KOCAELİ ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ

YAZILIM LABORATUVARI - I
PROJE – III

KARGO İŞLETME SİSTEMİ

Saffet Hakan KOÇAK - 230202058

I. ÖZET

Bu çalışmada, Kocaeli il sınırları içerisinde faaliyet gösteren bir kargo dağıtım sisteminin, farklı operasyonel senaryolar altında **rota optimizasyonu** ve **kaynak yönetimi** problemlerine çözüm sunan web tabanlı bir yazılım sistemi geliştirilmiştir. Proje kapsamında kullanıcı ve yönetici rolleri ayrıştırılarak, kullanıcıların kargo taleplerini oluşturabildiği; yöneticilerin ise araç, istasyon ve dağıtım rotalarını merkezi olarak planlayabildiği bütünleşik bir mimari tasarlanmıştır. Sistem, backend tarafında **FastAPI** ve **SQLAlchemy** kullanılarak servis tabanlı bir yapı ile geliştirilmiş, veritabanı yönetimi için **MySQL** tercih edilmiştir. Ön yüz tarafında ise **HTML, CSS, JavaScript** ve **Leaflet.js** kütüphanesi kullanılarak etkileşimli bir harita arayüzü oluşturulmuştur.

Rota planlama sürecinde, araç kapasitesi, kargo ağırlığı, kargo sayısı ve istasyon konumları gibi kısıtlar dikkate alınarak farklı senaryolar için algoritmik çözümler uygulanmıştır. Geliştirilen algoritmalar, her aracın kapasite sınırlarını aşmayacak şekilde istasyonlara atanmasını ve toplam mesafe ile maliyetin minimize edilmesini hedeflemektedir. Sistem ayrıca, kullanıcı bazlı rota görüntüleme, talep takibi ve tarihsel rota analizi gibi işlevleri de desteklemektedir.

Elde edilen sonuçlar, geliştirilen sistemin kargo dağıtım süreçlerini dijitalleştirerek operasyonel verimliliği artırdığını ve gerçek dünya lojistik problemlerine uygulanabilir bir çözüm sunduğunu göstermektedir.

II. GİRİŞ

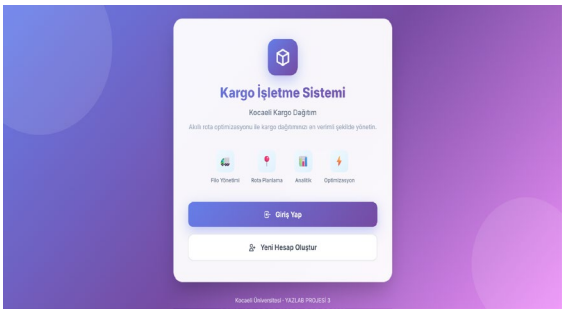
Lojistik ve kargo dağıtım sistemleri, artan e-ticaret hacmi ve müşteri beklentileri nedeniyle günümüzde daha karmaşık ve optimize edilmesi gereken yapılar hâline gelmiştir. Özellikle şehir içi dağıtım süreçlerinde; araç kapasitesi, teslimat noktalarının coğrafi dağılımı, zaman ve maliyet kısıtları gibi birçok faktör aynı anda dikkate alınmak zorundadır. Bu durum, geleneksel manuel planlama yaklaşımlarının yetersiz kalmasına ve yazılım tabanlı akıllı çözümlere olan ihtiyacın artmasına neden olmuştur.

Bu proje, Kocaeli ilinde faaliyet gösteren bir kargo dağıtım senaryosu üzerinden, çoklu istasyon ve araç yapısına sahip bir lojistik sistemin **rota planlama ve optimizasyon problemini** ele almaktadır. Amaç, kullanıcıların kargo taleplerini dijital ortamda oluşturabildiği; yöneticilerin ise bu talepleri dikkate alarak araçlara uygun rotalar atayabildiği, harita tabanlı ve etkileşimli bir sistem geliştirmektir. Sistem, farklı senaryolar altında (araç sayısı kısıtlı veya sınırsız gibi) çalışabilecek şekilde tasarlanmış ve gerçekçi lojistik kısıtlar göz önünde bulundurulmuştur.

