1 Gezgin Satıcı Problemi (Traveling Salesman Problem, TSP) – Giriş

1.1 TSP Probleminin Tanımı

Gezgin Satıcı Problemi (TSP), bir satıcının belirli bir şehirden başlayarak **her bir şehri yalnızca bir kez** ziyaret ettikten sonra **tekrar başlangıç noktasına** döndüğü **en kısa rotayı** bulma problemidir. Başka bir deyişle, TSP, verilen n adet şehir (düğüm) ve bu şehirler arasındaki mesafeler veya maliyetlerle tanımlanan bir ağ içerisinde, toplam yol uzunluğunu (veya maliyeti) minimize eden **kapalı bir tur** (rota) arayışıdır.

Matematiksel bakımdan, TSP şehirlerin düğümler; şehirlerarası yolların kenarlar olarak temsil edildiği bir çizge üzerinde **en kısa Hamilton döngüsünü** (her düğümü tam bir kez ziyaret edip başlangıç noktasına dönen yol) bulmayı hedefler. TSP farklı biçimlerde matematiksel olarak modellenebilir. Örneğin bir *tamsayılı doğrusal programlama* (integer linear programming) problemi şeklinde ifade edilmesi yaygındır ve bu kapsamda **Miller–Tucker–Zemlin (MTZ)** ile **Dantzig–Fulkerson–Johnson (DFJ)** gibi formülasyonlar öne çıkar.

Bu modellerde, her şehrin sadece bir defa ziyaret edilmesi ve başlangıç noktasına geri dönülmesi, karar değişkenleri ve kısıtlar yardımıyla sağlanır. MTZ ve DFJ formülasyonları, özellikle **alt turları** engelleyen kısıtlar sayesinde tek bir Hamilton turu elde edilmesini garanti eder. Dolayısıyla TSP, hem sezgisel bir rota optimizasyonu yaklaşımıyla anlaşılabilir hem de katı matematiksel kısıtlarla modellenebilen bir problemdir.

1.2 TSP'nin Önemi

TSP, **hesaplama kuramında** ve **optimizasyon** alanında önemli bir yere sahip olup, **NP-zor (NP-hard)** problemler sınıfında bulunur. Dolayısıyla şehir sayısı arttıkça (özellikle büyük n değerlerinde) tam çözüm bulmak üstel (exponential) karmaşıklığa yol açtığından **pratik olarak imkânsız** hale gelebilir. Bunun yanı sıra TSP, **P = NP?** gibi büyük tartışmaların odak noktalarından biri olarak **teorik bilgisayar bilimi** açısından da merkezi bir konumdadır.

TSP'nin bir diğer önemli noktası, **araştırma ve karşılaştırma** çalışmalarında bir mihenk taşı (benchmark) görevi görmesidir. Özellikle **yöneylem araştırması** ve **meta-sezgisel (metaheuristic) algoritmalar** alanında, yeni geliştirilen yöntemlerin test edilmesi için ilk başvurulan klasik problemlerdendir. TSP ayrıca, **kombinatoryal optimizasyon** ve **graf teorisi** konularında önemli bir referans noktasıdır.

1.3 Gerçek Hayattaki Uygulamaları

TSP, soyut tanımına rağmen çeşitli sektörlerde ve gerçek hayattaki uygulamalarda sıkça karşımıza çıkar. Aşağıda bunlardan bazıları özetlenmiştir:

- **Lojistik ve Ulaşım:** Kargo veya teslimat şirketlerinin bir depodan çıkıp farklı müşteri noktalarına uğrayıp tekrar depoya dönmesini planlama. Servis araçları, otobüsler veya benzeri taşıtların durak rotalarının optimize edilmesi.

- **Devre Üretimi (Mikroçip ve PCB Tasarımı):** PCB üretiminde, matkap makinesinin delme sıralamasını belirleme. Mikroçip wafer'larında işlem noktalarının sıralanması. Bu sayede üretim süresi ve makine ayar değişiklikleri minimize edilir.

- **Robotik ve Otomasyon:** Otonom robotların depo içindeki raflardan ürün toplama rotası veya boya/bakım robotlarının farklı noktalara uğrayıp başlangıç pozisyonuna dönmesi.

