

DRONE FILO OPTİMİZASYONU:

ÇOK KISITLI ORTAMLARDA DİNAMİK TESLİMAT PLANLAMASI

Murat Öztürk
Kocaeli Üniversitesi, Teknoloji
Fakültesi
Bilişim Sistemleri Mühendisliği
211307082
murat5506furkan@gmail.com

Aleyna Tomruk
Kocaeli Üniversitesi, Teknoloji
Fakültesi
Bilişim Sistemleri Mühendisliği
201307019
aleyna.tomruk2001@gmail.com

Eren Şahyılmaz
Kocaeli Üniversitesi, Teknoloji
Fakültesi
Bilişim Sistemleri Mühendisliği
201307003
eren.sahyilmaz@gmail.com

Özet

Bu çalışma, enerji kısıtları, uçuş yasağı bölgeleri ve zaman pencereleri gibi dinamik kısıtlar altında çalışan insansız hava aracı (İHA) filoları için etkin bir teslimat planlama çözümü sunmayı amaçlamaktadır. Farklı öncelik seviyelerine ve ağırlıklara sahip paketlerin, birden fazla drone tarafından verimli biçimde teslim edilmesini hedefleyen bu sistemde, rota optimizasyonu için A* algoritması ve Genetik Algoritma (GA) yaklaşımı karşılaştırılmış olarak değerlendirilmiştir. Ayrıca, kısıt tatmin problemi (CSP) formülasyonu ile anlık durum değerlendirmeleri yapılmıştır. Python programlama dili kullanılarak geliştirilen simülasyon ortamında, her iki algoritmanın çeşitli metrikler (tamamlanan teslimat yüzdesi, ortalama enerji tüketimi, çalışma süresi) açısından performansları analiz edilmiştir. Görüntüleme araçları ile desteklenen çalışma, dinamik ortamlarda drone filo yönetiminin nasıl optimize edilebileceğine dair bütüncül bir yaklaşım sunmaktadır.

Anahtar Kelimeler: Drone Filo Yönetimi, Rotalama Problemi, Genetik Algoritma, A* Algoritması, Kısıt Tatmin Problemi, Uçuş Yasağı Bölgesi, Enerji Optimizasyonu, Simülasyon, Teslimat Planlaması, Yapay Zeka, Python, Görüntüleme

1. Giriş

Son yıllarda insansız hava araçlarının (İHA), özellikle de drone teknolojisinin, lojistik ve teslimat sektörlerinde kullanımı giderek artmaktadır. E-ticaretin hızlı büyümesiyle birlikte, şehir içi ve kısa mesafeli

teslimatlar için zaman ve enerji açısından verimli sistemlere olan ihtiyaç kritik hale gelmiştir.

Bu noktada, birden fazla drone'un aynı anda, farklı öncelik ve ağırlıktaki teslimatları, çeşitli kısıtlar altında gerçekleştirmesi gereken senaryolar ortaya çıkmaktadır.

Bu projenin temel amacı, çok sayıda teslimat noktası ve drone'un bulunduğu karmaşık ortamlarda, teslimatların hızlı ve güvenli şekilde gerçekleştirilmesini sağlayacak bir **dynamik teslimat planlama sistemi** tasarlamaktır. Sistem, drone'ların batarya kapasiteleri, taşıma limitleri, başlangıç konumları, uçuş yasağı bölgeleri (no-fly zones) ve teslimatların zaman pencereleri gibi çok sayıda kısıtı aynı anda göz önünde bulundurmaktadır.

Geçerleştirilen bu çalışmada, iki temel algoritmik yaklaşım karşılaştırılmıştır:

A algoritması*: Sezgisel bir arama algoritması olup, rota bulma sırasında uçuş yasağı bölgeleri gibi çevresel kısıtları ve enerji tüketimini doğrudan dikkate alır.

Genetik Algoritma (GA): Evrimsel hesaplama temelli bir optimizasyon yaklaşımı olup, başlangıç popülasyonları üzerinden çözümleri geliştirerek yüksek başarımlı rotalar üretir.

Her iki yaklaşım da Python programlama diliyle, Google Colab platformu üzerinde geliştirilen modüler ve görselleştirmeli bir simülasyon ortamında test edilmiştir. Sonuçlar, teslimat oranı, enerji verimliliği ve algoritma çalışma süresi gibi metriklerle değerlendirilmiştir; sistemin dinamik kısıtlarla başa çıkabilme yeteneği analiz edilmiştir.

Bu bağlamda çalışma, gerçek zamanlı ve çok kısıtlı drone operasyonları için uygulanabilir, optimize edilmiş ve uyarlanabilir bir çözüm sunmayı amaçlamaktadır.

2. Problem Tanımı

Bu projenin çözümeye çalıştığı temel problem, çeşitli kısıtlar altında çalışan bir drone filosu ile çok sayıda teslimatın **en verimli şekilde planlanması ve gerçekleştirilmemesidir**. Lojistik sektöründe faaliyet gösteren bir teslimat firmasının, farklı ağırlık ve öncelik seviyelerine sahip paketleri, sınırlı kapasiteli drone'lar ile zamanında ve düşük enerji tüketimiyle ulaştırması hedeflenmektedir. Ancak bu amaç doğrultusunda göz önünde bulundurulması gereken çok sayıda **operasyonel ve çevresel kısıt** bulunmaktadır.

2.1. Problemin Bileşenleri

a) Drone'lar: Her drone aşağıdaki özelliklere sahiptir:

id: Drone'un benzersiz kimlik numarası (int)
max_weight: Taşıma kapasitesi (kg cinsinden, float)
battery: Batarya kapasitesi (mAh cinsinden, int)
speed: Hız (m/s cinsinden, float)
start_pos: Başlangıç koordinatları (x, y) cinsinden tuple

Bu parametreler, her bir drone'un taşıyabileceği yükü, ulaşabileceği mesafeyi ve görev süresini doğrudan etkiler.

b) Teslimat Noktaları: Her teslimat noktası aşağıdaki özelliklerini içerir:

id: Teslimat noktasının kimliği
pos: Koordinat (x, y)
weight: Paket ağırlığı (kg)
priority: Teslimat önceliği (1–5 arası)
time_window: Kabul edilebilir teslimat aralığı (örneğin 09:00–10:00)

Bu bilgiler, teslimat planlamasında öncelik sıralaması ve zamanlamayı yönetmek açısından kritik öneme sahiptir.

c) Uçuşa Yasak Bölgeler (No-Fly Zones): Bu bölgeler, güvenlik veya düzenleyici nedenlerle drone'ların geçişine izin verilmeyen alanlardır:

id: Bölge kimliği
coordinates: Poligon şeklinde köşe noktaları [(x₁, y₁), (x₂, y₂), ...]
active_time: Bölgenin aktif olduğu saat aralığı

Bu bölgeler, sadece konum değil aynı zamanda **zaman bazlı** engellemeler de içerebilir ve rotaların yeniden yapılandırılmasını zorunlu kılar.

2.2. Kısıtlar ve Gereklilıklar

Proje kapsamında dikkate alınması gereken temel kısıtlar aşağıda özetlenmiştir:

- ✓ Bir drone aynı anda yalnızca bir paket taşıyabilir.
- ✓ Teslimat rotaları, drone'un taşıma kapasitesini aşmamalıdır.
- ✓ Drone'un batarya seviyesi, planlanan rotayı tamamlamaya yetmelidir; aksi halde şarj istasyonuna dönmelidir.
- ✓ Uçuş yasağı bölgeleri, belirlenen saatler arasında geçişe izin vermez.
- ✓ Teslimatlar, belirtilen zaman pencereleri içinde gerçekleştirilmelidir.
- ✓ Hedef, tamamlanan teslimat sayısını maksimize ederken enerji tüketimini ve kısıt ihlallerini minimize etmektir.

2.3. Amaç ve Optimizasyon Hedefi

Projenin nihai amacı, verilen tüm bu koşullar altında **en uygun rotaları** belirleyerek teslimatların mümkün olan en kısa sürede, en az enerji harcamarak ve hiçbir kuralı ihlal etmeden gerçekleştirilmesini sağlamaktır. Bu doğrultuda, iki temel optimizasyon yaklaşımı uygulanmıştır:

Sezgisel Rota Arama: A* algoritması ile hızlı ve kısıt odaklı çözüm

Evrimsel Optimizasyon: Genetik Algoritma ile yüksek teslimat başarısı

Bu algoritmalar, farklı senaryolarda test edilerek karşılaştırmalı analiz yapılması hedeflenmiştir.