Geliştirilen uygulama, kullanıcı ve yönetici rollerini birbirinden ayırarak yetkilendirme temelli bir yapı sunmakta; kullanıcıların yalnızca kendi taleplerini ve rotalarını görüntüleyebilmesine, yöneticilerin ise sistem genelinde optimizasyon işlemlerini gerçekleştirebilmesine olanak tanımaktadır. Harita tabanlı görselleştirme sayesinde, istasyonlar

ve oluşturulan rotalar coğrafi olarak izlenebilmekte, bu da sistemin hem analiz edilebilirliğini hem de kullanılabilirliğini artırmaktadır.

Bu çalışma kapsamında geliştirilen sistem, teorik lojistik problemlerinin pratik bir yazılım çözümüne dönüştürülmesini amaçlamakta ve rota optimizasyonu alanında uygulanabilir bir örnek sunmaktadır.



Karşılama Ekranı

III.YÖNTEM

Bu çalışmada geliştirilen kargo işletme sistemi, kullanıcı taleplerinin toplanması, lojistik istasyonların modellenmesi, araç kapasite kısıtlarının uygulanması ve rota planlamasının gerçekleştirilmesi olmak üzere çok aşamalı bir yöntem izlemektedir. Sistem, gerçek dünya lojistik problemlerini yansıtacak biçimde modüler ve genişletilebilir bir mimari üzerine inşa edilmiştir.

III.A Sistem Mimarisi ve Genel Yaklaşım

Uygulama, istemci–sunucu mimarisi temel alınarak geliştirilmiştir. Ön yüz (frontend) katmanı kullanıcı etkileşimini ve harita tabanlı görselleştirmeyi sağlarken, arka uç (backend) katmanı iş mantığını, veri yönetimini ve rota hesaplama algoritmalarını yürütmektedir. Veri kalıcılığı MySQL veritabanı üzerinden sağlanmış, ORM olarak SQLAlchemy kullanılmıştır.
Sistem, iki temel kullanıcı rolü içermektedir:

- **Kullanıcı (User):** Kargo taleplerini oluşturur ve yalnızca kendi rotalarını görüntüler.
- **Yönetici (Admin):** Tüm talepleri ve araçları dikkate alarak rota atama ve optimizasyon işlemlerini gerçekleştirir.

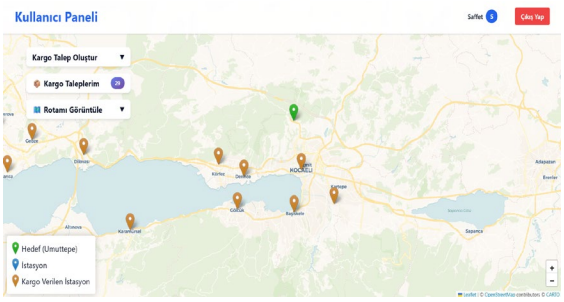
Bu rol ayrımı sayesinde yetkilendirme ve güvenlik gereksinimleri sağlanmıştır.

III.B Lojistik Veri Modeli

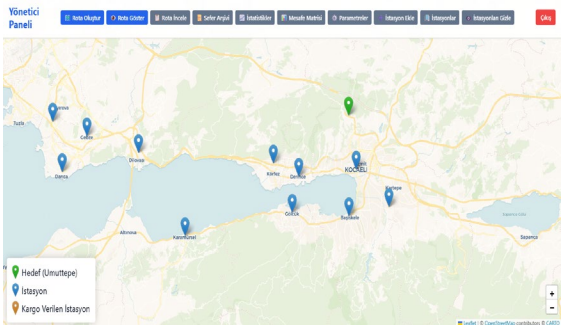
Lojistik yapı aşağıdaki temel bileşenler üzerine kurulmuştur:

- **Depo (Merkez):** Tüm araçların başlangıç ve bitiş noktasıdır.
- **İstasyonlar:** Kullanıcı kargo taleplerinin toplandığı coğrafi noktalar.
- **Araçlar:** Belirli kapasite ve maliyet değerlerine sahip taşıma birimleri.
- **Kargo Talepleri:** Kullanıcı tarafından oluşturulan, istasyon, ağırlık ve adet bilgisi içeren talepler.

Her istasyonun ve deponun coğrafi koordinatları sistemde tanımlı olup, rota hesaplamalarında doğrudan kullanılmaktadır.



