

Kap. 7.2 Approximative Algorithmen
und Gütegarantien

Kap. 7.3 Enumerationsverfahren

Kap. 7.4 Branch-and-Bound

Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

24. VO

4. Juli 2006

Überblick

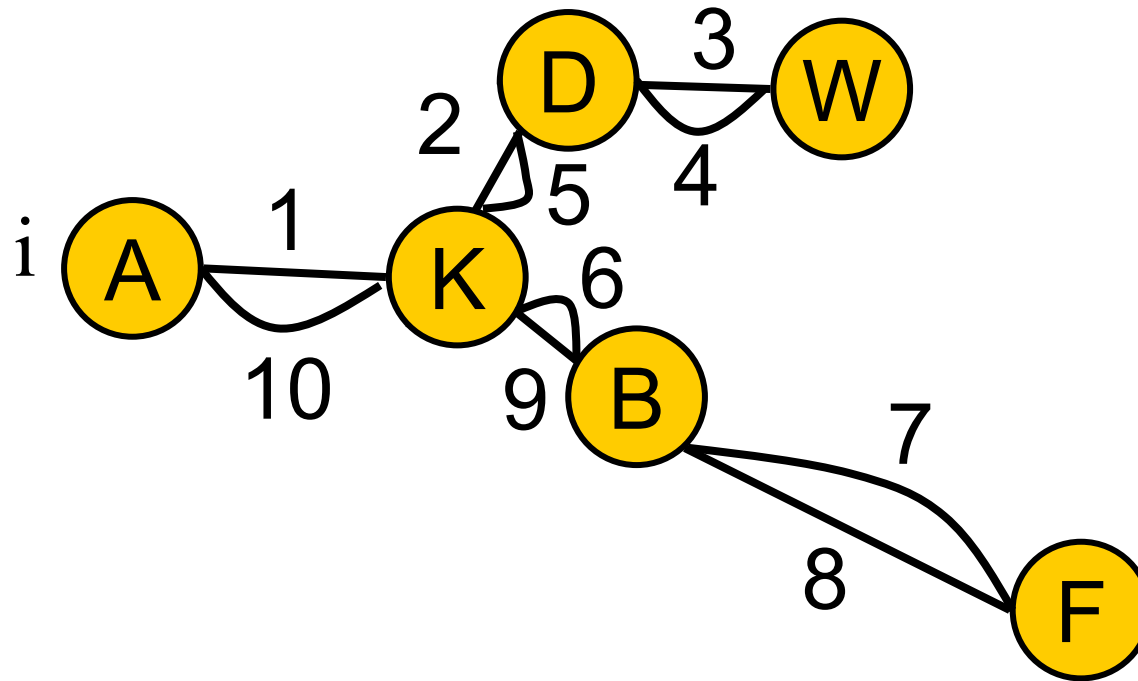
- Approximative Algorithmen für TSP:
 - Spanning Tree Heuristik
 - Christofides Heuristik

- Enumerationsverfahren
 - Beispiel: 0/1-Rucksackproblem
- Branch-and-Bound
 - Beispiel: 0/1-Rucksackproblem
 - Beispiel: ATSP

