



**FATİH
SULTAN
MEHMET**
VAKIF ÜNİVERSİTESİ

Student:

Students Name: Ayman Saeid – Ekrem Baş

ID Number: 2221221366 - 2221221051

Department: Computer Engineering

Project:

Topic: Genetik Algoritma ile Gezgin Satıcı Problemi (TSP) Çözümü

Course:

Name: Heuristic Optimization Algorithms

Instructor: Dr. Cumali TÜRKMENOĞLU

Content

1- Project Topic.....	Error! Bookmark not defined.
2- Tasks Completed During the Project.....	Error! Bookmark not defined.
3- Additional Notes.....	Error! Bookmark not defined.
4- References.....	Error! Bookmark not defined.



**FATİH
SULTAN
MEHMET**
VAKIF ÜNİVERSİTESİ

1- Giriş

Bu rapor, NP-Zor (NP-Hard) bir optimizasyon problemi olan Gezgin Satıcı Problemi'nin (Traveling Salesman Problem - TSP) çözümü için geliştirilen Genetik Algoritma (GA) tabanlı bir yaklaşımı detaylandırmaktadır. Problem, belirli bir şehirler kümesi için her şehre tam olarak bir kez uğrayıp başlangıç noktasına geri dönülen en kısa rotanın bulunmasını hedefler. Temel amaç, toplam seyahat mesafesini minimize etmektir.

Geliştirilen Python tabanlı program, standart TSPLIB formatındaki problem örneklerini okuyabilmekte, farklı genetik algoritma parametrelerini (popülasyon büyüklüğü, çaprazlama ve mutasyon oranları vb.) **Izgara Arama (Grid Search)** yöntemiyle sistematik olarak test ederek en iyi çözümü bulmakta ve elde edilen sonuçları hem rota grafiği hem de yakınsama eğrisi olarak görselleştirmektedir.

2- Problemin Tanımı ve Matematiksel Formülasyonu

2.1. Problemin Formülasyonu

Gezgin Satıcı Problemi, genellikle bir Tamsayılı Doğrusal Programlama (Integer Linear Programming - ILP) problemi olarak formüle edilir. Amaç, tüm şehirleri ziyaret eden ve toplam maliyeti (mesafeyi) minimize eden bir tur bulmaktır.

2.1.1. Parametreler ve Kümeler

- **n**: Toplam şehir sayısı.
- **V**: Şehirlerin (düğümlerin) kümesi, $V=\{1,2,...,n\}$.
- **c_{ij}**: Şehir *i*'den şehir *j*'ye gitmenin maliyeti (Öklid mesafesi).

2.1.2. Karar Değişkenleri

Karar değişkeni, bir şehirden diğerine giden yolun turda kullanılıp kullanılmadığını belirten ikili (binary) bir değişkendir.

$$x_{ij} = \begin{cases} 1, & \text{eğer rota } i \text{ şehirden } j \text{ şehrine direkt gidiyorsa} \\ 0, & \text{aksi halde} \end{cases}$$

Bu değişken $i, j \in V$ ve $i \neq j$ için tanımlıdır. Bir şehirden kendisine giden bir yol olamaz ($x_{ii}=0$).

2.1.3. Amaç Fonksiyonu

Amaç, turdaki tüm yolların toplam maliyetini minimize etmektir.

$$\text{Minimize } Z = \sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij} x_{ij}$$

Bu formül, tüm olası şehir çiftleri (i,j) için, eğer o yol turda kullanılıyorsa ($x_{ij}=1$) mesafeyi (c_{ij}) toplama ekler.

