



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Diskrete Optimierung mit Dynamischer Programming

Bachelor Medieninformatik
Wintersemester 2019/20

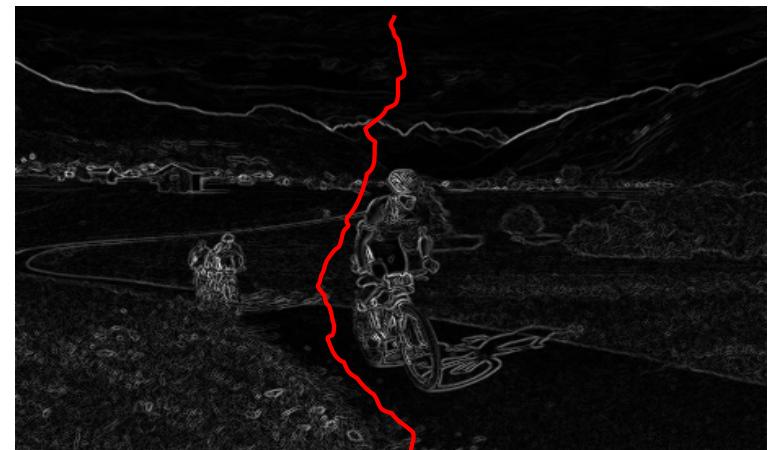
Prof. Dr.-Ing. Kristian Hildebrand
khildebrand@beuth-hochschule.de

Lernziele

- Recap
 - Floating Point
 - Lineare Algebra
 - Bildverarbeitung
- Einführung in Optimierung
- Dynamische Programmierung am Beispiel des Knapsack Problems

Ziel: Entfernen eines Pixelpfades

- Was macht einen geeigneten Pixelpfad aus?
 - Wo sind wichtige Änderungen im Bild?



Welche Informationen im Bild sind wichtig?
Wo ist die geringste Energie, um einen Pfad zu entfernen?

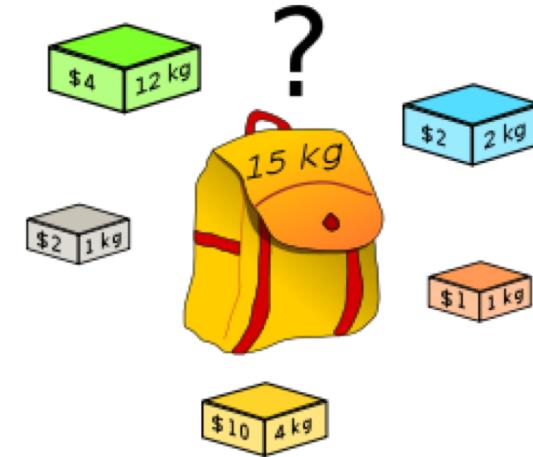
- Wie findet man einen optimalen Pixelpfad schnell?
 - Finden des optimalen Pfades durch Dynamische Programmierung

1D - Knapsack Problem



1D Knapsack Problem

- **Gegeben ein Set von Objekten I**, mit $i \in I$, wobei für jedes Objekt gilt:
 - Gewicht / Größe w_i
 - Wert v_i
- **Gesamtkapazität K** für den Rucksack
- **Ziel:** Finden Sie das Set von Objekten (Subset?), dass den größten Wert hat und in den Rucksack passt
 - Maximiere den Wert des Rucksacks



Optimization Model (Optimierungsmodell)

- Wie modelliert man dieses Problem als Optimierungsproblem?
 - **Auswahl der Entscheidungsvariablen**
 - typischerweise das Resultat, für das wir uns interessieren
 - **Auswahl der Bedingungen für die Entscheidungsvariablen (Constraints)**
 - spezifizieren die Lösungen zu dem Problem
 - **Zielfunktion (Objective Function)**
 - drückt die Qualität jeder Lösung aus
- **Wie stellt man ein gutes Optimierungsmodell auf?**
- **Optimierungsmodell** **Wie löst man ein Optimierungsmodell effizient?**
 - Was wollen wir optimieren?!

