Drone Filosu Optimizasyonu: Yüksek Kısıtlı Ortamlarda Dinamik Teslimat Planlaması

FRANCKY RONSARD SAAH
TEKNOLOJİ FAKÜLTESİ
BİLGİ SİSTEMİ İNGİLİZCE
KOCAELİ, TÜRKİYE
francky877832@gmail.com

KHADIM DIEYE
TEKNOLOJİ FAKÜLTESİ
BİLGİ SİSTEMİ İNGİLİZCE
KOCAELİ, TÜRKİYE
Khadimd978@gmail.com

TURAN ASGARLI
TEKNOLOJİ FAKÜLTESİ
BİLGİ SİSTEMİ YÖNETİMİ
KOCAELİ, TÜRKİYE
turanaskerov2004@gmail.com

Özet- *Son yıllarda,* ***drone teslimat sistemleri*** *son kilometre lojistik zorluklarına bir çözüm olarak büyük ilgi görmüştür. Ancak,* ***gerçek dünyadaki uygulamalar*** *sınırlı* ***enerji kapasitesi****, hava sahası kısıtlamaları (uçuşa yasak bölgeler),* ***teslimat önceliği*** *ve drone kullanılabilirliği gibi çok sayıda dinamik faktör tarafından kısıtlanmaktadır. Bu makale****, A* arama algoritması, Kısıt Tatmin Problemleri (CSP) ve Genetik Algoritmaları (GA)*** *birleştirerek bu tür kısıtlamalar altında drone teslimat rotası planlaması için hibrit bir yaklaşım sunmaktadır. Çizge tabanlı bir ortam, teslimat noktalarını temsil eden dinamik olarak güncellenen düğümler ve enerji tüketimi ve teslimat önceliği için maliyet cezalarını yansıtan kenarlarla modellenmiştir. Kısıtlı bölgelerden kaçınırken teslimat düğümleri arasındaki en uygun yolları hesaplamak için A* kullanılır. Bir* ***CSP*** *modülü, oluşturulan planların temel kısıtlamalara uymasını sağlar ve birden fazla dron arasında teslimat sıralarını optimize etmek için bir GA kullanılır. Deneysel sonuçlar, bu hibrit yöntemin enerji tüketimini ve kısıt ihlallerini en aza indirirken teslimat başarı oranını etkili bir şekilde en üst düzeye çıkardığını ve kısıtlı kentsel ortamlarda otonom teslimat rotası planlaması için umut verici bir çözüm sunduğunu göstermektedir.*

Anahtar Kelimeler - *Drone Teslimatı, gerçek dünya uygulamaları, . enerji kapasitesi, teslimat önceliği, A* arama algoritması, Kısıt Tatmin Problemleri ve Genetik Algoritmalar, CSP.*

1. Giriş

Hızlı, uygun maliyetli ve otonom teslimat sistemlerine yönelik artan talep, drone tabanlı lojistik ağlarının gelişimini hızlandırmıştır. Dronlar, doğrudan hava sahasında seyrederek ve seyahat süresini önemli ölçüde azaltarak geleneksel kara taşımacılığına umut verici bir alternatif sunmaktadır. Bu potansiyele rağmen, drone'ların gerçek dünyadaki teslimat senaryolarında kullanılması bir dizi karmaşık ve dinamik kısıtlamayı beraberinde getirmektedir. Bunlar arasında sınırlı pil ömrü, ağırlık kısıtlamaları, zamana duyarlı teslimatlar ve uçuşa yasak bölgeler veya hava koşulları gibi çevresel kısıtlamalar yer almaktadır.

Teslimat görevlerini sınırlı bir drone filosuna verimli bir şekilde atamak ve rotalarını planlamak, dinamik değişkenlerle başa çıkabilen ve optimalite ile fizibilite arasında ödünleşim yapabilen akıllı algoritmalar gerektirir. Geleneksel yol bulma yöntemleri, çözüm uzayı büyük ölçüde kısıtlandığında veya zaman içinde dinamik olarak değiştiğinde zorlanabilir.

Bu makale, üç temel algoritmik strateji kullanarak kısıtlı drone teslimat rotası planlaması için hibrit bir sistem önermektedir: Yol bulma için A*, kısıtlama uygulaması için CSP ve küresel optimizasyon için GA. Sistem ilk olarak batarya, bölge ve atama kısıtlarını göz önünde bulundurarak teslimatları geçerli bir şekilde atamak için CSP'yi kullanır. Ardından A*, enerji maliyetini ve uçuşa yasak bölgeleri hesaba katarak drone'un mevcut konumundan atanan teslimat konumuna giden en uygun yolu hesaplar. Son olarak, bir genetik algoritma, çok amaçlı bir uygunluk fonksiyonuna

dayalı olarak teslimat atamalarını ve sıralarını geliştirerek genel sistem performansını iyileştirir. Amaç, enerji kullanımını ve kısıtlama ihlallerini en aza indirirken başarılı teslimat sayısını en üst düzeye çıkarmaktır.

Bu hibrit yaklaşım, teslimat noktalarını ve bunların bağlantılarını temsil eden seyrek bir grafiğe dayalı simüle edilmiş bir ortamda uygulanmış ve test edilmiştir. Deneysel kurulum, önerilen sistemin birden fazla drone ve dinamik olarak değişen kısıtlamalar içeren senaryolardaki etkinliğini değerlendirmektedir.

2. Problem Tanımı

Bu proje, çoklu dinamik kısıtlamalar altında çalışan bir drone filosu için optimum rota planlama problemini ele almaktadır. Amaç, operasyonel sınırlamalara uyarken toplam verimliliği en üst düzeye çıkaracak şekilde teslimatları atamak ve yürütmektir.

a. Amaç

Bir lojistik şirketi, otonom dronlardan oluşan bir filo kullanarak farklı ağırlık ve önceliğe sahip paketleri teslim etmeyi amaçlamaktadır. Sistem, geçerli ve verimli teslimat rotalarını hesaplamalıdır:

Tamamlanan teslimat sayısını en üst düzeye çıkarın,

Toplam enerji tüketimini en aza indirin,

Kısıtlı hava sahasından (uçuşa yasak bölgeler) kaçının,

Bireysel drone kapasite kısıtlamalarına saygı gösterin,

Zaman pencerelerine ve teslimat önceliklerine riayet edin.

b. Kısıtlamalar ve Veri Yapıları

Giriş verileri dronları, teslimatları ve uçuşa yasak bölgeleri içerir. Bunlar bir JSON senaryo dosyasından ayrıştırılan yapılandırılmış nesneler olarak temsil edilir.

Dronlar

Her bir drone şu şekilde tanımlanır:

id: Benzersiz tanımlayıcı.

max_weight: Kilogram cinsinden maksimum yük kapasitesi.

pil: mAh cinsinden pil kapasitesi.

hız: Saniyede metre cinsinden hız.

start_pos: (x, y) koordinatı olarak başlangıç konumu.

Teslimatlar

Her teslimat görevi şunları içerir:

id: Benzersiz tanımlayıcı.

pos: (x, y) koordinatı olarak teslimat konumu.

Ağırlık: Kilogram cinsinden paket ağırlığı.

öncelik: Öncelik seviyesi (1: düşük ila 5: yüksek).

time_window: Teslimat için kabul edilebilir zaman aralığı.

Uçuşa Yasak Bölgeler

Her bir kısıtlı bölge şu şekilde tanımlanır:

id: Benzersiz tanımlayıcı.

koordinatlar: Bölgeyi tanımlayan çokgen köşeleri.

aktif_zaman: Bölgenin kısıtlandığı zaman aralığı.

c. Zorluklar

Problem hem *atama* (hangi dronun hangi teslimatı gerçekleştirmesi gerektiği) hem de *yönlendirme* (hangi yolun izlenmesi gerektiği) konularını içermektedir. Başlıca zorluklar şunlardır:

Dinamik Kısıtlamalar: Belirli saatlerde etkinleşen uçuşa yasak bölgeler gibi zamana duyarlı kısıtlamalar.

Kaynak Sınırlamaları: Dronlar sınırlı pil kapasitesine sahiptir ve bir seferde yalnızca bir paket taşıyabilir.

Karmaşık Maliyet Ölçütleri: Teslimat maliyeti mesafeye, enerji kullanımına ve teslimat aciliyetine bağlıdır.