2.1.4. Kısıtlar

Modelin geçerli bir tur oluşturmasını sağlamak için aşağıdaki kısıtlar gereklidir:

Her Şehre Tam Olarak Bir Kez Girilmelidir: Her şehir için, o şehre gelen yolların sayısı tam olarak 1 olmalıdır.

$$\sum_{i \in V, i \neq j} x_{ij} = 1 \quad \forall j \in V$$

Her Şehirden Tam Olarak Bir Kez Çıkılmalıdır: Benzer şekilde, her şehirden ayrılan yolların sayısı da tam olarak 1 olmalıdır.

$$\sum_{i \in V, i \neq j} x_{ij} = 1 \quad \forall j \in V$$

Alt Tur Eliminasyon Kısıtları (Subtour Elimination Constraints): Yukarıdaki iki kısıt, tek bir tam tur yerine birden fazla bağımsız döngü (alt tur) oluşmasına izin verebilir. Bunu önlemek için, şehirlerin herhangi bir alt kümesinde (S) kapalı bir döngü oluşmasını engelleyen kısıtlar eklenir.

$$\sum_{i \in S} \sum_{j \in S, j \neq i} x_{ij} \leq |S| - 1 \quad \forall S \subset V, 2 \leq |S| \leq n - 1$$

Not: Bu kısıtların sayısı, şehir sayısı ile birlikte üssel olarak (2^n) arttığı için, TSP'yi NP-zor bir problem yapan temel neden budur. Bu yüzden büyük problemler için Genetik Algoritmalar gibi sezgisel (heuristic) yöntemler tercih edilir.

Değişken Türü Kısıtı: Karar değişkenleri ikili (binary) olmalıdır.

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V, i \neq j$$

3- Kullanılan Kütüphaneler ve Araçlar

Projenin geliştirilmesinde aşağıdaki Python kütüphanelerinden yararlanılmıştır:

math: Şehirler arası Öklid mesafesini hesaplamak gibi temel matematiksel işlemler için kullanılmıştır.

random: Genetik algoritmanın temelini oluşturan rastgelelik için kritik bir kütüphanedir. Rastgele turlar (bireyler) oluşturmak, seçim ve mutasyon işlemleri için kullanılmıştır.

matplotlib.pyplot: Güçlü bir 2D çizim kütüphanesidir. Sonuçları ve yakınsama grafiğini görselleştirmek için kullanılmıştır.

networkx: Karmaşık ağları (grafları) oluşturmak, işlemek ve görselleştirmek için kullanılır. Bu projede, şehirleri ve rotaları bir graf yapısı olarak modellemek ve çizmek için tercih edilmiştir.

time: Algoritmanın çalışma süresini (hesaplama verimliliğini) ölçmek için kullanılmıştır.

itertools: Verimli döngüler oluşturmak için kullanılan bir kütüphanedir. Özellikle, `itertools.product` fonksiyonu, parametre optimizasyonu aşamasında farklı parametre kombinasyonlarını denemek için kullanılmıştır.

4- Genetik Algoritma Yaklaşımı ve Uygulaması

TSP'nin karmaşıklığı nedeniyle, probleme sezgisel bir yaklaşım olan Genetik Algoritma ile çözüm aranmıştır. GA, biyolojik evrimden esinlenen bir optimizasyon tekniğidir. Projede uygulanan GA'nın temel bileşenleri aşağıda açıklanmıştır.

4.1. Kromozom Temsili (Chromosome Representation)

- Ne Olduğu:** Problemin bir çözüm adayının nasıl kodlandığıdır. Bu projede **permutasyon tabanlı bir kromozom temsili** seçilmiştir. Her bir kromozom (birey), gezginin şehirleri ziyaret edeceği sırayı temsil eden bir tamsayı listesidir. Örneğin, 5 şehirlik bir problem için `[2, 0, 4, 1, 3]` şeklindeki bir kromozom, turun 2. şehirden başlayıp sırasıyla 0, 4, 1 ve 3. şehirlere giderek tamamlandığını ifade eder.
- Neden Seçildi:** Bu temsil, TSP'nin temel kısıtlarından biri olan "her şehrin tam olarak bir kez ziyaret edilmesi" kuralını doğal olarak sağlar. Her şehir listede yalnızca bir kez yer aldığı için geçersiz çözüm (bir şehre birden fazla uğrayan) üretme riski ortadan kalkar. Bu yapı, kodda `create_individual(num_cities)` fonksiyonu ile gerçekleştirilir.