Spanning-Tree Heuristik (ST) für TSP

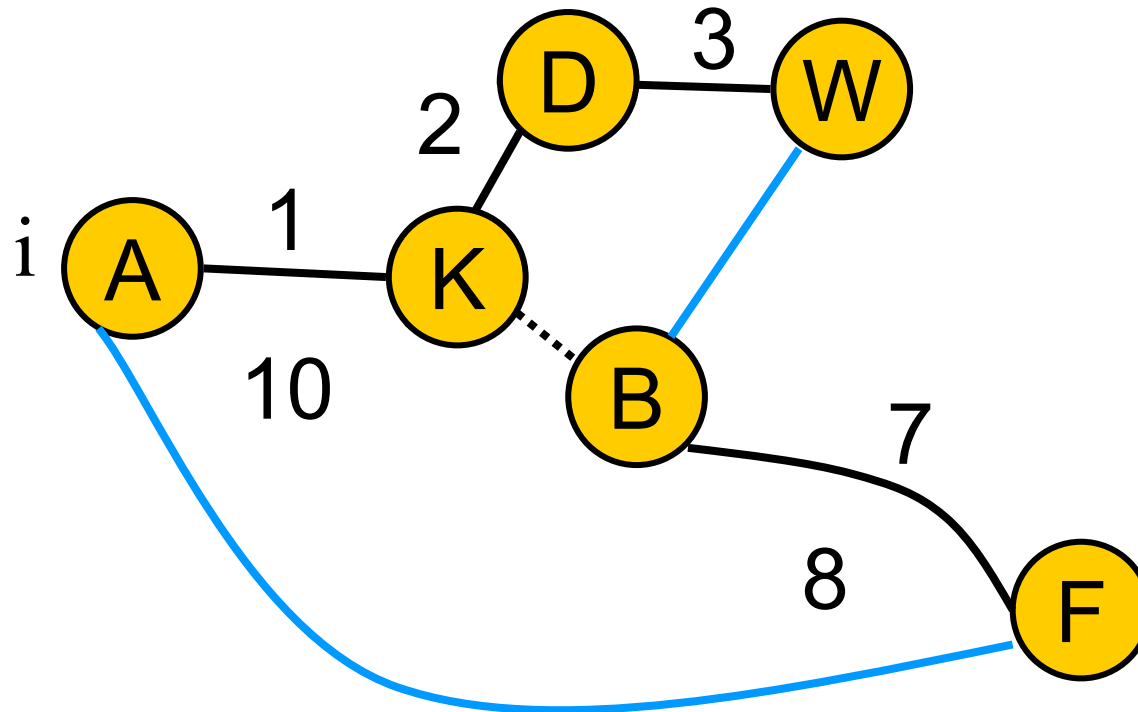
- (1) Bestimme einen MST B von G
- (2) Verdopple alle Kanten aus $B \rightarrow$ Graph $G_2 := (V, B_2)$
- (3) Bestimme eine Eulertour C in G_2 (Tour, die jede Kante genau einmal enthält); gib ihr Orientierung, wähle einen Knoten $i \in V$, markiere i , setze $p := i$ und $T := \emptyset$
- (4) Sind alle Knoten markiert, setze $T := T \cup \{(p, i)\} \rightarrow$ STOP; T ist Tour.
- (5) Laufe von p entlang der Orientierung von F bis ein unmarkierter Knoten q erreicht ist. Setze $T := T \cup \{(p, q)\}$, markiere q und gehe zu (4)

Beispiel für Spanning-Tree Heuristik



- MST mit verdoppelten Kanten
- Zahlen zeigen die Orientierung der Euler-Tour

Beispiel für Spanning-Tree Heuristik



Tour entsteht durch Wanderung entlang der Euler-Tour:
z.B. entsteht die Kante von W nach B, weil Knoten
D und K bereits markiert sind

Diskussion der Gütegarantie

- Auch für die Spanning-Tree Heuristik gibt es eine „schlechte“ Instanz, bei der Lösungen produziert werden, die beliebig weit vom optimalen Lösungswert entfernt sind.
- Man kann zeigen: Das Problem, das TSP-Problem für beliebiges $\epsilon > 1$ zu approximieren ist NP-schwierig.
- Aber: wenn man nur spezielle TSP-Instanzen betrachtet, dann kann man eine Gütegarantie finden.

Das Metrische TSP-Problem

- Ein TSP heißt **metrisch**, wenn für die gegebene Distanzmatrix alle $c_{ii}=0$ sind und die Dreiecksungleichung erfüllt ist, d.h. für alle Knoten i,j,k gilt: $c_{ik} \leq c_{ij} + c_{jk}$

- „Der direkte Weg von i nach k kann nicht länger sein als der Weg von i nach k über j .“

- TSP-Probleme aus der Praxis sind sehr oft metrisch (z.B. Wegeprobleme) --- sogar **euklidisch**: d.h. die Städte besitzen Koordinaten im 2-dim.Raum und die Distanzmatrix ist durch die euklidischen Distanzen gegeben.

Gütegarantie für ST-Heuristik

- Für metrische TSP und für die Spanning-Tree Heuristik gilt: $c_{ST}(P) / c_{opt}(P) \leq 2$ für alle $P \in \Pi$

- Beweis: $c_{ST}(P) \leq c_{B2}(P) = 2 \text{ MST}(P) \leq 2 c_{opt}(P)$

wegen
Dreiecksungleichung:
ST-Lösung läuft direkt

denn: in TSP-Lösung müssen
u.a. alle Knoten miteinander
verbunden sein;
der billigste Weg hierfür ist MST

Christophides-Heuristik (CH) für das TSP

Ähnlich wie die Spanning-Tree Heuristik, ersetze jedoch Schritt (2) durch folgenden:

(2) Sei W die Menge der Knoten in (V, B) mit ungeradem Grad.