3. Algoritma Tasarımı

Rota planlama problemini çoklu kısıtlar altında çözmek için çizge arama (A*), kısıt tatmini (CSP) ve genetik algoritmaları (GA) birleştiren hibrit bir yaklaşım kullanılmaktadır. Genel mimari aşağıdaki bileşenleri içermektedir:

a. Grafik Yapımı

Teslimat alanı bir grafik olarak modellenmiştir:

Düğümler teslimat konumlarını temsil eder.

Kenarlar bu konumlar arasındaki potansiyel seyahat yollarını temsil eder.

Senaryoya bağlı olarak birden fazla grafik türü oluşturulabilir:

Tam Grafik: Her düğüm diğer tüm düğümlerle bağlantılıdır.

Seyrek Grafik: Her düğüm sınırlı sayıda komşuya bağlıdır (örneğin 3).

Yönlendirilmiş Grafikler: Bağlantılar yönlüdür ve tek yönlü hava koridorlarını simüle eder.

Kenarlar için Maliyet Fonksiyonu

İki teslimat noktası iii ve jjj arasındaki bir kenarın maliyeti şu şekilde hesaplanır:

Maliyet fonksiyonu (maliyet) = mesafe × ağırlık + (öncelik × 100)

Bu fonksiyon seyahat mesafesi, yük ağırlığı ve teslimat aciliyeti ile artar ve geciken veya verimsiz bir şekilde yönlendirilen yüksek öncelikli teslimatları cezalandırır.

b. A* Algoritması ile Rota Arama

A*, dinamik ve statik kısıtlamaları dikkate alarak grafikteki iki nokta arasındaki en kısa geçerli yolu bulmak için kullanılır.

Sezgisel İşlev

$$f(n)= g(n)+ h(n)$$

Nerede?

g(n) kaynaktan nnn düğümüne olan gerçek maliyettir,

h(n) = mesafe + nofly_zone_penalty, hedefe yönelik sezgisel tahmindir: *Öklid mesafesi, Bir yol aktif bir uçuşa yasak bölgeyle kesişiyorsa ceza.*

Algoritma, drone'un ağırlık kapasitesini aşan, drone'un sağlayabileceğinden daha fazla pil gerektiren yolları atar. Ve kısıtlı zamanlarda aktif uçuşa yasak bölgelere girenleri cezalandırır,

c. Kısıt Tatmin Problemi (CSP)

Optimizasyondan önce, bir CSP modeli dronlara herhangi bir teslimat atamasının uygulanabilir olmasını sağlar. CSP aşaması kontrol eder:

- Bir drone aynı anda yalnızca bir teslimat taşıyabilir.
- Atanan teslimatların toplam ağırlığı drone'un kapasitesini aşmaz.
- Kısıtlı alanlar cezalandırılır.
- Teslimatlar için zaman aralıklarına riayet edilir.

CSP filtreleme geçersiz başlangıç popülasyonlarının GA aşamasına aktarılmasını önler.

c. Optimizasyon için Genetik Algoritma

Geçerli bir popülasyon oluşturulduktan sonra, teslimat programlarını optimuma doğru geliştirmek için bir genetik algoritma kullanılır.

İlk Nüfus - kullanılarak oluşturulmuştur:

CSP kontrolüne dayalı rastgele geçerli atamalar.

Popülasyondaki her birey, dronlardan teslimat dizilerine bir eşlemedir.

Genetik Operatörler

Çaprazlama: Yeni yavrular oluşturmak için iki ebeveyn bireyin doğum atamalarını birleştirir.

Mutasyon: Bir teslimatı rastgele farklı bir drone'a yeniden atar veya teslimat sırasını değiştirir.

Seçim: Turnuva seçimi, bir sonraki nesil için ebeveynleri seçmek için kullanılır.

Uygunluk = (tamamlanan teslimat × 50) - (toplam enerji × 0,1) (ihlaller × 1000)

Bu fonksiyon, daha fazla teslimatı verimli bir şekilde tamamlayan bireyleri ödüllendirir ve ağırlık limitleri, zaman pencereleri veya uçuş kısıtlamaları gibi kısıtlamaları ihlal edenleri cezalandırır.

4. Proje Yapısı

```
drone/
├── algoritmalar/
│   ├── __pycache__/
```

```
│ │ ├── a_star.python-3.11.pyc
│ │ └── graph_builder.python-3.11.pyc
│ ├── a_star.py
│ └── graph_builder.py
├── varlıklar/
│ ├── a_star_algorithm_flowchart.png
│ ├── a_star_algorithm_flowchart.svg
│ ├── a_star_vs_ga_diagram.png
│ ├── a_star_vs_ga_diagram.svg
│ ├── evaluate_individual_flowchart.png
│ ├── evaluate_individual_flowchart.svg
│ ├── generate_random_flowchart.png
│ ├── generate_random_flowchart.svg
│ ├── program_workflow_flowchart.png
│ └── program_workflow_flowchart.svg
├── veri/
│ └── sample_data.txt
├── ga/
│ ├── __pycache__/
│ ├── fitness.py
│ ├── ga.py
│ └── population.py
├── grafik/
│ ├── __pycache__/
│ └── graph.py
├── modeller/
├── testler/
│ ├── test_a_star.py
│ └── test_route_planning.py
├── utils/
│ ├── __pycache__/
│ ├── constraints.py
│ ├── helpers.py
│ ├── scenario_1_generator.py
│ └── scenario_2_generator.py
│ ├── simulate_delivery.py
├── main.py
├── scenario1.json
└── scenario2.json
```

5. Proje Ana Alogirthms Açıklaması

a. a_start algoritması (algoritms/a_start.py) - *a_start fonksiyonu algoritması, mevcut teslimat için en az maliyetli yolu (en uygun yol) bulur.*

a_star işlevi, drone'a özgü kısıtlamaları ve uçuşa yasak bölgeleri göz önünde bulundurarak bir başlangıç konumundan hedef konuma en uygun yolu bulmak için kullanılan A* arama algoritmasının bir uygulamasıdır. A* algoritması, hedefe giden olası en iyi yolu tahmin etmek için gerçek hareket maliyeti ve sezgisel bir kombinasyon kullanan iyi bilinen bir yol bulma tekniğidir.

Algoritma aşağıdaki şekilde uygulanmaktadır:

Fonksiyon Girişi ve Kurulumu

Fonksiyon birkaç girdi alır:

grafik: Düğümler (konumlar) arasındaki bağlantıları ilgili maliyetleriyle birlikte temsil eden bir sözlük.

başlangıç: Drone'un başlangıç konumu.

hedef: Drone'un hedef noktası (örneğin, bir teslimat yeri).

nofly_zones: Drone'un kaçınması gereken uçuşa yasak bölgelerin bir listesi.

drone: Maksimum ağırlık kapasitesi ve pil ömrü gibi nitelikleri içeren, drone'u temsil eden bir nesne.

teslimatlar: Drone'un ziyaret etmesi gereken teslimat konumlarının bir listesi.

Algoritma, başlangıç ve hedef konumlarını çiftlere dönüştürerek başlar ve arama süreci için bir başlangıç yapılandırması oluşturur.

Sezgisel İşlev

Sezgisel fonksiyon, mevcut düğümden hedefe ulaşmak için kalan maliyetin bir tahminini hesaplar. A* algoritmasının önemli bir bileşenidir:

Öklid Mesafesi: Mevcut düğümden hedefe olan düz çizgi mesafesi euclidean_distance(n, target) kullanılarak hesaplanır.

Uçuşa Yasak Bölgeler için Ceza: Geçerli düğüm bir uçuşa yasak bölge içindeyse, ceza

apply_penality(n, target, nofly_zones) işlevi kullanılarak eklenir.

Sezgisel, mesafe ve cezanın toplamını verir ve kısıtlı alanlardan kaçınırken aramayı hedefe doğru yönlendirir.

Yolun Yeniden Yapılandırılması

Hedefe ulaşıldığında, redraw_path işlevi hedeften başlangıca kadar olan en uygun yolu yeniden oluşturmak için kullanılır:

came_from sözlüğünde saklanan üst düğümleri takip ederek hedeften başlangıca doğru yinelemeli olarak izler.

Hiçbir yol bulunamazsa (yani hedefe ulaşılamazsa), boş bir liste döndürür.

