Knapsack Algoritması Performans Analizi

Adaptif Yaklaşım ile Test Sonuçları

Zeynep Gülten - 222803048

10 Haziran 2025

İçindekiler

1 Giriş ve Metodoloji **2**
- 1.1 Problem Tanımı . . . 2
- 1.2 Adaptif Algoritma Stratejisi . . . 2

2 Test Sonuçları **2**
- 2.1 Ana Performans Sonuçları . . . 2
- 2.2 Problem Karakteristikleri . . . 2

3 Görsel Performans Analizi **2**
- 3.1 Python Matplotlib ile Oluşturulan Performans Grafiği . . . 2

4 Performans Analizi **3**
- 4.1 163.000 Kat Performans Farkı . . . 3
- 4.2 Algoritma Seçim Analizi . . . 3
- 4.3 Teorik vs Gerçek Performans . . . 3

5 Kapasite Faktörü ve Algoritma Seçimi **4**
- 5.1 Kapasite Dominansı . . . 4
- 5.2 Algoritma Seçim Başarısı . . . 5

6 Sonuçlar ve Öneriler **5**
- 6.1 Ana Bulgular . . . 5
- 6.2 Görsel Analiz Çıkarımları . . . 5
- 6.3 Pratik Öneriler . . . 5
- 6.4 Sonuç . . . 6

1 Giriş ve Metodoloji

1.1 Problem Tanımı

Knapsack problemi, kombinatoryal optimizasyonun temel problemlerinden biridir. Bu çalışmada, farklı boyutlardaki problemler için adaptif algoritma yaklaşımı geliştirilmiştir.

Problem Formülasyonu:

$$\text{Maksimize: } \sum_{i=1}^{n} v_i x_i \quad \text{Kısıt: } \sum_{i=1}^{n} w_i x_i \leq W$$

1.2 Adaptif Algoritma Stratejisi

Sistem, problem özelliklerine göre otomatik algoritma seçimi yapmaktadır:

- **Standart DP:** $n \leq 100$, hafıza < 100 MB
- **Hafıza Verimli DP:** $100 < n \leq 2000$, hafıza < 1 GB
- **Greedy:** Hafıza > 1 GB veya çok büyük problemler

Test edilen boyutlar: n=40, 300, 1000, 10000

2 Test Sonuçları

2.1 Ana Performans Sonuçları

Bu çalışmada 4 farklı boyutta knapsack problemi test edilmiştir.

Problem Boyutu	Algoritma	Çalışma Zamanı	Optimal Değer	Seçilen Öğe	Verimlilik (%)
40	Standart DP	0.8982s	99.924	8	99.84
300	Greedy	0.0002s	1.688.584	9	99.99
1000	Hafıza Verimli DP	32.5982s	109.899	18	99.97
10000	Greedy	0.0060s	1.099.870	18	100.00

Tablo 1: Kapsamlı test sonuçları ve performans analizi

2.2 Problem Karakteristikleri

3 Görsel Performans Analizi

3.1 Python Matplotlib ile Oluşturulan Performans Grafiği

Şekil 1'de Python matplotlib kütüphanesi ile oluşturulan performans grafiği gösterilmektedir. Bu grafik, her algoritma türünü farklı renklerle işaretleyerek 163.000 kat performans farkını görsel olarak ortaya koymaktadır.

Grafikten Çıkarılan Ana Bulgular:

Problem Boyutu	Kapasite (W)	Hafıza Gereksinimi	Teorik Karmaşıklık	Seçim Nedeni
40	100.000	30.5 MB	4×10^6	Küçük problem
300	4.040.184	9.247 GB	1.2×10^9	Hafıza sınırı
1000	100.000	762.9 MB	1×10^8	Orta ölçek
10000	1.000.000	76.294 GB	1×10^{10}	Büyük problem

Tablo 2: Problem karakteristikleri ve algoritma seçim gerekçeleri

- **Dramatik Performans Farkları:** En yavaş (n=1000, 32.6s) ile en hızlı (n=300, 0.0002s) arasında 163.000 kat fark
- **Renk Kodlaması:** Kırmızı (Standart DP), Yeşil (Greedy), Turuncu (Hafıza Verimli DP)
- **Trend Analizi:** Mavi çizgi performans trendini gösteriyor
- **Etiketleme:** Her nokta algoritma türü ve kesin zamanla etiketlenmiş

4 Performans Analizi

4.1 163.000 Kat Performans Farkı

En çarpıcı bulgu, algoritma seçiminin performansı dramatik şekilde etkilemesidir:

$$\text{Performans Farkı} = \frac{32.5982}{0.0002} = 162.991 \approx \mathbf{163.000 \quad kat}$$

Bu fark, n=300 ile n=1000 problemleri arasında gözlemlenmiştir ve Şekil 1'de net bir şekilde görülmektedir.