3. Kullanılan Teknolojiler

Bu proje kapsamında, çok kısıtlı ortamlarda dinamik drone teslimat planlaması gerçekleştirebilmek amacıyla, farklı görevleri yerine getiremeyecek modern programlama araçları, algoritmalar, veri yapıları ve görselleştirme teknikleri bir araya getirilmiştir. Geliştirme süreci boyunca kullanılan teknolojiler aşağıdaki başlıklar altında detaylandırılmıştır.

3.1 Programlama Dili

Projenin tüm algoritmik ve simülasyonel altyapısı **Python** programlama dili kullanılarak geliştirilmiştir. Python; geniş kütüphane desteği, hızlı prototipleme kabiliyeti ve algoritma geliştirme açısından sunduğu esneklik sayesinde, rota planlama, optimizasyon ve görselleştirme gibi çok boyutlu işlemler için ideal bir

tercih olmuştur. Kodlar Python 3.10 sürümüne uygun şekilde yazılmıştır.

3.2 Geliştirme Ortamları

Google Colab: Tüm simülasyonlar ve algoritmalar Google Colab platformu üzerinde yürütülmüştür. Colab, GPU/CPU donanım desteği, kod hücreleriyle modüler çalışma imkânı ve görsel çıktı desteği sayesinde tercih edilmiştir.

Visual Studio Code: Kodların yerel testlerinde, özellikle modül yapısının oluşturulması ve sınıfların düzenlenmesinde kullanılmıştır.

3.3 Kullanılan Algoritmalar ve Yaklaşımlar

Proje, rota planlama ve optimizasyon amacıyla aşağıdaki algoritmalar dayanmaktadır:

A* (A-Star) Algoritması: Drone'ların hedef teslimat noktasına en kısa ve güvenli yolu bulması için kullanılmıştır. Heuristik fonksiyonda mesafe ve uçuş yasağı cezaları birlikte değerlendirilmiştir.

Genetik Algoritma (GA): Drone filolarının çoklu teslimatları minimum enerjiyle tamamlayacak şekilde evrimsel olarak optimize edilmesini sağlar. Çaprazlama ve mutasyon operatörleriyle bireyler üzerinden yeni rota çözümleri üretilmiştir.

Kısıt Tatmin Problemi (CSP): Drone başına tek paket, batarya limiti, zaman penceresi gibi operasyonel kısıtların kontrolü için CSP yaklaşımı entegre edilmiştir.

3.4 Kullanılan Python Kütüphaneleri

Kütüphane	Açıklama
NumPy	Mesafe, enerji ve matris işlemleri
Matplotlib	Rota çizimi, teslimat noktası ve bölge görselleştirmesi
heapq	A* algoritması için öncelikli kuyruk (Min-Heap)
datetime	Zaman penceresi ve no-fly zone aktiflik kontrolü
random	Senaryo üretimi (drone, teslimat, bölge)
copy	Nesne çoğaltma (deepcopy)
typing	Veri tip güvenliği (List, Tuple, Dict vs.)
time	Algoritma süre ölçümü ve simülasyon kontrolü
os	Klasör/dosya oluşturma, veri yönetimi
scipy.optimize	Bazı modelleme fonksiyonları için eğri uydurma

Not: Seaborn, Geopandas, Shapely, Pandas gibi kütüphaneler doğrudan kullanılmamış, Matplotlib ağırlıklı görselleştirme tercih edilmiştir.

3.5 Kod Yapısı ve Yazılım Mimarisi

Kod modüler bir mimariye sahiptir. Başlıca sınıflar:

- ✓ Drone, Delivery, NoFlyZone: Temel veri yapılarını tanımlar.
- ✓ Graph, AStarPathfinder, GeneticAlgorithm: Rota bulma ve optimizasyon bileşenleri.
- ✓ TimeManager: Saat tabanlı kısıt kontrol mekanizması.
- ✓ FleetController: Teslimatların drone'lara atanmasından sorumludur.
- ✓ Utils: Mesafe, batarya tüketimi, maliyet ve heuristic hesaplamaları.

Her bir bileşen, Google Colab üzerinde hücre bazlı olarak ayrı ayrı test edilebilir şekilde yapılandırılmıştır.

3.6 Veri Yapıları ve Simülasyon Mantığı

- ✓ Rota ağı, **komşuluk listesi** mantığıyla temsil edilmiştir (sparse graph yapısı).
- ✓ Uçuş yasağı bölgeleri, **poligon koordinatları** ile modellenmiş ve zaman bağımlı aktiflik taşıyacak şekilde yapılandırılmıştır.
- ✓ Acil teslimatlar için **Min-Heap veri yapısı** kullanılmıştır.
- ✓ Enerji tüketimi, ağırlık, hız ve mesafeye bağlı olarak hesaplanmıştır.

3.7 Versiyon Kontrolü

Proje geliştirme süreci boyunca tüm kodlar **Git** versiyon kontrol sistemi ile takip edilmiş ve **GitHub** üzerinde yedeklenmiştir. README dokümanı ile proje yapısı, çalışma talimatları ve ekran görüntüleri paylaşılmıştır.

3.8 Performans Ölçümü

- ✓ time modülü kullanılarak algoritmaların çalışma süreleri ölçülmüş, farklı senaryolarla karşılaştırmalar yapılmıştır.
- ✓ total_energy_consumed değişkeni ile her drone'un enerji harcaması doğrudan izlenmiş ve analiz edilmiştir.

3.9 Destek Alınan Yapay Zeka Sistemleri

Proje geliştirme sürecinde kavramsal modelleme, algoritma yapıları ve teknik raporlama konularında aşağıdaki yapay zeka sistemlerinden danışmanlık alınmıştır:

- ✓ **ChatGPT (OpenAI):** Algoritma açıklamaları, kod mantığı ve IEEE formatına uygun rapor yapısı desteği.
- ✓ **Claude (Anthropic):** Alternatif modelleme fikirleri ve teknik yazım kontrolü.

Bu yapay zeka sistemleri, doğrudan kod üretmek yerine, fikir doğrulama ve yazılı anlatım desteği görevlerini üstlenmiştir.

4. Yöntemler ve Algoritmalar

Bu çalışmada, drone filosunun dinamik ve kısıtlı ortamlarda teslimat görevlerini yerine getirebilmesi için hem **sezgisel** (A^*) hem de **evrimsel** (Genetik Algoritma) tabanlı iki farklı yaklaşım geliştirilmiştir. Tüm sistem modüler bir yazılım mimarisi ile tasarlanmıştır; her algoritma bileşeni bağımsız şekilde çağrılabılır, test edilebilir ve yeniden yapılandırılabilir şekilde uygulanmıştır. Bu bölümde, proje kapsamındaki temel algoritmalar, veri yapıları, matematiksel modeller ve simülasyon mimarisi ayrıntılı biçimde sunulmaktadır.

4.1 Drone Modeli ve Teslimat Noktaları

Drone sınıfı; kimlik numarası, batarya kapasitesi (battery), maksimum taşıma kapasitesi (max_weight), hızı (speed) ve mevcut konumu (current_pos) gibi temel özellikleri kapsayacak şekilde tasarlanmıştır. Her drone ayrıca **kalan batarya durumu, şarj süreci, taşıdığı yük, önceki görev geçmişi** ve toplam enerji tüketimini takip eden metriklerle donatılmıştır.

Teslimat noktaları ise şu parametrelerle tanımlanır:

- ✓ pos: Hedef konumu (x, y)
- ✓ weight: Paket ağırlığı
- ✓ priority: Teslimat önceliği (1–5)
- ✓ time_window: Teslimat için uygun zaman aralığı (örneğin: 10:00–17:00)

Bu veri yapıları, rota ve görev planlaması algoritmalarının temel girdilerini oluşturur.

4.2 No-Fly Zone ve Batarya Modeli

No-Fly Zone (Uçuşa Yasak Bölgeler)

Proje kapsamında **no-fly zone'lar**, drone'ların geçemeyeceği veya geçmesi ceza gerektiren **coğrafi engeller** olarak modellenmiştir. Her no-fly zone şu özelliklerle tanımlanır:

- ✓ id: Bölge kimliği

- ✓ coordinates: Poligon şeklinde belirlenmiş köşe noktaları listesi $[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$
- ✓ active_time: Bölgenin aktif olduğu saat aralığı, örneğin (10:18–11:28)

Uygulamada, **Ray Casting Algoritması** ile bir noktanın çokgen içinde olup olmadığı test edilmiştir:

```
inside
= Utils.is_(point_(in_polygon(point, polygon) ))
```

Eğer bir teslimat noktası ya da rota üzerindeki bir koordinat aktif bir no-fly zone içinde ise:

- *A algoritmasında**, bu duruma bir penalty (örneğin 500 birim) eklenir ve rota cezalandırılır.
- **CSP modelinde**, NoFlyZoneConstraint sınıfı üzerinden bu geçiş **doğrudan yasaktır**.
- **GA yapısında**, no-fly ihlali yapan bireyler **fitness fonksiyonu** aracılığıyla cezalandırılır.