Ana Arama *Süreci

Ana arama döngüsü, başlangıç pozisyonunun open_set öncelik kuyruğuna itilmesiyle başlatılır. Kuyruk, düğümlerinin ilişkili maliyetleri (f_score), gerçek hareket maliyetleri (g_score) ve konumlarıyla birlikte depolandığı bir yığın olarak uygulanır.

Arama şu şekilde ilerler:

Düğüm Keşfi: En düşük f_score'a (tahmini toplam maliyet) sahip düğüm açık kümeden çıkarılır. Mevcut düğüm hedefle eşleşiyorsa, optimum yol yeniden oluşturulur ve döndürülür.

Komşu Keşfi: Mevcut düğümün komşuları değerlendirilir:

Her bir komşunun ağırlığı drone'un maksimum ağırlık kapasitesini aşmamalıdır.

Komşuya gidiş-dönüş için gereken batarya Öklid mesafesine göre hesaplanır ve drone'un mevcut bataryasını aşmamalıdır.

Uçuşa yasak bölgelerden geçiş için bir ceza eklenir.

Komşu henüz ziyaret edilmemişse veya ona giden daha iyi bir yol bulunursa, güncellenmiş bir maliyetle açık kümeye eklenir.

Yol Bulma Sonlandırma

Açık küme boş olursa, yani geçerli bir yol yoksa, fonksiyon uygulanabilir bir yol bulunamadığını belirten None değerini döndürür. Bir yol bulunursa, başlangıçtan hedefe giden en uygun rotayı temsil eden konumların bir listesi olarak döndürülür.

b. Graph_builder.py

Bu dosya, farklı türde grafikler oluşturmak için çeşitli işlevler içerir.

build_graph *- Tüm Noktaların Tam Grafiği*

build_graph işlevi, tüm teslimat noktalarını ve başlangıç noktasını içeren **tam** bir grafik oluşturur:

Her noktanın (başlangıç ya da teslimat) diğer tüm noktalara bağlı olduğu bir grafik oluşturur.

Çıktı: Her bir düğümün diğer tüm düğümlere bağlı olduğu ve bu düğümler arasında seyahat etmenin maliyetinin saklandığı tam bir grafik.

Bu tür bir grafik, tüm olası rotaları dikkate almanın önemli olduğu senaryolar için yararlı olabilir, ancak teslimat noktalarının sayısı arttıkça hesaplama açısından pahalı olabilir.

generate_complete_graph *- Teslimat Noktaları ile Komple Grafik*

generate_complete_graph işlevi, teslimat noktalarının **eksiksiz** bir **grafiğini** oluşturur:

Grafik, her bir teslimat noktasını diğer tüm teslimat noktalarına bağlar ve her bir çift için compute_base_cost fonksiyonunu kullanarak maliyeti hesaplar.

Çıktı: İlgili seyahat maliyetleri ile teslimat noktalarının tam bağlantılı bir grafiği.

generate_oriented_sparse_graph *- k En Yakın Komşu ile Seyrek Yönlendirilmiş Grafik*

generate_oriented_sparse_graph işlevi, her düğümün (teslimat noktası) yalnızca k en yakın komşusuna bağlandığı **seyrek** bir **yönlendirilmiş grafik** oluşturur:

Algoritma, her teslimat noktası için diğer tüm noktalara olan Öklid mesafesini hesaplar ve bunları mesafeye göre sıralar.

Daha sonra her düğüm için yalnızca en yakın k komşuyu tutar ve her düğümden en yakın k komşusuna yönlendirilmiş kenarlar oluşturur.

Çıktı: Her düğümün en yakın k komşusuna işaret eden kenarlara sahip olduğu ve ilişkili hareket maliyetleri olan seyrek, yönlendirilmiş bir grafik.

Bu yaklaşım, bağlantıları en yakın noktalarla sınırlandırarak grafiğin karmaşıklığını azaltır. Tüm olası bağlantıları tutmanın verimsiz olduğu ve en ilgili komşulara odaklanıldığı büyük veri kümeleri için uygundur.

generate_sparse_graph *- k En Yakın Komşu ile Seyrek Yönlendirilmemiş Çizge*

generate_sparse_graph işlevi **seyrek** bir **yönlendirilmemiş çizge** oluşturur:

generate_oriented_sparse_graph'a benzer şekilde, tüm noktalar arasındaki Öklid mesafesini hesaplar ve en yakın k komşuyu tutar.

Yönlendirilmiş grafiğin aksine, bu versiyon her nokta ile en yakın komşuları arasına çift yönlü (yönlendirilmemiş) kenarlar ekler. Noktalar arasındaki seyahatin her iki yönü de dikkate alınır.

Çıktı: Her düğümün k en yakın komşusuna bağlı olduğu seyrek, yönlendirilmemiş bir grafik.

Bu tür bir grafik, noktalar arasındaki ilişkiler çift yönlü olduğunda kullanışlıdır (yani, seyahat her iki yönde de aynı maliyetle mümkündür). Ayrıca sadece en yakın komşuları dikkate alarak grafiğin boyutunu küçültür.

Drone Teslimatında Uygulama

Bu grafikler, dronların teslimat noktaları arasındaki hareketini modellemek için kullanılır. Sistem, farklı grafik türlerini göz önünde bulundurarak, hesaplama verimliliği ile modelin doğruluğu arasında denge kurarak rota planlamasını optimize edebilir. Seyrek graflar özellikle büyük ölçekli ortamlarda hesaplama yükünü azaltmada yardımcı olurken, tam graflar gerektiğinde tüm olası rotaların dikkate alınmasını sağlar.

c. Evaluate_individual(ga/fitness.py)

Genetik Algoritma için Değerlendirme Fonksiyonu: evaluate_individual

evaluate_individual işlevi, drone teslimat rotalarını optimize etmek için kullanılan genetik algoritmanın önemli bir parçasıdır. Bu işlev, enerji tüketimi, kısıtlama ihlalleri ve başarılı teslimatlar gibi çeşitli faktörlere dayalı olarak belirli bir bireyin (yani her drone'a atanan belirli bir teslimat sırası) uygunluğunu hesaplar.

Amaç

evaluate_individual fonksiyonunun amacı, her bir drone için teslimat sürecini simüle ederek ve ilgili performans metriklerini hesaplayarak genetik algoritmadaki aday çözümün kalitesini değerlendirmektir. Drone tarafından kullanılan enerjiyi, ihlal sayısını (örneğin, zaman pencereleri veya kısıtlamalar nedeniyle) ve başarılı teslimat sayısını dikkate alır.

Girişler

birey: Her bir anahtarın bir drone tanımlayıcısı (örneğin, "D1", "D2") ve her bir değerin o drone'a atanan teslimatların sırasını temsil eden teslimat kimliklerinin bir listesi olduğu bir sözlük.

grafik: Düğümlerin teslimat noktaları ve kenarların bu noktalar arasındaki maliyetler (örneğin, mesafeler veya zaman) olduğu çevreyi temsil eden grafik.

no_fly_zones: Drone'un uçmasına izin verilmeyen kısıtlı bölgelerin bir listesi.

dronlar: Her biri kimliği, pili, hızı ve diğer özellikleri gibi niteliklere sahip bir drone'u temsil eden drone nesnelerinin bir listesi.

teslimatlar: Her biri teslimat noktasının konumu, ağırlığı ve önceliği gibi bilgileri içeren bir teslimat nesneleri listesi.

Süreç

Değişkenleri Başlatın:

total_energy: Tüm dronlar tarafından tüketilen toplam enerjinin kaydını tutar.

total_violations: Kısıtlama ihlallerinin sayısını izler (örneğin, drone ağırlık sınırlarını aşarsa, zaman pencerelerini ihlal ederse, vb.)

successful_deliveries: Yapılan başarılı teslimatların sayısını sayar.

Her Drone ve Teslimat Sırası Üzerinde Yineleme Yapın:

Bireyin teslimat sırasındaki her bir drone için:

Drone'u **Sıfırla**: Drone'un durumu (örn. pil, ağırlık) drone.reset() kullanılarak ilk değerlerine sıfırlanır.

Teslimatlar Üzerinde Yineleme: Drone'un sırasındaki her teslimat için:

Yol bulma: a_star algoritması, drone için mevcut konumu ile hedef (teslimat konumu) arasındaki en uygun yolu hesaplamak için kullanılır.

