Approximationsalgorithmen - Zusammenfassung

Einführung

→ Ein Algorithmus ist schnell, genau dann wenn seine Laufzeit polynomiell in der Eingabegröße ist.
⇒ Allerdings nur abstrakte Definition, da auch bei Polynomen extrem lange Laufzeiten auftreten!
⇒ Grund für Polynome: durch Verdoppelung der Rechengeschwindigkeit wächst handhabbare Problemgröße um einen konstanten Faktor.
→ Ein Problem ist hartnäckig, genau dann wenn seine Entscheidungsvariante NP-vollständig ist.

Definition 1 (*Kombinatorisches Optimierungsproblem*)

Kombinatorische Optimierungsprobleme Π sind durch 4 Komponenten charakterisiert:

1. *Domain D: Menge der Instanzen/Eingaben*
2. *S(I) für $I \in D$: Menge der zu I zulässigen Lösungen*
3. *Bewertungsfunktion $f : S(I) \to \mathbb{N}^{\neq 0}$*
   → $\mathbb{N}^{\neq 0}$ um Tricks mit Bitlänge zu verhindern und dividieren durch Lösung zu ermöglichen.
4. *ziel $\in \{min, max\}$*

Gesucht ist zu $I \in D$ eine zulässige Lösung $\sigma_{opt} \in S(I)$, so dass $OPT(I) = f(\sigma_{opt})$ der Wert einer optimalen Lösung ist.

Beispiel 2 (*Knotenfärbungsproblem*)

1. $D = \{\langle G\rangle \mid G$ *ung. Graph mit min. einer Kante*$\}$.
2. $S(\langle G\rangle) = \{c_V \mid c_V$ *Knotenfärbung von* $G\}$.
3. $f(c_V) = |c_V(V)|$.
4. *min*

Definition 3 (*Approximationsalgorithmus*)

Sei Π gegeben. Ein $t(n)$-Zeit-Approximationsalgorithmus A berechnet zu einer Eingabe $I \in D$ in Zeit $t(|I|)$ eine Ausgabe σ_I^A mit A(I) Wert der approximierten Lösung.

Definition 4 (*Absolute Güte*)

- *Absolute Güte:*

$$\boxed{\kappa_A(I) = |A(I) - OPT(I)|}$$

- *Absolute worst-case Güte:*

$$\boxed{\kappa_A^{WC}(n) = \max\{\kappa_A(I) \mid I \in D, |I| \leq n\}}$$

- *Garantierte absolute Güte von $\kappa_A(n)$:*

$$\boxed{\kappa_A^{WC}(n) \leq \kappa_A(n) \qquad \forall n}$$

- *Absolute Abweichung $\kappa'_A(n)$:*

$$\boxed{\kappa'_A(n) \leq \kappa_A^{WC}(n) \qquad \text{für unendlich viele } n}$$

Approx mit absoluter Gütegarantie

Graphfärbarkeit

→ $G = (V, E)$ ist ungerichteter Graph.
→ **Menge der Nachbarn von u**: $\Gamma_G(u) = \{v \mid \{u, v\} \in E\}$
→ **Grad von u**: $deg_G(u) = |\Gamma_G(u)|$
→ **Grad von G**: $\Delta(G)$, höchster Grad unter allen Knoten.
→ **G ist r-regulär**, wenn alle Knoten in G Grad r haben.
→ Knotenfärbung: Abbildung $c_v : V \rightarrowtail \mathbb{N}$ wo $c_V(u) \neq c_V(v)\ \forall\{u, v\} \in E$.
⇒ Keine benachbarten Knoten haben die gleiche Farbe!
→ Kantenfärbung: Abbildung $c_E : E \rightarrowtail \mathbb{N}$ wo $c_E(\{u, v\}) \neq c_E(\{u, w\})\ \forall\{u, v\}, \{u, w\} \in E$.
⇒ Keine benachbarten Kanten haben die gleiche Farbe.

Definition 5 (*Knoten-/Kantenfärbung*)

Die Aufgabe des Knoten- bzw. Kantenfärbungsproblems besteht darin, eine möglichst farbenminimale Färbung zu finden.

→ Chromatische Zahl $\chi(G)$: minimale Farbanzahl Knoten.
→ Chromatischer Index $\chi'(G)$: minimale Farbanzahl Kanten.