Knapsack Optimierungsmodell

■ Entscheidungsvariablen

- x_i sagt ob ein Objekt i ausgesucht wird oder nicht
 - $x_i = 1$ wird ausgesucht
 - $x_i = 0$ wird nicht ausgesucht

■ Problembedingungen

- ausgesuchtes Objekt darf nicht die Kapazität des Rucksacks übersteigen

$$\sum_{i \in I} w_i x_i \leq K$$

■ Zielfunktion

- Summe aller Werte der ausgewählten Objekte
- soll maximiert werden

$$\sum_{i \in I} v_i x_i$$

Knapsack Optimierungsmodell

maximiere $\sum_{i \in I} v_i x_i$

unter der Bedingung $\sum_{i \in I} w_i x_i \leq K$

$$x_i \in \{0, 1\} \quad (i \in I)$$

Exponentielles Wachstum

- **Wieviele mögliche Konfigurationen gibt es?**
 - $(0,0,0,\dots,0), (1,0,0,\dots,0), (1,1,0,\dots,0) \dots (1,1,1,\dots,1)$
 - Auswahl der Konfiguration die Bedingung erfüllt und Zielfunktion maximiert **ohne** alle durchzuprobieren
 - Manche Konfigurationen passen nicht
- **Wieviele gibt es?**
 - $2^{|I|}$ **Exponentielles Wachstum!!!**
- **Wie lange dauert es alle auszuprobieren?**
 - 1 ms zur Berechnung einer Konfiguration
 - $|I| = 50$ ergibt dann ~ 35000 Jahre!

Dynamische Programmierung

Dynamische Programmierung

- **Was ist dynamisches Programmieren?**
 - häufig genutzte Optimierungstechnik
 - wenn es anwendbar ist, dann ist es supergut!!
- **Wann ist es anwendbar?**
 - Computergrafik, Computer Vision, Computational Biology etc.
- **Einfaches Prinzip:**
 - Divide and Conquer
 - Bottom-up Berechnungen

Was ist dynamisches Programmieren?

- **1957 von R. Bellmann definiert für mathematische Optimierungsprobleme:**
 - die rekursiv beschreibbar sind,
 - die dem Optimalitätsprinzip genügen, d.h. eine optimale Lösung für das Ausgangsproblem setzt sich aus optimalen Lösungen für die kleineren Probleme, auf die rekurriert wird, zusammen,
 - die zur Berechnung mehrfach gleiche Teillösungen benötigen.

Knapsack Optimierungsmodell

$$\text{maximiere } \sum_{i \in I} v_i x_i$$

$$\text{unter der Bedingung } \sum_{i \in I} w_i x_i \leq K$$

$$x_i \in \{0, 1\} \quad (i \in I)$$

- **Notation:**
 - $I = \{1, 2, \dots, n\}$
 - $O(k, j)$ gibt mir die optimale Lösung des Knapsack Problems für Kapazität k und Objekte $[1, \dots, j]$
- **Wollen die beste Lösung für $O(K, n)$ finden**

Schlüssel zur Lösung

Rekursionsbeziehung (Bellmann Gleichung)

- **Angenommen wir wissen die Lösung für**
 - $O(k,j-1)$ für alle k in $0..K$
 - **Wollen die Lösung für $O(k,j)$ finden:**
 - also nur für ein Objekt mehr – Objekt j
 - wenn $w_j > k$ bleibt die Lösung $O(k,j-1)$
 - wenn $w_j \leq k$ gibt es zwei Fälle:
 - wir selektieren Objekt j nicht, dann bleibt die Lösung $O(k,j-1)$
 - wir selektieren Objekt j und die beste Lösung ist $v_j + O(k-w_j, j-1)$
- $O(k,j) = \max(O(k,j-1), v_j + O(k-w_j, j-1))$ wenn $w_j \leq k$
 - $O(k,j) = O(k,j-1)$

Rekursionsbeziehung

- Einfaches Programm mit top-down Ansatz:

```
def O(k,j):  
    if j == 0:  
        return 0  
    elif wi < k:  
        return max( O(k,j-1), vj + O(k-wj,j-1) )  
    else:  
        return O(k,j-1)
```

- Wie effizient ist diese Lösung?
 - Analogie – Fibonacci Zahlen

Rekursionsbeziehung – Fibonacci Zahlen

- Einfaches Programm um Fibonacci Zahlen zu finden:

```
def fib(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return fib(n-2) + fib(n-1)
```

- Ist das effizient?
 - Lösen wiederholt das gleiche Teilprobleme
 - $\text{fib}(n-1)$ benötigt $\text{fib}(n-2)$ – das haben wir bereits gelöst
 - $\text{fib}(n-3)$ benötigt $\text{fib}(n-4)$ – das haben wir bereits gelöst
 - ...