Maliyet Hesaplama: Yol boyunca seyahat etmenin maliyeti grafiğe göre hesaplanır.

Kısıtlama Kontrolü: check_all_csp_for_violation işlevi, drone'nun herhangi bir kısıtlamayı (örneğin pil, ağırlık veya zaman penceresi) ihlal edip etmediğini kontrol etmek için kullanılır.

Teslimat Simülasyonu: Drone teslimat konumuna hareket eder, pilini günceller ve teslimat tamamlandı olarak işaretlenir.

Şarj etme: Bir teslimatı tamamladıktan sonra drone yeniden şarj olmaya başlar ve tahmini varış süresi drone'un hızına ve kat edilen mesafeye göre hesaplanır.

Fitness Hesaplaması:

Bireyin uygunluğu şu şekilde hesaplanır:

Uygunluk = (teslimat sayısı × 50) - (toplam enerji × 0,1) - (ihlal edilen kısıtlamalar × 1000)

Başarılı teslimatların her biri 50 puan ile ödüllendirilir.

Enerji tüketimi, kullanılan enerji birimi başına 0,1 oranında cezalandırılır.

İhlaller, kısıtlamaları ihlal eden çözümleri caydırmak için yüksek bir maliyetle (ihlal başına 1000) ağır bir şekilde cezalandırılır.

Uygunluğu Geri Döndür: Drone için teslimat sırasının kalitesini temsil eden uygunluk puanı döndürülür.

Çıktılar

uygunluk: Bireyin (yani teslimat dizisinin) uygunluk puanı. Daha yüksek bir uygunluk puanı, daha başarılı teslimatlar, daha az enerji tüketimi ve daha az ihlal ile daha iyi bir çözümü gösterir.

Drone Teslimat Optimizasyonunda Uygulama

Bu değerlendirme fonksiyonu, her bir "bireyin" (teslimat dizisi) enerji verimliliği, kısıtlamalara bağlılık ve başarılı teslimat sayısı açısından ne kadar iyi performans gösterdiğine göre puanlandığı genetik algoritmanın ayrılmaz bir parçasıdır. Algoritma, dronlar için en uygun teslimat rotalarını bulmak amacıyla bireyleri seçerek, çaprazlayarak ve mutasyona uğratarak daha iyi çözümlerin evrimine rehberlik etmek için bu uygunluk puanını kullanır.

Sistem, enerji tüketimi, kısıt ihlalleri ve teslimat başarısı gibi faktörleri bir araya getirerek, ortaya çıkan rotaların yalnızca uygulanabilir değil, aynı zamanda verimli ve operasyonel kısıtlamalara (pil ömrü, ağırlık sınırları ve zaman pencereleri gibi) uygun olmasını sağlar.

d. Ga/ga.py

Çaprazlama Fonksiyonu: çaprazlama

Çaprazlama işlevi, yavruları oluşturmak için iki ebeveyn çözümünü (teslimat dizileri) birleştirir. Rastgele bir çaprazlama noktası seçer, bu noktadan sonraki teslimatları ebeveynler arasında değiştirir ve ardından yavruların teslimat dizisini yeniden oluşturur. Bu, genetik algoritmalarda yeni çözümlerin keşfedilmesine yardımcı olan bir rekombinasyon şeklidir.

Girdiler: Teslimat dizilerini temsil eden iki ana sözlük.

Çıktılar: Her biri yeni bir teslimat dizisini temsil eden iki yavru sözlük.

Mutasyon Fonksiyonu: mutate

Mutasyon işlevi, tek bir çözüme küçük değişiklikler getirir. Rastgele bir drone ve bir teslimat seçer, bir teslimatı kaldırır ve mevcut başka bir teslimatla değiştirir. Bu, popülasyon içindeki çeşitliliğin korunmasına yardımcı olur ve algoritmanın çok hızlı bir şekilde optimal olmayan çözümlere yakınsamasını önler.

Girdiler: Birey (drone atamaları sözlüğü) ve teslimat listesi.

Çıktılar: Mutasyona uğramış birey (güncellenmiş drone atamaları).

Turnuva Seçimi İşlevi: tournament_selection

Bu fonksiyon, popülasyonun rastgele seçilen bir alt kümesinden en iyi bireyi seçer. Seçim, evaluate_individual işlevi kullanılarak hesaplanan bireylerin uygunluğuna dayanır. Bu, en iyi performans gösteren bireylerin üreme için seçilme olasılığının daha yüksek olmasını sağlar.

Girdiler: Nüfus, grafik, uçuşa yasak bölgeler, dronlar, teslimatlar ve turnuva büyüklüğü.

Çıktılar: Turnuvadaki en iyi birey.

Sonraki Nesli Oluştur: generate_next_generation

Fonksiyon bir sonraki nesil bireyleri oluşturur. Ebeveynleri seçmek için turnuva seçimi kullanır, yavruları oluşturmak için çaprazlama yapar ve belirli bir olasılıkla mutasyon uygular. Yeni nesil, en iyi bireyler birleştirilerek ve çaprazlama ve mutasyon yoluyla yeni genetik materyal eklenerek oluşturulur.

Girdiler: Nüfus, grafik, uçuşa yasak bölgeler, dronlar, teslimatlar ve mutasyon olasılığı.

Çıktılar: Yeni nüfus (bir sonraki nesil bireyler).

Ga/population.py

Rastgele Birey Oluştur: generate_random_individual

Bu fonksiyon her teslimatı bir drone'a atayarak rastgele bir birey (çözüm) oluşturur. Teslimatlar karıştırılır ve her drone'a bir teslimat atanır (hiçbir drone'un birden fazla teslimat taşımaması sağlanır). İşlev, hiçbir drone'a birden fazla teslimat atanmadığından emin olmak için her drone'un atamalarını kontrol eder.

Girdiler: Dronların ve teslimatların listesi.

Çıktılar: Anahtarların drone ID'leri ve değerlerin teslimat ID'leri listesi olduğu bir sözlük olarak temsil edilen rastgele bir birey.

Rastgele Tam Birey Oluştur: generate_random_full_individual

Bu fonksiyon, bir dronun başlangıç pozisyonuna teslimat yapamayacağı ek kısıtlamasıyla rastgele bir birey oluşturur. Karıştırılan teslimatlar dronun başlangıç pozisyonuna atanmış bir teslimat içeriyorsa, fonksiyon tüm dronlara geçerli teslimatlar atanana kadar tekrar dener. Hiçbir dronun kendi başlangıç pozisyonuna teslimat yapmamasını ve tüm dronlara en az bir teslimat atanmasını sağlar.

Girdiler: Dronların ve teslimatların listesi.

Çıktılar: Her bir drone'a geçerli bir teslimat atanmasını sağlayan rastgele bir tam birey.

İlk Nüfusu Oluştur: generate_initial_population

Bu fonksiyon bireylerden oluşan bir başlangıç popülasyonu oluşturur. Rastgele bireylerden oluşan bir liste oluşturmak için generate_random_individual işlevini kullanır. Popülasyondaki bireylerin sayısı size parametresi ile tanımlanır.

Girdiler: Dronların listesi, teslimatlar ve nüfus boyutu (varsayılan değer 5'tir).

Çıktılar: Her biri olası bir çözümü temsil eden rastgele bireylerden oluşan bir liste.

İlk Tam Nüfusu Oluştur: generate_initial_full_population

Bu fonksiyon, her bireyin hiçbir dronun başlangıç konumuna teslimat yapmaması kısıtlamasına bağlı olduğu bir başlangıç birey popülasyonu oluşturur. Bireyleri oluşturmak için generate_random_full_individual işlevini kullanır. Popülasyon büyüklüğü size parametresi ile tanımlanır.

Girdiler: Dronların listesi, teslimatlar ve nüfus boyutu (varsayılan değer 5'tir).

Çıktılar: Her biri drone kısıtlamalarına uyan rastgele tam bireylerin bir listesi.

Neden generate_initial_full_population **Entegre Edilmeli?**

Algoritmanın ilk tasarımında, dronlar başlangıç konumlarıyla başlatılır ve her drona mevcut teslimatlar listesinden bir teslimat atanır. Ancak bu yaklaşım kritik bir soruna yol açmıştır: **bazı dronlara teslimatlar başladıkları noktadan atanmıştır**. Örneğin, drone 1'in delivery.pos adresinde bir başlangıç konumu varsa, zaten bulunduğu noktaya teslimat yapmaya çalışacaktır. Bir drone kendi başlangıç konumuna teslimat yapamayacağı için bu **mantıksal bir kusurdur**.

