Drone Teslimat Rotalarının Optimizasyonu İçin Python Uygulaması Raporu

Ömer Hasan Gülsoy - 221307064

1. Giriş

Bu projede, birden fazla drone kullanarak teslimat noktalarına en uygun rotaların belirlenmesi amaçlanmıştır. Ayrıca, uçuş yasak bölgeleri göz önüne alınarak, güvenli ve etkili bir rota planlaması yapılmaktadır. Projede A* algoritması ile engellerden kaçınan yol bulunması ve genetik algoritma ile rota optimizasyonu gerçekleştirilmiştir.

2. Veri Modelleri

- **Ucak (Drone) Sınıfı:** Her drone için maksimum taşıma ağırlığı, batarya kapasitesi, hız ve başlangıç konumu gibi özellikler tanımlanmıştır.
- **TeslimatNoktasi:** Teslim edilecek noktaların konumu, ağırlığı, öncelik seviyesi ve teslimat için zaman penceresi bulunmaktadır.
- **UcusYasakBolgesi:** Uçuşun yasak olduğu bölgeler, koordinatları ve aktif oldukları zaman aralıklarıyla modellenmiştir.

```
class Ucak:  5 usages
    def __init__(self, idx: int, maks_agirlik: float, batarya: int, hiz: float, baslangic: Tuple[float, float]):
        self.idx = idx
        self.maks_agirlik = maks_agirlik
        self.batarya = batarya
        self.hiz = hiz
        self.baslangic = baslangic
        self.anlik_konum = baslangic
        self.kalan_batarya = batarya
        self.toplam_mesafe = 0
        self.toplam_enerji = 0
        self.tamamlanan_teslimat = 0

    def __str__(self):
        return f"Ucak {self.idx} (Max Ağırlık: {self.maks_agirlik}kg, Batarya: {self.batarya}mAh, Hız: {self.hiz}m/s)"
```

```
class TeslimatNoktasi:  9 usages
    def __init__(self, idx: int, konum: Tuple[float, float], agirlik: float, oncelik: int,
                 zaman_araligi: Tuple[datetime.time, datetime.time]):
        self.idx = idx
        self.konum = konum
        self.agirlik = agirlik
        self.oncelik = oncelik
        self.zaman_araligi = zaman_araligi
        self.teslim_edildi = False

    def __eq__(self, diger):
        return isinstance(diger, TeslimatNoktasi) and self.idx == diger.idx

    def __hash__(self):
        return hash(self.idx)

    def __str__(self):
        return f"Teslimat {self.idx} (Konum: {self.konum}, Ağırlık: {self.agirlik}kg, Öncelik: {self.oncelik})"


class UcusYasakBolgesi:  10 usages
    def __init__(self, idx: int, koordinatlar: List[Tuple[float, float]],
                 aktif_zaman: Tuple[datetime.time, datetime.time]):
        self.idx = idx
        self.koordinatlar = koordinatlar
        self.aktif_zaman = aktif_zaman

    def __str__(self):
        return f"Yasak Bölge {self.idx} (Köşe sayısı: {len(self.koordinatlar)})"
```

3. Yardımcı Fonksiyonlar

- **Mesafe Hesaplama:** İki nokta arasındaki Öklid mesafesi hesaplanır.
- **Poligon İçinde Nokta Kontrolü:** Ray casting algoritması kullanılarak bir noktanın yasak bölge içinde olup olmadığı kontrol edilir.
- **Çizgi Parçası ve Poligon Kesişim Kontrolü:** Yolun herhangi bir yasak bölgeyi kesip kesmediği belirlenir.
- **Yolun Yasak Bölgeden Geçip Geçmediği:** Hem noktaların hem de yolun yasak bölge içinden geçip geçmediği denetlenir.
- **Kenar Maliyeti ve Enerji Hesabı:** Yol maliyeti ve enerji tüketimi mesafe, hız, teslimat önceliği ve ağırlık gibi parametrelere göre hesaplanır.

4. A Algoritması*

- **Amaç:** Başlangıç ve hedef noktaları arasında yasak bölgelerden kaçınarak en uygun yolu bulmak.
- **Yöntem:** Açık ve kapalı kümeler kullanılarak, toplam maliyet ($f = g + h$) minimize edilir.
- **Heuristik:** Öklid mesafesi temel alınmıştır. Yasak bölgelerden geçiş durumunda büyük ceza puanı eklenerek alternatif yollar tercih edilir.
- **Sınırlandırma:** Maksimum iterasyon ile algoritmanın sonsuz döngüye girmesi önlenir.

5. Rota ve Genetik Algoritma

- **Rota Sınıfı:** Her drone için teslimat noktaları ve bu teslimatlar arasındaki yol detayları tutulur.

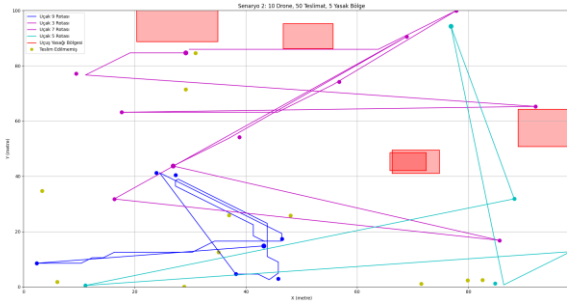
- **Kısıtlar:** Maksimum ağırlık, zaman pencereleri, batarya kapasitesi ve yasak bölge kısıtları değerlendirilir.
- **Uygunluk Fonksiyonu:** Tamamlanan teslimat sayısı, enerji tüketimi ve kural ihlalleri göz önünde bulundurulur.
- **Genetik Algoritma İşleyişi:**
    - Başlangıç popülasyonu rastgele oluşturulur.
    - Her nesilde çaprazlama ve mutasyon işlemleri ile yeni çözümler üretilir.
    - En uygun çözümler seçilerek sonraki nesle aktarılır.
    - Algoritma belirlenen nesil sayısına ulaşana kadar devam eder.

