TSP with Genetic Algorithm

November 6, 2021

1 Getting input

saving Adjacency matrix with weight of edges

[2]:
```
graph = []
n = int(input("Number of cities: "))

for i in range(n):
    l = [0 for i in range(n)]
    graph.append(l)

l=list(map(int, input().split()))
cnt = 0
for i in range(n-1):
    for j in range(n-1-i):
        graph[i][j+i+1] = l[cnt]
        graph[j+i+1][i] = l[cnt]
        cnt += 1

#sample view
for i in range(4):
    for j in range(4):
        print(graph[i][j], end=' ')
    print('\n')
```

```
Number of cities: 29
97 205 139  86  60 220  65 111 115 227  95  82 225 168 103 266 205 149 120  58
257 152  52 180 136  82  34 145 129 103  71 105 258 154 112  65 204 150  87 176
137 142 204 148 148  49  41 211 226 116 197  89 153 124  74 219 125 175 386 269
134 184 313 201 215 267 248 271 274 236 272 160 151 300 350 239 322  78 276 220
60 167 182 180 162 208  39 102 227  60  86  34  96 129  69  58  60 120 119 192
114 110 192 136 173 173  51 296 150  42 131 268  88 131 245 201 175 275 218 202
119  50 281 238 131 244  51 166  95  69 279 114  56 150 278  46 133 266 214 162
302 242 203 146  67 300 205 111 238  98 139  52 120 178 328 206 147 308 172 203
165 121 251 216 122 231 249 209 111 169  72 338 144 237 331 169 151 227 133 104
242 182  84 290 230 146 165 121 270  91  48 158 200  39  64 210 172 309  68 169
286 242 208 315 259 240 160  90 322 260 160 281  57 192 107  90 140 195  51 117
```

```
72 104 153  93  88  25  85 152 200 104 139 154 134 149 135 320 146  64  68 143
106  88  81 159 219  63 216 187  88 293 191 258 272 174 311 258 196 347 288 243
192 113 345 222 144 274 124 165  71 153 144  86  57 189 128  71  71  82 176 150
56 114 168  83 115 160  61 165  51  32 105 127 201  36 254 196 136 260 212 258
234 106 110  56  49  91 153  91 197 136  94 225 151 201 205 215 159  64 126 128
190  98  53  78 218  48 127 214  61 155 157 235  47 305 243 186 282 261 300 252
105 100 176  66 253 183 146 231 203 239 204 113 152 127 150 106  52 235 112 179
221  79 163 220 119 164 135 152 153 114 236 201  90 195  90 127  84  91 273 226
148 296 238 291 269 112 130 286  74 155 291 130 178  38  75 180 281 120 205 270
213 145  36  94 217 162
0 97 205 139

97 0 129 103

205 129 0 219

139 103 219 0
```

2 Creating random chromosomes

generating a random permutation

[3]:
```python
import random
def first_population(number_of_samples, size):
    l = []
    for i in range(number_of_samples):
        chromosome = []
        for j in range(size):
            a = random.randint(0,size-1)
            while a in chromosome:
                a = random.randint(0,size-1)
            chromosome.append(a)
        l.append(chromosome)
    return l

first_population(3,5)
```

[3]:
```
[[0, 4, 1, 2, 3], [1, 2, 0, 4, 3], [1, 2, 4, 3, 0]]
```

3 Fitness function (sum of weights in chromosome path)

[4]:
```
def fitness(chromosome):
    w = 0
    for i in range(len(chromosome)):
        w += graph[chromosome[i]][chromosome[(i+1) % len(chromosome)]]
    return w

chromosome = first_population(1,29)[0]
print(chromosome)
print(fitness(chromosome))
```

```
[24, 5, 1, 8, 18, 2, 20, 9, 4, 22, 25, 13, 11, 6, 3, 10, 14, 16, 23, 15, 7, 17,
0, 26, 28, 27, 12, 21, 19]
5091
```