Bu sorunu çözmek ve **geçerli atamaları** sağlamak için **generate_initial_full_population işlevini** entegre ettik. Bu işlev, hiçbir drone'a başladığı yerde bir teslimat atanmadığından emin olmak için ek kontroller gerçekleştirir. Böyle bir atama gerçekleşirse, geçerli bir atama yapılana kadar yeniden dener. Sonuç olarak, başlangıç popülasyonundaki her drone'un, bir drone'un kendi başlangıç konumuna teslimat yapamayacağı kısıtlamasına bağlı kalarak **geçerli** bir teslimat **atamasına** sahip olması garanti edilir.

generate_initial_full_population **Nasıl Çalışır**?

Geçersiz Atamaların Önlenmesi: İşlev, bir drone'a atanan teslimatın başlangıç konumundan olup olmadığını kontrol eder. Bir dronun kendi başlangıç konumuna teslimat yapmak üzere olduğunu tespit ederse, teslimatları yeniden karıştırır ve atamayı yeniden dener.

Tüm Drone'lara Teslimat Atanmasını Sağlar: İşlev, her drone'a bir teslimat atanmasını ve hiçbir drone'un boşta bırakılmamasını sağlar.

Geliştirilmiş Popülasyon Geçerliliği: Bu fonksiyonla, oluşturulan popülasyon tamamen geçerli bireylerden oluşur ve evrim süreci sırasında atılması gerekebilecek geçersiz çözümler üretme riskini azaltır.

Bu yaklaşımı kullanarak, geçersiz bireylerin oluşturulduktan sonra kontrol edilmesi ve atılmasına ilişkin hesaplama yükünden kaçınıyoruz ve bu **da daha verimli** bir **genetik algoritma** sağlıyor.

e. Graphics/graphc.py

plot_combined_graph_and_path()

Bu fonksiyon, aşağıdakileri içeren **birleşik** bir **grafik** çizer:

Uçuşa yasak bölgeler: Bunlar harita üzerinde kırmızı çokgenler (şeffaf) olarak çizilir.

Teslimat noktaları: Yanlarında kimlikleri olan mavi noktalar olarak çizilmiştir.

Grafik kenarları: Teslimat noktaları arasındaki bağlantıları temsil eder.

Yol: Drone'un planlanan rotasını temsil eden, yönü belirtmek için oklarla mavi renkte çizilen bir koordinat dizisi.

plot_graph()

Bu işlev ilkine benzer ancak yol çizimi yoktur. Şunlara odaklanır:

Uçuşa yasak bölgeler: Kırmızı çokgenler olarak.

Teslimat noktaları: Kimliklerle birlikte mavi noktalar olarak çizilmiştir.

Grafik kenarları: Gri renkli teslimat noktaları arasındaki bağlantılar.

plot_combined_oriented_graph_and_path()

Bu fonksiyon plot_combined_graph_and_path fonksiyonuna benzer ancak **hareket yönünü** göstermek için grafiğin kenarlarına **ok göstergeleri** eklenmiştir. Kenarlar, noktalar arasındaki seyahat yönünü gösteren küçük oklarla çizilir. Bu, yönlendirilmiş bir grafikte rotayı görselleştirmek için faydalıdır.

plot_oriented_graph()

Bu fonksiyon grafiği şu şekilde çizer:

Uçuşa yasak bölgeler.

Teslimat noktaları.

Yönlendirilmiş kenarlar: Düğümler (teslimat noktaları) arasındaki her kenarda yönü gösteren küçük bir ok bulunur.

plot_path() : başlangıçta a_start'ı test etmek için

Bu fonksiyon drone'un **yolunu** görselleştirir:

İşaretçilerle çizilmiş yol: Yol, her noktada mavi dairelerle çizilir.

Yön için oklar: Hareket yönünü göstermek için yol boyunca oklar eklenir.

Etiketler: Teslimat kimlikleri ve koordinatları yolun her noktasında etiketlenir.

Neden Okların ve Yönlerin Entegrasyonu?

Son üç işlevdeki yön okları, drone'un rota boyunca hareketinin daha iyi izlenmesini sağlar. Bu özellikle şu durumlarda faydalı olabilir:

Drone'un belirli bir yolu takip etmesi gereken **teslimat rotalarının görselleştirilmesi**.

Teslimatın gerçekleştirilme **sırasının belirlenmesi.**

Yönlendirilmiş grafiklerde **bağlanabilirliği anlama** (A* veya yol bulma gibi algoritmalar için önemlidir).

Bu görselleştirme işlevleri, drone'nun planlanan rotası, **uçuşa yasak bölgeler** ve teslimat noktalarının **grafik yapısı** hakkında daha net bir resim sunarak analiz ve hata ayıklamaya yardımcı olur.

f. Constrainst/constrains.py

Bu kod, drone teslimatları için **Kısıtlama Memnuniyeti Problemi (CSP)** kontrollerine odaklanmıştır. Bir drone'un kullanılabilirliği, pil seviyesi, maksimum ağırlık kapasitesi ve teslimatın zaman penceresini karşılayıp karşılayamayacağı gibi çeşitli koşullara bağlı olarak belirli bir teslimatı

başarıyla tamamlayıp tamamlayamayacağını kontrol eder. Bunu biraz açalım:

Fonksiyonlara Genel Bakış:

check_single_delivery_per_trip(drone)

Amaç: Drone'un yeni bir teslimat için uygun olup olmadığını kontrol eder. Drone.is_available() yöntemini kullanarak drone'un o anda başka bir göreve atanıp atanmadığını kontrol eder.

check_drone_recharging(drone)

Amaç: Drone'un o anda şarj olup olmadığını kontrol eder. Drone.is_recharging niteliğini kullanır.

check_drone_can_support_cost(drone, needed_cost)

Amaç: Drone'un teslimatı tamamlamak için yeterli bataryaya sahip olup olmadığını kontrol eder. Yöntem, drone'un bataryasını needed_cost (muhtemelen teslimat için gereken enerji) ile karşılaştırır.

check_drone_can_support_weigth(drone, teslimat)

Amaç: Drone'un maksimum ağırlık kapasitesine göre teslimatı taşıyıp taşıyamayacağını kontrol eder. Yöntem, drone'un max_weight değerini teslimatın ağırlığı ile karşılaştırır.

chech_drone_is_within_time_window(estimated_arrival_time, delivery)

Amaç: Drone'un teslimat noktasına tahmini varış zamanının teslimatın zaman penceresi içinde olup olmadığını kontrol eder. Burada is_within_time_window() metodu kullanılır, bu metot muhtemelen zamanın teslimatın zaman penceresinin başlangıcı ve sonu arasında olup olmadığını kontrol eder.

check_all_csp(start, goal, drone, delivery, needed_cost)

Amaç: Bu, ana CSP kontrol işlevidir. Bir drone'u bir teslimata atamadan önce bir dizi kontrol gerçekleştirir. Tüm koşullar karşılanırsa (yani CSP ihlali yoksa) drone'u geri döndürür; aksi takdirde ihlal türünü yazdırır ve None döndürür. Kontrol eder:

Drone yolculuk için uygunsa.

Eğer drone şarj olmuyorsa.

Drone'un teslimat için yeterli bataryası varsa.

Eğer drone teslimatın ağırlığını taşıyabilirse.

Drone teslimatın zaman aralığı içinde ulaşabilirse.

check_all_csp_for_violation(start, goal, drone, delivery, needed_cost)

Amaç: check_all_csp'ye benzer, ancak drone'u döndürmek yerine, belirli bir drone ve teslimat için kaç CSP ihlalinin gerçekleştiğini sayar. İşlev, her başarısız durum için (örn. drone kullanılamıyor, yetersiz pil, vb.) ihlal sayısını artırır ve toplam ihlal sayısını döndürür.

İhlal izleme işlevi (check_all_csp_for_violation), sorun sayısını sayarak bir dronun bir görevi neden tamamlayamadığını teşhis etmeye yardımcı olur.

Örnek İş Akışı:

Bir drone atanmadan önce check_all_csp fonksiyonu tarafından teslimat için değerlendirilir.