Dynamische Programmierung (Knapsack)

- **Berechnung der Rekursion Bottom-up**
 - Erst mit 0 Objekten,
 - dann mit 1 Objekt,
 - dann 2 Objekte
 - ...
- **Bauen eine Tabelle (Lookup-Table) mit Lösungen des Teilproblems**

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1
0	0	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	

$$v_1 = 5$$

$$w_1 = 4$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1
0	0	0
1	0	0
2	0	0
3	0	0
4	0	5
5	0	5
6	0	5
7	0	5
8	0	5
9	0	5

$$v_1 = 5$$

$$w_1 = 4$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	5	
5	0	5	
6	0	5	
7	0	5	
8	0	5	
9	0	5	

$$\begin{array}{ll} v_1 = 5 & v_2 = 6 \\ w_1 = 4 & w_2 = 5 \end{array}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	5	5
5	0	5	
6	0	5	
7	0	5	
8	0	5	
9	0	5	

$$\begin{array}{ll} v_1 = 5 & v_2 = 6 \\ w_1 = 4 & w_2 = 5 \end{array}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	5	5
5	0	5	6
6	0	5	6
7	0	5	6
8	0	5	6
9	0	5	

$$\begin{array}{ll} v_1 = 5 & v_2 = 6 \\ w_1 = 4 & w_2 = 5 \end{array}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	5	5
5	0	5	6
6	0	5	6
7	0	5	6
8	0	5	6
9	0	5	11

$$\begin{array}{ll} v_1 = 5 & v_2 = 6 \\ w_1 = 4 & w_2 = 5 \end{array}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	5	5	
5	0	5	6	
6	0	5	6	
7	0	5	6	
8	0	5	6	
9	0	5	11	

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	
3	0	0	0	
4	0	5	5	
5	0	5	6	
6	0	5	6	
7	0	5	6	
8	0	5	6	
9	0	5	11	

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	
5	0	5	6	
6	0	5	6	
7	0	5	6	
8	0	5	6	
9	0	5	11	

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	
5	0	5	6	
6	0	5	6	
7	0	5	6	
8	0	5	6	
9	0	5	11	

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	5
5	0	5	6	6
6	0	5	6	
7	0	5	6	
8	0	5	6	
9	0	5	11	

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	5
5	0	5	6	6
6	0	5	6	6
7	0	5	6	
8	0	5	6	
9	0	5	11	

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	5
5	0	5	6	6
6	0	5	6	8
7	0	5	6	9
8	0	5	6	9
9	0	5	11	11

Das geht dann
weiter für alle
Objekte

$$\begin{aligned}v_1 &= 5 \\w_1 &= 4\end{aligned}$$

$$\begin{aligned}v_2 &= 6 \\w_2 &= 5\end{aligned}$$

$$\begin{aligned}v_3 &= 3 \\w_3 &= 2\end{aligned}$$

Dynamische Programmierung

Welche Objekte sollen wir auswählen?

Done!

Kapazität	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	3
3	0	0	0	3
4	0	5	5	5
5	0	5	6	6
6	0	5	6	8
7	0	5	6	9
8	0	5	6	9
9	0	5	11	11

$v_1 = 5$
 $w_1 = 4$

$v_2 = 6$
 $w_2 = 5$

$v_3 = 3$
 $w_3 = 2$

Regeln:

1. Nur nach links: Wir nehmen das Objekt nicht!
2. Nach links und hoch: Wir nehmen das Objekt!
3. Verfolgen bis nach oben links! Fertig!

Wir wissen, dass wir maximal einen Wert von 11 erzielen können.

Das sagt uns die Zielfunktion!

Aber mit welchen Objekten?

Unter Nutzung des Wertes und der Tabelle finden wir raus, welche Werte man nehmen muss

Wir verfolgen den Algorithmus zurück!

Dynamische Programmierung (Knapsack)

$$\text{maximiere } 16x_1 + 19x_2 + 23x_3 + 28x_4$$

$$\text{unter der Bedingung } 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$$
$$x_i \in \{0, 1\} \quad (i \in 1..4)$$

Kapazität	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					

Dynamische Programmierung (Knapsack)

maximiere $16x_1 + 19x_2 + 23x_3 + 28x_4$

unter der Bedingung $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$

$x_i \in \{0, 1\} \quad (i \in 1..4)$

Wichtigste Erkenntnis?
Wir lösen niemals das gleiche Teilproblem mehrfach!

Done!