4 Cross over(ordered crossover)

[40]:
```
def cross_over(parent1, parent2):

    size = len(parent1)
    # result
    res = [-1 for i in range(size)]
    a = random.randint(0, size-1)
    b = -1
    while b == a or b == -1:
        b = random.randint(0, size-1)
    if (a > b):
        a, b = b, a
    # putting parent1 genes
    i = a
    while i != b:
        res[i] = parent1[i]
        i = (i + 1) % size

    # putting parent2 genes
    cnt = 0
    i = b
    while cnt < size:

        if parent2[cnt] not in res:
            res[i] = parent2[cnt]
            i = (i + 1) % size
        cnt += 1

```

```
    return res
parents = first_population(2,5)
print(parents)
print(cross_over(parents[0], parents[1]))
```

```
[[4, 3, 1, 2, 0], [3, 4, 0, 1, 2]]
[0, 3, 1, 2, 4]
```

5 Mutation

[6]:
```
def mutation(chromosome, mutation_rate):
    for i in range(len(chromosome)):
        if (random.random() < mutation_rate):
            x = random.randint(0,len(chromosome)-1)
            # swap:
            chromosome[i], chromosome[x] = chromosome[x], chromosome[i]
    return chromosome

print(mutation([1,2,3,4,5], 0.5))
```

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

6 Main Function

in every iteration i printed the two best answers (weight) in next expriments they are comment(so there is no more printing just showing the plot...) at first my answers didn't converge to enything and the plot was like a sin(x) graph. (the first examines were like second plot below. then i tried to change numbers but nothing changed. first tries were somthing like *genentic(100, 100, 0.1, 20) and genentic(1000, 100, 0.5, 20) and ...)* genentic(iterations, population_number, mutation_rate, elite_size)

then i radicaly changed almost evrything to be more stable like below: genentic(100, 1000, 0.01, 100) mutation rate was 10times bigger and elite size waz 10times smaller... and you can see the result plot that finaly converged!

[43]:
```
import matplotlib.pyplot as plt
def reverse(l):
    return float(1) / float(l)

def genentic(iterations, population_number, mutation_rate, elite_size):
    population = first_population(population_number, n)
    progress = []
    for i in range(iterations):
```

```python
        print(i)
        population.sort(key = fitness)
        print(fitness(population[0]), fitness(population[1]))
        progress.append(fitness(population[0]))
        child = []
        # moving elites to next generation
        for j in range(elite_size):
            child.append(population[j])

        # selection based on fitness(normalize fitness)
        select = []
        normalized_fitness = [(reverse(fitness(j))) for j in population]
        reversed(population)
        sum_fitness = sum(normalized_fitness)
        while(len(select) < population_number - elite_size):
            for j in range(population_number):
                if (random.random() < normalized_fitness[j] / sum_fitness):
                    select.append(population[j])
        # making new child from selected parents
        j = 0
        #print(select)
        while (len(child) < population_number):
            child.append(cross_over(select[j], select[len(select) - 1 - j]))
            j = (j + 1) % len(select)
        # mutate next generation
        for j in range(population_number):
            child[j] = mutation(child[j], mutation_rate)
        population = child

    population.sort(key = fitness)
    print(population[0], fitness(population[0]))

    plt.plot(progress)
    plt.ylabel('Distance')
    plt.xlabel('Generation')
    plt.show()

genentic(100, 1000, 0.01, 100)
```

```
0
3765 3770
1
3621 3622
2
3549 3577
3
3447 3462
```

4
3230 3447
5
3230 3403
6
3230 3281
7
3282 3334
8
3282 3334
9
3166 3282
10
3132 3166
11
3001 3132
12
2938 2952
13
2938 2952
14
2938 2952
15
2938 2952
16
2885 2952
17
2885 2952
18
2885 2957
19
2957 2996
20
2923 2957
21
2923 2953
22
2923 2953
23
2732 2953
24
2732 2812
25
2732 2812
26
2812 2863
27
2799 2809

28
2747 2799
29
2747 2809
30
2747 2809
31
2747 2777
32
2747 2752
33
2747 2765
34
2776 2777
35
2776 2777
36
2705 2724
37
2443 2705
38
2443 2705
39
2443 2687
40
2443 2586
41
2283 2443
42
2246 2283
43
2246 2283
44
2246 2283
45
2246 2283
46
2246 2380
47
2246 2441
48
2246 2441
49
2246 2454
50
2246 2423
51
2246 2290