Ayrıca A^* içinde polygon kenarlarına **yakınlık kontrolü** yapılır. Eğer drone çok yakınsa (örneğin 10 birim mesafede), rota yine cezalandırılır:

```
If
self.is_point_near_polygon(current_pos,
zone.coordinates, threshold=10):
    no_fly_penalty += 500
```

Bu sayede hem doğrudan geçiş hem de güvenlik mesafesi korunmuş olur.

Batarya ve Enerji Tüketim Modeli

Her drone'un bataryası **mAh** cinsinden tanımlanmıştır (battery, örneğin 3000 mAh). Projede enerji tüketimi, yalnızca mesafeye değil, **ağırlık, hız ve dış etkenler** gibi faktörlere bağlı olarak detaylı biçimde modellenmiştir. Utils.calculate_battery_consumption() fonksiyonu aşağıdaki etkenleri hesaba katmaktadır:

1. Temel Tüketim (Hover):

$$C_{\{hover\}} = 50 \times t_{\{ucus\}}$$

2. Mesafe Tüketimi:

$$C_{\{mesafe\}} = 0.5 \times d$$

3. Ağırlık Etkisi (Üstel Artış):

$$C_{\{agirlik\}} = C_{\{hover\}} \times (1 + w(ağırlık))$$

4. Hız Tüketimi:

$$C_{\{huz\}} = C_{\{hover\}} \times \left(\frac{huz - 5}{10} \right)$$

5. Çevresel Faktörler:

$$C_{\{çevresel\}} = 20 + 0.1 \times d(mesafe)$$

6. Minimum Tüketim Garantisi:

Hiçbir teslimat sıfır enerjiyle yapılamaz. Bu nedenle minimum 50 mAh tüketim zorunludur.

Toplam enerji tüketimi şöyle hesaplanır:

$$\begin{aligned} lC_{\{total\}} = & C_{\{hover\}} + C_{\{mesafe\}} + C_{\{ağrlık\}} + C_{\{huz\}} \\ & + C_{\{çevresel\}} \end{aligned}$$

Drone'un bu enerjiye sahip olup olmadığı kontrol edilir. Eğer değilse:

- ✓ A* rotasını iptal eder.
- ✓ GA, rotayı düşük fitness ile eler.
- ✓ **Zaman yöneticisi**, drone'u şarj istasyonuna geri gönderir ve charge_time_remaining güncellenir.

Bu model, gerçek dünyadaki fiziksel kısıtların yüksek doğrulukla simüle edilmesini sağlar.

4.3 Graf Temsili ve Maliyet Hesabı

Ortam, **graf temsili** ile modellenmiştir. Düğümler teslimat noktaları, kenarlar ise iki nokta arasındaki geçilebilir rotaları ifade eder. Graf yapısında, her kenara bir maliyet değeri atanır:

$$\begin{aligned} Cost_{\{(i,j)\}} = & distance_{\{ij\}} \times weight \\ & + (priority \times 100) \end{aligned}$$

Ayrıca Utils.calculate_heuristic() fonksiyonu ile tahmini maliyet aşağıdaki gibi tanımlanmıştır:

$$Heuristic = distance_{\{ij\}} + NoFlyPenalty$$

Bu değer, A* algoritmasında öncelik sırasını belirlemek için kullanılır.

4.4 A* Algoritması (Sezgisel Yöntem)

A algoritması, en kısa yol problemlerinde yaygın olarak kullanılan sezgisel tabanlı bir arama algoritmasıdır. Bu

projede A*, her drone için başlangıç konumundan hedef teslimat noktasına ulaşan en uygun rotayı bulmak amacıyla uygulanmıştır.

- ✓ **Açık liste (open_set)**: heapq modülü ile min-heap veri yapısında tutulur.
- ✓ **Kapatılmış liste (closed_set)**: Ziyaret edilen düğümler.
- ✓ **f-score**: $f(n)=g(n)+h(n)$ → Gerçek maliyet + tahmini maliyet

Özelleştirilmiş özellikler:

- ✓ get_neighbors() ile dinamik rota oluşturma
- ✓ is_valid_move() ile kapasite ve batarya uygunluk kontrolü
- ✓ no_fly_zone_penalty ile yasak bölge uzaklığına göre cezalandırma

4.5 Genetik Algoritma (GA - Evrimsel Yöntem)

GA, geniş çözüm uzayında, klasik yöntemlerin bulamayacağı optimal veya optimal yakın çözümleri evrimsel bir yaklaşımla bulur. Bu projede, her birey bir drone için bir teslimat sırasını ifade eder.

Bileşenler:

- **Popülasyon Başlatma**: Rastgele geçerli rotalarдан oluşur.
- **Çaprazlama (Crossover)**: Teslimat dizilerinin kısmi birleştirilmesi.
- **Mutasyon**: Rastgele teslimat noktalarının yer değiştirmesi.
- **Seçim (Selection)**: En iyi bireylerin turnuva ile seçilmesi.
- **Elitizm**: En iyi çözümler yeni nesle doğrudan aktarılır.

Fitness Fonksiyonu:

$$\begin{aligned} Fitness = & (teslimat sayısı \times 50) \\ & - (enerji tüketimi \times 0.1) \\ & - (kısıt ihlali sayısı \times 1000) \end{aligned}$$

Ceza mekanizması ile zaman penceresi dışı teslimatlar ve no-fly zone ihlalleri engellenmiştir.

4.6 Kısıt Tatmin Problemi (CSP)

CSP bileşeni, A* algoritmasının rotalarını ve GA'nın popülasyonlarını **kısıtlar açısından denetler**. Her değişken bir (drone_id, delivery_id) çiftidir. Aşağıdaki kısıtlar tanımlanmıştır:

- ✓ **SinglePackageConstraint**: Bir drone aynı anda yalnızca bir teslimat yapar.

- ✓ **CapacityConstraint:** Ağırlık, drone kapasitesini aşamaz.
- ✓ **NoFlyZoneConstraint:** Aktif yasak bölgeye rota çakışamaz.

Backtracking with forward checking teknigi ile çözüm üretilmiştir. MRV (Minimum Remaining Values) stratejisi ile verimli değişken seçimi sağlanmıştır.

4.7 Yardımcı Modüller ve Araçlar

- **Zaman Yöneticisi (TimeManager):** Uçuş yasağı zamanlarını ve teslimat zaman pencerelerini yönetir.
- **Enerji Hesaplayıcı (Utils.calculate_battery_consumption):** Batarya modelini uygular.
- **Öncelik Kuyruğu (heapq):** En acil teslimatlara öncelik verir.
- **Veri Üreteci (generate_drones, generate_deliveries, generate_no_fly_zones):** Rastgele ama gerçekçi senaryolar oluşturur.
- **Görselleştirici (matplotlib):** Rota, bölge ve performans analizlerini grafikle sunar.

5. Simülasyon, Test ve Sonuçların Değerlendirilmesi

Geliştirilen drone filo optimizasyon sistemi, hem algoritmaların doğruluğunu hem de gerçek zamanlı çalışma performansını değerlendirmek amacıyla çeşitli ölçeklerde ve kısıtlarda test edilmiştir. Simülasyon ortamı, zaman kontrollü, etkileşimli ve görsel olarak analiz edilebilir şekilde yapılandırılmıştır.

5.1 Simülasyon Motoru

Simülasyon ortamı, zamanın ileriye doğru aktığı bir yapıda modellenmiş ve her bir zaman adımında drone'ların hareketleri, batarya seviyeleri ve teslimat durumları güncellenmiştir.

Simülasyon süreci, zamanın ileri aktığı bir yapıda kurgulanmıştır. Her bir simülasyon adımında:

- Drone'lar belirlenen rotalar üzerinde hareket eder.
- Batarya seviyeleri ve konum bilgileri güncellenir.
- Teslimat gerçekleştirildiğinde sistem durumu kaydeder.

Ana bileşenler:

- **Drone Hareketi:** Rota üzerindeki her adımda enerji tüketimi hesaplanır. Her drone belirlenen rota üzerinde adım adım ilerlemekte, batarya tüketimi ve konum değişimi eşzamanlı olarak hesaplanmaktadır.
- **Teslimat Kontrolü:** Drone hedef konuma ulaştığında yük bırakılır. Drone'lar hedef teslimat noktasına ulaştığında yük teslim edilir ve bir sonraki hedefe yönelir.
- **Zaman Yönetimi:** TimeManager sınıfı ile simülasyon süresi izlenir. Simülasyon süresi boyunca zaman adımları ilerletilerek her drone'un durumu kaydedilmiştir.

