**Travelling Salesman Problem – Oğuz Kaan Satan**

**Sınıflar**

1) City

* **\_\_init\_\_(self, x, y):** City sınıfını initialize eder; x ve y koordinatlarını atar.
* **distance(self, city):** İlgili şehrin başka bir şehir objesi ile arasındaki mesafeyi ölçer.
* **\_\_repr\_\_(self):** Şehre ait koordinat bilgilerini “(x, y)” formatında string döndürür.

2) Fitness

* **\_\_init\_\_(route):** route atar, distance=0, fitness=0 set eder.
* **routeDistance(self):** Sınıfa ait route’ın distance (uzaklık) değerini hesaplar.
* **routeFitness(self):** Sınıfa ait route’ın fitness (uyumluluk) değerini hesaplar.

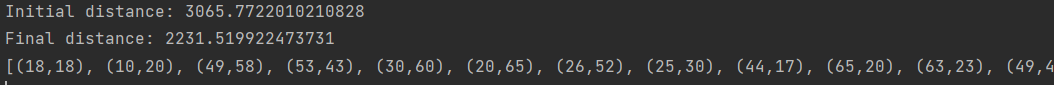
**Fonksiyonlar**

* **getCityListFromFile():** CitiesCoordinates.txt dosyasındaki verileri okuyarak her satırdan bir City sınıfı yaratıp, liste döndürür.
* **createRoute(cityList):** cityList’in “karıştırır” (shuffle) ve elde edilen yeni listeyi döndürür.
* **initialPopulation(popSize, cityList):** popSize değişkeninin büyüklüğü kadarlık bir topluluk oluşturmayı amaçlar. Popülasyonun her elemanı farklı bir yol alma şeklini ifade ettiği için her elemana createRoute metoduyla elde edilen route’ı atar.
* **rankRoutes(population):** population parametresinin içerisindeki tüm rotaların (route) uyumluluk (fitness) değerlerini hesaplar. Elde edilen sonuçları en uyumludan en uyumsuza doğru sıralar (sort).
* **selection(popRanked, eliteSize):** popRanked değişkeni sort edilmiş popülasyonu ifade eder, eliteSize parametresi de seçim sırasında korunan eleman sayısını ifade eder. Bu fonksiyon eşleşme sırasında kullanılacak ebeveyn listesini sağlamaktadır.
* **matingPool(population, selectionResults):** Seçim listesindeki popülasyon üyeleri sırayla eşleşme havuzuna eklenir.
* **breed(parent1, parent2):** Eşleşme işlemi iki farklı ebeveynin genlerini sentezleyerek yeni bir birey oluşturur.
* **breedPopulation(matingpool, eliteSize):** Belli bir sayıda birey yeni jenerasyona direkt olarak aktarılır. Diğer kişiler ise art arda gelen bireylerin breed fonksiyonuyla yeni çocuk türetmeleriyle oluşturulur. Breed ile oluşturulan her yeni kişi döndürülecek çocuklar listesine eklenir.
* **mutate(individual, mutationRate):** Mutasyon oranı baz alınarak, bir bireye ait genin bazı alanları yer değiştirilerek mutasyona uğratılır (Örneğin; rastgele bir sayı belirlenir, ve 3. ve 5. Şehirlerin yerleri ilgili bireyin (rotanın) geninde yer değiştirir).
* **mutatePopulation(population, mutationRate):** Bir popülasyona ait tüm bireyler için mutate fonksiyonu çağrılır; mutasyonlu yeni popülasyon fonksiyon tarafından döndürülür.
* **nextGeneration(currentGen, eliteSize, mutationRate):** Gelecek nesili hesaplayan bu fonksiyon sırasıyla algoritmanın akışını yerine getiren fonksiyonları çağırır. Bunlar:
  + **rankRoutes**
  + **selection**
  + **matingPool**
  + **breedPopulation**
  + **mutatePopulation**

fonksiyonlarıdır. Bu işlem sonucunda yeni bir popülasyon oluşturulmuş olur.

* **runGA(population, popSize, eliteSize, mutationRate, generations):** Kurmuş olduğumuz genetik algoritmayı çalıştıran fonksiyondur. Sağlanan iterasyon sayısı (generations) kadar yeni jenerasyon üretir; her seferinde ilgili neslin performansını ölçerek en başarılı rotayı (route) seçer. Elde edilen değerleri bir array’e kaydederek, algoritma sonlandığında bunların bir grafiğe basılmasını sağlar. En başarılı rotanın uzunluğunu ve ziyaret ettiği koordinatların bilgisini döndürür.

**Yorumlama**

****

Şekil 1

Şekil 1’de görüldüğü üzere 500 nesil sonunda başlangıçtaki rotaya kıyasla çok daha kısa bir yol alınmaktadır. Bu da algoritmanın gerçekten de başarılı noktaya doğru yakınsama gerçekleştirdiğini göstermektedir.Chart, histogram

Description automatically generated

Şekil 2

Şekil 2’den de görüleceği üzere Başlangıç noktasından ilk yola çıkıldığında yüksek miktarda yakınsama gerçekleşmiştir. Fakat 300. nesilden sonra tekrardan değerlerde yükselme olduğu görülmektedir. Bu yükselmelerin “başarısız mutasyonlar” kaynaklı olduğu savunulabilir.

Text

Description automatically generated

Şekil 3 - 1000 Nesil, 2000 Nesil ve 5000 Nesil Sonuçları

Chart

Description automatically generated

Şekil 4 - 5000 Nesil Boyunca Uzaklık Değerleri

Final distance: 2275.2252437983425

[(55,45), (6,38), (40,25), (17,34), (20,50), (11,31), (49,42), (15,47), (12,24), (25,21), (55,20), (63,23), (32,12), (37,31), (26,27), (47,16), (42,7), (30,5), (49,58), (61,52), (55,54), (22,27), (22,22), (28,18), (30,60), (26,52), (2,48), (10,43), (15,60), (31,67), (55,60), (65,55), (62,77), (57,68), (63,65), (53,43), (49,73), (40,60), (23,3), (35,17), (19,21), (37,56), (46,13), (60,12), (20,20), (4,18), (18,24), (11,14), (5,30), (14,37), (45,65), (41,37), (35,40), (36,26), (65,35), (57,29), (45,30), (16,22), (10,20), (31,52), (24,58), (37,47), (30,25), (15,10), (35,35), (50,35), (35,69), (21,24), (15,19), (5,5), (24,12), (20,40), (13,52), (15,77), (6,68), (27,69), (20,65), (26,35), (27,43), (8,56), (2,60), (47,47), (67,5), (18,18), (45,10), (49,11), (53,12), (57,48), (65,20), (55,5), (25,24), (20,26), (15,30), (56,37), (44,17), (25,30), (53,52), (64,42), (56,39), (45,20), (41,49)]