52
2290 2423
53
2290 2423
54
2290 2423
55
2290 2423
56
2290 2424
57
2290 2306
58
2290 2306
59
2290 2306
60
2290 2290
61
2290 2290
62
2280 2290
63
2280 2306
64
2306 2311
65
2306 2311
66
2306 2311
67
2306 2346
68
2306 2346
69
2306 2346
70
2306 2350
71
2306 2350
72
2350 2355
73
2350 2355
74
2272 2340
75
2272 2340

76
2272 2340
77
2074 2272
78
2074 2272
79
2118 2215
80
2118 2126
81
2118 2126
82
2118 2215
83
2118 2215
84
2215 2247
85
2215 2265
86
2166 2215
87
2166 2215
88
2166 2215
89
2166 2215
90
2166 2215
91
2106 2156
92
2106 2156
93
2007 2106
94
2007 2146
95
2007 2098
96
2007 2098
97
2007 2098
98
2007 2098
99
2007 2079

```
[2, 25, 8, 28, 19, 1, 9, 20, 12, 3, 14, 10, 21, 17, 16, 13, 18, 24, 6, 22, 23,
15, 26, 7, 0, 27, 11, 5, 4] 2007
```

output_12_1.png

[44]:
```
genentic(100, 1000, 0.1, 20)
```

```
0
3612 3678
1
3495 3610
2
3343 3546
3
3445 3469
4
3591 3606
5
3450 3560
6
3556 3560
7
3342 3358
8
3542 3572
9
3437 3460
10
3417 3444
11
3466 3515
12
3573 3598
13
3343 3432
14
3560 3579
15
3490 3648
16
3238 3415
```

17
3246 3471
18
3238 3246
19
3492 3555
20
3638 3689
21
3650 3686
22
3410 3546
23
3427 3482
24
3524 3584
25
3586 3608
26
3430 3572
27
3532 3669
28
3543 3596
29
3633 3653
30
3493 3588
31
3365 3509
32
3483 3659
33
3481 3596
34
3579 3631
35
3401 3549
36
3260 3598
37
3330 3539
38
3540 3569
39
3577 3595
40
3532 3676

41
3525 3532
42
3379 3716
43
3562 3564
44
3147 3549
45
3131 3415
46
3305 3453
47
3227 3431
48
3441 3503
49
3457 3607
50
3450 3497
51
3412 3539
52
3437 3633
53
3471 3595
54
3561 3579
55
3507 3522
56
3522 3552
57
3563 3565
58
3550 3552
59
3443 3474
60
3539 3604
61
3501 3633
62
3434 3522
63
3387 3499
64
3451 3499

65
3501 3513
66
3456 3533
67
3503 3561
68
3454 3519
69
3217 3390
70
3284 3545
71
3345 3456
72
3538 3616
73
3320 3648
74
3236 3490
75
3174 3512
76
3500 3556
77
3382 3397
78
3516 3594
79
3505 3601
80
3489 3627
81
3662 3710
82
3526 3640
83
3452 3627
84
3546 3673
85
3559 3662
86
3523 3605
87
3547 3684
88
3318 3360

```
89
3318 3557
90
3456 3458
91
3436 3556
92
3377 3557
93
3631 3640
94
3386 3497
95
3520 3570
96
3494 3609
97
3442 3482
98
3428 3544
99
3533 3606
[22, 8, 27, 10, 21, 13, 16, 17, 18, 28, 20, 11, 15, 0, 3, 4, 19, 12, 14, 23, 1,
2, 25, 5, 9, 6, 24, 7, 26] 3503
```

output_13_1.png

now i take the iterations from 100 to 500 because in the last plot it seems to be decreasing in the futre iterations! this time running time of code was very longer but answer didn't change a lot.