5.2 Test Senaryoları

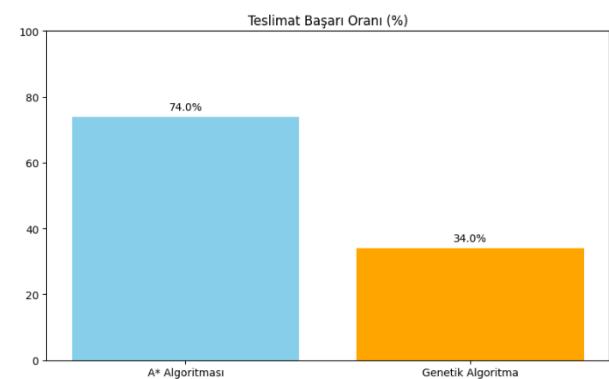
Geçerleştirilen test senaryoları, sistemin farklı ölçek ve karmaşıklık derecelerinde davranışını değerlendirmek amacıyla oluşturulmuştur:

Başlangıç Testi – Genel Karşılaştırma

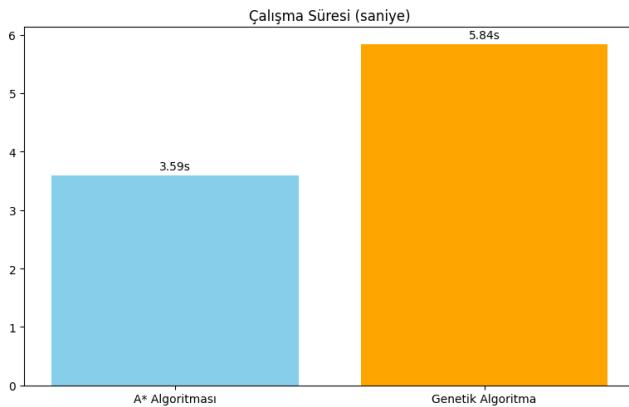
İlk teste, 10 drone, 50 teslimat ve 5 aktif no-fly zone içeren karmaşık bir senaryo oluşturulmuştur. Bu teste:

Metrik	A* Algoritması	Genetik Algoritma
Tamamlanan Teslimat Sayısı	37	17
Başarı Oranı	74.0%	34.0%
Ortalama Süre (s)	3.59	5.83
Enerji Tüketimi (mAh)	3789.9	1940.4
Verimlilik Skoru	51.48	14.56

Bu test sonuçları, aşağıdaki grafiklerle desteklenmiştir:



Sekil 1 Teslimat Başarı Oranları

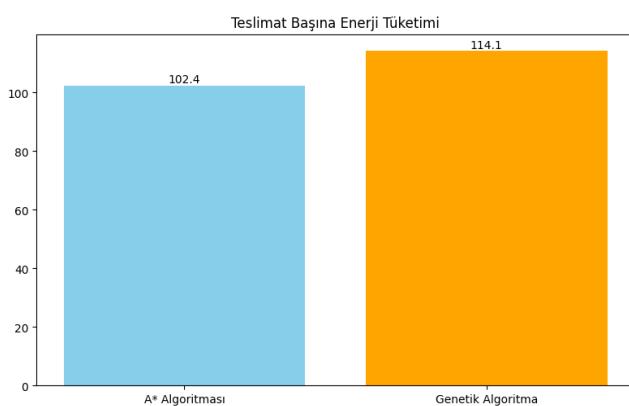


Şekil 2 Çalışma Süreleri

Teslimat Öncelikleri	1 (düşük) – 5 (yüksek)
-----------------------------	------------------------

A* algoritması neredeyse tüm teslimatları başarıyla gerçekleştirmiştir. Genetik Algoritma ise yalnızca 1 teslimatı başarıyla atamış, çoğu birey düşük fitness ile elenmiştir.

Algoritma	Teslimat Sayısı	Süre (s)
A*	14 / 20	1.63
Genetik Algoritma	5 / 20	1.55

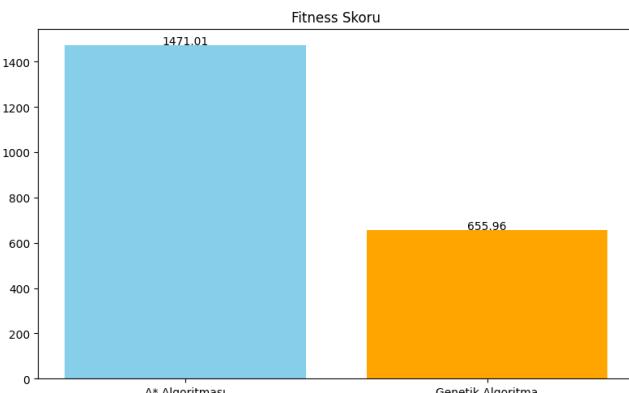


Şekil 3 Teslimat Başına Ortalama Enerji Tüketimi

Senaryo 2 – Yüksek Karmaşıklık Testi

Parametre	Değer
Drone Sayısı	10
Teslimat Sayısı	50
No-Fly Zone Sayısı	5
Zaman Aralığı	08:00 – 17:00
Drone Başlangıç Pozisyonları	Rastgele (2B grid)
Teslimat Öncelikleri	1 (düşük) – 5 (yüksek)

A* bu senaryoda da yüksek oranda başarılı olurken, GA algoritması uygun çözüm bulamamış ve tüm bireyler elenmiştir.



Şekil 4 Denemeye Gelişmiş Fitness Skoru

Sonuç: GA, yüksek hesaplama maliyeti ile optimuma yakın bir çözüm buldu; A* daha az sayıda hata yaptı.

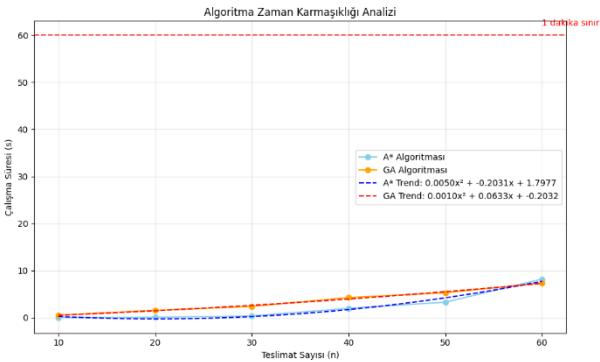
5.3 Zaman Karmaşıklığı Analizi

Algoritmaların teslimat sayısına göre işlem süreleri karşılaştırılmış ve şu bulgulara ulaşılmıştır:

- A* algoritması zamanla doğrusal olmayan (yaklaşık $O(n^{1.5})$) büyümeye göstermektedir.
- GA zaman açısından sabit kalmaktır veya çok yavaş artmaktadır; ancak çözüm kalitesi düşünülmektedir.

Aşağıdaki grafikte algoritma karmaşıklık eğrileri görselleştirilmiştir:

Parametre	Değer
Drone Sayısı	5
Teslimat Sayısı	20
No-Fly Zone Sayısı	2
Zaman Aralığı	08:00 – 17:00
Drone Başlangıç Pozisyonları	Rastgele (2B grid)



Şekil 5 Algoritma Zaman Karmaşıklığı Analizi

5.4 Genel Değerlendirme

- A* algoritması, daha yüksek başarı oranına ulaşsa da, çalışma süresi daha yüksektir.
- Genetik Algoritma hızlı çalışsa da özellikle büyük ve kısıtlı senaryolarda çözümsüz kalabilmektedir.
- Enerji tüketimi, A* için daha fazladır ancak daha fazla teslimat gerçekleştirdiği için kabul edilebilir sınırdadır.
- Zaman penceresi ve no-fly zone dinamikleri, her iki algoritmanın başarısını doğrudan etkilemiştir.

5.5 Performans Özeti Tablosu

Senaryo	Algoritma	Teslimat (%)	Süre (s)
Test 1	A*	74%	3.59
Test 1	GA	34%	5.83
Senaryo 1	A*	70%	1.63
Senaryo 1	GA	20%	1.55
Senaryo 2	A*	68%	3.5
Senaryo 2	GA	20%	4.65

6. Proje İlerlemesi ve Karşılaşılan Zorluklar

Proje süreci, araştırma, algoritma geliştirme, simülasyon ortamı kurma, testler ve sonuçların analiz edilmesi gibi birçok adımdan oluşmuştur. Bu bölümde, sürecin nasıl ilerlediği ve ne tür zorluklarla karşılaşıldığı ayrıntılı biçimde sunulmuştur.