- Bestimme im von W induzierten Untergraphen von K_n (vollständiger Graph auf n Knoten) ein perfektes Matching M kleinsten Gewichts
- Setze $B_2 := B \cup M$.

Ein **perfektes Matching** M ist eine Kantenmenge, die jeden Knoten genau einmal enthält. Sie ordnet jedem Knoten einen eindeutigen Partnerknoten zu.

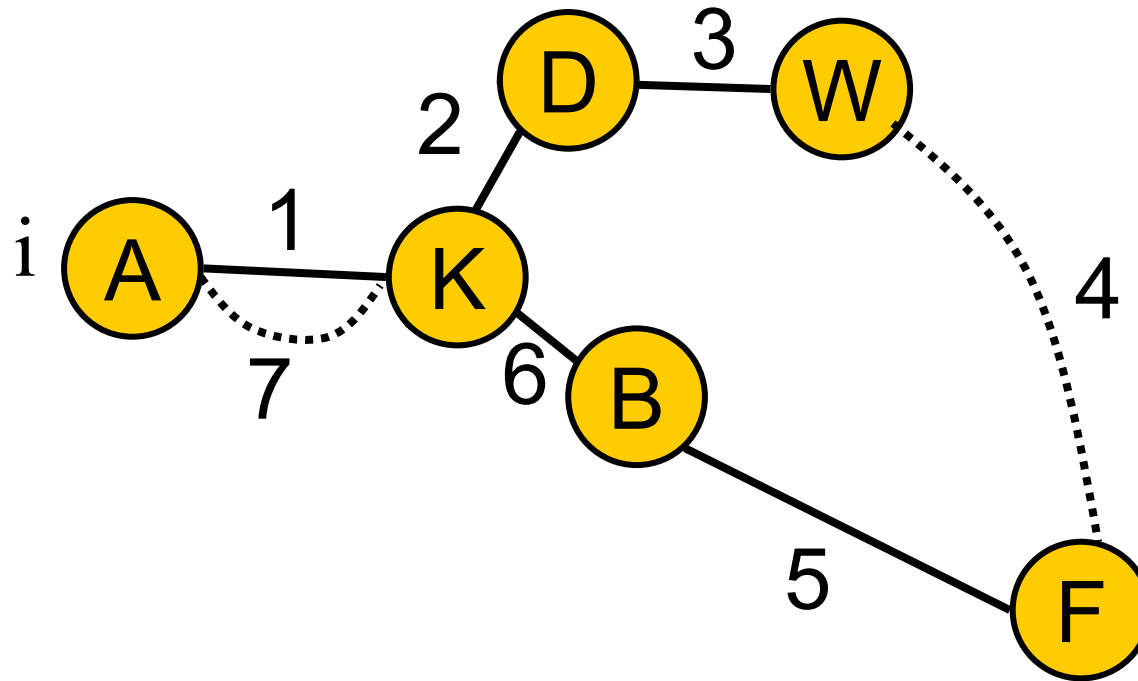
Christophides-Heuristik (CH) für das TSP

(2) Sei W die Menge der Knoten in (V, B) mit ungeradem Grad.

- Bestimme im von W induzierten Untergraphen von K_n (vollständiger Graph auf n Knoten) ein perfektes Matching M kleinsten Gewichts
- Setze $B_2 := B \cup M$.

- Das Matching „geht auf“, denn: $|W|$ ist gerade.
- Ein perfektes Matching M mit kleinstem Gewicht kann in polynomieller Zeit gefunden werden.
- Da eine Eulertour existiert, wenn alle Knoten geraden Grad haben, kann wie bisher vorgegangen werden.