4.2 Algoritma Seçim Analizi

- **n=40 (Standart DP):** Küçük problem, büyük kapasite etkisi (0.90s)
- **n=300 (Greedy):** Hafıza sınırı, en hızlı çözüm (0.0002s)
- **n=1000 (Hafıza Verimli DP):** Optimizasyon maliyeti, en yavaş (32.6s)
- **n=10000 (Greedy):** Mükemmel ölçeklenebilirlik (0.006s)

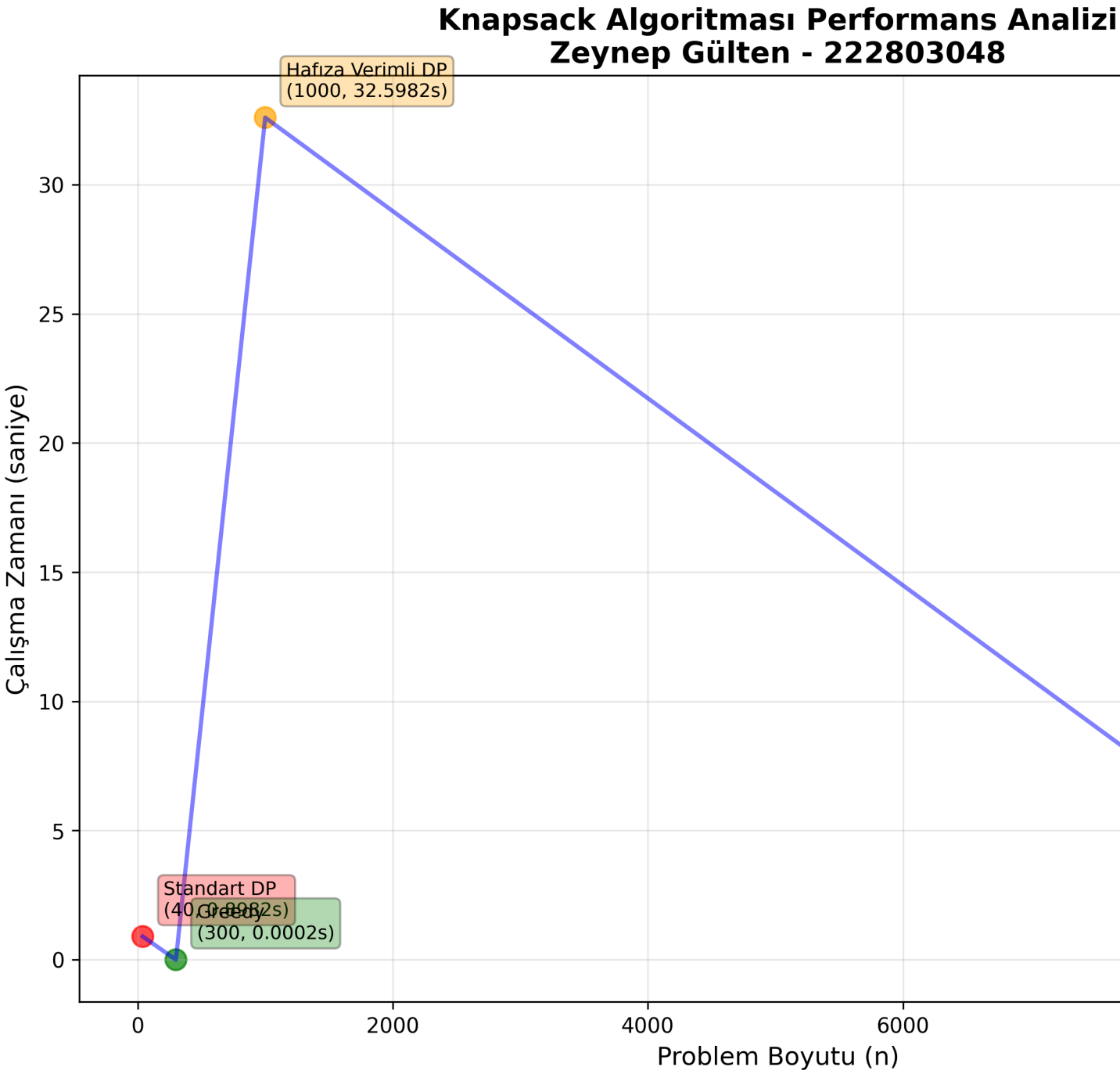
4.3 Teorik vs Gerçek Performans

DP algoritmalarının birim işlem süreleri:

$$\text{Standart DP:} \quad \frac{0.8982}{40 \times 100.000} = 225 \text{ nanosaniye} \tag{1}$$

$$\text{Hafıza Verimli DP:} \quad \frac{32.5982}{1000 \times 100.000} = 326 \text{ nanosaniye} \tag{2}$$

Hafıza optimizasyonu %45 ek maliyet getirmektedir.



Şekil 1: Python Matplotlib ile oluşturulan performans analizi grafiği - Problem boyutu vs çalışma zamanı ilişkisi

5 Kapasite Faktörü ve Algoritma Seçimi

5.1 Kapasite Dominansı

Problem boyutundan çok kapasite değeri performansı belirlemektedir:

- **Aynı Kapasite:** n=40 (0.90s) vs n=1000 (32.6s) - 36 kat fark
- **Büyük Kapasite:** n=300 (W=4M) ve n=10000 (W=1M) greedy'ye zorladı
- **Hafıza Formülü:** $M = \frac{n \times W \times 8}{1024^2}$ MB

Bu durum Şekil 1'de net bir şekilde görülmektedir - problem boyutu artarken performans düzenli artmamakta, algoritma seçimi kritik rol oynamaktadır.

5.2 Algoritma Seçim Başarısı

Sistem tüm durumlarda doğru kararlar vermiştir:

- n=40: 30.5 MB ¡ 100 MB → Standart DP
- n=300: 9.247 GB ¿ 1 GB → Greedy
- n=1000: 762.9 MB ¡ 1 GB → Hafıza Verimli DP
