

Bursa Şehir İçi Rota Optimizasyonu: Gezgin Satıcı Problemi için Algoritma Karşılaştırması

Elif Yılmaz - 25435004004

GitHub Repository: <https://github.com/elifylmaz/TSP-with-Neighborhoods>

1 Veri Üretilimi ve Problem Modelleme

Çalışmada Bursa ili Osmangazi merkezli (40.19°N , 29.06°E) 7 km yarıçaplı bir yol ağı OSMnx kütüphanesi ile gerçek zamanlı veriler kullanılarak oluşturulmuştur. Simülasyon süreci şu teknik adımlardan oluşmaktadır:

- **Noktaların Dağılımı:** 5 km yarıçaplı dairesel bir alanda homojen dağılım sağlamak amacıyla polar koordinat sistemi (r, θ) kullanılarak 60 adet rastgele nokta üretilmiştir. Bu yöntem, noktaların merkez etrafında yoğunlaşmasını önleyerek daha gerçekçi bir kentsel yayılım sunar.
- **Bölgeleme ve Kümeleme:** Üretilen noktalar, K-Means algoritması ile 10 farklı bölgeye ayrılmıştır. Algoritma, noktaların birbirine olan Öklid uzaklıklarını minimize ederek optimum bölge merkezlerini (centroids) belirlemiştir.
- **Yol Ağı Eslestirmesi (Snapping):** Hesaplanan geometrik merkezler, NetworkX kütüphanesi kullanılarak gerçek yol ağındaki en yakın fiziksel düğümlere (nodes) yansıtılmıştır.
- **Mesafe Matrisi ve Dijkstra:** Belirlenen 10 merkez arasındaki ulaşım maliyetleri, yol ağı topolojisi üzerinde Dijkstra algoritması çalıştırılarak hesaplanmıştır. Kuş uçuşu mesafe yerine gerçek yol mesafelerinin kullanılması, modelin doğruluğunu artırmıştır.
- **Deney Tasarımı:** İstatistiksel anlamlılık için 30 bağımsız deney koşturulmuştur. Deneylerin tekrarlanabilirliği için `seed=42` parametresi ile rastgelelik kontrol altında tutulmuştur.

1.1 Algoritmaların Kurulumu

Greedy Insertion: Başlangıç düğümünden (0) başlayarak, her adımda henüz ziyaret edilmemiş düğümler arasından mevcut tura eklenmesi en az mesafe artışına neden olan düğümü seçer. Mesafe artışı, yeni düğümün tura eklendiği konumdaki iki komşu düğüm arasındaki mesafe farkı ile hesaplanır. Algoritma tüm düğümler ziyaret edilene kadar devam eder.

OR-Tools: Google'ın Routing kütüphanesi kullanılmıştır. Mesafe matrisi transit callback fonksiyonu ile modele aktarılmış, en ucuz yol stratejisi ile başlangıç çözümü oluşturulmuştur. 10 saniyelik zaman limiti içinde dallandır-sınırla ve yerel arama ile optimal veya optimale yakın çözüm aranmıştır.

Genetik Algoritma: 100 birimlik popülasyonda her birey rastgele sıralanmış şehir listesi olarak başlatılmıştır. Her nesilde fitness değerleri (toplam tur uzunluğu) hesaplanmış, en iyi 10 birey doğrudan sonraki nesle aktarılmıştır (elitizm). Yeni bireyler için turnuva seçimi (5 birey arasından en iyisi) ile ebeveynler seçilmiş, Ordered Crossover (OX1) ile çocuk üretilmiş ve %2 olasılıkla Swap mutasyonu uygulanmıştır. 200 nesil boyunca bu işlem tekrarlanmıştır.