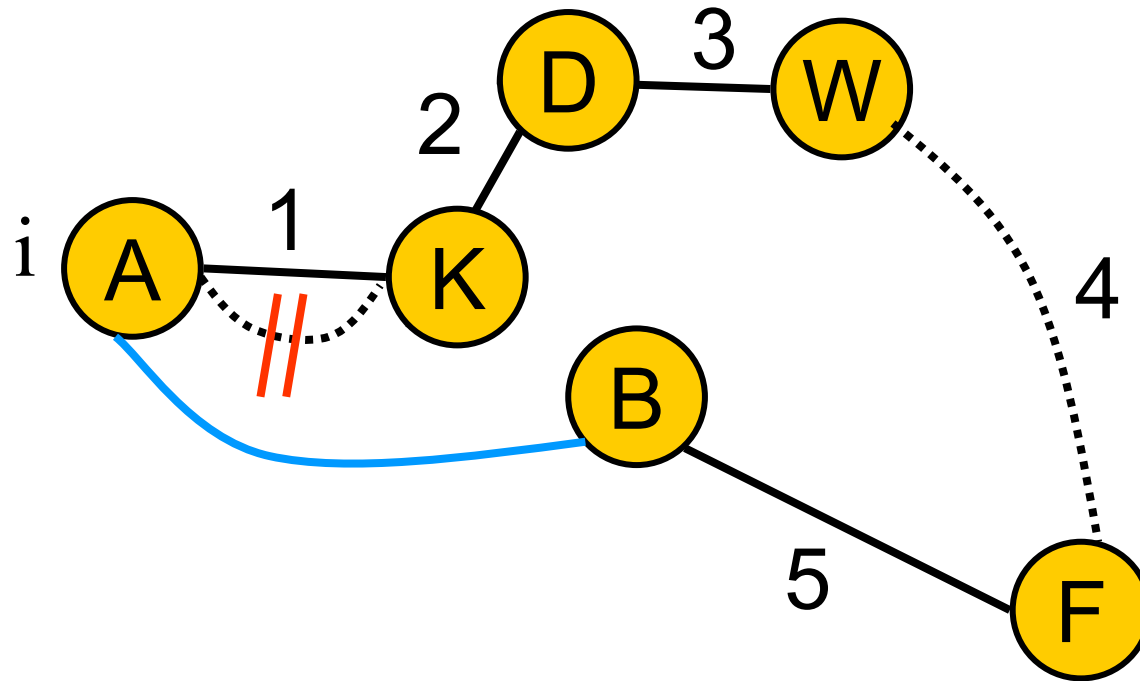
Beispiel für CH-Heuristik



Matching-Kanten: 

- MST wird mit Matching-Kanten erweitert um die ungeraden Knoten zu verhindern.
- Zahlen zeigen die Orientierung der Euler-Tour

Beispiel für CH-Heuristik



Matching-Kanten: 

- Die mit der CH-Heuristik berechnete Tour.

Gütegarantie für CH-Heuristik

- Für das metrische TSP und die CH-Heuristik gilt: $c_{CH}(P) / c_{opt}(P) \leq 3/2$ für alle $P \in \Pi$

- Beweis: Seien i_1, i_2, \dots, i_{2M} die Knoten von B mit ungeradem Grad so nummeriert, wie sie in einer optimalen Tour T_{opt} vorkommen.
- Sei $M_1 := \{(i_1, i_2), (i_3, i_4), \dots\}$ und $M_2 := \{(i_2, i_3), (i_4, i_5), \dots, (i_{2M}, i_1)\}$.
- Es gilt: $c_{opt}(P) \geq c_{M_1}(P) + c_{M_2}(P) \geq c_M(P) + c_M(P)$

wg. Dreiecksungleichung:
ST-Lösung läuft direkt

denn: M ist Matching
kleinsten Gewichts

Gütegarantie für CH-Heuristik

- Für das metrische TSP und die CH-Heuristik gilt: $c_{CH}(P) / c_{opt}(P) \leq 3/2$ für alle $P \in \Pi$

- Beweis: Seien i_1, i_2, \dots, i_{2M} die Knoten von B mit ungeradem Grad so nummeriert, wie sie in einer optimalen Tour T_{opt} vorkommen.
- Sei $M_1 := \{(i_1, i_2), (i_3, i_4), \dots\}$ und $M_2 := \{(i_2, i_3), (i_4, i_5), \dots, (i_{2M}, i_1)\}$.
- Es gilt: $c_{opt}(P) \geq c_{M_1}(P) + c_{M_2}(P) \geq c_M(P) + c_M(P)$
- Weiterhin gilt: $c_{CH}(P) \leq c_{B_2}(P) = c_B(P) + c_M(P) \leq c_{opt}(P) + 1/2 c_{opt}(P) = 3/2 c_{opt}(P)$