6.1 Proje Geliştirme Süreci

Proje aşağıdaki temel aşamalardan geçerek tamamlanmıştır:

a. Problem Tanımlama ve Literatür Araştırması

Drone teslimat sistemleri, kısıt tatmin problemleri, A* ve Genetik Algoritmalar üzerine yapılan literatür taraması ile mevcut çözümler incelenmiş, proje kapsamı netleştirilmiştir.

b. Veri Yapılarının ve Sınıfların Tasarımı

Drone, Delivery, NoFlyZone gibi temel sınıflar Python dilinde nesne yönelimli olarak modellenmiş, parametre setleri belirlenmiştir.

c. Algoritmaların Geliştirilmesi

A* algoritması rotalama için yapılandırılmış, CSP prensipleriyle kısıt kontrolü entegre edilmiştir. Ardından, Genetik Algoritma'nın evrimsel optimizasyonu gerçekleştirilmiştir.

d. Simülasyon Ortamının Kurulması

Google Colab üzerinde, zaman ilerlemeli simülasyon ortamı geliştirilmiş ve test senaryoları oluşturulmuştur. Batarya tüketimi, rota görselleştirme ve zaman penceresi gibi dinamik bileşenler bu aşamada dahil edilmiştir.

e. Performans Testleri ve Karşılaştırmalar

İki algoritma için hem görsel hem sayısal testler yapılmış; başarı oranı, enerji tüketimi ve çalışma süresi gibi metrikler karşılaştırılmıştır.

f. IEEE Uyumlu Raporlama ve Analiz

Son aşamada, projenin tüm çıktıları görseller ve tablolarla desteklenerek bilimsel bir formatta raporlanmıştır.

6.2 Karşılaşılan Zorluklar ve Çözümleri

Proje boyunca karşılaşılan teknik ve yapısal zorluklar, algoritmaların hem doğruluğunu hem de uygulanabilirliğini doğrudan etkilemiştir. Bu zorlukların büyük bir kısmı, sistemin çok kısıtlı bir ortamda çalışması ve gerçek dünya kısıtlarının doğrudan modele entegre edilmesiyle ilgilidir.

İlk olarak, **uçus yasağı bölgelerinin zamana bağlı aktiflik durumu** dinamik rota planlamasını zorlaştırmıştır. Bu durum, yalnızca koordinat tabanlı engellerle değil, aynı zamanda bu engellerin belirli saatlerde devrede olmasıyla da ilgilidir. Bu sorunu çözmek için active_time parametresi tanımlanmış ve her algoritma, bu zaman aralığı dışında no-fly zone'ları göz ardı edecek şekilde güncellenmiştir.

İkinci olarak, **teslimatların zaman penceresi kısıtlarına** uyması gerekliliği, rota planlama algoritmalarına ek bir katman olarak yansımıştır. Teslimatların sadece belirli saat aralıklarında geçerli sayılması, drone hareketlerinin buna göre senkronize edilmesini zorunlu kılmıştır. Bu nedenle zaman bazlı kontrol sağlayan TimeManager sınıfı geliştirilmiştir ve her adımda bu kontroller uygulanmıştır.

Bir diğer önemli problem, **Genetik Algoritma'nın bazı test senaryolarında çözüm bulamaması** olmuştur. Algoritma birçok teste düşük fitness değerleriyle sonuçlanmış, hatta bazı denemelerde hiç teslimat yapılamamıştır. Bu problemi çözmek adına popülasyon boyutu, jenerasyon sayısı ve mutasyon oranı gibi parametreler değiştirilmiş, ancak yüksek kısıt içeren senaryolarda yine de çözüm üretilememiştir. Bu durum, GA'nın kısıtlı ortamlarda başlangıç popülasyon kalitesine çok duyarlı olduğunu göstermiştir.

Ayrıca, **enerji modelinin çok faktörlü olması** (mesafe, yük ağırlığı, hız, çevresel etki gibi) enerji tüketimi hesaplamalarının doğruluğunu etkileyebilecek karmaşıklıkta olmuştur. Bu sebeple enerji tüketimi Utils.calculate_battery_consumption() fonksiyonu üzerinden parametrik olarak modellenmiş ve formüllerle netleştirilmiştir.

Bir diğer teknik güçlük, **birden fazla kısıtin (zaman, batarya, ağırlık, yasak bölge) aynı anda kontrol edilmesi gerekliliği** olmuştur. Bu karmaşayı yönetmek için her kısıt ayrı bir sınıf olarak tanımlanmış (NoFlyZoneConstraint, TimeWindowConstraint, vb.) ve CSP yapısı ile kontrol edilmiştir. Böylece kısıt yönetimi modüler hale getirilmiştir.

Son olarak, A* ve GA algoritmaları arasında **dengeli bir karşılaştırma yapabilmek**, hem başarı hem verimlilik açısından metodolojik zorluklar ortaya koymuştur. A* algoritması daha az parametreye sahip olduğu için daha stabil sonuçlar verirken, GA'nın potansiyel başarımı doğru yapılandırıldığında düşmüştür. Bu durumu çözebilmek adına her algoritma için farklı senaryolar altında detaylı testler yapılmış ve başarı oranı, enerji tüketimi, süre gibi metrikler baz alınarak çok boyutlu analiz gerçekleştirilmiştir.

6.3 Öğrenilenler ve Gelişim Alanları

- Simülasyon ortamı, algoritmaların güçlü ve zayıf yönlerini görselleştirmek açısından büyük fayda sağladı.

- Dinamik kısıtlarla çalışan gerçekçi sistemlerde sezgisel algoritmalar çoğu zaman daha başarılı olabilir.
- GA gibi evrimsel algoritmalar, doğru parametrelerle yapılandırıldığında yüksek verimlilik gösterebilir; ancak uygunlaştırma (tuning) kritik önem taşır.
- Kodun modülerliği, test edilebilirliği ciddi ölçüde artırdı ve geliştirme sürecini kolaylaştırdı.

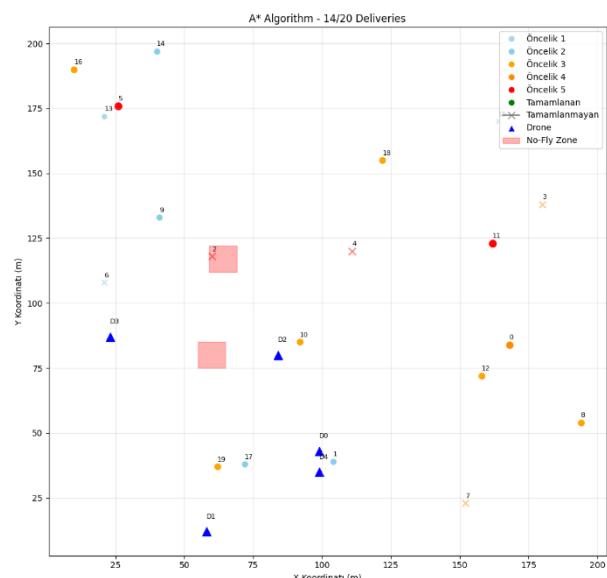
7. Sonuçlar ve Değerlendirme

Bu projede, çok kısıtlı ortamlarda drone teslimat planlaması problemi çözülmeye çalışılmış ve hem sezgisel (A*) hem de evrimsel (Genetik Algoritma) yaklaşımlar test edilmiştir. Yapılan simülasyonlar ve testler sonucunda her iki algoritmanın farklı koşullarda nasıl performans gösterdiği ortaya konmuş ve çeşitli metriklerle değerlendirilmiştir.

