BURSA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERSİN ADI Algoritma Analizi Ve Tasarımı

ÖDEV KONUSU: Kaba Kuvvet (Brute Force) Ve Graham Scan Algoritmaları İle Kapalı Çevrim (Convex Hull) Probleminin Teorik Ve Deneysel Analizi

ÖĞRENCİ NUMARASI: 23360859083

ÖĞRENCİ ADI SOYADI: Yağmur Eraslan

TESLİM TÜRÜ: Rapor + Canlı Uygulama Gösterimi

TESLİM TARİHİ: 08.01.2026

SORUMLU ÖĞRETİM ELEMANI: Dr. Öğretim Üyesi Seçkin YILMAZ

1. UYGULAMANIN GENEL MİMARİSİ VE TASARIMI

Bu çalışmada, kapalı çevrim (Convex Hull) probleminin çözümü için iki farklı algoritmanın karşılaştırılmasını sağlayan **grafik tabanlı bir masaüstü uygulaması** geliştirilmiştir. Uygulama Python programlama dili kullanılarak oluşturulmuş olup, kullanıcı etkileşimini sağlamak amacıyla **Tkinter** grafik arayüz kütüphanesi tercih edilmiştir.

Uygulamanın temel bileşenleri şu şekildedir:

- **Kontrol Paneli:**
  Kullanıcıdan nokta sayısının (N) alınması, yeni rastgele nokta üretilmesi ve algoritmaların çalıştırılması bu panel üzerinden gerçekleştirilir.
- **Çizim Alanı (Canvas):**
  Rastgele oluşturulan noktalar ve hesaplanan kapalı çevrim (convex hull) sonuçları, Matplotlib kütüphanesi aracılığıyla görselleştirilir.
- **Zaman Ölçüm Mekanizması:**
  Algoritmaların çalışma süreleri `time.perf_counter()` fonksiyonu kullanılarak hassas biçimde ölçülür ve milisaniye cinsinden kullanıcıya sunulur.

Bu yapı sayesinde, teorik karmaşıklık analizlerinin deneysel sonuçlarla doğrudan karşılaştırılması mümkün hâle gelmiştir.

ŞEKİL 1 - GUI ARAYÜZÜ

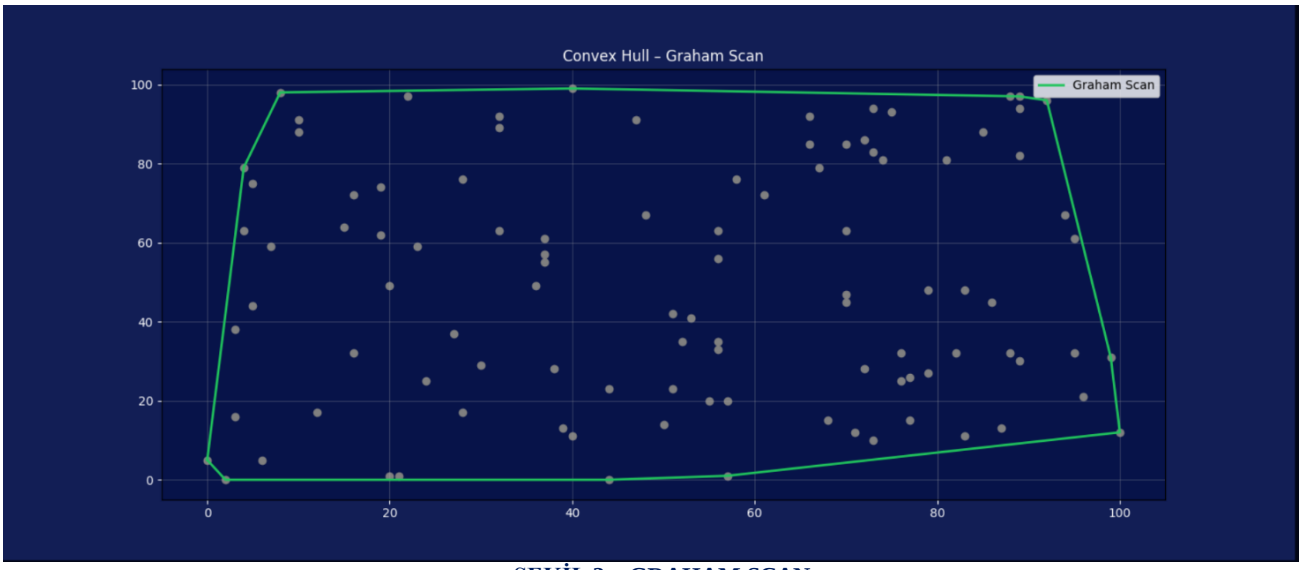
Şekil 1. Uygulamanın grafik arayüzü; rastgele oluşturulan nokta kümesi ve kullanıcı etkileşimini sağlayan kontrol bileşenleri görülmektedir.