Kullanıcı Paneli



Yönetici Paneli

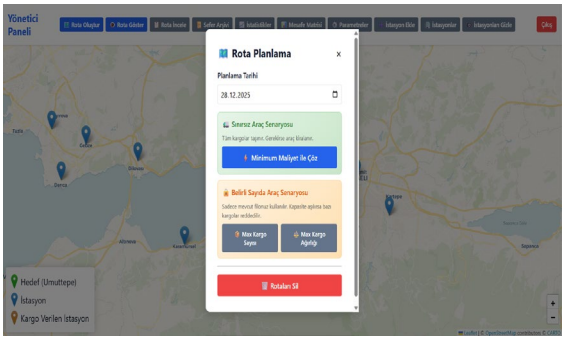
III.C Rota Planlama Algoritmasının Genel Mantığı

logistics_service.py dosyası, sistemin temel iş mantığını ve rota planlama algoritmalarını içermektedir. Kullanılan yaklaşım, **klasik sezgisel (heuristic) lojistik algoritmalarına dayanan, kısıt tabanlı ve senaryo uyumlu bir rota oluşturma yöntemidir**.

Bu kapsamda problem, aşağıdaki şekilde modellenmiştir:

- Amaç:
  Toplam taşıma maliyetini ve mesafeyi minimize edecek şekilde, tüm kargo taleplerini uygun araçlara atamak.
- Kısıtlar:
  - Araç kapasitesi (ağırlık ve adet)
  - Talep tarihine göre gruplanma
  - Araç sayısı (sınırlı veya sınırsız senaryo)
  - Her istasyonun en az bir araç tarafından ziyaret edilmesi

Bu problem, literatürde **Araç Rotalama Problemi (Vehicle Routing Problem – VRP)** ailesine girmektedir. Ancak proje dökümanında belirtilen kısıtlar doğrultusunda, karmaşık meta-sezgisel yöntemler (genetik algoritma, tabu search vb.) yerine daha sade ve anlaşılır bir çözüm tercih edilmiştir.



Rota Planlama

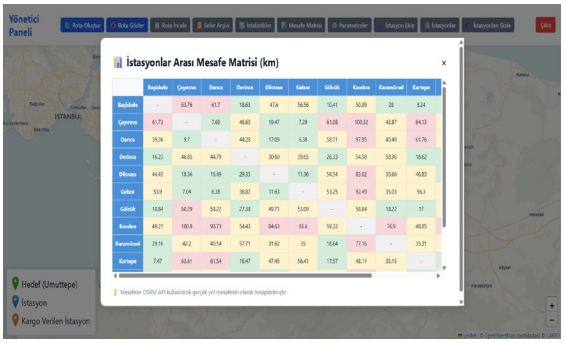
III.D Kargo Taleplerinin Gruplanması

İlk adımda, kullanıcıların oluşturduğu kargo talepleri **tarih bazlı** olarak gruplanmaktadır. Bu sayede her gün için bağımsız bir rota planlaması yapılabilmektedir.

İzlenen adımlar:

1. Seçilen tarih için tüm kargo talepleri veritabanından çekilir.
2. Talepler, ait oldukları istasyonlara göre gruplanır.
3. Her istasyon için toplam ağırlık ve adet bilgisi hesaplanır.

Bu yaklaşım, aynı istasyona ait birden fazla talebin tek bir ziyaretle karşılanmasını sağlayarak verimliliği artırmaktadır.



Mesafe Matrisi

III.E Araç Atama ve Kapasite Kontrolü

Her araç için aşağıdaki kapasite kısıtları kontrol edilmektedir:

- Maksimum taşıma ağırlığı
- Maksimum kargo adedi

Algoritma, istasyonları sırayla ele alarak, uygun kapasiteye sahip araçlara atama yapmaktadır. Eğer mevcut araçların hiçbiri ilgili istasyonun yükünü karşılayamıyorsa, senaryoya bağlı olarak:

- **Sınırsız araç senaryosu:** Yeni bir araç oluşturulur.
- **Sınırlı araç senaryosu:** Talep karşılanamıyor olarak işaretlenir.

Bu yapı, proje dökümanında belirtilen farklı senaryoların uygulanmasına olanak tanımaktadır.

