating=2,
                       sol_per_pop=4,
                       num_genes=2,
                       crossover_type="single_point", # Values: 'single_point',␣
 ↪'two_points', 'uniform', 'scattered'
                       mutation_type="random",        # Values: 'random',␣
 ↪'swap', 'inversion', 'scramble', 'adaptive'
                       mutation_probability=0.9,      # Values: Value between 0.
 ↪0 and 1.0
                       parent_selection_type="sss",   # Values: 'sss', 'rws',␣
 ↪'sus', 'rank', 'random', 'tournament'
                    #    gene_space={"low": 0, "high": 10},
                       gene_space=[0,1,2,3,4,5,6,7,8,9],
                       mutation_by_replacement=True,
                       fitness_func=fitness_func,
                       on_generation=on_generation,
                       save_solutions=True)

ga_instance.run()
```

```
-> Generation=  1   Fitness=0.00055   Change=0.00055  Best Solution:[6, 4]
Population: [6, 4] [5, 0] [8, 9] [2, 7]
-> Generation=  2   Fitness=0.00055   Change=2.7e-06  Best Solution:[6, 5]
Population: [6, 4] [2, 8] [6, 7] [6, 5]
-> Generation=  3   Fitness=0.00055   Change=    0.0  Best Solution:[6, 5]
Population: [6, 5] [4, 9] [5, 8] [2, 4]
-> Generation=  4   Fitness=0.00055   Change=    0.0  Best Solution:[6, 5]
Population: [6, 5] [5, 8] [8, 0] [4, 7]
-> Generation=  5   Fitness=0.00055   Change=    0.0  Best Solution:[6, 5]
Population: [6, 5] [8, 8] [7, 4] [7, 3]
-> Generation=  6   Fitness=0.00055   Change=    0.0  Best Solution:[6, 5]
Population: [6, 5] [7, 4] [9, 6] [8, 8]
-> Generation=  7   Fitness=0.00058   Change=2.8e-05  Best Solution:[5, 4]
Population: [6, 5] [5, 8] [5, 4] [1, 6]
-> Generation=  8   Fitness=0.00059   Change=3.6e-06  Best Solution:[5, 5]
Population: [5, 4] [5, 5] [7, 1] [7, 2]
```

```
-> Generation=  9   Fitness=0.00059   Change=    0.0  Best Solution:[5, 5]
Population: [5, 5] [9, 7] [2, 1] [7, 9]
-> Generation= 10   Fitness=0.00059   Change=    0.0  Best Solution:[5, 5]
Population: [5, 5] [5, 6] [5, 2] [9, 9]
-> Generation= 11   Fitness=0.00059   Change=    0.0  Best Solution:[5, 5]
Population: [5, 5] [4, 6] [8, 8] [5, 6]
-> Generation= 12   Fitness=0.00059   Change=    0.0  Best Solution:[5, 5]
Population: [5, 5] [1, 3] [3, 5] [5, 9]
-> Generation= 13   Fitness=0.00059   Change=    0.0  Best Solution:[5, 5]
Population: [5, 5] [2, 0] [6, 8] [1, 3]
-> Generation= 14   Fitness=0.00059   Change=2.9e-06  Best Solution:[4, 4]
Population: [5, 5] [6, 1] [3, 3] [4, 4]
-> Generation= 15   Fitness=0.00059   Change=    0.0  Best Solution:[4, 4]
Population: [4, 4] [8, 3] [4, 2] [0, 2]
-> Generation= 16   Fitness=0.00059   Change=    0.0  Best Solution:[4, 4]
Population: [4, 4] [3, 2] [9, 0] [9, 8]
-> Generation= 17   Fitness=0.00059   Change=    0.0  Best Solution:[4, 4]
Population: [4, 4] [0, 7] [8, 0] [5, 6]
-> Generation= 18   Fitness=0.00059   Change=    0.0  Best Solution:[4, 4]
Population: [4, 4] [3, 9] [6, 3] [1, 3]
-> Generation= 19   Fitness=0.00059   Change=    0.0  Best Solution:[4, 4]
Population: [4, 4] [5, 8] [0, 8] [3, 5]
-> Generation= 20   Fitness=0.00059   Change=    0.0  Best Solution:[4, 4]
Population: [4, 4] [8, 3] [5, 3] [2, 4]
```

IMPRIME SOLUCIÓN ENCONTRADA:

[32]:
```
solution, solution_fitness, solution_idx = ga_instance.
  ↪best_solution(ga_instance.last_generation_fitness)
print("Solution", solution)
print("Fitness value of the best solution = {solution_fitness}".
  ↪format(solution_fitness=solution_fitness))
```