CH-Heuristik: Bemerkungen

- Die Christophides-Heuristik (1976) war lange Zeit die Heuristik mit der besten Gütegarantie.
- Vor kurzem zeigte Arora (1996): das euklidische TSP kann beliebig nah approximiert werden: die Gütegarantie $\epsilon > 1$ kann mit Laufzeit $O(N^{1/(\epsilon-1)})$ approximiert werden (PTAS: polynomial time approximation scheme)
- Konstruktionsheuristiken für das symmetrische TSP erreichen in der Praxis meist eine Güte von ca. 10-15% Abweichung von der optimalen Lösung. Die CH-Heuristik liegt bei ca. 14%.

Kap. 7.3: Enumerationsverfahren

Enumerationsverfahren

- Exakte Verfahren, die auf einer vollständigen Enumeration beruhen
- Eignen sich für kombinatorische Optimierungsprobleme: hier ist die Anzahl der zulässigen Lösungen endlich.

Definition: Kombinatorisches Optimierungsproblem

Gegeben sind:

- endliche Menge E (Grundmenge)
- Teilmenge I der Potenzmenge 2^E von E (zulässige Mengen)
- Kostenfunktion $c: E \rightarrow K$

Gesucht ist: eine Menge $I^* \in I$, so dass

$$c(I^*) = \sum_{e \in I^*} c(e)$$

so groß (klein) wie möglich ist.

Enumerationsverfahren

„Exhaustive Search“

- Idee: Enumeriere über die Menge aller zulässigen Lösungen und bewerte diese mit der Kostenfunktion
 - Die am besten bewertete Lösung ist die optimale Lösung.
-
- Problem: Bei NP-schwierigen OP ist die Laufzeit dieses Verfahrens nicht durch ein Polynom in der Eingabegröße beschränkt (sondern exponentiell).

Enumerationsverfahren für 0/1-Rucksackproblem

- Idee: Enumeriere über alle Teilmengen einer N -elementigen Menge.

- Für alle Gegenstände $i=1\dots N$:
 - Fixiere Gegenstand i : Löse das kleinere Problem, das nur noch aus $N-i$ Gegenständen besteht.

Divide & Conquer-Prinzip

Realisierung der Enumeration für 0/1-Rucksackproblem

- x : Lösungsvektor mit $x_i=1 \Leftrightarrow$ Gegenstand i eingepackt wird, und $x_i=0$ sonst
- z : Anzahl der bereits fixierten Variablen in x
- x_{cost} : Gesamtkosten (Wert) der Lösung x
- x_{weight} : Gesamtgewicht der Lösung x