4.2. Uygunluk Fonksiyonu (Fitness Function)

- Ne Olduğu:** Bir kromozomun (çözümün) ne kadar "iyi" olduğunu ölçen bir fonksiyondur. Bu projede uygunluk değeri, turun toplam mesafesidir.
- Neden Seçildi:** Algoritmanın amacı toplam tur mesafesini minimize etmek olduğundan, uygunluk fonksiyonu olarak toplam mesafenin kendisi kullanılmıştır. **Daha düşük bir mesafe değeri, daha iyi (daha uygun) bir çözümü ifade eder.** Bu fonksiyon, kodda `total_distance(tour, cities_coords)` olarak implemente edilmiştir ve ardışık şehirler arasındaki Öklid mesafelerini toplayarak sonucu verir.

4.3. Popülasyonun Başlatılması

- Ne Olduğu:** Algoritmanın başlangıcında, çok sayıda rastgele çözüm adayından (kromozomdan) oluşan bir başlangıç popülasyonu oluşturma işlemidir.
- Neden Gerekli:** Genetik algoritma, tek bir çözümde başlamak yerine bir çözüm havuzu (popülasyon) üzerinden çalışır. Bu, çözüm uzayının daha geniş bir alanını keşfetmeye ve daha

iyi sonuçlar bulma şansını artırmaya olanak tanır. `create_initial_population` fonksiyonu bu işlemi gerçekleştirir.

4.4. Seçim Mekanizması: Turnuva Seçimi (Tournament Selection)

- **Ne Olduğu:** Popülasyondan, bir sonraki nesli oluşturmak üzere "ebeveyn" olacak bireyleri seçme yöntemidir. **Turnuva Seçimi** yönteminde, popülasyondan rastgele k adet birey seçilir ve bunlar arasından en iyi uygunluk değerine (en düşük mesafeye) sahip olan birey ebeveyn olarak seçilir.
- **Neden Seçildi:** Bu yöntem oldukça etkilidir çünkü k parametresi ile **seçim baskısı (selection pressure)** ayarlanabilir. k değeri arttıkça daha iyi bireylerin seçilme şansı artar (yakınsama hızlanır), k değeri düştükçe daha zayıf bireylerin de şansı olur (genetik çeşitlilik korunur). Bu esneklik, `tournament_selection` fonksiyonu ile sağlanmıştır.

4.5. Çaprazlama Operatörü: Sıralı Çaprazlama (Order Crossover - OX1)

- **Ne Olduğu:** İki ebeveyn kromozomdan genetik bilgileri birleştirerek yeni "çocuk" kromozomlar üreten bir operatördür. Permütasyon tabanlı kromozomlar için özel olan **Sıralı Çaprazlama (OX1)** kullanılmıştır. Bu yöntemde, birinci ebeveynden rastgele bir alt dizi alınır ve doğrudan çocuğa kopyalanır. Çocuğun kalan genleri ise ikinci ebeveynden, mevcut genlerle çakışmayacak şekilde sırayla alınarak doldurulur.
- **Neden Seçildi:** OX1, ebeveynlerin genetik bilgisini (özellikle şehirlerin göreceli sıralamasını) korurken, permütasyon yapısını bozmadan geçerli yeni turlar (çocuklar) üretir. Bu, TSP gibi sıralama problemlerinde çok önemlidir. Bu operatör, `order_crossover_ox1` fonksiyonunda implemente edilmiştir.