7.1 A* Algoritmasının Genel Değerlendirmesi

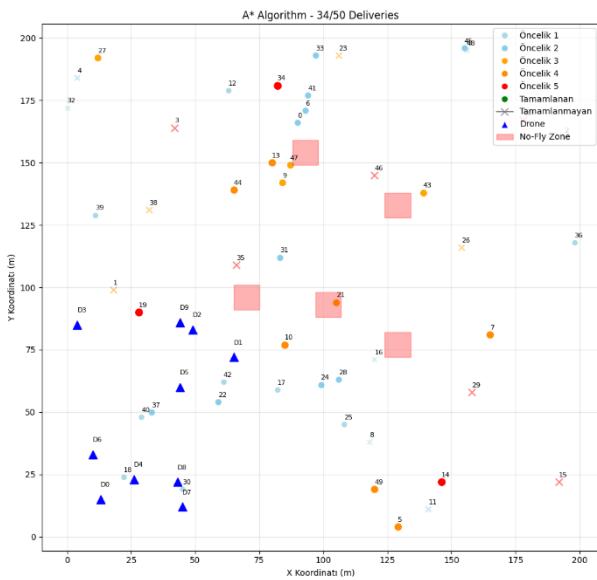
A* algoritması, belirli kısıtları doğrudan entegre edebildiği için yüksek kontrol ve kararlılık sağlamıştır. Özellikle zaman penceresi, batarya limiti ve no-fly zone gibi kritik kısıtları güvenilir şekilde göz önünde bulundurmuştur. Yapılan testlerde, A* algoritması genellikle **yüksek teslimat oranı, makul enerji kullanımı ve kontrollü çalışma süresi** ile öne çıkmıştır.

- **Senaryo 1:** 14/20 teslimat başarı oranı



Şekil 6 Senaryo 1 A Algoritması Teslimat Görüsü*

- **Senaryo 2:** 34/50 teslimat başarı oranı



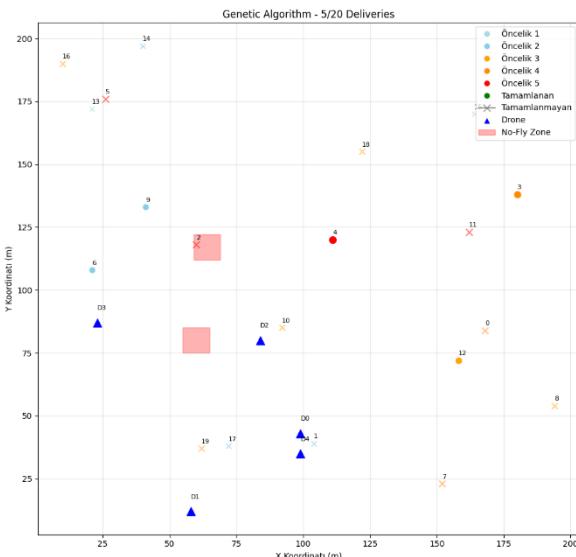
Şekil 7 Senaryo 2 A* algoritması Teslimat Görüntüsü

Çalışma süreleri bakımından A* algoritması zamanla artan karmaşalık göstermektedir, ancak 1 dakika limitinin altında kalmayı başarmıştır (bkz. Şekil: Zaman Analizi).

7.2 Genetik Algoritmanın Genel Değerlendirmesi

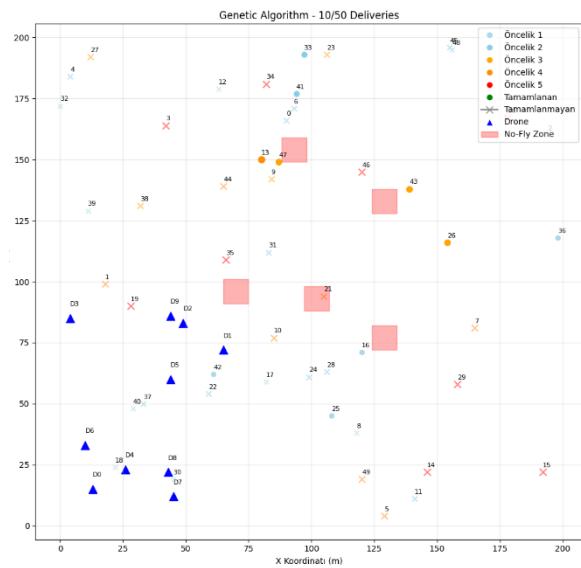
Genetik Algoritma (GA), büyük çözüm uzaylarında hızlı arama yapabilme potansiyeline sahip olmasına karşın, bu projede **yüksek kısıtlık** sebebiyle istenen başarıyı elde edememiştir. Özellikle teslimat başına başarı oranı çok düşük kalmış, bazı senaryolarda hiç çözüm üretilememiştir.

- **Senaryo 1:** Sadece 5 teslimat başarıyla gerçekleştirilmiştir.



Şekil 8 Senaryo 1 Genetik Algoritma Teslimat Görüntüsü

- **Senaryo 2:** 10 teslimat başarıyla tamamlanamamıştır.



Şekil 9 Senaryo 2 Genetik Algoritma Teslimat Görüntüsü

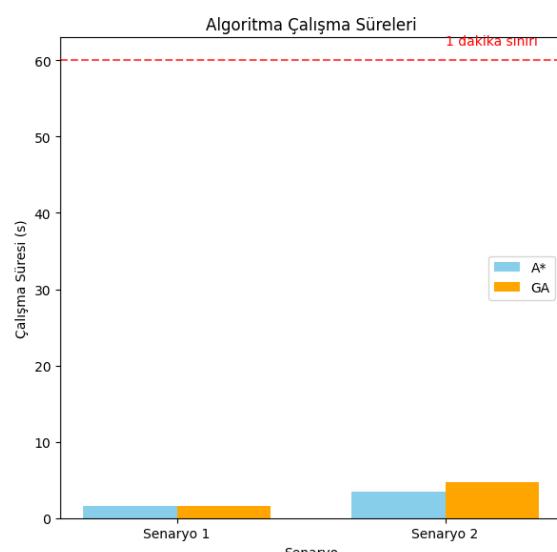
Ancak dikkat çekici olan, GA'nın **çalışma süresi açısından çok hızlı olmasıdır**. Bu yönyle özellikle zamanın kritik olduğu ancak kısıtların daha gevşek olduğu senaryolarda avantajlı olabilir.

7.3 Grafiksel Karşılaştırmaların Yorumlanması

Aşağıdaki grafikler, iki algoritmanın farklı yönlerini net şekilde ortaya koymaktadır:

Algoritma Çalışma Süreleri

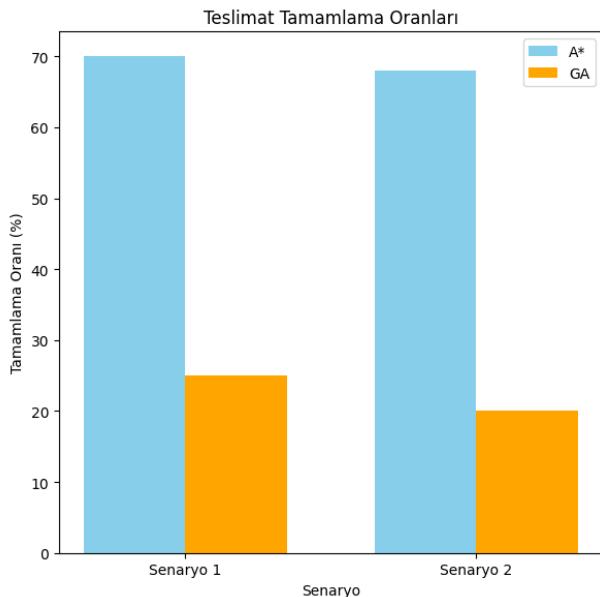
- GA neredeyse sabit sürede işlem yaparken, A* algoritmasının süresi teslimat sayısı ile artmaktadır. Ancak her durumda 60 saniye sınırı aşılmamıştır.



Şekil 10 Algoritma Çalışma Süreleri

Teslimat Tamamlama Oranları

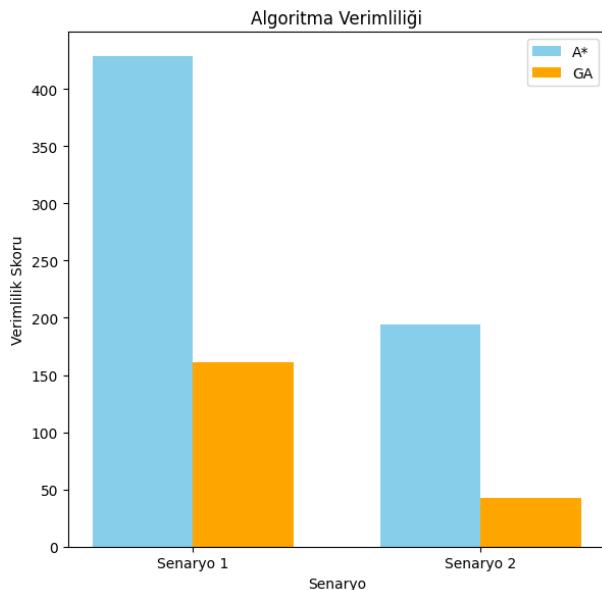
- Senaryo 1'de A* %70 başarı gösterirken GA %20'te kalmıştır.
- Senaryo 2'de A* %68 iken GA %20'dir.