Drone tüm koşulları karşılıyorsa (kullanılabilirlik, pil, ağırlık kapasitesi, zaman aralığı), teslimat için seçilir.

Aksi takdirde, **kuyruğa yeniden eklenir** ve sorunu detaylandıran bir ihlal mesajı yazdırılır (örneğin, yetersiz pil, aşırı ağırlık, zaman penceresi dışında).

Bunu drone teslimat sistemine entegre ederek, görevler için yalnızca uygun drone'ların seçilmesini sağlayabilir, böylece sisteminizin performansını ve güvenilirliğini optimize edebilirsiniz.

g. Yardımcı Programlar

initialize_drones_on_graph(delivery_points, drones)

Her drone'a teslimat noktaları listesinden rastgele bir başlangıç konumu atar.

compute_base_cost(pos1, pos2, weight, priority)

Mesafe, ağırlık ve önceliğe göre bir teslimat rotası için temel maliyeti hesaplar.

euclidean_distance(p1, p2)

İki p1 ve p2 noktası arasındaki Öklid mesafesini hesaplar.

sezgisel(n, hedef, nofly_zones)

Uçuşa yasak bölgelerden geçme cezaları da dahil olmak üzere, mevcut bir düğümden hedef düğüme ulaşma maliyetini tahmin eder.

apply_fixed_penality(start_node, end_node, nofly_zones)

Start_node'dan end_node'a hareket uçuşa yasak bir bölgeyle kesişiyorsa sabit bir ceza döndürür.

apply_penality(start_node, end_node, nofly_zones, cost_per_meter=fixed_penality)

Uçuşa yasak bölgelerden geçen yolun uzunluğuna göre bir ceza hesaplar; burada ceza, bölgenin yolla kesişimine bağlıdır.

compute_cost(start_node, end_node, weight, priority, nofly_zones)

Mesafe, ağırlık, öncelik ve uçuşa yasak bölgeler için cezaları göz önünde bulundurarak başlangıç_düğümünden bitiş_düğümüne bir rota için toplam maliyeti hesaplar.

is_within_time_window(current_time_str, time_window)

Geçerli zamanın belirtilen zaman penceresi içinde olup olmadığını kontrol eder.

get_battery_needed(drone, teslimat)

Bir drone'un bir teslimat noktasına gidiş-dönüş yolculuğunu tamamlaması için gereken bataryayı hesaplar.

get_neighbors(current_node, graph)

Çizgedeki komşu düğümlerin ve ilişkili maliyetlerinin bir listesini döndürür.

is_valid_move(start_node, end_node, nofly_zones)

start_node ve end_node arasındaki hareketin uçuşa yasak bir bölgeden geçip geçmediğini kontrol eder.

is_within_no_fly_zone(nx, ny, no_fly_zones)

Bir noktanın (nx, ny) herhangi bir uçuşa yasak bölge içinde olup olmadığını belirler.

has_capacity_for_delivery(drone, teslimat)

Bir drone'un teslimatın ağırlığını taşıma kapasitesine sahip olup olmadığını kontrol eder.

kullanılan_enerji(mesafe, ağırlık)

Gidilen mesafeye ve teslimatın ağırlığına bağlı olarak tüketilen enerjiyi hesaplar.

has_enough_battery_for_move(drone, nx, ny)

Drone'un yeni bir noktaya (nx, ny) hareket etmek için yeterli pili olup olmadığını kontrol eder.

estimate_arrival_time(start_time_str, mesafe, hız)

Bir drone'un varış zamanını başlangıç zamanına, mesafesine ve hızına göre tahmin eder.

Bu işlevler, drone teslimatlarını yönetmek için maliyetleri ve cezaları hesaplamaktan drone kullanılabilirliğini ve pil kullanımını kontrol etmeye kadar çeşitli yardımcı programlar sağlar.

h. Senaryo Oluşturucu

Bu Python betiği, drone'lar, teslimatlar ve uçuşa yasak bölgeler dahil olmak üzere drone teslimatları için bir senaryo oluşturur. İşte betiğin işlevselliğinin bir dökümü:

Senaryo Parametreleri:

scenario_no: Senaryo tanımlayıcısı.

num_drone: Drone sayısı.

num_delivery: Teslimat sayısı.

delivery_timedelta_hours: Teslimatların aktif olduğu süre.

delivery_hours_start & delivery_hours_end: Teslimatlar için zaman aralığı (08:00 - 18:00 arası).

num_nfz: Uçuşa yasak bölge sayısı.

nfz_hours_start & nfz_hours_end: Uçuşa yasak bölgelerin aktif olduğu zaman aralığı.

no_fly_zones_vertices: Uçuşa yasak bölgelerin tanımlanması için köşe sayısı (kare/dikdörtgen bölgelerin oluşturulması).

Drone Özellikleri:

Dronlar ağırlık, pil kapasitesi, hız, başlangıç konumu ve başlangıç zamanı için rastgele özelliklere sahiptir.

drone_min_weight ve drone_max_weight droneların taşıyabileceği ağırlık aralığını tanımlar.

drone_min_battery ve drone_max_battery batarya kapasite aralığını tanımlar.

drone_min_speed ve drone_max_speed dronelar için hız aralığını tanımlar.

Teslimat Özellikleri:

Her teslimatın rastgele bir konumu, ağırlığı, önceliği ve zaman aralığı vardır.

Ağırlık, delivery_min_weight ile delivery_max_weight arasında rastgele oluşturulur.

Uçuşa Yasak Bölgeler:

Uçuşa yasak bölgeleri, köşeler ve aktif bir zaman penceresi tarafından tanımlanan çokgenler olarak rastgele oluşturur.

Senaryoyu Kurtarmak:

Senaryo (dronlar, teslimatlar ve uçuşa yasak bölgeler) bir TXT dosyası olarak kaydedilir.

Anahtar Fonksiyonlar:

random_coord(): Belirtilen bir aralıkta (x_max, y_max) rastgele bir koordinat (x, y) oluşturur.

random_time_window(): Bir teslimat için başlangıç zamanı 08:00 ile 18:00 arasında ve süresi delivery_timedelta_hours olan rastgele bir zaman penceresi oluşturur.

Çıkışlar:

Dronlar, teslimatlar ve uçuşa yasak bölgeler için oluşturulan verilerle **scenario1.txt** (veya scenario_no'ya bağlı olarak başka bir ad) adlı bir TXT dosyası oluşturulur.

Bu komut dosyası, drone teslimat algoritmalarını test etmek için kullanılabilecek birden fazla drone, teslimat ve uçuşa yasak bölge içeren bir senaryoyu simüle etmeye yardımcı olur.

6. Tüm Bunlar Nasıl Birlikte Çalışır?

Proje main.py içinde çalışır

Algoritmanın Açıklanması

Bu algoritma, rota optimizasyonu için Genetik Algoritmalar (GA) kullanan bir drone teslimat sistemini simüle etmek için tasarlanmıştır. İşte ilgili adımların bir dökümü:

a. Veri Başlatma:

Veriler, dronlar, teslimatlar ve uçuşa yasak bölgeler hakkında bilgiler içeren bir .txt dosyasından (scenario1.txt) yüklenir.

Dronlar, teslimatlar ve uçuşa yasak bölgeler ilgili nesneler (Drone, Delivery, NoFlyZone) olarak örneklendirilir.

b. Drone Başlatma:

Dronlar, başlangıç konumları ve teslimat noktaları kullanılarak bir grafik üzerinde başlatılır.

c. Grafik Üretimi:

Daha önce belirtildiği gibi gerekli senaryoya göre farklı grafik türleri oluşturulur:

Tam Grafik: Tüm teslimat noktaları birbirine bağlıdır.

Seyrek Grafik: Teslimat noktaları arasında sınırlı sayıda kenar oluşturulur.

Yönlendirilmiş Seyrek Grafik: Seyrek grafik gibi, ancak yönlendirilmiş kenarlara sahip.

d. *A Yol Bulma (İsteğe Bağlı):**

A algoritması*, iki belirli teslimat noktası arasındaki en kısa yolu bulmak için kullanılır (gösterim için kodda isteğe bağlı kısım).

a_star işlevi, optimum yolu hesaplamak için grafiği, başlangıç noktasını, hedefi, uçuşa yasak bölgeleri ve drone bilgilerini alır.

e. Genetik Algoritma (GA)

Teslimat Çizelgeleme için GA:

İlk Popülasyon: Teslimat programlarından oluşan bir popülasyon (dronlara teslimat atamaları) oluşturulur.