4.6. Mutasyon Operatörü: Tersine Çevirme Mutasyonu (Inversion Mutation)

- **Ne Olduğu:** Bir kromozom üzerinde küçük, rastgele değişiklikler yaparak genetik çeşitliliği artıran bir operatördür. **Tersine Çevirme Mutasyonu** yönteminde, bir bireydeki genlerin rastgele seçilen bir alt dizisi (örneğin, 3. ve 7. şehirler arasındaki rota parçası) tersine çevrilir.
- **Neden Seçildi:** Bu mutasyon türü, kromozomun geçerliliğini (her şehrin bir kez bulunması kuralını) bozmadır. Sadece küçük bir rota parçasını tersine çevirerek yeni ama yapısal olarak benzer rotalar oluşturur. Bu, algoritmanın yerel optimumlardan kaçmasına ve çözüm uzayında yeni bölgeler keşfetmesine yardımcı olur. `inversion_mutation` fonksiyonu, bu işlemi belirli bir mutasyon olasılığına bağlı olarak uygular.

4.7. Yerine Koyma Stratejisi: Elitizm

- **Ne Olduğu:** Yeni bir jenerasyonun nasıl oluşturulacağını belirleyen stratejidir. Bu projede **Elitizm ile Güçlendirilmiş Jenerasyonel Değişim** stratejisi benimsenmiştir. Her jenerasyonun sonunda, mevcut jenerasyonun en iyi bireylerinden belirli bir orandaki kısmı (`elite_rate` ile belirlenen) doğrudan yeni jenerasyona aktarılır. Geri kalan popülasyon ise seçim, çaprazlama ve mutasyon adımları ile üretilen yeni çocuklardan oluşur.
- **Neden Seçildi:** Elitizm, çok güçlü bir tekniktir çünkü evrim süreci boyunca bulunan en iyi çözümün asla kaybolmamasını garanti eder. Bu, algoritmanın yakınsama sürecini hızlandırır ve kararlılığını artırır.

5- Kod Yapısı ve Fonksiyonların Açıklamaları

Proje, modüler bir yaklaşımla tasarlanmış ve her bir görevi yerine getiren özel fonksiyonlardan oluşmuştur.

5.1. Yardımcı Fonksiyonlar

- **calculate_distance(city1_coords, city2_coords)**: İki şehrin koordinatlarını alarak aralarındaki Öklid mesafesini hesaplar.
- **total_distance(tour, cities_coords)**: Bir tur (şehir listesi) ve şehir koordinatlarını alarak turun toplam mesafesini hesaplar. Bu, GA'daki uygunluk fonksiyonudur.
- **read_tsp_file(file_path)**: TSPLIB formatına benzer yapıdaki bir .txt dosyasını okuyarak şehirlerin koordinatlarını bir sözlük (dictionary) yapısında döndürür.

5.2. Genetik Algoritma Bileşenleri

- **create_individual(num_cities)**: Rastgele karıştırılmış bir şehir listesi (birey/kromozom) oluşturur.
- **create_initial_population(num_cities, population_size)**: Belirtilen popülasyon büyüklüğü kadar rastgele birey oluşturarak başlangıç popülasyonunu meydana getirir.
- **tournament_selection(...)**: Turnuva seçilimini uygular.
- **order_crossover_ox1(parent1, parent2)**: İki ebeveynden iki yeni çocuk üretmek için sıralı çaprazlamayı uygular.
- **inversion_mutation(individual, mutation_rate)**: Bir bireye tersine çevirme mutasyonunu uygular.

5.3. Ana Algoritma ve Deney Fonksiyonları

- **genetic_algorithm_tsp(...)**: Tüm GA bileşenlerini bir araya getiren ana fonksiyondur. Belirtilen parametrelerle popülasyonu başlatır, jenerasyonlar boyunca evrimsel süreci yönetir (seçilim, çaprazlama, mutasyon, elitizm), en iyi çözümü takip eder ve sonuçları (en iyi tur, mesafe, süre, yakınsama verisi) döndürür.
- **parameter_tuning_experiment(cities_coords, problem_name)**: Projenin en önemli fonksiyonlarından biridir. param_grid içinde tanımlanan farklı GA parametrelerinin tüm olası kombinasyonlarını sistematik olarak dener. Her kombinasyon için genetic_algorithm_tsp fonksiyonunu çalıştırır, sonuçları karşılaştırır ve en iyi sonucu veren parametre setini raporlar.