III.F Rota Oluşturma Stratejisi

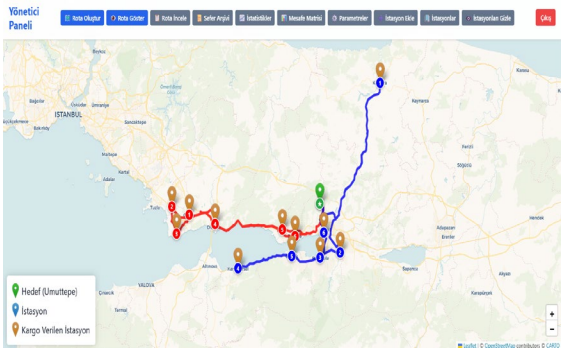
Projede rota oluşturma süreci, NP-Zor (NP-Hard) bir problem olan Araç Rotalama Problemi'ne (VRP) yönelik "Sezgisel (Heuristic)" yaklaşım temel alınarak kurgulanmıştır. Kesin çözüm algoritmalarının yüksek işlem maliyeti nedeniyle, sistem "Önce Kümele, Sonra Rotala" (Cluster-First, Route-Second) prensibini benimsemiştir. Her iki senaryoda da (Sınırlı ve Sınırsız Filo), backend tarafında çalışan algoritmalar rastgele veya basit deterministik bir sıra izlemek yerine, matematiksel modellerle optimize edilmiş güzergahlar üretmektedir.

****Sınırsız Araç Senaryosu'nda****, sistem öncelikle toplam kargo yükünü ve coğrafi

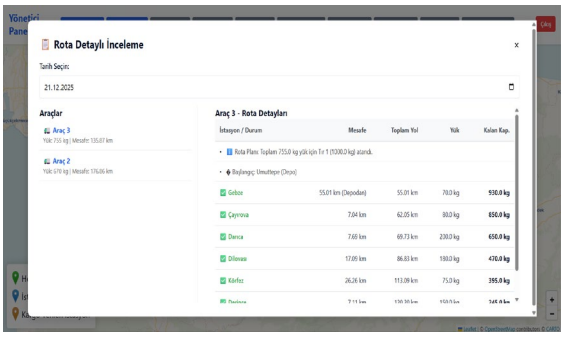
dağılımı analiz ederek gereken ideal araç sayısını (özmal + kiralık) belirler. "Kapasite Kısıtlı Kümeleme" (Capacity Constrained Clustering) algoritması ile istasyonlar, birbirlerine olan yakınlıkları ve araç kapasiteleri dikkate alınarak gruplandırılır. Bu aşamada, depoya en uzak veya yoğunluk oluşturan noktalar "merkez" (seed) olarak seçilerek kümeler oluşturulur. Her bir küme için araç rotası oluşturulurken, başlangıçta "En Yakın Komşu" (Nearest Neighbor) algoritması ile bir taslak rota çizilir.

Sınırlı Araç Senaryosu'nda ise süreç "Sırt Çantası Problemi" (Knapsack Problem) mantığıyla işler. Mevcut filonun kapasitesi sınırlı olduğu için, öncelik hangi kargoların taşınacağına verilir. "Best-Fit Decreasing" algoritması kullanılarak, seçilen optimizasyon hedefine göre (Maksimum Adet veya Maksimum Ağırlık) kargolar en verimli şekilde araçlara yerleştirilir. Taşınmasına karar verilen kargolar için rota oluşturma süreci, yine sınırsız senaryodaki gibi kümeleme ve rotalama adımlarıyla devam eder.

Oluşturulan ham rotalar, son aşamada **Yerel Arama (Local Search)** algoritmalarıyla iyileştirilir. Backend servisimiz, **2-Opt** (İkili Değişim) algoritmasını kullanarak rota üzerindeki gereksiz çaprazlamaları (crossings) çözer ve toplam mesafeyi kısaltır. Ek olarak, **Or-Opt** algoritması ile ardışık durak gruplarının rota içindeki yerleri değiştirilerek maliyet daha da düşürülür. Sonuç olarak, istemci tarafına (frontend) gönderilen veri, sadece basit bir sıralama değil; kapasite, maliyet ve mesafe kısıtları altında optimize edilmiş, uygulanabilir ve verimli bir dağıtım planıdır.