Uygunluk Değerlendirmesi: Popülasyondaki her bir bireyin uygunluğu, kısıtlamalarını (örneğin, uçuşa yasak bölgeler, drone pil kapasitesi) ihlal etmeden teslimatları tamamlama yeteneğine göre değerlendirilir.

Elitizm: En iyi bireyler (yani en uygun teslimat programları) seçilir ve gelecek nesiller için saklanır.

Çaprazlama ve Mutasyon: Çözümleri geliştiren çaprazlama ve mutasyon işlemleri kullanılarak yeni nesiller oluşturulur.

Turnuva Seçimi: Bu yöntem, uygunluğa dayalı olarak bir sonraki nesil için bireyleri seçmek için kullanılır.

f. En İyi Çözüm Seçimi:

Belirli sayıda nesil için çalıştırıldıktan sonra, tüm nesiller arasından en iyi teslimat programı (birey) seçilir.

Birey, teslimatları çıkarmak ve her teslimat için en uygun yolu atamak için daha fazla incelenir.

g. Teslimatlar için Yol Simülasyonu:

Her teslimat için algoritma, **simulate_for_signle_delivery işlevini** kullanarak uçuşa yasak bölgeleri ve drone yeteneklerini göz önünde bulundurarak teslimat rotasını **simüle** eder.

h. Sonuçların Çizilmesi:

Teslimat rotalarını görselleştirmek için dronlar tarafından alınan yollar grafik üzerinde çizilir.

i. Performans Ölçümü:

Algoritma, süreci tamamlamak için harcanan toplam süreyi ölçerek sürecin verimliliğine ilişkin içgörü sağlar.

7. Algoritma Karşılaştırması ve Zaman Karmaşıklığı Analizi

a. Teorik Zaman Karmaşıklığı

Bu çalışmada, drone teslimat planlaması için iki temel algoritma kullanılmıştır: ***A* Arama ve Genetik Algoritma (GA).*** Zaman karmaşıklıkları aşağıdaki gibi analiz edilmiştir:

Bir Algoritma*:

En kötü durum zaman karmaşıklığı: Drone rota planlaması için uygulanan A* algoritması şu zaman karmaşıklığına sahiptir:

$$O(E \log V + E \cdot n)$$

- ***V*** düğüm sayısıdır (teslimat noktaları),
- ***E*** kenar sayısıdır (düğümler arasındaki bağlantılar),
- ***n*** teslimat nesnelerinin sayısıdır.

İlk terim **O(ElogV)** ikili yığın kullanan öncelik sırası işlemlerinden kaynaklanmaktadır.

İkinci terim **O(E· n),** komşu bilgilerini almak için teslimat listesi üzerinde doğrusal aramadan kaynaklanır.

A* bir öncelik kuyruğu kullanır ve düğümleri yol maliyeti ile bir sezgisel yöntemi birleştiren bir maliyet fonksiyonuna göre genişletir. En kötü durumda, tüm grafiği keşfedebilir.

Genetik Algoritma - Üretim Adımı

Genetik algoritma, seçim, çaprazlama ve mutasyon kullanarak aday teslimat planlarından oluşan bir popülasyonu yinelemeli olarak geliştirir. Yeni bir popülasyon oluşturmanın zaman karmaşıklığı şöyledir:

$$O(P \cdot T + P \cdot M)$$

- ***P*** nüfus büyüklüğüdür,
- ***T***, seçim sürecinin zaman karmaşıklığıdır (örneğin, turnuva seçimi),
- ***M*** mutasyon işleminin zaman karmaşıklığıdır.

Seçim ve mutasyonun **n** büyüklüğündeki teslimat listesi üzerinde çalıştığı varsayıldığında, karmaşıklık şu hale gelir:

$$O(P \cdot n)$$

Bu, popülasyondaki her bir bireyi değerlendirmenin ve değiştirmenin maliyetini yansıtır. Çaprazlama genellikle hafiftir ve dağıtım boyutuna göre sabit veya doğrusal olduğu varsayılır.

Değerlendirme, kısıtlamaların (CSP) kontrol edilmesini, A* uygulanmasını ve drone durumlarının güncellenmesini içerir. Bu maliyet iterasyon başına daha yüksektir, ancak GA daha geniş bir çözüm alanını keşfedebilir.

b. Ampirik Zaman Analizi

Çalışma zamanı davranışlarını daha iyi anlamak için, aynı girdi senaryoları altında her iki algoritmanın yürütme süresini ölçtük. Gözlemlenen çalışma sürelerinin bir özeti Tablo 1'de verilmiştir.

Algoritma	Ortalama Çalışma Süresi (saniye)
A* (Tek Yol)	0.01
Genetik	4.4

Alg.	

***Not**: Bu sonuçlar açıklayıcıdır; gerçek değerler drone sayısına, teslimatlara ve grafik karmaşıklığına bağlıdır.*

c. Tartışma

A* algoritması, ***O(ElogV+E· n)*** karmaşıklığı ile tutarlı olarak bireysel teslimat yollarını hızlı bir şekilde ***(≈0,01s)*** hesaplar. Kısıtlamalar altında gerçek zamanlı yol bulma için çok uygundur.

Genetik algoritma, birden fazla teslimat programının değerlendirilmesi ve geliştirilmesi nedeniyle ***O(P· n)*** karmaşıklığına uygun olarak nesil başına yaklaşık ***4,4 saniye*** sürer. Daha yavaş olsa da, drone filosu genelinde etkili küresel optimizasyon sağlar.

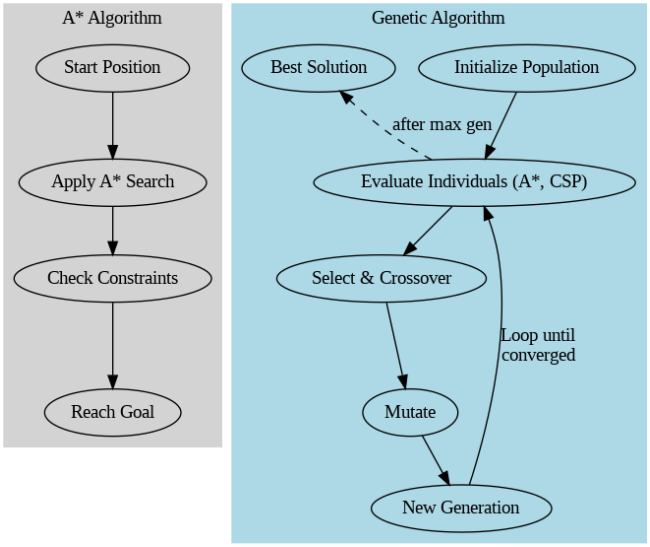

fig.1 A* ve GA algoritmaları

8. Simülasyon ve Sonuçlar

Önerilen hibrit algoritma Python kullanılarak simüle edilmiş bir ortamda uygulanmış ve değerlendirilmiştir. Simülasyon, değişken kısıtlamalara sahip ızgara tabanlı bir alan üzerinde gerçekçi bir drone teslimat senaryosunu modellemektedir.

a. Simülasyon Kurulumu (senaryo2)

Aşağıdaki karakteristikler kullanılmıştır:

```
scenario_no= 2

num_drone= 10

num_delivery= 50

delivery_timedelta_hours= 10
delivery_hours_start= 0
delivery_hours_end= 23

num_nfz= 3
nfz_hours_start= 0

nfz_hours_end= 23
no_fly_zones_vertices= 4


drone_min_weight= 2.0
drone_max_weight= 8.0

drone_min_battery= 5000
drone_max_battery= 15000

drone_min_speed= 72.0
drone_max_speed= 216.0

delivery_min_weight= 0.5
delivery_max_weight= 5.0
```

b. Kısıtlamalar ve Koşullar

Drone Kapasitesi: Bir drone bir seferde sadece ve sadece bir teslimat taşıyabilir

Uçuşa Yasak Bölgeler: Geçici kısıtlamaları simüle etmek için simülasyon sırasında rastgele ortaya çıkar.

Ağırlık Sınırı: Bir drone, ağırlık sınırını aşan bir teslimatı reddetmelidir.