- **Biyoinformatik (DNA Dizileme):** DNA fragmanlarının örtüşme oranlarına dayalı olarak doğru sıralamasını bulma (sequence assembly). Büyük genetik verileri sıralarken, TSP benzeri yöntemlerle en uyumlu sıralama hedeflenir.

2 Literatürde TSP İçin Yapılan Bazı Araştırmalar

2.1 TSP Problemi İçin Yeni Bir Kaos Serçesi Arama Algoritması (*A Novel Chaos Sparrow Search Algorithm for TSP Problem*)

Bu çalışmada, Gezgin Satıcı Problemi için geliştirilen **Yeni Kaos Serçesi Arama Algoritması (NCSSA)** tanıtılmıştır. NCSSA, *kaotik başlangıç*, gelişmiş konum güncelleme ve çeşitli mutasyon stratejileri ile yerel optimumlardan kaçınmayı hedeflemektedir. TSPLIB verileriyle yapılan karşılaştırmalarda, NCSSA hem **çözüm doğruluğu** hem de **hız** açısından ACO ve DBA algoritmalarına üstünlük sağlamıştır. Yöntem, gerçek dünya uygulamaları için etkili bir optimizasyon aracı olarak öne çıkmaktadır.

(İlgili IEEE Xplore Kaynağı için bkz. [1])

2.2 TSP'yi Maksimize Etme Uygulamalarına Çözüm Olarak En İyi Hamilton Döngüsünü Bulmak (*Finding the Best Hamiltonian Cycle as a Solution to Applications of Maximizing the TSP*)

Bu çalışma, **Max TSP** için geliştirilen **SAMA algoritmasını** tanıtmaktadır. SAMA, tam ve ağırlıklı grafiklerde optimal veya optimal yakın **Hamilton döngüsü** bulmayı amaçlayan, sistematik ve verimli bir yaklaşımdır. K6 grafiği üzerindeki deneyler, SAMA'nın geleneksel yöntemlere kıyasla **daha kaliteli ve hızlı** çözümler ürettiğini göstermektedir. Böylece Max TSP gibi **NP-zor** problemler için etkili bir alternatif sunulur.

(İlgili IEEE Xplore Kaynağı için bkz. [2])

2.3 Genetik Algoritma Tabanlı TSP Algoritması (*Genetic Algorithm-based TSP Algorithm*)

Burada, Gezgin Satıcı Problemi'nin çözümünde **genetik algoritmanın**, geleneksel dinamik programlama yöntemine göre **daha kısa rotalar** ve **daha yüksek verimlilik** sağladığı gösterilmektedir. 31 şehirli veri seti üzerinde yapılan deneylerde, genetik algoritma daha **kısa**

ve çakışmasız çözümler üretmiştir. Bu bulgular, GA'nın TSP gibi **NP-tam** problemler için önemli bir alternatif olduğuna işaret etmektedir.

(İlgili IEEE Xplore Kaynağı için bkz. [3])

2.4 Sezgisel Tekniklerin Karşılaştırılması: TSP Örneği (*Comparison of Heuristic Techniques: A Case of TSP*)

Bu çalışma, TSP çözümünde yaygın olarak kullanılan üç sezgisel algoritmayı – **Genetik Algoritma (GA)**, **Uyum Arama Algoritması (HSA)** ve **Firefly Algoritması (FA)** – karşılaştırmaktadır. Deneysel sonuçlar:

- FA'nın küçük ölçekli TSP problemlerinde başarılı olduğunu,
- GA'nın ise büyük veri setlerinde **daha istikrarlı** ve **daha başarılı** sonuçlar verdiğini

göstermektedir. Dolayısıyla TSP çözümlerinde algoritma seçiminin **problem boyutu** ve **yapısına** göre dikkatle yapılması gerektiği ortaya konmuştur.