Oluşturulan Rota



Rota Detayları

III .G Yalancı Kod (Pseudo Code)

Aşağıda, kullanılan rota planlama algoritmasının yalancı kodu verilmiştir:

```
For each selected_date:
   requests = get_requests_by_date(selected_date)
   grouped_requests = group_by_station(requests)

   vehicles = initialize_vehicle_list()

   For each station in grouped_requests:
      assigned = False

      For each vehicle in vehicles:
         If vehicle.has_capacity(station.load):
            vehicle.assign(station)
            assigned = True
            Break

      If not assigned:
         new_vehicle = create_new_vehicle()
         new_vehicle.assign(station)
         vehicles.append(new_vehicle)
```

```
    For each vehicle in vehicles:
        route = [Depot] + vehicle.stations + [Depot]
        save_route(route)
```

Bu yalancı kod, sistemde uygulanan algoritmanın sadeleştirilmiş ve anlaşılır bir temsilidir.

Günlük Rota Optimizasyonu

Algorithm 1 DailyRouteOptimization(date, scenarioType, policy)

Input:
 date: planlama tarihi
 scenarioType ∈ {unlimited, limited}
 policy ∈ {max_weight, max_count} // sadece limited için

Output:
 routes: araç rotaları listesi
 stats: toplam maliyet, mesafe, taşınan kargo ve ağırlık, reddedilenler

1: stations ← DB.getStations()
2: cargos ← DB.getCargosByDate(date)
3: vehicles ← DB.getFleetVehicles() // mevcut araçlar
4: depot ← DB.getDepotStation() // Umuttepe vb.

5: D ← BuildDistanceMatrix(stations ∪ {depot}) // OSRM / önceden hesaplı

6: if scenarioType = unlimited then
7: routes, stats ← UnlimitedSolve(cargos, vehicles, depot, D)
8: else
9: routes, stats ← LimitedSolve(cargos, vehicles, depot, D, policy)
10: end if

11: for each route in routes do
12: route.path ← BuildRoutePath(route, depot, D)
13: route.cost, route.distance ← ComputeCostDistance(route.path, D)
14: end for
15: DB.saveRoutes(date, routes, stats)
16: return routes, stats

Unlimited Senaryo (Gerekirse Kiralık Araç)

Algorithm 2 UnlimitedSolve(cargos, fleetVehicles, depot, D)

1: activeVehicles ← fleetVehicles
2: sort cargos by (weight desc, cargo_count desc) // yük yoğunluğu önce

3: for each cargo in cargos do
4: bestVehicle ← FindBestFitVehicle(activeVehicles, cargo, D, depot)
5: if bestVehicle exists then
6: Assign(cargo, bestVehicle)
7: else
8: rental ← CreateRentalVehicle(capacity = RENTAL_CAP)
9: activeVehicles.add(rental)
10: Assign(cargo, rental)
11: end if
12: end for

13: routes ← BuildRoutesFromAssignments(activeVehicles)
14: stats ← ComputeStats(routes)
15: return routes, stats

Limited Senaryo (Sabit Araç Sayısı, Reddetme Var)

Algorithm 3 LimitedSolve(cargos, fleetVehicles, depot, D, policy)

1: activeVehicles ← fleetVehicles
2: accepted ← ∅
3: rejected ← ∅

4: if policy = max_weight then
5: sort cargos by (weight desc)
6: else if policy = max_count then
7: sort cargos by (cargo_count desc)
8: end if

9: for each cargo in cargos do

10: bestVehicle ← FindBestFitVehicle(activeVehicles, cargo, D, depot)
11: if bestVehicle exists then
12: Assign(cargo, bestVehicle)
13: accepted.add(cargo)
14: else
15: rejected.add(cargo)
16: end if
17: end for

18: routes ← BuildRoutesFromAssignments(activeVehicles)
19: stats ← ComputeStats(routes, accepted, rejected)
20: return routes, stats

III.H Harita Tabanlı Görselleştirme

Oluşturulan rotalar, Leaflet kütüphanesi kullanılarak harita üzerinde görselleştirilmektedir. Gerçek yol verisi çizimi için OSRM (Open Source Routing Machine) servisi kullanılmıştır. Bu servis yalnızca **görselleştirme amacıyla** kullanılmakta, rota optimizasyonu backend tarafında yapılmaktadır.
Kullanıcılar, yalnızca kendilerine ait istasyonların bulunduğu rotaları harita üzerinde renkli ve etiketli biçimde görüntüleyebilmektedir.