6. Veri Üretimi

- Rastgele drone, teslimat noktası ve uçuş yasak bölgeleri oluşturulur.
- Teslimat noktaları için rastgele konum, ağırlık, öncelik ve zaman aralıkları atanır.
- Yasak bölgeler genellikle dörtgen şeklinde ve günün tamamında aktif olarak belirlenmiştir.

7. Görselleştirme

- Matplotlib kütüphanesi kullanılarak drone rotaları, teslimat noktaları ve yasak bölgeler görselleştirilir.
- Farklı dronelar farklı renklerle gösterilir.
- Yasak bölgeler kırmızı, teslim edilmemiş noktalar sarı renkle vurgulanır.



8. Performans Analizi

- Tamamlanan teslimat sayısı ve yüzdesi raporlanır.
- Ortalama enerji tüketimi ve ortalama uçuş mesafesi hesaplanır.
- Her drone için ayrı ayrı teslimat sayısı, mesafe ve enerji tüketimi sunulur.
- Algoritmanın çalışma süresi kullanıcıya bildirilir.

```
def performans_analizi(rotalar: List[Rota], teslimatlar: List[TeslimatNoktasi], calisma_suresi: float):  2 usages
    teslim_edilen = set()
    for r in rotalar:
        for t in r.teslimatlar:
            teslim_edilen.add(t.idx)

    toplam_teslimat = len(teslim_edilen)
    yuzde = (toplam_teslimat / len(teslimatlar)) * 100

    gecerli_rotalar = [r for r in rotalar if r.gecerli]
    toplam_enerji = sum(r.enerji_tuketimi for r in gecerli_rotalar)
    ort_enerji = toplam_enerji / len(gecerli_rotalar) if gecerli_rotalar else 0

    toplam_mesafe = sum(r.toplam_mesafe for r in gecerli_rotalar)
    ort_mesafe = toplam_mesafe / len(gecerli_rotalar) if gecerli_rotalar else 0

    print("\nPerformans Analizi:")
    print(f"Tamamlanan teslimat sayısı: {toplam_teslimat}/{len(teslimatlar)}")
    print(f"Teslimat yüzdesi: %{yuzde:.2f}")
    print(f"Ortalama enerji tüketimi: {ort_enerji:.2f} birim")
    print(f"Ortalama mesafe: {ort_mesafe:.2f} metre")
    print(f"Algoritma çalışma süresi: {calisma_suresi:.2f} saniye")

    print("\nUçak Bazında Sonuçlar:")
    for r in rotalar:
        if r.gecerli:
            print(f"Uçak {r.ucak.idx}: {len(r.teslimatlar)} teslimat, "
                  f"{r.toplam_mesafe:.2f} metre, {r.enerji_tuketimi:.2f} enerji")

    return {
        "yuzde": yuzde,
        "ort_enerji": ort_enerji,
        "ort_mesafe": ort_mesafe,
        "calisma_suresi": calisma_suresi,
        "toplam_teslimat": toplam_teslimat
    }
```

9. Zaman Karmaşıklığı

- *A Algoritması:** O(E log V) — Kenar ve düğüm sayısına bağlıdır.
- **Genetik Algoritma:** O(G * P * D * A) — Nesil sayısı, popülasyon büyüklüğü, drone sayısı ve A* çağrı sayısına bağlıdır.
- **Kısıt Kontrolleri:** O(D * T) — Drone ve teslimat sayısına bağlıdır.
- **Toplam Karmaşıklık:** O(G * P * D * T * E log V)

```
Zaman Karmaşıklığı Analizi:
1. A* Algoritması: O(E log V), E = kenar sayısı, V = düğüm sayısı
2. Genetik Algoritma: O(G * P * D * A), G = nesil sayısı, P = popülasyon boyutu, D = drone sayısı, A = A* çağrı sayısı
3. Kısıt Kontrolleri: O(D * T), D = drone sayısı, T = teslimat sayısı
4. Toplam: O(G * P * D * T * E log V)
```

10. Sonuçlar ve Değerlendirme

Bu proje, çoklu drone kullanımı ve uçuş kısıtları göz önüne alınarak karmaşık rota planlama problemini çözmek için etkili bir yöntem

sunmaktadır. A* algoritması ile engellerden kaçınan yollar bulunurken, genetik algoritma rota optimizasyonunu sağlar. Gerçekçi zaman pencereleri ve yasak bölgeler simülasyonu, çözümün uygulamaya uygunluğunu artırmaktadır.

Kaynakça

Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th Edition). Pearson.
— Yapay zeka ve A* algoritması hakkında kapsamlı bir referans.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
— Genetik algoritmaların temel prensipleri ve uygulamaları.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.
— Robotik ve yol planlama algoritmaları üzerine detaylı bilgiler.

O'Rourke, J. (1998). *Computational Geometry in C*. Cambridge University Press.
— Geometrik hesaplamalar ve poligon işlemleri için referans.

Github:https://github.com/omer-gulsoy/Grup17_YazLab2_Drone