**KOCAELİ ÜNİVERSİTESİ PROJE RAPORU**

**BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ**

**YAZ LAB 2**

**Drone Teslimat Optimizasyonu Proje Raporu**

**1. Giriş**

Bu proje, birden fazla drone kullanarak otonom teslimat operasyonları için bir rota planlama ve optimizasyon sistemi sunmaktadır. Sistemin temel amacı, belirli kısıtlar (drone kapasitesi, batarya ömrü, uçuşa yasak bölgeler, teslimat öncelikleri) altında en verimli teslimat planını oluşturmaktır. Bu hedefe ulaşmak için **Genetik Algoritma (GA)** ve **A\* (A-Star) patika bulma algoritması** gibi yapay zeka teknikleri kullanılmıştır. Sistem, oluşturulan planın performansını analiz eder ve sonuçları görsel bir harita üzerinde sunar.

Projenin ana hedefleri şunlardır: - Teslimatların başarı oranını maksimize etmek. - Drone’ların toplam enerji tüketimini minimize etmek. - Uçuşa yasak bölgeler (no-fly zone) ve drone’ların fiziksel limitleri gibi kısıtlara uymak. - Teslimat önceliklerini ve zaman pencerelerini dikkate almak.

**2. Sistem Mimarisi ve Modüller**

Proje, her biri belirli bir sorumluluğa sahip modüler bir yapıda tasarlanmıştır. Bu yapı, kodun okunabilirliğini, bakımını ve genişletilebilirliğini artırır.

* **main.py**: Projenin ana giriş noktasıdır. Verileri yükler, optimizasyon sürecini başlatır, performans analizini tetikler ve sonuçları görselleştirir. Ayrıca farklı senaryoların test edilmesini sağlar.
* **models.py**: Sistemin temel varlıklarını (Drone, Delivery, NoFlyZone) temsil eden veri sınıflarını içerir. Bu sınıflar, varlıkların özelliklerini ve temel davranışlarını (örn. batarya tüketimi, bir noktanın yasak bölgede olup olmadığı) tanımlar.
* **data\_generator.py**: Test ve simülasyonlar için rastgele drone, teslimat ve yasak bölge verileri üreten bir yardımcı modüldür.
* **star\_algorithms.py**: A\* patika bulma algoritmasını içeren AStarPathfinder sınıfını barındırır. İki nokta arasında, yasak bölgelerden kaçınarak en kısa rotayı bulmaktan sorumludur.
* **genetic\_algorithms.py**: Teslimatların drone’lara atanmasını optimize eden GeneticAlgorithm sınıfını içerir. En iyi rota planını bulmak için evrimsel bir yaklaşım kullanır.
* **performance.py**: Optimizasyon sonuçlarını analiz eden PerformanceAnalyzer sınıfını içerir. Tamamlanma oranı, ortalama enerji tüketimi ve çalışma süresi gibi metrikleri hesaplar.
* **visualizer.py**: MapVisualizer sınıfı ile optimizasyon sonuçlarını (drone başlangıç noktaları, teslimat hedefleri, yasak bölgeler ve drone rotaları) matplotlib kullanarak bir harita üzerinde görselleştirir.

*Şekil 1: Sistem Mimarisi ve Modül Etkileşimleri*

**3. Modüllerin Detaylı Analizi**

**3.1. models.py - Veri Modelleri**

* **Drone Sınıfı**: Bir drone’u temsil eder.
* **Özellikler**: id, max\_weight (taşıma kapasitesi), battery (mAh), speed, start\_pos (başlangıç konumu).
* **Metotlar**:
* can\_carry(weight): Verilen ağırlığı taşıyıp taşıyamayacağını kontrol eder.
* battery\_consumption(distance): Kat edilen mesafeye ve mevcut yüke göre batarya tüketimini hesaplayan basit bir model içerir.
* **Delivery Sınıfı**: Bir teslimat görevini temsil eder.
* **Özellikler**: id, pos (teslimat konumu), weight, priority (öncelik seviyesi), time\_window (teslimatın yapılması gereken zaman aralığı).
* **NoFlyZone Sınıfı**: Uçuşa yasak bir bölgeyi temsil eder.
* **Özellikler**: id, coordinates (bölgeyi tanımlayan çokgenin köşe noktaları), active\_time (bölgenin aktif olduğu zaman aralığı).
* **Metotlar**:
* is\_point\_inside(point): Verilen bir noktanın, **Ray Casting Algoritması** kullanarak yasak bölge poligonu içinde olup olmadığını kontrol eder.

**3.2. star\_algorithms.py - A\* Patika Bulma**

AStarPathfinder sınıfı, bir drone’un başlangıç noktasından hedef teslimat noktasına en uygun maliyetli rotayı bulur.