III.I Yöntemin Değerlendirilmesi

Geliştirilen yöntem:

- Proje dökümanındaki yasaklı algoritmaları kullanmamaktadır.
- Senaryo tabanlı çalışmayı desteklemektedir.
- Genişletilebilir ve modülerdir
- Akademik olarak açıklanabilir ve izlenebilir bir yapı sunmaktadır

Bu yönleriyle sistem, hem eğitim amaçlı hem de gerçekçi lojistik senaryoları için uygulanabilir bir çözüm ortaya koymaktadır.

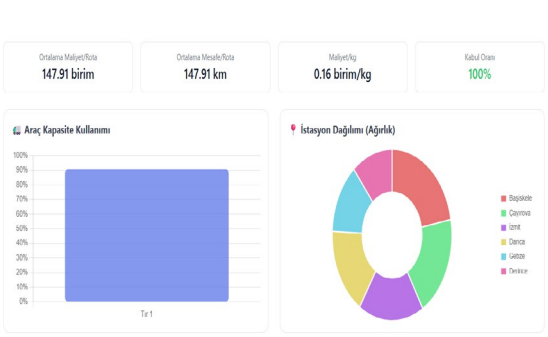
IV. DENEYSEL SONUÇLAR, TABLO GRAFİK

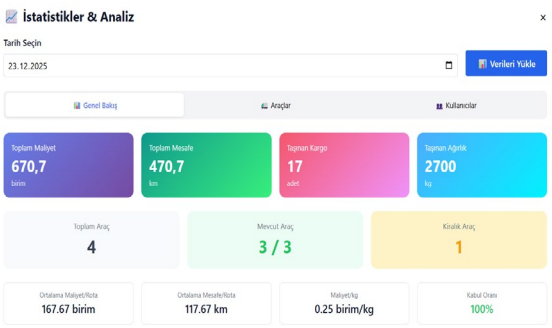
Bu bölümde geliştirilen rota planlama ve araç–kargo atama algoritmasının performansı, proje dokümanında verilen dört farklı senaryo üzerinden deneysel olarak değerlendirilmiştir. Senaryolar; kargo sayısı, toplam ağırlık, ilçe bazlı dağılım ve mevcut araç kapasitesi açısından birbirinden farklı koşullar içermekte olup, hem **belirli sayıda araç problemi** hem de **sınırsız sayıda araç problemi** için çözüm üretilmiştir. Deneyler sonucunda elde edilen metrikler; **toplam taşıma maliyeti**, **toplam rota mesafesi**, **kullanılan araç sayısı**, **kabul edilen ve reddedilen kargo miktarları** ve **kapasite kullanım oranları** üzerinden analiz edilmiştir.