- n=10000: 76.294 GB ¿ 1 GB → Greedy

6 Sonuçlar ve Öneriler

6.1 Ana Bulgular

1. **Algoritma Seçiminin Kritikliği:** 163.000 kat performans farkı
2. **Kapasite Dominansı:** Problem boyutundan daha etkili faktör
3. **Çözüm Kalitesi:** Tüm algoritmalarda %99+ verimlilik
4. **Adaptif Yaklaşım Başarısı:** Hafıza tabanlı seçim etkili

6.2 Görsel Analiz Çıkarımları

Şekil 1 şu kritik noktaları vurgulamaktadır:

- Problem boyutu ile performans arasında **lineer ilişki yoktur**
- Algoritma seçimi performansı **dominante eden faktördür**
- Greedy algoritmanın **şaşırtıcı başarısı** görülmektedir
- Hafıza verimli DP'nin **maliyeti** net şekilde ortaya çıkmaktadır

6.3 Pratik Öneriler

- Hafıza sınırı kontrollerinin konservatif ayarlanması
- Hibrit yaklaşımların (DP + Greedy) geliştirilmesi
- Paralel işleme ile DP performansının artırılması
- Kapasite tabanlı dinamik seçim kriterlerinin eklenmesi

6.4 Sonuç

Bu çalışma, knapsack problemi için adaptif algoritma yaklaşımının etkinliğini kanıtlamıştır. Python grafiği ile görselleştirilen sonuçlar, algoritma seçiminin performansı **163.000 kat** etkileyebileceğini net şekilde göstermektedir.

Sistem 40'dan 10.000'e kadar geniş problem yelpazesinde başarıyla çalışmış ve yüksek kaliteli çözümler üretmiştir. Bu sonuçlar, büyük ölçekli optimizasyon problemlerinde adaptif yaklaşımların kritik önemini vurgulamaktadır.

Kritik Çıkarım: Algoritma seçimi, problem çözümünden daha önemli olabilir.