```
Solution [4. 4.]
Fitness value of the best solution = 0.0005883826017753597
```

IMPRIME LAS SOLUCIONES OBTENIDAS EN CADA GENERACIÓN:

[33]:
```
# arr = ga_instance.solutions
# arr2 = ga_instance.solutions_fitness

# print(arr)
x = [x[0]+0.5 for x in arr]
y = [x[1]+0.5 for x in arr]
print(x)
```

```
[6.5, 6.5, 6.5, 6.5, 6.5, 6.5, 5.5, 5.5, 5.5, 5.5, 5.5, 5.5, 5.5, 4.5, 4.5, 4.5,
4.5, 4.5, 4.5, 4.5]
```

GRÁFICO DE SEQUENCIA DE SOLUCIONES ENCONTRADAS:

[34]:
```
def removeDuplicates(seq):
    seen = set()
    seen_add = seen.add
    return [x for x in seq if not (x in seen or seen_add(x))]

colors= [1-(x/max(arr2)) for x in arr2]
colors2 = [[x]*3 for x in colors]
plt.grid(True)
# plt.scatter(x, y, color=colors2)
z = list(removeDuplicates(zip(x,y)))
x = [x[0] for x in z]
y = [x[1] for x in z]
plt.plot(y, x)
plt.xticks(range(0,10))
plt.yticks(range(0,10))
for  it, (xi, yi) in enumerate(list(zip(x,y))):
    plt.text(x=yi, y=xi,s=it, ha='center', va='center', color='black',)
plt.show()
```

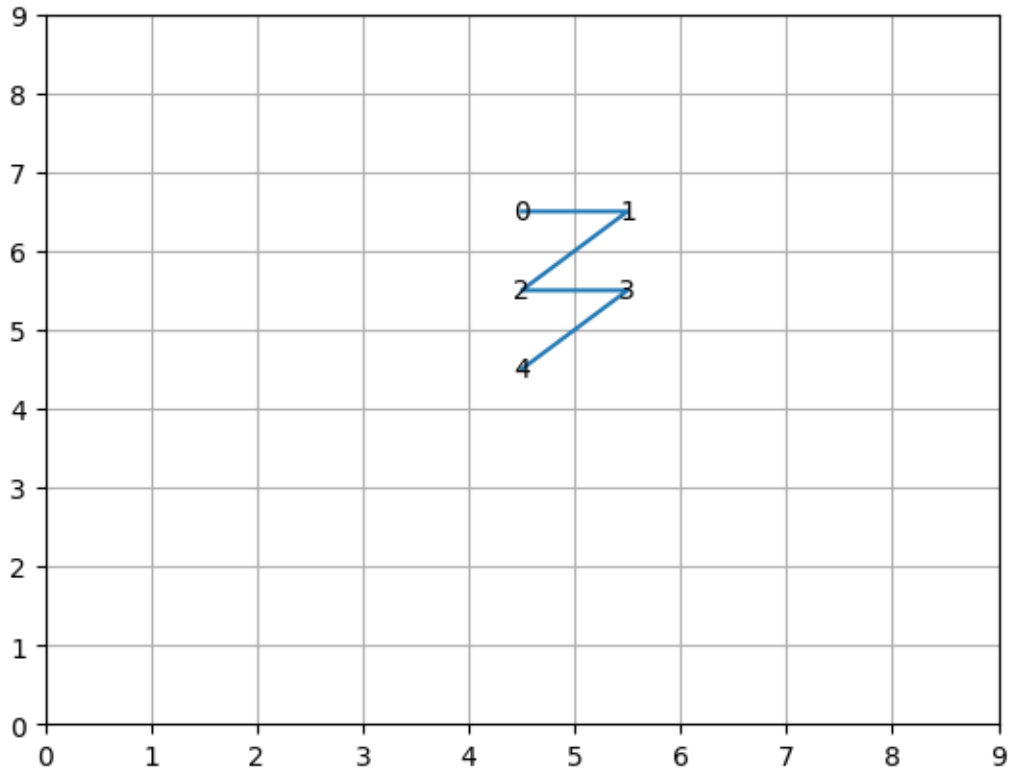


GRÁFICO DE VALORES DE LA FUNCIÓN DE COSTOS A LO LARGO DE LAS GENERACIONES:

[35]:
```
tmp = ga_instance.plot_fitness()
```

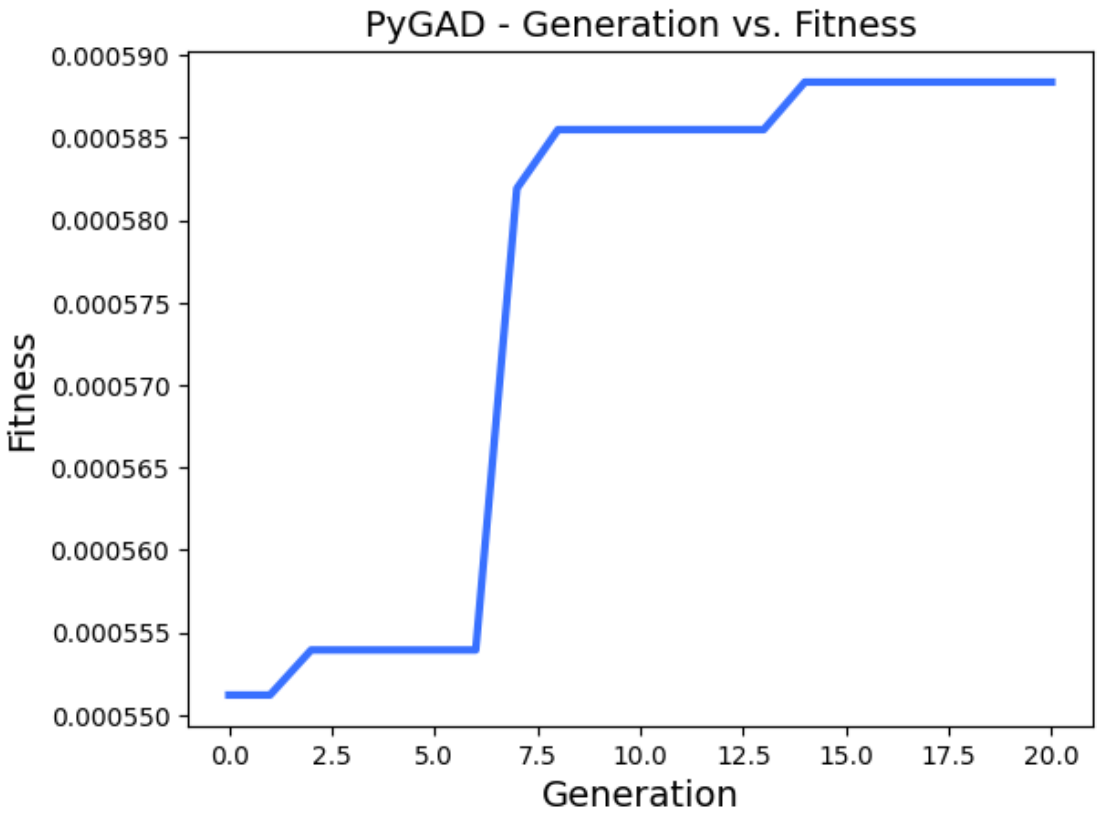


GRÁFICO DE VALORES DE LOS GENES :

[36]:
```
tmp =ga_instance.plot_genes()
```