2. UYGULAMANIN ÇALIŞMA AKIŞI

Uygulama başlatıldığında, kullanıcı tarafından belirlenen nokta sayısına göre iki boyutlu düzlem üzerinde rastgele noktalar oluşturulur. Bu noktalar mavi renk ile çizim alanında gösterilir. Kullanıcı, ilgili butonlar aracılığıyla aşağıdaki işlemleri gerçekleştirebilir:

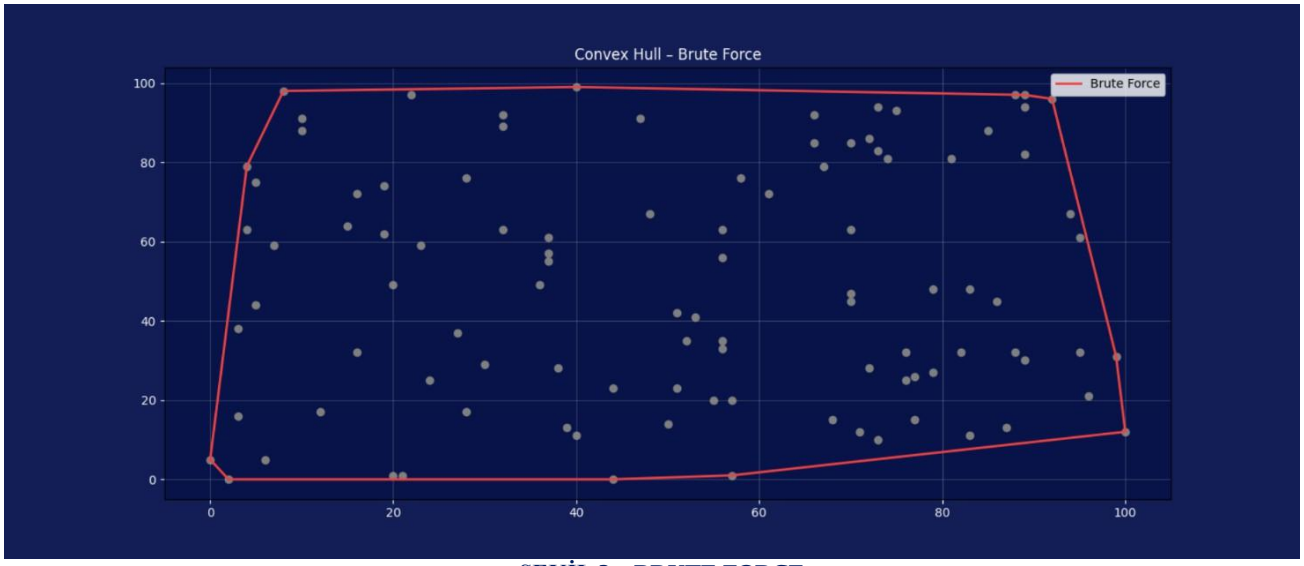
1. **Yeni Noktalar Oluşturma:**
   Mevcut nokta kümesi silinir ve aynı düzlemde yeni rastgele noktalar üretilir.
2. **Graham Scan Algoritmasını Çalıştırma:**
   Nokta kümesi üzerinde Graham Scan algoritması uygulanır ve elde edilen kapalı çevrim yeşil renk ile çizilir.
3. **Brute Force Algoritmasını Çalıştırma:**
   Tüm nokta çiftleri değerlendirilerek kapalı çevrim hesaplanır ve sonuç kırmızı renk ile gösterilir.
4. **Performans Testi:**
   Farklı N değerleri için her iki algoritmanın çalışma süreleri ölçülerek karşılaştırmalı bir performans grafiği oluşturulur.