Şekil 11 Teslimat Tamamlama Oranları

Verimlilik Skoru

- A* algoritması daha çok teslimat yaparak yüksek skor elde ederken, GA'nın skoru oldukça düşüktür.

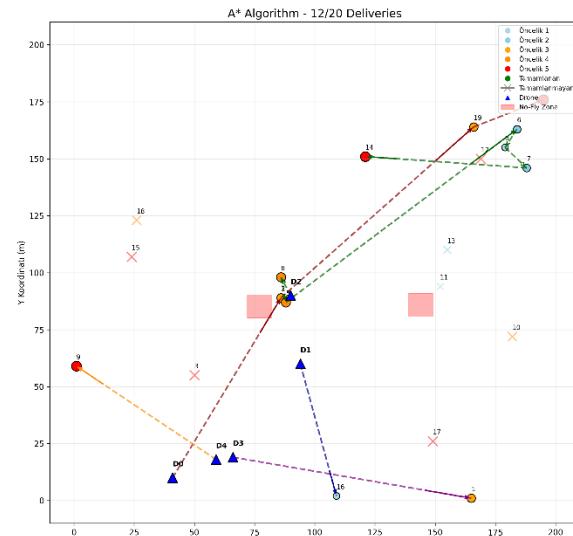


Şekil 12 Algoritmaların Verimlilik Grafiği

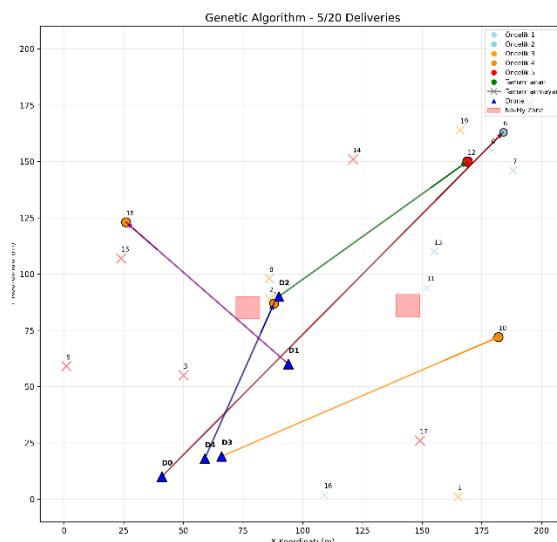
Bu bölümde, A* algoritması ve Genetik Algoritma'nın farklı senaryolar altında davranışları, **teslimat yolları** (rotalar) üzerinden analiz edilmiştir. Her bir teslimat noktası, öncelik düzeyine göre renklendirilmiş ve başarılı teslimatlar yeşil işaretle, başarısızlar ise kırmızı çarpı ile gösterilmiştir. No-fly zone bölgeleri şeffaf kırmızı alanlarla haritaya eklenmiştir.

Senaryo 1: 5 Drone, 20 Teslimat Noktası, 2 No-Fly Zone

- A* algoritması, çoğu teslimatı başarıyla tamamlamış ve drone'ların rotalarını no-fly zone'lardan kaçırarak planlamıştır.
- Genetik Algoritma ise daha az teslimat gerçekleştirmiş ve bazı rotalarda yasa dışı bölgelere tehlikeli yakınlık gözlemlenmiştir.



Şekil 13 Senaryo 1 – A algoritmasına göre çizilen teslimat yolları

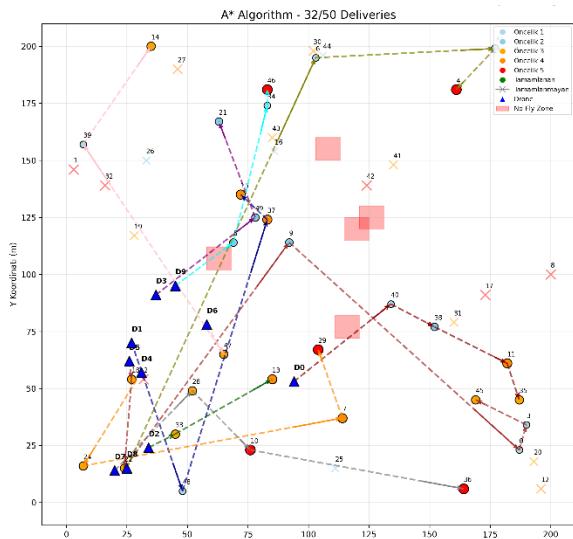


Şekil 14 Senaryo 1 – GA algoritmasına göre çizilen teslimat yolları

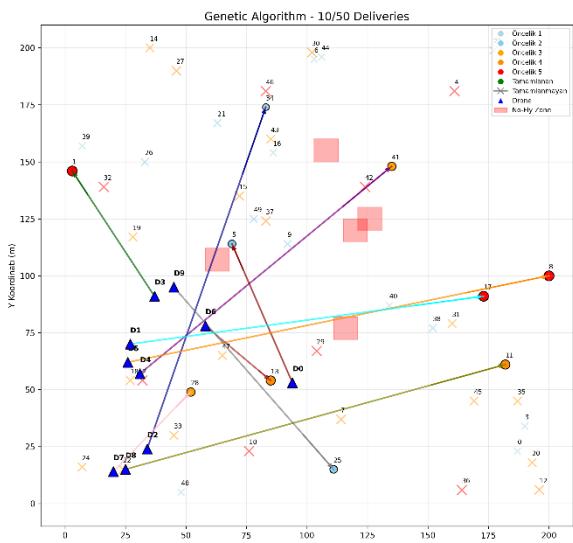
7.4 Drone Rota Görselleştirmeleri

Senaryo 2: 10 Drone, 50 Teslimat Noktası, 5 No-Fly Zone

- A* algoritması, daha büyük senaryo ölçüğünde de yüksek teslimat başarısı sağlamış; rotalar optimize şekilde çizilmiştir.
- Genetik Algoritma yine düşük teslimat oranı ile sınırlı sayıda rota üretmiştir ve bazı drone'lar yalnızca tek görev alabilmistiştir.



Şekil 15 Senaryo 2 – A* algoritmasına göre çizilen teslimat yolları



Şekil 16 Senaryo 2 – GA algoritmasına göre çizilen teslimat yolları

Bu görseller, yalnızca algoritmaların sayısal performanslarını değil, **harita üzerinde operasyonel başarımını** da ortaya koymak açısından oldukça değerlidir. A* algoritması, rota güvenliği ve görev kapsama açısından çok daha etkili bir davranış göstermiştir. Genetik Algoritma

ise farklı parametreler altında iyileştirilmeye açıktır.

7.5 Genel Sonuçlar

1. **A* algoritması***, kısıtlı, gerçekçi ortamlarda tutarlı ve yüksek başarı oranlı çözümler üretmiştir. Zaman ve enerji açısından kabul edilebilir seviyelerdedir.
2. **Genetik Algoritma**, yeterli parametre optimizasyonu yapılmadığında özellikle sıkı kısıtlarla başa çıkmakta zorlanmıştır. Ancak çok hızlı çalışması sayesinde daha gevşek sistemler için uygun bir çözüm olabilir.
3. Enerji tüketimi bakımından A*, GA'ya kıyasla daha fazla enerji harcasa da teslimat başarısı göz önüne alındığında bu durum tolere edilebilirdir.
4. Senaryo bazlı değerlendirmeler, algoritma seçiminin problemin doğasına göre yapılması gerektiğini göstermiştir.

7.6 Gelecek Çalışmalar İçin Öneriler

- Genetik Algoritma için **dinamik parametre ayarlama**, ön çözüm üretimi gibi hibrit yaklaşım test edilmelidir.
- A* algoritması ile **machine learning temelli rota ön tahmini** entegrasyonu performansı artırabilir.
- Simülasyon ortamı daha büyük senaryolarla test edilerek sistemin ölçeklenebilirliği analiz edilebilir.