Kapazität	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44

Code

Dynamische Programmierung

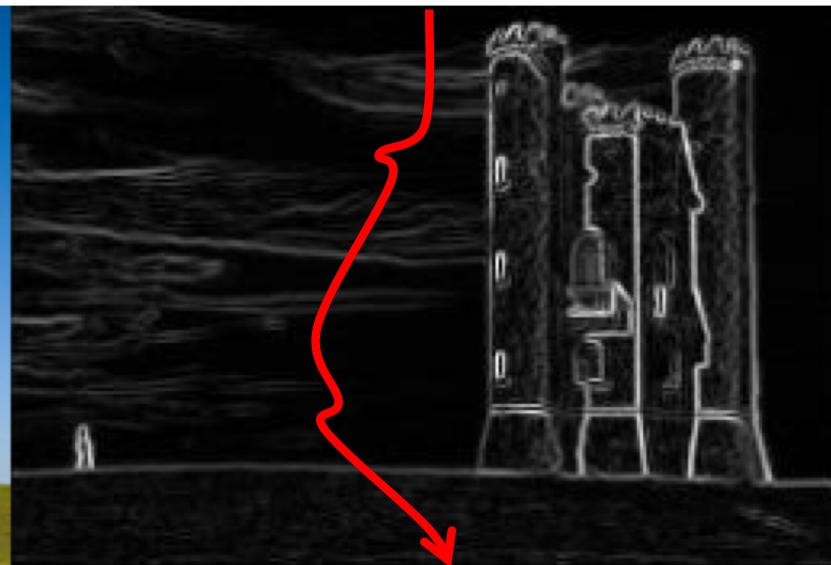
- **Was ist die Komplexität des Algorithmus?**
 - $O(K n)$
- **Ist das Polynomial?**
 - Nein.
 - Algorithmus ist exponentiell bezogen auf die Eingabegröße
 - Pseudo-exponentieller Algorithmus
 - Effizient wenn K ausreichend klein ist

Seam Carving

Seam Carving



Was ist ein Pixelpfad?



Zusammenhängender Pfad von oben nach unten!

Berechnung akkumulierter Energien:

1. Von oben nach unten aufsummieren
2. Unten das Minimum suchen
3. Minimum von unten nach oben verfolgen
4. Fertig

E: 40 aE: 40	E: 60 aE: 60	E: 40 aE: 40	E: 10 aE: 10	E: 20 aE: 20
E: 53,3 aE: 93,3	E: 50 aE: 90	E: 25 aE: 35	E: 47,5 aE: 57,5	E: 40 aE: 50
E: 50 aE: 140	E: 40 aE: 75	E: 40 aE: 75	E: 60 aE: 95	E: 90 aE: 140
E: 30 aE: 105	E: 70 aE: 145	E: 75 aE: 150	E: 25 aE: 100	E: 50 aE: 145
E: 65 aE: 170	E: 70 aE: 175	E: 30 aE: 130	E: 30 aE: 130	E: 10 aE: 110

Dynamische Programmierung im 3D Druck

Adaptive Slicing (3D Printing)

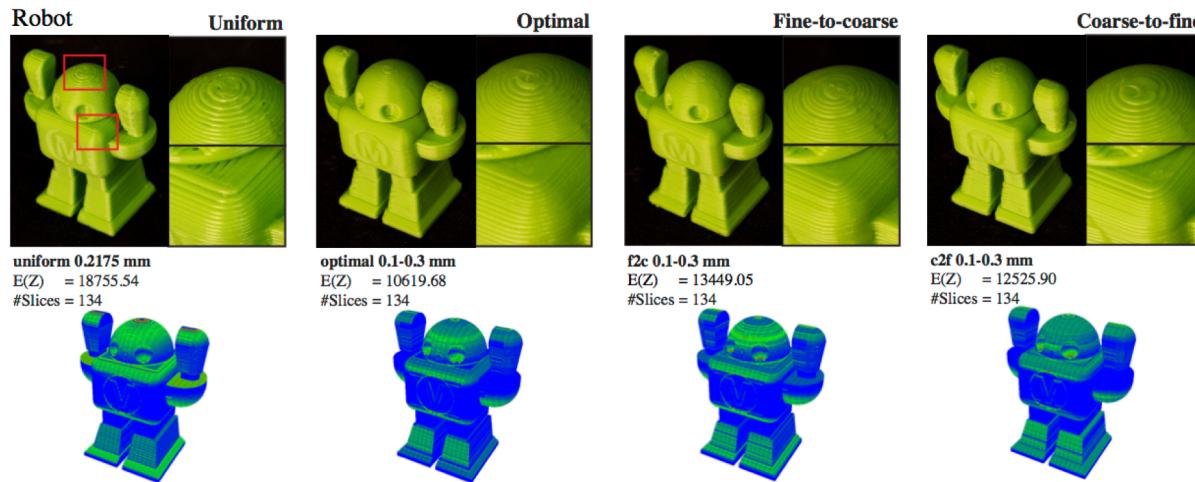
Idee: Wie kann man 3D Druckprozess beschleunigen?



(a) Adaptive

(b) Uniform, same error

(c) Uniform, same time



Optimal Discrete Slicing. ACM Transactions on Graphics. 2017. Alexa, Marc and Hildebrand, Kristian and Lefebvre, Sylvain.

Dynamische Programmierung in der Bewegungsanalyse

Dynamic Time Warping

Idee: Wie kann man zwei Zeitserien miteinander vergleichen