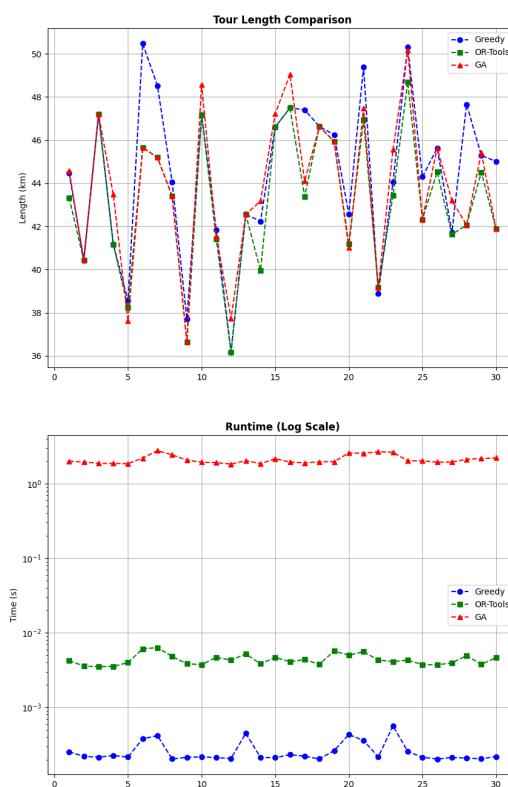
2 Deneysel Sonuçlar

2.1 Genel Performans

Tablo 1: Algoritma Performans Analizi (30 Deney Ortalaması)

Algoritma	Ort. Uzunluk (km)	Ort. Süre (s)	En İyi Sonuç	Başarı Oranı
Greedy Insertion	44.39	0.00026	1/30	3.3%
OR-Tools	43.17	0.00448	27/30	90.0%
Genetik Algoritma	43.82	2.07000	2/30	6.7%

Tablo 1, 30 deney boyunca algoritmaların performansını özetlemektedir. OR-Tools, %90 başarı oranı ve 43.17 km ortalama tur uzunluğu ile açık ara en etkili çözümü olmuştur. Greedy algoritması 0.26 milisaniye gibi çok kısa sürede çalışmasına rağmen, %2.8 daha uzun rotalar üretmiştir. Genetik Algoritma ise OR-Tools'a göre 460 kat daha yavaş çalışmış, ancak iki deneyde en iyi sonucu vermiştir.



Şekil 1: 30 Deney Boyunca Performans Dağılımı

Şekil 1'deki üst grafik, OR-Tools'un (yeşil kareler) neredeyse tüm deneylerde en kısa rotayı bulduğunu göstermektedir. Greedy (mavi daireler) genellikle %3-5 daha uzun rotalar üretirken, Genetik Algoritma (kırmızı üçgenler) ikisi arasında değişken performans sergilemiştir. Alt grafik, çalışma sürelerini logaritmik ölçekte göstermektede ve algoritmaların zaman maliyetindeki büyük farkları ortaya koymaktadır.

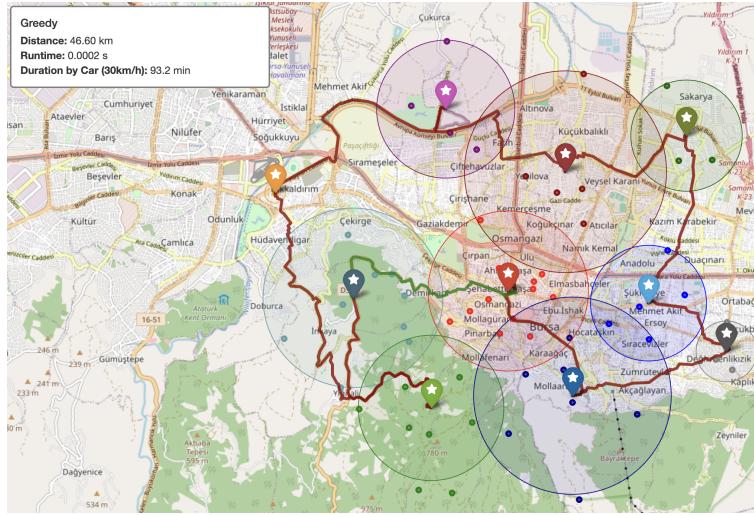
2.2 Kritik Deneylerin Detaylı Analizi

Deney 5 - Genetik Algoritmanın Zaferi: Greedy 38.55 km, OR-Tools 38.24 km, GA **37.61 km**. Genetik Algoritma, OR-Tools'tan 0.63 km (%1.6) daha kısa rota bulmuştur. Bu, GA'nın rastgele mutasyonları sayesinde yerel optimumlardan kaçabildiğini gösterir.