Bu akış sayesinde, kullanıcı hem algoritmaların görsel çıktısını hem de performans farklarını eş zamanlı olarak inceleyebilmektedir.



ŞEKİL 2 – GRAHAM SCAN

Şekil 2. Graham Scan algoritması kullanılarak hesaplanan kapalı çevrim sonucu; dışbükey yapı yeşil renkli çizgi ile gösterilmiştir.



ŞEKİL 3 – BRUTE FORCE

Şekil 3. Kaba Kuvvet (Brute Force) yaklaşımı ile elde edilen kapalı çevrim sonucu; hesaplanan dışbükey yapı kırmızı renkli çizgi ile ifade edilmiştir.

3. ORIENTATION (YÖN TESTİ) FONKSİYONUNUN ROLÜ

Convex Hull probleminin temelinde, üç noktanın birbiriyle oluşturduğu yön (orientation) ilişkisi yer almaktadır. Bu çalışmada kullanılan `orientation(p, q, r)` fonksiyonu, üç noktanın saat yönünde mi, saat yönünün tersinde mi yoksa doğrusal mı olduğunu belirlemek için kullanılmaktadır.

Matematiksel olarak bu işlem, iki boyutlu vektörler arasındaki çapraz çarpım (cross product) mantığına dayanmaktadır:

- Pozitif değer → Saat yönünün tersine dönüş (counter-clockwise)
- Negatif değer → Saat yönünde dönüş (clockwise)
- Sıfır → Doğrusal (collinear)

Bu yön testi, Graham Scan algoritmasında içbükey noktaların elenmesini sağlarken, Brute Force yaklaşımında bir doğru parçasının kapalı çevrime ait olup olmadığının belirlenmesinde kritik rol oynamaktadır.

4. EN İYİ DURUM – EN KÖTÜ DURUM KARŞILAŞTIRMASI

Graham Scan algoritması için en iyi durum (best case), noktaların zaten dışbükey bir yapı oluşturduğu senaryodur. Ancak bu durumda dahi, noktaların başlangıçta sıralanması gerektiğinden zaman karmaşıklığı O(N log N) olarak kalmaktadır.

En kötü durumda (worst case) ise tüm noktalar kapalı çevrime dahil olur. Buna rağmen, eleme işlemleri doğrusal zamanlı olduğu için algoritmanın baskın maliyeti yine sıralama adımıdır.

Brute Force algoritmasında ise hem en iyi hem de en kötü durumda üçlü döngü yapısı değişmediğinden zaman karmaşıklığı $O(N^3)$ olarak kalmaktadır. Bu durum, algoritmanın pratik kullanımını ciddi biçimde sınırlandırmaktadır.

5. PERFORMANS GRAFİĞİ VE DENEYSEL BULGULAR

Deneysel performans testleri, farklı nokta sayıları (N = 50, 100, 200, 300, 400, 500) için gerçekleştirilmiştir. Elde edilen sonuçlar, iki algoritmanın teorik karmaşıklıklarıyla birebir uyumlu olduğunu göstermektedir.

Özellikle N değeri arttıkça:

- Graham Scan algoritmasının çalışma süresi kontrollü ve yavaş bir artış göstermiştir.
- Brute Force algoritmasının çalışma süresi ise kübik artış nedeniyle çok hızlı yükselmiştir.

Bu sonuçlar, Graham Scan algoritmasının büyük veri kümeleri için tercih edilmesi gerektiğini açıkça ortaya koymaktadır.

6. KABA KUVVET (BRUTE FORCE) YAKLAŞIMINA YÖNELİK TEORİK VE UYGULAMA TEMELLİ HESAPLAMA KARMAŞIKLIĞI ANALİZİ

Teorik olarak algoritmanın çalışma süresi şu şekildedir:

- Nokta çiftlerinin sayısı $O(N^2)$'dir.
- Her nokta çifti için tüm noktalar tekrar kontrol edildiğinden ek olarak O(N) maliyet oluşur.

Bu nedenle Kaba Kuvvet algoritmasının **toplam zaman karmaşıklığı**: $O(N^3)$

şeklindedir.