[48]:
```
def genentic_just_plot(iterations, population_number, mutation_rate, elite_size):
    population = first_population(population_number, n)
    progress = []
    for i in range(iterations):
        #print(i)
        population.sort(key = fitness)
        #print(fitness(population[0]), fitness(population[1]))
        progress.append(fitness(population[0]))
        child = []
        # moving elites to next generation
        for j in range(elite_size):
```

```
            child.append(population[j])

        # selection based on fitness(normalize fitness)
        select = []
        normalized_fitness = [(reverse(fitness(j))) for j in population]
        reversed(population)
        sum_fitness = sum(normalized_fitness)
        while(len(select) < population_number - elite_size):
            for j in range(population_number):
                if (random.random() < normalized_fitness[j] / sum_fitness):
                    select.append(population[j])
        # making new child from selected parents
        j = 0
        #print(select)
        while (len(child) < population_number):
            child.append(cross_over(select[j], select[len(select) - 1 - j]))
            j = (j + 1) % len(select)
        # mutate next generation
        for j in range(population_number):
            child[j] = mutation(child[j], mutation_rate)
        population = child

    population.sort(key = fitness)

    plt.plot(progress)
    plt.ylabel('Distance')
    plt.xlabel('Generation')
    plt.show()

genentic_just_plot(500, 1000, 0.01, 100)
```

output_15_0.png

so this time i changed the number of iterations back to first value and increased the size of population

[49]:
```
genentic_just_plot(100, 2000, 0.01, 100)
```

output_17_0.png

this time i changed the mutation rate to 10times smaller and this time was the best answer so far!

```
[50]: genentic_just_plot(100, 1000, 0.001, 100)
```

output_19_0.png

and now increasing the elite size x2.even beat the previous result!

```
[51]: genentic_just_plot(100, 1000, 0.001, 200)
```

output_21_0.png

elite*2 and iteratio*2 again better result!

```
[52]: genentic_just_plot(200, 1000, 0.001, 400)
```

output_23_0.png

I end my exprience with this plot that looks like a human face :)) and i think it can't get better result...

[57]:
```
genentic_just_plot(500, 1000, 0.001, 400)
```

output_25_0.png

[59]:
```
genentic(500, 1000, 0.001, 400)
```

```
0
3642 3753
1
3642 3734
2
3642 3670
3
3559 3642
4
3467 3481
5
3111 3328
6
3111 3401
7
3111 3376
8
3111 3376
9
2810 3111
10
2810 3111
11
2810 3055
12
2810 3055
13
2810 2892
14
2810 2892
15
2810 2916
16
2810 2849
17
```

2810 2849
18
2810 2849
19
2810 2849
20
2810 2820
21
2810 2820
22
2779 2810
23
2779 2783
24
2779 2783
25
2779 2783
26
2777 2779
27
2690 2777
28
2680 2690
29
2664 2680
30
2646 2664
31
2546 2551
32
2546 2551
33
2546 2551
34
2546 2551
35
2516 2527
36
2513 2516
37
2410 2473
38
2410 2471
39
2326 2410
40
2326 2410
41

```
2326 2410
42
2326 2332
43
2326 2332
44
2326 2332
45
2332 2407
46
2290 2332
47
2290 2332
48
2290 2332
49
2290 2332
50
2290 2332
51
2290 2332
52
2290 2332
53
2244 2267
54
2244 2267
55
2244 2267
56
2232 2244
57
2232 2244
58
2211 2232
59
2211 2232
60
2211 2232
61
2211 2232
62
2211 2211
63
2178 2211
64
1978 2140
65
```

1978 2140
66
1978 2140
67
1978 2087
68
1978 2087
69
1978 2035
70
2035 2087
71
2035 2140
72
2035 2044
73
2035 2044
74
2035 2044
75
2035 2044
76
2035 2044
77
2006 2035
78
2006 2035
79
2006 2035
80
1981 2006
81
1964 1981
82
1964 1981
83
1964 1978
84
1964 1978
85
1978 1978
86
1929 1978
87
1916 1929
88
1916 1929
89

1916 1926
90
1916 1926
91
1916 1926
92
1916 1926
93
1916 1926
94
1916 1926
95
1916 1926
96
1916 1926
97
1915 1916
98
1915 1916
99
1915 1916
100
1915 1916
101
1915 1916
102
1895 1915
103
1895 1915
104
1895 1915
105
1895 1915
106
1895 1908
107
1779 1873
108
1779 1857
109
1779 1856
110
1779 1856
111
1779 1853
112
1779 1844
113