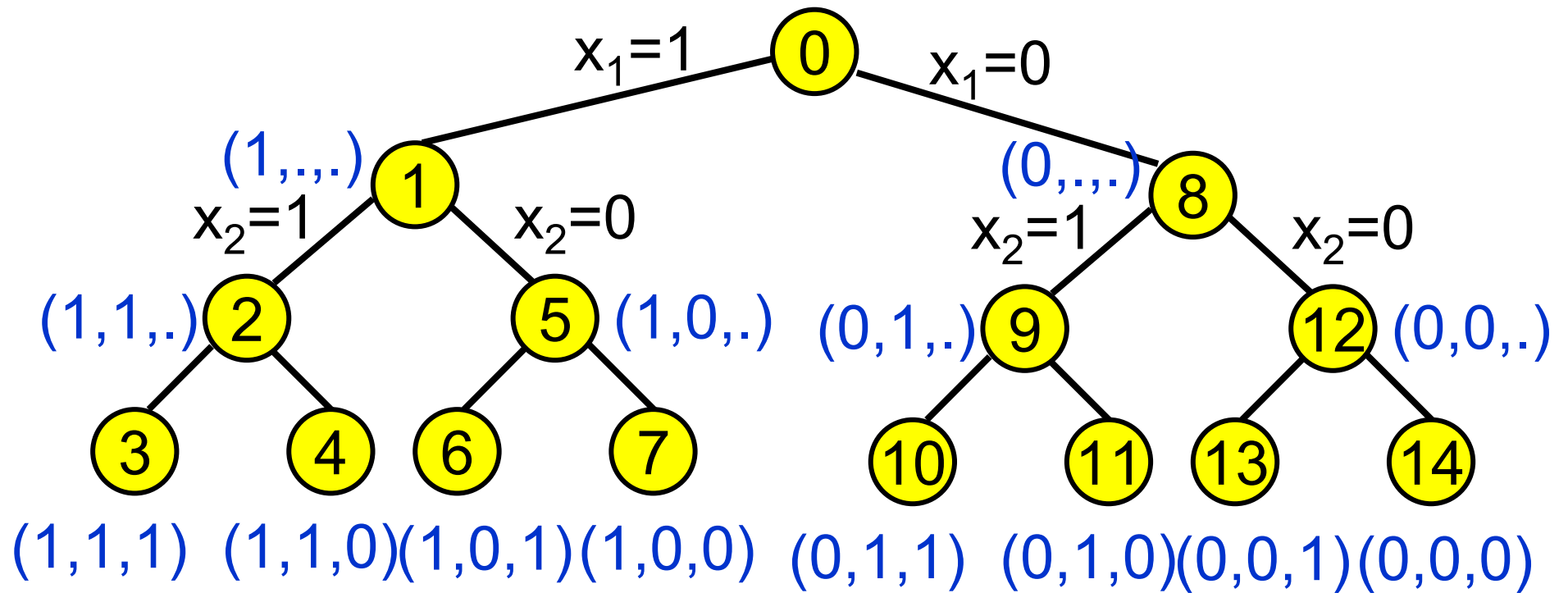
- $\text{Enum}(z, x_{\text{cost}}, x_{\text{weight}}, x)$
- Aufruf: $\text{Enum}(0, 0, 0, x)$

Diskussion Enumerationsverfahren

- Laufzeit: $O(2^N)$

- Wegen der exponentiellen Laufzeit sind Enumerationsverfahren i.A. nur für kleine Instanzen geeignet. Bereits für $N \geq 50$ ist das Verfahren nicht mehr praktikabel.

Erzeugter Suchbaum für N=3



„Backtracking-Verfahren“

Diskussion

Enumerationsverfahren

- Zusatzidee: N-elementige Mengen, die nicht in den Rucksack passen, müssen nicht aufgezählt werden
- In vielen Fällen kann die Anzahl der ausprobierten Möglichkeiten deutlich verringert werden.
- Systematische Verkleinerung des Suchraumes bieten Branch-and-Bound Algorithmen

Kap. 7.4: Branch-and-Bound

Branch-and-Bound

- Idee: Eine spezielle Form der beschränkten Enumeration, die auf dem Divide&Conquer Prinzip basiert.
- Frühes Ausschliessen ganzer Lösungsgruppen in der Enumeration durch „Bounding“.
- Man sagt: L ist eine **untere Schranke** für eine Instanz P eines OPs, wenn für den optimalen Lösungswert gilt: $c_{\text{opt}}(P) \geq L$ ist. U ist eine **obere Schranke**, wenn gilt $c_{\text{opt}}(P) \leq U$.

Branch-and-Bound

- Idee: An den entstehenden Knoten des Suchbaums wird geprüft, ob die dazugehörige Teillösung x' weiterverfolgt werden muss.
- Hierzu wird an jedem Knoten eine untere Schranke L der Teillösung (lower bound) und eine obere Schranke U (upper bound) berechnet.
- L (für Max.probleme) kann mit einer beliebigen Heuristik (z.B. Greedy) berechnet werden
- U (für Max.probleme) gibt eine Schranke für den besten Wert an, der von x' erreicht werden kann.

Gerüst für Branch-and-Bound Algorithmen für Maximierungsprobleme

- Berechne eine zulässige Startlösung mit Wert L und eine obere Schranke U für alle möglichen Lösungen.
- Falls $L = U \rightarrow \text{STOP}$: gefundene Lösung ist optimal.

- **Branching**: Partitioniere die Lösungsmenge (in zwei oder mehr Teilprobleme).

- **Search**: Wähle eines der bisher erzeugten Teilprobleme.

- **Bounding**: Berechne für eine dieser Teilmengen T_i je eine untere und obere Schranke L_i und U_i . Sei L der Wert der besten bisher gefundenen Lösung. Falls $U_i \leq L$, braucht man die Teillösungen in der Teilmenge T_i nicht weiter betrachten.