8. Kaynaklar (References)

Bu projedeği temel kaynaklar:

- [1] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*. MIT press, 2005.
- [2] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT Press, 2004.
- [3] R. Dechter, “Constraint Processing,” *Morgan Kaufmann*, 2003.
- [4] P. Toth and D. Vigo, “The Vehicle Routing Problem,” *Society for Industrial and Applied Mathematics*, vol. 9, no. 2, pp. 1–31, 2002.
- [5] OpenAI, “GPT-4 Technical Report,” *arXiv preprint*, arXiv:2303.08774, 2023.
- [6] Python Software Foundation, “Python Language Reference, version 3.10,” [Online]. Available: <https://www.python.org>

9. Ekler

Ek-A Yardımcı Modüller ve Fonksiyon Açıklamaları

Proje kapsamında ana algoritmaların yanı sıra, sistemi destekleyen birçok yardımcı modül geliştirilmiştir. Bu modüller, kodun yeniden kullanılabilirliğini, test edilebilirliğini ve genişletilebilirliğini artırmak amacıyla modüler şekilde yapılandırılmıştır. Aşağıda, en kritik yardımcı fonksiyonlar ve sınıflar detaylı biçimde açıklanmıştır:

1. battery_check(drone, delivery)

Bu fonksiyon, bir drone'un belirli bir teslimat görevini gerçekleştirebilmesi için yeterli enerjiye sahip olup olmadığını kontrol eder. Fonksiyon, hem mesafe bazlı tüketimi hem de yük, hız ve çevresel faktörleri dikkate alır.

Girdi:

drone: Drone nesnesi

delivery: Teslimat nesnesi

Çıktı:

True / False: Batarya yeterli mi?

İlgili Hesaplama:

python

KopyalaDüzenle

consumption =

Utils.calculate_battery_consumption(drone,
delivery)

return drone.battery_level >= consumption

2. drone_assignment(deliveries, drones)

Tüm teslimat noktaları için uygun drone eşleştirmeğını gerçekleştirir. CSP mantığına dayanır ve aşağıdaki kısıtları gözetir:

Batarya yeterliliği

Zaman penceresi uyumu

Ağırlık kapasitesi

No-fly zone çıkışmaları

Strateji:

- ❖ Her teslimat için uygun drone'lar liste şeklinde alınır.
- ❖ Önceliği yüksek olan teslimatlar önce atanır.
- ❖ Teslimat tamamlandıktan sonra drone durumu güncellenir.

3. visualize_delivery_map(drones, deliveries, no_fly_zones)

Simülasyon sonuçlarını 2B harita üzerinde göstermektedir. Farklı teslimat öncelikleri renklerle kodlanır; başarıyla tamamlananlar, iptal edilenler ve drone pozisyonları ayrı sembollerle gösterilir.

Özellikler:

matplotlib ile harita çizimi

seaborn ile görsel iyileştirmeler

No-fly zone'lar şeffaf kırmızı poligonlarla çizilir

Başarılı teslimatlar: işaretli, başarısızlar:

Drone konumları: ▲ mavi üçgen

Kullanım:

```
visualize_delivery_map(drone_list, delivery_list,  
no_fly_zones)
```

4. calculate_battery_consumption() – (Utils modülü içinde)

Enerji tüketimini 5 farklı değişkene göre hesaplayan gelişmiş bir modeldir:

Hover Süresi

Uçuş Mesafesi

Taşınan Yük

Hız Faktörü

Çevresel Etki (sabittir)

Kodun enerji farkındalığı yüksek bir simülasyon sağlaması için bu modül temel alınmıştır.

5. calculate_heuristic(start, goal)

A* algoritmasında tahmini maliyet hesabını yapar. Kullanılan varsayılan yaklaşım, düz çizgi (Euclidean) mesafedir. Eğer rota bir no-fly zone'a temas ediyorsa, NoFlyPenalty puanı eklenir.

Fonksiyon Gövdesi:

```
def calculate_heuristic(start, goal):  
    distance = euclidean_distance(start, goal)  
    penalty = no_fly_zone_penalty(start, goal)  
    return distance + penalty
```

6. TimeManager Sınıfı

Tüm zaman tabanlı süreçleri yönetir. Özellikle zaman penceresi kısıtlarında teslimatın uygun olup olmadığını kontrol eder.

Özellikleri:

- ❖ Simülasyon zamanını dakika/dakika ilerletir.
- ❖ Teslimat için uygun zaman aralığını kontrol eder.
- ❖ Zaman uyumsuzluğu durumunda drone görevlendirmesini engeller.

Kullanım:

```
if  
time_manager.in_time_window(delivery.time_window):  
    assign_drone(...)
```

Bu modüller, projenin doğru, esnek ve test edilebilir şekilde çalışmasına büyük katkı sağlamıştır. Her biri bağımsız test edilebilir olarak tasarlandığı için, ileride başka optimizasyon problemlerine de kolayca adapte edilebilir.

Ek B – Örnek Veri Seti Çıktısı

Veri başarıyla 'data/dataset.txt' dosyasına kaydedildi.

Dataset başarıyla yüklendi.

Drone sayısı: 3

Teslimat sayısı: 5

No-fly zone sayısı: 3

Drones:

Drone(id=0, max_weight=4.93, battery=6341, speed=9.38, pos=(55, 22), remaining_battery=6341.0, charging=False)
Drone(id=1, max_weight=3.55, battery=9975, speed=5.08, pos=(97, 47), remaining_battery=9975.0, charging=False)
Drone(id=2, max_weight=3.49, battery=6562, speed=8.07, pos=(44, 7),

Deliveries:

Delivery(id=0, pos=(64, 126), weight=2.13, priority=1)
Delivery(id=1, pos=(189, 143), weight=1.13, priority=5)
Delivery(id=2, pos=(18, 171), weight=1.55, priority=3)
Delivery(id=3, pos=(46, 62), weight=1.33, priority=3)
Delivery(id=4, pos=(115, 60), weight=2.33, priority=3)

No-Fly Zones:

NoFlyZone(id=0, active=(datetime.time(9, 17), datetime.time(10, 28)))
NoFlyZone(id=1, active=(datetime.time(12, 8), datetime.time(13, 15)))
NoFlyZone(id=2, active=(datetime.time(9, 15), datetime.time(10, 6)))

Ek B – Örnek DeliveryManager Delivery Priority Çıktısı

Aciliyet: 5 | Teslimat ID: 9

Aciliyet: 4 | Teslimat ID: 12

Aciliyet: 4 | Teslimat ID: 18

Aciliyet: 4 | Teslimat ID: 41

Aciliyet: 3 | Teslimat ID: 2

Aciliyet: 2 | Teslimat ID: 48

Aciliyet: 1 | Teslimat ID: 0

Ek C – Örnek Sistem Test Çıktıları

Test Bloğu Çıktısı:

SİSTEM TESTİ BAŞLIYOR...

Mesafe testi: (0, 0) -> (3, 4) = 5.00 metre

Polygon testi: (5, 5) polygon içinde mi? True

Zaman: 09:00:00

30 dk sonra: 09:30:00

Graf testi: 50 teslimat noktası ile graf oluşturuluyor...

Graf oluşturuldu! Depot'un komşu sayısı: 5

A* testi: Depot -> Teslimat 0

Rota: ['depot', 0]

Maliyet: 264.60

Tam veri ile Csp Testi Çıktısı:

TAM VERİ SETİ İLE CSP TESTİ:

TAM VERİ İSTATİSTİKLERİ:

Drone sayısı: 10

Teslimat sayısı: 50

No-fly zone sayısı: 5

CSP değişken sayısı: 360

CSP kısıt sayısı: 57

TAM CSP çözülüyor...

CSP Çözümü (ESNEK MOD, Tolerans: 120dk):

Drone 0 -> Teslimat 0

Ağırlık: 2.13kg, Öncelik: 1, Durum: YAKLAŞAN

Drone 1 -> Teslimat 1

Ağırlık: 1.13kg, Öncelik: 5, Durum: YAKLAŞAN

Ek D – Örnek Senaryo 1 ve Senaryo 2 Sonuç Çıktıları

SENARYO 1 TEST SONUÇLARI (5 drone, 20 teslimat, 2 no-fly zone):

A* Algoritması: 14/20 teslimat, 1.63 saniye

GA Algoritması: 5/20 teslimat, 1.55 saniye

SENARYO 2 TEST SONUÇLARI (10 drone, 50 teslimat, 5 no-fly zone):

A* Algoritması: 34/50 teslimat, 3.50 saniye

GA Algoritması: 10/50 teslimat, 4.65 saniye

PERFORMANS KRİTERİ KONTROLÜ (50+ teslimat için <1 dakika):

A* Algoritması: BAŞARILI (3.50 saniye)

GA Algoritması: BAŞARILI (4.65 saniye)

Ek E – Proje Linkleri Ve Çalıştırma Bilgileri

Proje Not Defteri (Google Colab):

<https://colab.research.google.com/drive/1wublP0dxUKPdEq1xTyb6ALofQE7kx32?usp=sharing#scrollTo=mUKvpdpK5zR2>

Çalıştmak için:

- Python 3.10+

- Google Colab ortamı veya Jupyter Notebook

- Gereken kütüphaneler: numpy, pandas, matplotlib.pyplot, random, time, math, heapq, copy