1765 1779
114
1765 1779
115
1765 1779
116
1756 1765
117
1756 1765
118
1756 1765
119
1765 1773
120
1765 1773
121
1765 1773
122
1765 1773
123
1765 1773
124
1765 1773
125
1765 1773
126
1765 1773
127
1765 1773
128
1765 1773
129
1741 1761
130
1741 1761
131
1741 1757
132
1741 1761
133
1741 1761
134
1741 1761
135
1741 1761
136
1741 1756
137

1741 1756
138
1741 1756
139
1741 1742
140
1736 1741
141
1735 1741
142
1735 1735
143
1735 1735
144
1735 1735
145
1735 1735
146
1731 1735
147
1731 1735
148
1731 1735
149
1731 1734
150
1731 1734
151
1731 1734
152
1728 1731
153
1682 1727
154
1682 1719
155
1682 1719
156
1682 1719
157
1682 1702
158
1682 1702
159
1682 1702
160
1682 1702
161

1682 1702
162
1682 1702
163
1682 1702
164
1682 1702
165
1682 1702
166
1682 1702
167
1682 1702
168
1682 1702
169
1682 1702
170
1682 1702
171
1682 1702
172
1682 1699
173
1682 1699
174
1682 1699
175
1682 1699
176
1682 1691
177
1682 1691
178
1682 1691
179
1682 1691
180
1682 1690
181
1682 1690
182
1682 1690
183
1678 1682
184
1677 1678
185

1677 1677
186
1676 1677
187
1676 1677
188
1676 1677
189
1676 1677
190
1676 1677
191
1676 1677
192
1665 1676
193
1665 1676
194
1665 1676
195
1665 1676
196
1662 1665
197
1662 1665
198
1662 1665
199
1665 1675
200
1665 1665
201
1658 1665
202
1658 1665
203
1654 1658
204
1654 1658
205
1654 1658
206
1654 1658
207
1654 1658
208
1654 1658
209

1654 1658
210
1654 1658
211
1654 1658
212
1654 1658
213
1654 1658
214
1654 1658
215
1654 1658
216
1654 1658
217
1654 1658
218
1654 1658
219
1654 1658
220
1654 1658
221
1650 1654
222
1650 1654
223
1650 1650
224
1650 1650
225
1650 1650
226
1650 1650
227
1650 1650
228
1650 1650
229
1650 1650
230
1650 1650
231
1650 1650
232
1650 1650
233

```
1643 1643
234
1643 1643
235
1627 1643
236
1627 1643
237
1627 1643
238
1627 1643
239
1627 1643
240
1627 1643
241
1627 1634
242
1627 1634
243
1627 1634
244
1627 1634
245
1622 1627
246
1622 1627
247
1622 1627
248
1622 1627
249
1622 1622
250
1615 1622
251
1615 1622
252
1615 1622
253
1615 1622
254
1615 1622
255
1615 1622
256
1615 1615
257
```

1615 1615
258
1615 1615
259
1615 1615
260
1615 1615
261
1615 1615
262
1615 1615
263
1615 1615
264
1615 1615
265
1615 1615
266
1615 1615
267
1615 1615
268
1615 1615
269
1615 1615
270
1615 1615
271
1615 1615
272
1615 1615
273
1610 1615
274
1610 1610
275
1610 1610
276
1610 1610
277
1610 1610
278
1610 1610
279
1610 1610
280
1610 1610
281

1610 1610
282
1610 1610
283
1610 1610
284
1610 1610
285
1610 1610
286
1610 1610
287
1610 1610
288
1610 1610
289
1610 1610
290
1610 1610
291
1610 1610
292
1610 1610
293
1610 1610
294
1610 1610
295
1610 1610
296
1610 1610
297
1610 1610
298
1610 1610
299
1610 1610
300
1610 1610
301
1610 1610
302
1610 1610
303
1610 1610
304
1610 1610
305