* **Heuristic Fonksiyon (heuristic)**: Rota arama sürecini yönlendirmek için kullanılır. Maliyeti tahmin ederken iki bileşeni birleştirir:
* **Öklid Mesafesi**: İki nokta arasındaki en kısa düz çizgi mesafesi.
* **Ceza Puanı**: Eğer bir nokta yasak bölge içindeyse, o yola yüksek bir ceza eklenir. Bu, algoritmanın yasak bölgelerden kaçınmasını sağlar.
* **Rota Bulma (find\_path)**: Standart A\* algoritmasını uygular. Bir öncelik kuyruğu (heapq) kullanarak en umut verici yolları keşfeder. Komşu düğümleri (get\_neighbors) 8 yönde (yukarı, aşağı, sağ, sol ve çaprazlar) arar. Bir rota bulunduğunda, yol ve toplam maliyet döndürülür.
* **Maliyet Hesaplama (calculate\_path\_cost)**: Bir rotanın maliyeti, sadece mesafeye değil, aynı zamanda teslimatın özelliklerine de bağlıdır. Formül: Maliyet = (Mesafe × Ağırlık Faktörü) + Öncelik Cezası. Bu, algoritmanın ağır ve düşük öncelikli teslimatlar için daha “pahalı” yollar hesaplamasına neden olur.

**3.3. genetic\_algorithms.py - Genetik Algoritma ile Optimizasyon**

GeneticAlgorithm sınıfı, “en iyi” teslimat planını (hangi drone’un hangi teslimatları yapacağı) bulmak için evrimsel bir süreç kullanır.

* **Birey (create\_individual)**: Genetik algoritmadaki her bir “birey”, potansiyel bir çözümü, yani bir teslimat planını temsil eder. Bu projede bir birey, her bir drone’a atanmış teslimat ID’lerinin bir listesini içeren bir dictionary’dir. Örnek: {drone\_1: [teslimat\_3, teslimat\_5], drone\_2: [teslimat\_1]}.
* **Uygunluk Fonksiyonu (fitness)**: Bir bireyin (çözümün) ne kadar “iyi” olduğunu ölçer. Yüksek uygunluk puanı, daha iyi bir çözümü belirtir. Bu projede uygunluk şu şekilde hesaplanır: Uygunluk = (Tamamlanan Teslimat Sayısı × 500) - (Toplam Enerji Tüketimi × 0.1) - (Kural İhlali Sayısı × 1000) Bu formül, çok sayıda teslimatı tamamlayan, az enerji harcayan ve kuralları (örn. ağırlık limiti) ihlal etmeyen çözümleri ödüllendirir.
* **Çaprazlama (crossover)**: İki “ebeveyn” çözümden yeni bir “çocuk” çözüm oluşturur. Bu projede, her bir drone’un görev listesi rastgele olarak ebeveynlerden birinden alınarak basit bir çaprazlama uygulanır.
* **Mutasyon (mutate)**: Bir çözüme küçük, rastgele değişiklikler ekleyerek genetik çeşitliliği artırır. Bu, algoritmanın yerel optimumlara takılıp kalmasını önlemeye yardımcı olur. Burada mutasyon, rastgele bir teslimatı bir drone’dan alıp başka bir drone’a atayarak gerçekleştirilir.
* **Optimizasyon Süreci (optimize)**:
* Rastgele bireylerden oluşan bir başlangıç popülasyonu oluşturulur.
* Belirlenen nesil sayısı kadar döngü çalıştırılır.
* Her nesilde:
* Tüm bireylerin uygunluk puanları hesaplanır.
* En iyi bireyler (“elitler”) doğrudan yeni nesle aktarılır.
* Geri kalan bireyler, çaprazlama ve mutasyon yoluyla elitlerden yeniden üretilir.
* Son nesildeki en yüksek uygunluk puanına sahip birey, en iyi çözüm olarak döndürülür.

**4. Sonuç ve Gelecek Çalışmalar**

Bu proje, karmaşık kısıtlar altında drone teslimat rotalarını optimize etmek için Genetik Algoritma ve A\* algoritmasını başarılı bir şekilde birleştiren kapsamlı bir sistem sunmaktadır. Modüler mimarisi, gelecekteki geliştirmeler için sağlam bir temel oluşturur.

**Potansiyel Geliştirmeler:**

* **Dinamik Rota Değişikliği**: Gerçek zamanlı trafik veya hava durumu değişikliklerine yanıt olarak rotaların anında güncellenmesi.
* **Gelişmiş Enerji Modeli**: Rüzgar hızı, irtifa ve drone’un hızlanma/yavaşlama gibi faktörleri içeren daha gerçekçi bir batarya tüketim modeli.
* **Çoklu Teslimat Rotaları**: Bir drone’un tek bir sortide birden fazla teslimat yapıp üsse geri dönmesini optimize etme (Gezgin Satıcı Problemi varyasyonu).
* **Web Arayüzü**: Kullanıcıların teslimatları ve drone’ları interaktif bir harita üzerinden yönetebileceği bir web tabanlı kontrol paneli.
* **Makine Öğrenmesi Entegrasyonu**: Geçmiş teslimat verilerinden talep tahmini yaparak drone filosunun daha verimli konumlandırılması.