Uygulamada, bu durum iç içe döngülerle açıkça gözlemlenmektedir. Her `(p, q)` nokta çifti için üçüncü bir döngü ile tüm noktalar kontrol edilmekte, bu da algoritmanın özellikle büyük N değerlerinde pratikte çalışamaz hâle gelmesine neden olmaktadır.

Kaba Kuvvet yaklaşımında hesaplama maliyetini artıran temel unsurlar şunlardır:

- Nokta çiftlerinin karesel sayıda olması
- Her çift için tüm noktaların tekrar kontrol edilmesi
- Gereksiz kenar adaylarının elenmesinin geç aşamada yapılması

Bu nedenlerle Kaba Kuvvet yöntemi, teorik olarak basit olmasına rağmen deneysel olarak düşük performans göstermektedir.

7. GRAHAM SCAN ALGORİTMASINA YÖNELİK TEORİK VE UYGULAMA TEMELLİ HESAPLAMA KARMAŞIKLIĞI ANALİZİ

Graham Scan algoritması, kapalı çevrim problemini daha verimli şekilde çözmeyi amaçlayan bir yöntemdir. Algoritma, öncelikle noktaları belirli bir kritere göre sıralar ve ardından yalnızca dışbükey yapıyı bozan noktaları elemek suretiyle kapalı çevrimi oluşturur.

Algoritmanın temel adımları şu şekildedir:

1. Noktaların sıralanması
2. Alt ve üst kapalı çevrimin oluşturulması
3. İçbükeylik oluşturan noktaların yığından çıkarılması

Noktaların sıralanması adımı O(NlogN) zaman alırken, kalan işlemler doğrusal zamanlıdır. Bu nedenle Graham Scan algoritmasının **toplam zaman karmaşıklığı**:

O(NlogN) şeklindedir.

Graham Scan algoritması, Kaba Kuvvet yaklaşımına kıyasla çok daha hızlı çalışmaktadır. Bunun temel nedeni, gereksiz kenar adaylarının baştan elenmesi ve yalnızca potansiyel dışbükey noktaların değerlendirilmesidir.

En iyi durumda (best case) noktalar zaten dışbükey bir yapı oluşturuyorsa, algoritma minimum eleme işlemi yapar ancak sıralama adımı nedeniyle karmaşıklık yine O(NlogN) olarak kalır. En kötü durumda (worst case) ise tüm noktalar kapalı çevrime dahil olsa bile sıralama adımı baskın olduğundan karmaşıklık değişmez.