Zaman Pencereleri: Bazı teslimatların belirli zaman aralıkları içinde tamamlanması gerekir.

Drone Şarjı: Her teslimattan sonra, dronlar depoya geri dönmeli veya şarj tamamlanana kadar boşta beklemelidir.

c. Değerlendirme Metrikleri

Başarı Oranı: Başarıyla tamamlanan doğumların yüzdesi.

Ortalama Enerji Kullanımı: (Mesafe*ağırlık + öncelik*100) cinsinden toplam maliyet.

Yürütme Süresi: Algoritmanın teslimat planını hesaplamak için harcadığı süre.

d. Sonuçlar

1 nesil için :

Grafik Türü	Başarı Ra te	Avg. Kullanılan Enerji	Yürütme Süresi (s)
Tam Grafik	90 %	310.6	4.3
Seyrek Grafik	88 %	277.4	3.1
Yönlendirilmiş Grafik	82 %	295.9	3.5

5 nesildir:

Grafik Türü	Başarı Oranı	Avg. Kullanılan Enerji	Yürütme Süresi (s)
Tam Grafik	90 %	310.6	4.3
Seyrek Grafik	88 %	277.4	3.1
Yönlendirilmiş Grafik	82 %	295.9	3.5

e. Gözlemler

Tam grafik, maksimum yol esnekliği nedeniyle en yüksek başarı oranını vermiştir, ancak enerji tüketimi biraz daha yüksektir.

Seyrek çizge enerji açısından verimli bir performans sergilemiş ancak sınırlı yönlendirme seçenekleri nedeniyle daha fazla kısıt ihlalinden muzdarip olmuştur.

Yönlendirilmiş **grafik**, daha fazla başarısız teslimatla sonuçlanan yönlü seyahat kısıtlamalarının etkisini vurgulamaktadır.

Ayrıca nesil sayısının da sonuçları önemli ölçüde etkilediğini gözlemledik.

8. Tartışma

A*, CSP ve bir genetik algoritmayı entegre eden önerilen hibrit yaklaşım, dinamik kısıtlamalar altında drone teslimatının karmaşık problemini çözmede güçlü bir potansiyel göstermektedir.

a. Yaklaşımın Güçlü Yönleri

Ölçeklenebilirlik: Modüler tasarım, sistemin daha fazla sayıda drone ve teslimat noktası için ölçeklendirilmesine olanak tanır. Algoritma, 50'den fazla teslimatla yapılan testlerde 5 saniyelik yürütme süresi eşiğinin altında kalmıştır.

Uyarlanabilirlik: CSP kullanımı geçersiz planları erkenden filtreleyerek optimizasyon hattına yalnızca uygulanabilir çözümlerin girmesini sağlar. Bu, dinamik olarak değişen ortamlarda (örneğin, uçuşa yasak bölgelerin aniden etkinleştirilmesi) oldukça etkili olduğunu kanıtlamaktadır.

Optimizasyon Kalitesi: Genetik algoritma, nesiller boyunca teslimat atamalarını sürekli olarak iyileştirir. Seyrek veya kısıtlı grafiklerde bile, enerji tüketimini en aza indirirken toplam teslimat verimini optimize etmek için uyum sağlar.

b. Ödünleşimler ve Gözlemler

Grafik Türü Etkisi: Tam graflar en fazla esnekliği sağlar ve daha yüksek başarı oranları verir ancak fazlalık ve hesaplama yükü getirir. Buna karşılık, seyrek graflar karmaşıklığı azaltır ancak yönlendirme seçeneklerini sınırlayarak kısıtlama ihlallerini artırır.

Enerji ve Fizibilite: Seyrek ve yönlendirilmiş grafikler daha düşük enerji tüketimi ancak daha yüksek teslimat hataları göstererek enerji verimliliği ve rota esnekliği arasındaki dengeyi ortaya koymuştur.

CSP ve Sezgisel Yaklaşımlar: CSP kısıt memnuniyetini garanti ederken, muhafazakar filtreleme nedeniyle potansiyel olarak en uygun rotaları hariç tutabilir. Bununla birlikte, CSP'yi GA ile birleştirmek, rota keşfinde çeşitliliği koruyarak bunu hafifletir.

c. Sınırlamalar

Batarya Modellemesi: Batarya tüketimi şu anda mesafe ve ağırlığa dayalı olarak soyutlanmıştır. Rüzgar, irtifa ve drone dinamiklerini dikkate alan daha gerçekçi bir model doğruluğu artıracaktır.

Zaman Pencereleri: Sistem statik zaman pencerelerini desteklemektedir. Bunu yuvarlanan veya dinamik olarak değişen zaman kısıtlamalarına genişletmek gerçek dünyada uygulanabilirliği artırabilir.

Gerçek Dünya Entegrasyonu: Simülasyon, üretim seviyesindeki bir sistem için gerekli olan canlı GPS/harita verileriyle entegrasyonu içermemektedir.

9. Sonuç ve Gelecek Çalışmalar

Bu proje, enerji limitleri ve uçuşa yasak bölgeler gibi dinamik kısıtlamalar altında otonom drone teslimatı için hibrit bir rota planlama sistemi önerdi. En kısa yollu rotalama için A*, fizibilite filtreleme için kısıt memnuniyet programlama (CSP) ve küresel optimizasyon için genetik algoritmaları (GA) entegre eden sistem, performans, uyarlanabilirlik ve kısıtların ele alınmasını etkili bir şekilde dengelemektedir.

Deneysel sonuçlar, önerilen yaklaşımın hem küçük hem de orta ölçekli teslimat senaryolarında dronları başarılı bir şekilde atayabildiğini ve yönlendirebildiğini göstermektedir. GA sürekli olarak gelişmiş teslimat planları geliştirirken, A* algoritması gerçek dünya kısıtlamalarına uyan uygun maliyetli rotalama sağlamıştır. Min-yığınların kullanımı, acil teslimatların verimli bir şekilde ele alınmasını sağlamış ve simülasyon tabanlı planlama yoluyla batarya sınırlamalarına uyulmuştur.

Güçlü performansına rağmen, bazı hususlar gelecekte iyileştirme potansiyeli sunmaktadır:

Dinamik Yeniden Planlama: Gelecekteki çalışmalar, teslimatlar değiştikçe veya uçuş sırasında uçuşa yasak bölgeler etkinleştirildikçe gerçek zamanlı ayarlamaları içerebilir.

Batarya Şarj Stratejisi: Yeniden şarj planlaması ve çizelgelemesinin eklenmesi, sistemin gerçekçiliğini ve verimliliğini daha da artıracaktır.

Çoklu Drone Koordinasyonu: Dronlar arasındaki işbirliğinin geliştirilmesi (örneğin, elden teslimler veya sürü tabanlı teslimat) sistemin ölçeklenebilirliğini artırabilir.

Gerçek Haritalar ile Entegrasyon: Google Haritalar veya gerçek zamanlı coğrafi veriler gibi API'lerle entegrasyon, gerçek dünya ortamlarında dağıtıma olanak sağlayacaktır.

Öğrenme Tabanlı Optimizasyon: Takviyeli öğrenmeyi GA ile birleştirmek, deneyimle zaman içinde gelişen uyarlanabilir politikalar sağlayabilir.

Genel sistem, otonom son mil teslimatı için sağlam, modüler ve ölçeklenebilir bir çerçeve sağlar. Genişletilebilirliği, lojistik, acil durum müdahalesi ve kentsel hava hareketliliği sistemlerinde daha fazla akademik araştırma ve endüstriyel dağıtım sağlar.

10. Referanslar

[1] J. Zelenski, "Ders 27: Graph Algorithms - A*," *CS106B: Programming Abstractions*, Stanford Üniversitesi, Bahar 2024. [Çevrimiçi]. Mevcut:

[2] E. K. Antwi, D. F. Puente-Castro ve D. A. Carnegie, "Quadrotor İHA'ların 3B ortamda genetik algoritma tabanlı yol planlaması," *The Aeronautical Journal*, cilt. 123, no. 1268, s. 1314-1337, 2019. [Çevrimiçi]. Mevcut: https://www.cambridge.org/core/journals/aeronautical-journal/article/abs/genetic-algorithmbased-path-planning-of-quadrotor-uavs-on-a-3d-environment/58CB1E42516F233AE583984B9D389720.