5.4. Görselleştirme Fonksiyonu

- **plot_results_combined_animated(...)**: matplotlib ve networkx kullanarak iki panelli bir sonuç grafiği çizer.
 - **Sol Panel**: Jenerasyonlar ilerledikçe en iyi mesafenin nasıl düştüğünü gösteren **yakınsama grafiği**.
 - **Sağ Panel**: Bulunan en iyi rotanın şehirler üzerinde çizildiği **tur grafiği**.

6- Performans Analizi ve Sonuçlar

Algoritmanın performansı dört ana kritere göre analiz edilmek üzere tasarlanmıştır:

Yakınsama Analizi: `plot_results_combined_animated` fonksiyonu tarafından çizilen "Jenerasyon vs. Toplam Mesafe" grafiği, algoritmanın ne kadar hızlı bir şekilde iyi çözümlere yakınsadığını ve gelişiminin zamanla durma noktasına gelip gelmediğini görsel olarak analiz etme imkanı tanır.

Çözüm Kalitesi: `parameter_tuning_experiment` fonksiyonu, çok sayıda farklı parametre kombinasyonunu deneyerek algoritmanın ulaşabileceği en iyi çözümü (en kısa mesafeyi) bulmaya çalışır.

Hesaplama Verimliliği: `genetic_algorithm_tsp` fonksiyonu, her bir çalıştırmanın ne kadar sürdüğünü `time` modülü ile ölçer. Bu, farklı parametrelerin (özellikle popülasyon büyüklüğü ve jenerasyon sayısı) algoritmanın çalışma hızını nasıl etkilediğini analiz etmemizi sağlar.

Parametre Duyarlılık Analizi: `parameter_tuning_experiment` fonksiyonu, hangi parametrelerin çözüm kalitesi üzerinde daha etkili olduğunu anlamak için kapsamlı bir analiz sunar.

7- Sonuç ve Değerlendirme

Bu projede, Gezgin Satıcı Problemi'nin çözümü için kapsamlı bir Genetik Algoritma aracı başarıyla geliştirilmiştir. Projenin öne çıkan özellikleri şunlardır:



Sağlam GA Mimarisi: Permütasyon temsili, turnuva seçilimi, sıralı çaprazlama ve tersine çevirme mutasyonu gibi TSP için kanıtlanmış ve etkili yöntemler kullanılmıştır. Elitizm stratejisi, en iyi çözümlerin korunmasını sağlayarak performansı artırmıştır.

Sistemik Optimizasyon: Manuel deneme-yanılma yerine, **Izgara Arama (Grid Search)** tabanlı `parameter_tuning_experiment` fonksiyonu, algoritmanın belirli bir problem için en iyi hiperparametre ayarlarını otomatik olarak bulmasını sağlar.

Kapsamlı Görselleştirme: Hem algoritmanın içsel yakınsama dinamiğini hem de nihai rota çıktısını tek bir birleşik grafikte sunan `plot_results_combined_animated` fonksiyonu, sonuçların kolayca yorumlanmasına olanak tanır.

Sonuç olarak, geliştirilen program, TSP'ye Genetik Algoritma yaklaşımının nasıl uygulanacağını, performansının nasıl ölçüleceğini ve parametrelerinin nasıl optimize edileceğini gösteren bütüncül bir çözüm sunmaktadır. Bu araç, farklı TSP problem örnekleri üzerinde çalıştırılarak çeşitli senaryolar için en uygun rota ve en etkili GA parametre setlerini bulma yeteneğine sahiptir.

8- Grup Üyelerinin Katkıları

- 9-  Ekrem Baş: Algoritmanın temel kodu,, tuning yapısı , Deneylerin yürütülmesi
- 10-  Ayman Saeid: Rapor yazımı, görselleştirme, sunum hazırlanması