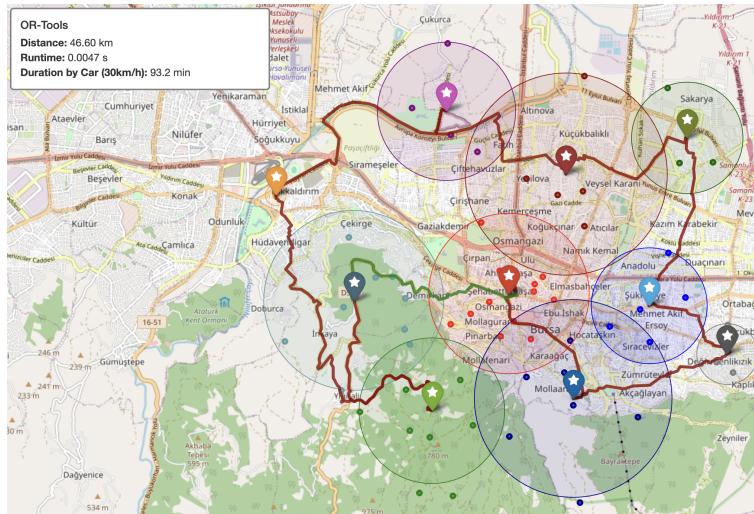
Deney 20 - Yakın Yarış: Greedy 42.56 km, OR-Tools 41.18 km, GA **41.02 km**. Genetik Algoritma 2.59 saniyede, OR-Tools'u sadece 0.16 km farkla geçmiştir. Bu, hesaplama süresinin bazen karşılığını verebildiğini göstermektedir.

Deney 22 - Greedy'nin Sürprizi: Greedy **38.89 km**, OR-Tools 39.15 km, GA 39.15 km. Bu istisnai durumda, basit açgözlü algoritma her iki sofistike yöntemi de geçmiştir. Şehirlerin geometrik düzeni, "en yakına git" mantığının optimal sonuç vermesini sağlamıştır.

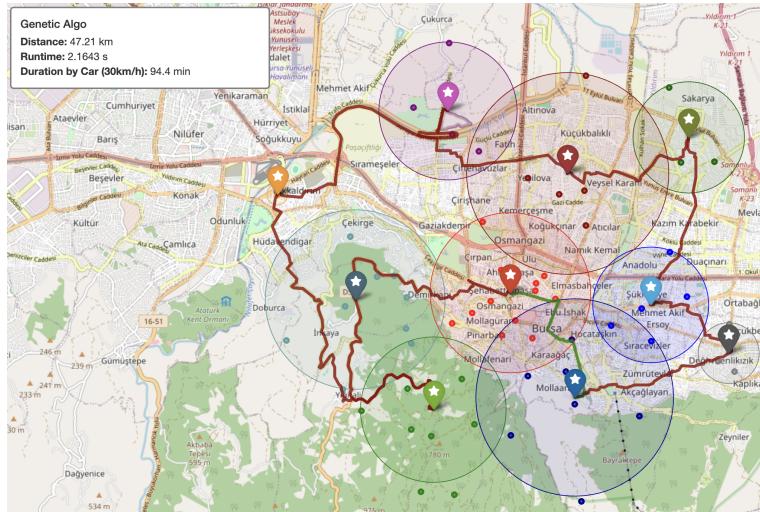
Deney 28 - En Büyük Fark: Greedy 47.66 km (%13.4 daha uzun), OR-Tools ve GA **42.07 km** (eşitlik). Bu deney, Greedy'nin yerel optimuma takıldığı, ancak GA'nın evrimsel araması sayesinde global optimuma ulaşabildiği durumu örnekler.



Şekil 2: Deney 15 - Greedy: 46.60 km



Şekil 3: Deney 15 - OR-Tools: 46.60 km



Şekil 4: Deney 15 - Genetik Algoritma: 47.21 km

Algoritma Rotaları. Renkli daireler kümelenmiş müşteri noktalarını, yıldızlar bölge merkezlerini, kırmızı çizgiler hesaplanan rotayı göstermektedir.

3 Sonuç

Bu kapsamlı karşılaştırma, gerçek dünya rota optimizasyonu problemleri için önemli çıkarımlar sunmaktadır:

OR-Tools, %90 başarı oranı ve 43.17 km ortalama tur uzunluğu ile endüstriyel uygulamalar için en güvenilir seçimdir. Milisaniyeler mertebesindeki çalışma süresi, gerçek zamanlı sistemler için idealdir.

Genetik Algoritma, nadiren (%6.7) OR-Tools'u geçebilmiş ve önemli hesaplama maliyeti (2+ saniye) gerektirmiştir. Ancak çok karmaşık problemlerde veya özel kısıtlar olduğunda, esnekliği nedeniyle avantajlı olabilir.

Greedy Insertion, hızına rağmen (%3.3 başarı oranı) güvenilir sonuçlar üretmemiştir. Sadece ön hesaplama veya başlangıç çözümü için kullanılabilir.