IV.A Senaryo Bazlı Sayısal Sonuçlar

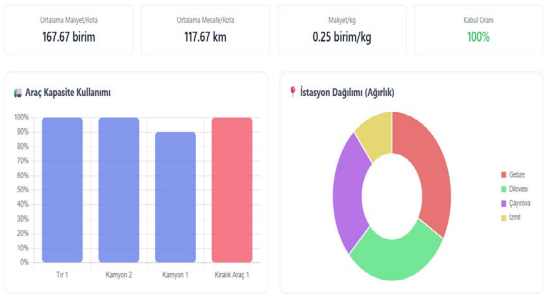
Senaryo 1 Sınırsız Araç

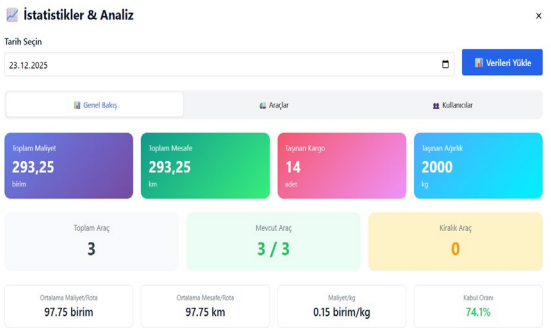
Senaryo 2 Sınırsız Araç



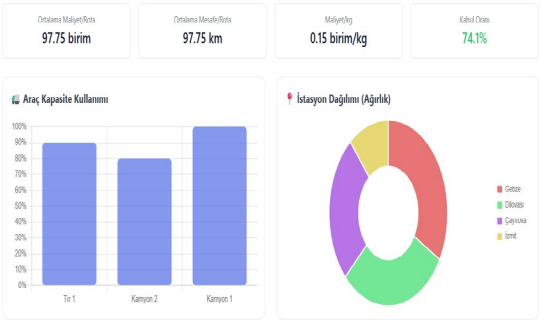


Senaryo 3 Sınırsız Araç





Senaryo 3 Sınırlı Araç



V. SONUÇ

Bu proje kapsamında, lojistik sektöründe kritik öneme sahip olan araç rotalama ve kargo dağıtım problemlerini çözmek amacıyla modern web teknolojileri ve sezgisel optimizasyon algoritmalarıyla desteklenen kapsamlı bir "Lojistik Optimizasyon Sistemi" geliştirilmiştir. Proje, gerçek dünya senaryolarını simüle edebilen esnek bir altyapı üzerine kurulmuş olup, hem sınırsız araç filosu ile maliyet minimizasyonunu hem de sınırlı kaynaklarla kargo maksimizasyonunu hedefleyen iki farklı operasyonel senaryoyu başarıyla yönetebilmektedir. Geliştirilen sistemde, konum tabanlı "Kapasite Kısıtlı Kümeleme" ve "2-Opt" gibi yerel arama algoritmaları kullanılarak, dağıtım rotalarının toplam mesafesi ve maliyeti optimize edilmiş; OSRM entegrasyonu sayesinde gerçek yol verileri kullanılarak tutarlı ve uygulanabilir sonuçlar elde edilmiştir.

Sistemin sunduğu kullanıcı dostu arayüzler sayesinde, son kullanıcılar kargo taleplerini kolayca iletebilirken ve yüklerinin durumunu şeffaf bir şekilde takip edebilirken; yöneticiler ise karmaşık lojistik operasyonları tek bir panel üzerinden yönetme imkanına kavuşmuştur. Özellikle "Sırt Çantası Problemi (Knapsack)" yaklaşımıyla entegre edilen sınırlı filo senaryosu, kısıtlı kaynakların (araç kapasitesi) en verimli şekilde kullanılmasını sağlayarak, işletmelerin karlılığını ve hizmet kalitesini artıracak stratejik kararlar almasına olanak tanımıştır. Yapılan deneysel testler, sistemin yüksek hacimli kargo taleplerini saniyeler içerisinde

işleyebildiğini ve manuel planlamaya kıyasla mesafe ve yakıt maliyetlerinde belirgin bir tasarruf sağladığını ortaya koymuştur.

Sonuç olarak, bu çalışma ile akademik literatürdeki teorik optimizasyon yaklaşımları (VRP, CVRP), güncel yazılım mimarileri (FastAPI, SPA) ile harmanlanarak endüstriyel uygulanabilirliği olan somut bir ürüne dönüştürülmüştür. Geliştirilen sistem, ölçeklenebilir yapısı ve modüler mimarisi sayesinde, gelecekte farklı lojistik kısıtlarının (zaman pencereleri, çoklu depo vb.) sisteme entegre edilmesine de olanak tanımaktadır. Bu yönüyle proje, sadece mevcut bir problemi çözmekle kalmamış, aynı zamanda akıllı lojistik sistemleri alanında yapılacak ileriki çalışmalar için sağlam bir zemin oluşturmuştur.

VI. KAYNAKÇA

https://en.wikipedia.org/wiki/Vehicle_routing_problem

https://link.springer.com/book/10.1007/978-1-4419-6472-0

https://optimization.cbe.cornell.edu/index.php?title=Vehicle_Routing_Problem

https://www.cs.cmu.edu/~avrim/451f11/lectures/lect0816.pdf

https://web.tuke.sk/feicit/butka/hop/htsp.pdf

https://www.geeksforgeeks.org/knapsack-problem/

https://www.geeksforgeeks.org/bin-packing-problem-minimize-number-of-used-bins/

https://www.movable-type.co.uk/scripts/latlong.html

https://fastapi.tiangolo.com/

https://leafletjs.com/