(İlgili IEEE Xplore Kaynağı için bkz. [4])

2.5 Uyarlanabilir Karınca Kolonisi Kümeleme Algoritması ve TSP'de Uygulaması (*An Adaptive Ant Colony Clustering Algorithm and Application in the TSP*)

Bu çalışmada, büyük ölçekli TSP sorununda çözüm süresini azaltmaya yönelik bir ön işleme yöntemi olarak **Uyarlanabilir Karınca Kolonisi Kümeleme Algoritması (ACCA)** önerilmektedir. ACCA, veri nesnelerini *karınca davranışlarını* taklit ederek kümelere ayırıp TSP problemini **daha küçük alt problemlere** bölmekte ve böylece hesaplama yükünü hafifletmektedir. Deneysel analizler, **farklı küme sayıları** denenerek **çözüm kalitesi** ve **hesaplama süresi** açısından olumlu sonuçlar elde edildiğini göstermiştir.

(İlgili IEEE Xplore Kaynağı için bkz. [5])

3 Metot

Bu çalışmada, **farklı boyutlardaki** TSP veri setlerine çözüm getirmek amacıyla çeşitli algoritmalar kullanılmıştır:

- **Küçük veri setleri** için:
  - *A Novel Chaos Sparrow Search Algorithm*
  - K-En Yakın Komşu (K-Nearest Neighbors, KNN)
- **Daha büyük veri setleri** için:
  - Greedy (Açgözlü) Algoritma
  - 2-opt Yerel Arama Algoritması

4 Sonuçlar

Farklı boyutlardaki veri setleri üzerindeki test sonuçları aşağıda özetlenmiştir:

4.1 4.1. 51 Şehir

- **Kullanılan Yöntem**: A Novel Chaos Sparrow Search Algorithm
- **Optimal Maliyet**: 474.42
- **En İyi Çözüm Rotası (ilk birkaç şehir)**:

```
41 -> 34 -> 23 -> 30 -> 12 -> 36 -> 6 -> 1 -> 25 -> 20 -> ...
```

4.2 4.2. 150 Şehir

- **Kullanılan Yöntem**: KNN (K-En Yakın Komşu) Algoritması
- **Optimal Maliyet**: 28456.91
- **Örnek Rota**:

```
[0, 45, 35, 144, 63, 55, 53, 18...]
```

4.3 4.3. 318 Şehir

- **Kullanılan Yöntem**: 2-opt (Two-Opt) Yerel Arama Algoritması
- **Elde Edilen Maliyet**: 45581.65
- **Rota Başlangıcı**:

```
0 -> 313 -> 311 -> 308 -> 312 -> 303 -> 196 -> 197 -> 193 -> ...
```

4.4 4.4. 3038 Şehir

- **Kullanılan Yöntem**: 2-opt (İki Opt) Yerel Arama Algoritması
- **Hesaplanan Maliyet**: 154932.41
- **Rotanın İlk Kısmı**:

```
[0, 1, 2, 3, 4, 9, 8, 5, 6, 7, 3017, 3022...]
```

4.5 4.5. 14051 Şehir

- **Kullanılan Yöntem**: Greedy (Açgözlü) Algoritma
- **Optimal Maliyet**: 575715.88
- **Rota Başlangıcı**:

```
[0, 42, 142, 117, 32, 269, 449, 413, 432, 460...]
```

4.6 4.6. 85900 Şehir

- **Proje Yapılandırması**: Greedy (Aç Gözlü) Algoritma
- **Hesaplanan Maliyet**: 176282696.83
- **Rota Başlangıcı**:

```
0 -> 84079 -> 84081 -> 84082 -> 42 -> 3287 -> 3288 -> 59 -> 84084 -> 76 -> 84085 -> ...
```

5 Kaynakça

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3. Baskı). MIT Press.
2. Laporte, G. (1992). The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research*, 59(2), 231–247.
3. Gutin, G., & Punnen, A. P. (2002). *The Traveling Salesman Problem and Its Variations*. Springer.
4. Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley.
5. Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Operations Research*, 2(4), 393–410.
6. Pevzner, P. A. (2000). *Computational Molecular Biology: An Algorithmic Approach*. MIT Press.

IEEE Xplore Bağlantıları:
[1] `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9701715&isnumber=9701587`
[2] `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10757196&isnumber=10757175`
[3] `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10665371&isnumber=10665312`
[4] `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9058211&isnumber=9057798`
[5] `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9933194&isnumber=9933134`