8. PERFORMANS TESTİ VE GRAFİKSEL KARŞILAŞTIRMA

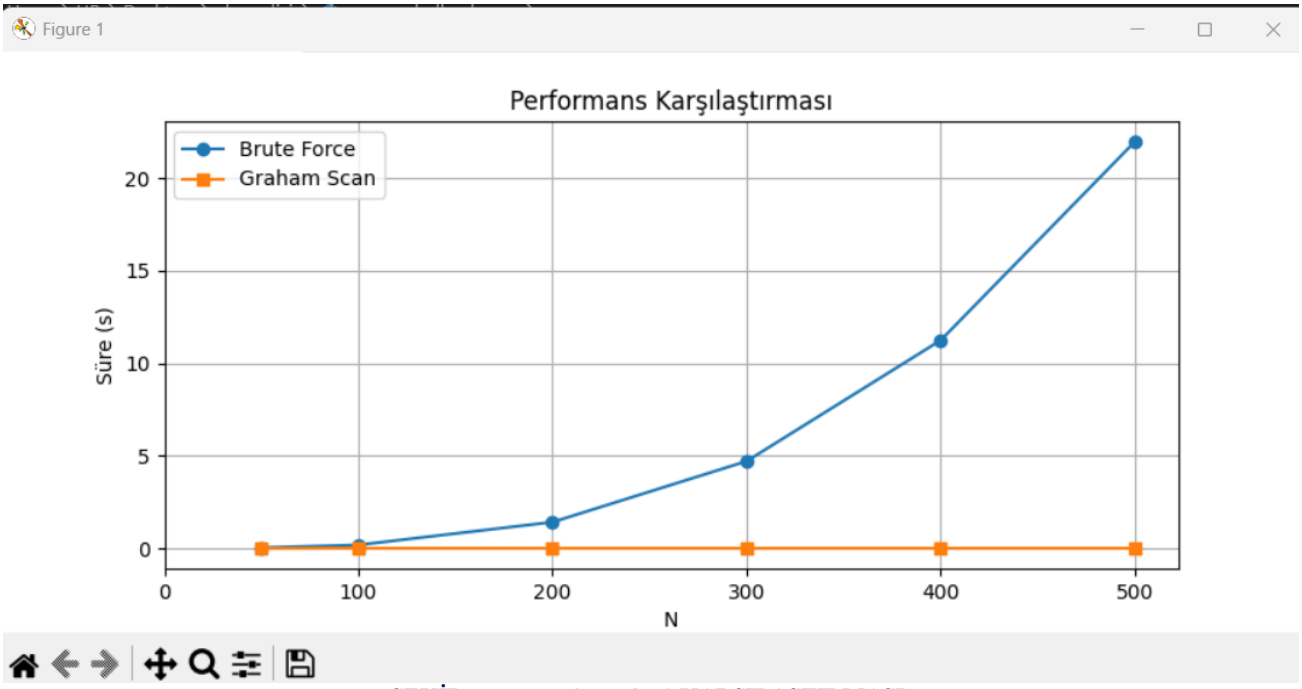
Bu çalışmada, Kaba Kuvvet ve Graham Scan algoritmalarının performansları deneysel olarak karşılaştırılmıştır. Farklı N değerleri için her iki algoritmanın kapalı çevrimi bulma süreleri ölçülmüş ve sonuçlar aynı grafik üzerinde gösterilmiştir.

Grafiklerde:

- X ekseni: Nokta sayısı (N)
- Y ekseni: Çalışma süresi (saniye)

olarak belirlenmiştir.

Deneysel sonuçlar incelendiğinde, Graham Scan algoritmasının çalışma süresinin N arttıkça logaritmik bir artış gösterdiği, Kaba Kuvvet algoritmasının ise kübik artış nedeniyle kısa sürede pratikte kullanılamaz hâle geldiği gözlemlenmiştir. Özellikle büyük N değerlerinde Kaba Kuvvet algoritmasının çalışma süresi belirgin biçimde artmış ve bu durum teorik karmaşıklık analizi ile örtüşmüştür.



ŞEKİL 4 – PERFORMANS KARŞILAŞTIRMASI

Şekil 4. Graham Scan ve Brute Force algoritmalarının farklı nokta sayıları için ölçülen çalışma sürelerinin karşılaştırılması.

9. UYGULAMA GELİŞTİRMEDE YAPILAN LLM KULLANIMI

Bu bölümde, uygulama geliştirme sürecinde ChatGPT, Grok, Gemini gibi Büyük Dil Modelleri (LLM) tabanlı araçlardan hangi aşama veya aşamalarda yararlanıldığı açıkça belirtilmelidir. LLM'lerin; algoritma tasarımı, kodlama, hata ayıklama, teorik analiz, rapor yazımı veya performans değerlendirmesi gibi süreçlerde nasıl ve ne amaçla kullanıldığı detaylandırılmalıdır.
Kullanılan örnek prompt'lar ve LLM'lerden elde edilen yanıtların kısa özetleri bu bölümde yer almalıdır.
Bu açıklamalar, LLM tabanlı araçların birer destek ve öğrenme aracı olarak nasıl kullanıldığını gösterecek şekilde, şeffaf ve doğrulanabilir biçimde sunulmalıdır.

Bu çalışmanın geliştirme sürecinde ChatGPT gibi Büyük Dil Modeli (LLM) tabanlı araçlardan destek alınmıştır. LLM'ler; algoritmaların teorik karmaşıklık analizlerinin doğrulanması, Python kodlarının düzenlenmesi, grafik arayüz tasarımının iyileştirilmesi ve rapor metninin akademik dilde hazırlanması aşamalarında yardımcı araç olarak kullanılmıştır.

LLM'ler, nihai karar verici veya otomatik kod üretici olarak değil; öğrenme sürecini destekleyen, açıklayıcı ve yönlendirici bir kaynak olarak değerlendirilmiştir.