Branch-and-Bound für 0/1-Rucksackproblem

- Berechnung einer zulässigen Startlösung:
- z.B. Greedy-Heuristik $\rightarrow L$

- Berechnung einer oberen Schranke U : später!

Branch-and-Bound für 0/1-Rucksackproblem

- **Branching:** Wir zerlegen das Problem in zwei Teilprobleme: Wir wählen Gegenstand i und wählen es für Teilproblem T_{1i} aus (d.h. $x_i=1$) und für T_{2i} explizit nicht aus (d.h. $x_i=0$).
- Für T_1 muss das Gewicht von Gegenstand i von der Rucksackgröße abgezogen werden und der Wert zum Gesamtwert hinzuaddiert werden.
- Danach streichen wir Gegenstand i aus unserer Liste.

Branch-and-Bound für 0/1-Rucksackproblem

- **Search:** Wähle eines der bisher erzeugten Teilprobleme, z.B. dasjenige mit der besten (größten) oberen Schranke (wir hoffen, dass wir hier die beste Lösung finden werden).

- **Bounding:** Berechne für eine dieser Teilmengen T_i je eine untere und obere Schranke L_i und U_i . Sei L der Wert der besten bisher gefundenen Lösung. Falls $U_i \leq L$, braucht man die Teillösungen in der Teilmenge T_i nicht weiter betrachten.

Berechnung einer oberen Schranke für 0/1-Rucksackproblem

- Sortierung nach Nutzen $f_i := \text{Wert } c_i / \text{Gewicht } w_i$
- Seien g_1, g_2, \dots, g_n die in dieser Reihenfolge sortierten Gegenstände.
- Berechne das maximale r mit $g_1 + \dots + g_r \leq K$
- Gegenstände $1, \dots, r$ werden eingepackt (also $x_i = 1$ für $i = 1, \dots, r$)
- Danach ist noch Platz für $K - (g_1 + \dots + g_r)$ Einheiten an Gegenständen. Diesen freien Platz „füllen“ wir mit $(K - (g_1 + \dots + g_r)) / g_{r+1}$ Einheiten von Gegenstand $r+1$ auf.

Behauptung: Es existiert keine bessere Lösung!

Berechnung einer oberen Schranke für 0/1-Rucksackproblem

- **Begründung:** Die ersten r Gegenstände mit dem höchsten Nutzen pro Einheit sind im Rucksack enthalten. Und zwar jeweils so viel davon, wie möglich ist.
- **Achtung:** vom letzten Gegenstand $r+1$ wird eventuell ein nicht-ganzzahliger Teil eingepackt
- deswegen ist die generierte Lösung i.A. nicht zulässig (sie erfüllt die Ganzzahligkeitsbedingung i.A. nicht)
- **Aber es gilt:** der berechnete Lösungswert der Gegenstände im Rucksack ist mindestens so groß wie die beste Lösung.

Berechnung einer oberen Schranke für unser Beispiel

K=17

Gegenstand	a	b	c	d	e	f	g	h
Gewicht	3	4	4	6	6	8	8	9
Wert	3	5	5	10	10	11	11	13
Nutzen	1	1,25	1,25	1,66	1,66	1,37	1,37	1,44

- Sortierung nach Nutzen:= Wert c_i / Gewicht w_i :
d,e,h,f,g,b,c,a
- Wir packen also in den Rucksack: $x_d=x_e=1$
- Platz frei für 5 Einheiten: $x_h = 5/9$ Einheiten dazu
- Wert: $10+10+5/9(13) < 20+7,3=27,3$
- Obere Schranke für beste Lösung: 27

Berechnung einer oberen Schranke für 0/1-Rucksackproblem

- FALL: Es besteht bereits eine Teillösung, denn wir befinden uns mitten im Branch-and-Bound Baum.
- Betrachte jeweils die aktuellen Teilprobleme, d.h. sobald ein Gegenstand als „eingepackt“ fixiert wird, wird K um dessen Gewicht reduziert. Falls $x_i=0$ fixiert wird, dann wird der Gegenstand aus der Liste der Gegenstände gestrichen.

Branch-and-Bound für ATSP

- ATSP: wie TSP, jedoch auf gerichtetem Graphen: die Kantenkosten sind hier nicht symmetrisch: $c(u,v)$ ist i.A. nicht gleich $c(v,u)$.

bis zum nächsten Mal überlegen!