1610 1610
306
1610 1610
307
1610 1610
308
1610 1610
309
1610 1610
310
1610 1610
311
1610 1610
312
1610 1610
313
1610 1610
314
1610 1610
315
1610 1610
316
1610 1610
317
1610 1610
318
1610 1610
319
1610 1610
320
1610 1610
321
1610 1610
322
1610 1610
323
1610 1610
324
1610 1610
325
1610 1610
326
1610 1610
327
1610 1610
328
1610 1610
329

```
1610 1610
330
1610 1610
331
1610 1610
332
1610 1610
333
1610 1610
334
1610 1610
335
1610 1610
336
1610 1610
337
1610 1610
338
1610 1610
339
1610 1610
340
1610 1610
341
1610 1610
342
1610 1610
343
1610 1610
344
1610 1610
345
1610 1610
346
1610 1610
347
1610 1610
348
1610 1610
349
1610 1610
350
1610 1610
351
1610 1610
352
1610 1610
353
```

1610 1610
354
1610 1610
355
1610 1610
356
1610 1610
357
1610 1610
358
1610 1610
359
1610 1610
360
1610 1610
361
1610 1610
362
1610 1610
363
1610 1610
364
1610 1610
365
1610 1610
366
1610 1610
367
1610 1610
368
1610 1610
369
1610 1610
370
1610 1610
371
1610 1610
372
1610 1610
373
1610 1610
374
1610 1610
375
1610 1610
376
1610 1610
377

1610 1610
378
1610 1610
379
1610 1610
380
1610 1610
381
1610 1610
382
1610 1610
383
1610 1610
384
1610 1610
385
1610 1610
386
1610 1610
387
1610 1610
388
1610 1610
389
1610 1610
390
1610 1610
391
1610 1610
392
1610 1610
393
1610 1610
394
1610 1610
395
1610 1610
396
1610 1610
397
1610 1610
398
1610 1610
399
1610 1610
400
1610 1610
401

1610 1610
402
1610 1610
403
1610 1610
404
1610 1610
405
1610 1610
406
1610 1610
407
1610 1610
408
1610 1610
409
1610 1610
410
1610 1610
411
1610 1610
412
1610 1610
413
1610 1610
414
1610 1610
415
1610 1610
416
1610 1610
417
1610 1610
418
1610 1610
419
1610 1610
420
1610 1610
421
1610 1610
422
1610 1610
423
1610 1610
424
1610 1610
425

```
1610 1610
426
1610 1610
427
1610 1610
428
1610 1610
429
1610 1610
430
1610 1610
431
1610 1610
432
1610 1610
433
1610 1610
434
1610 1610
435
1610 1610
436
1610 1610
437
1610 1610
438
1610 1610
439
1610 1610
440
1610 1610
441
1610 1610
442
1610 1610
443
1610 1610
444
1610 1610
445
1610 1610
446
1610 1610
447
1610 1610
448
1610 1610
449
```

1610 1610
450
1610 1610
451
1610 1610
452
1610 1610
453
1610 1610
454
1610 1610
455
1610 1610
456
1610 1610
457
1610 1610
458
1610 1610
459
1610 1610
460
1610 1610
461
1610 1610
462
1610 1610
463
1610 1610
464
1610 1610
465
1610 1610
466
1610 1610
467
1610 1610
468
1610 1610
469
1610 1610
470
1610 1610
471
1610 1610
472
1610 1610
473

```
1610 1610
474
1610 1610
475
1610 1610
476
1610 1610
477
1610 1610
478
1610 1610
479
1610 1610
480
1610 1610
481
1610 1610
482
1610 1610
483
1610 1610
484
1610 1610
485
1610 1610
486
1610 1610
487
1610 1610
488
1610 1610
489
1610 1610
490
1610 1610
491
1610 1610
492
1610 1610
493
1610 1610
494
1610 1610
495
1610 1610
496
1610 1610
497
```

```
1610 1610
498
1610 1610
499
1610 1610
[14, 17, 13, 16, 21, 10, 18, 24, 6, 22, 7, 26, 15, 12, 23, 0, 27, 5, 11, 8, 25,
2, 28, 4, 20, 1, 19, 9, 3] 1610
```

output_26_1.png

[14, 17, 13, 16, 21, 10, 18, 24, 6, 22, 7, 26, 15, 12, 23, 0, 27, 5, 11, 8, 25, 2, 28, 4, 20, 1, 19, 9, 3] 1610 so this is my approximate answer for bayg29