Knotenfärbung beliebiger Graphen

Algorithmus 1 GreedyCol

Input: Graph $G = (V, E)$ mit $V = (u_1, \ldots, u_n)$.
Output: Zulässige Knotenfärbung.

```
for i = 1,...,n do
    c_V(u_i) = ∞                         ▷ Knoten ungefärbt
for i = 1,...,n do
    c_V(u_i) = min{N \ {c_V(Γ(u_i))}}    ▷ kleinste Farbe
return c_V
```

→ Garantierte absolute Güte:

$$\boxed{\kappa_{GreedyCol}(G) = GreedyCol(G) - OPT(G) \leq \Delta(G) - 1}$$

(Bewiesen in Übungen: $\kappa_{GreedyCol}(G) \leq \lceil\sqrt{2m}\rceil$)

Lemma 6 (*Herleitung garantierte absolute Güte*)

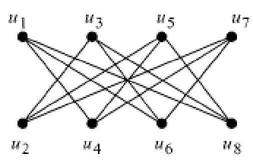
→ $OPT(G) \geq 2$, da Graphen mindestens eine Kante haben.
⇒ GreedyCol berechnet in $\mathcal{O}(|V|+|E|)$ eine Knotenfärbung aus höchstens $\Delta(G) + 1$ Farben, d.h.

$$\boxed{GreedyCol(G) \leq \Delta(G) + 1}$$

Beweis:
→ Obere Schranke für Anzahl der Farben:
⇒ Algorithmus sei bei Knoten u_i.
⇒ Es können nicht alle Farben aus $\{1, \ldots, deg(u_i) + 1\}$ an Nachbarn vergeben sein, da Menge eine Farbe mehr als Nachbarn hat.

□

→ Für Qualität der Analyse betrachte absolute Abweichung.
→ Für GreedyCol: Finde Graphenmenge $\mathcal{G}$, so dass für diese Graphen Färbung $\Delta(G_i) + 1$ Farben ausgegeben wird, obwohl 2 benötigt werden.
⇒ Absolute Abweichung von $\Delta(G) - 1$



$(\Delta(G) - 1)$-Zeuge

→ Färbung von GreedyCol hängt stark von der Reihenfolge ab.
⇒ Es existiert immer eine Knotenreihenfolge, so dass GreedyCol optimale Färbung ausgibt.
→ Variabler Algorithmus ohne Knotennummerierung:

Algorithmus 2 GreedyCol-Var

Input: Graph $G = (V, E)$.
Output: Zulässige Knotenfärbung.

for alle Knoten u **do**
    $c_V(u) = \infty$
while $\exists u : c_V(u) = \infty$ **do**
    $c_V(u_i) = \min\{\mathbb{N} \setminus \{c_V(\Gamma(u_i))\}\}$
return c_V

→ Absolute Gütegarantie unverändert.
→ Hier gehört zur Angabe eines Zeugen auch die Angabe der Färbungsreihenfolge.
⇒ Nicht zu zeigen: Algorithmus verhält sich unabhängig von der Auswahl schlecht.

Knotenfärbung planarer Graphen

→ Planarer Graph: Graph kann kreuzungsfrei in der Ebene dargestellt werden.
→ Facette: Eine von Kanten eingegrenzte Fläche ohne Knoten.
⇒ Bemerke: Fläche um den Graphen herum zählt als Facette.

Fakten:

1. Jeder planare Graph kann in Polyzeit mit 6 Farben knotengefärbt werden.
2. Jeder planare Graph kann mit 4 Farben gefärbt werden.
3. Das Entscheidungsproblem: Ist der planare Graph knoten-3-färbbar? ist NP-vollständig.
4. Es kann in Polyzeit gegebenenfalls eine Knoten-2-Färbarkeit gefunden werden.

→ Beweis zu 4-Farben-Satz kann zwar in Algorithmus umgewandelt werden, jedoch besteht dieser aus zu vielen Fällen.

Algorithmus 3 ColPlan: Knoten-6-Färbung

Input: Graph $G = (V, E)$.
Output: Gültige Färbung.

Teste ob G knoten-2-färbbar ist.
Falls nicht: Färbe Knoten mit 6 Farben.

→ Garantierte absolute Güte:

$$\boxed{\kappa_{ColPlan}(G) = |ColPlan(G) - OPT(G)| \leq 3}$$

⇒ Bei 4-Farben-Satz wäre absolute Güte 1.

Kantenfärbung beliebiger Graphen

⇒ Jeder Graph braucht mindestens $\Delta(G)$ und höchstens $\Delta(G) + 1$ Farben für die Kantenfärbung.
→ Untere Schranke ist klar.
→ Für obere Schranke betrachte folgendes Lemma:

Lemma 7 (*Umfärbarkeit von Kanten*)

Sei G kantengefärbt mit $\Delta(G) + 1$ Farben und seien u, v Knoten, die durch keine Kanten verbunden sind. Des Weiteren gilt $deg(u)$, $deg(v) < \Delta(G)$. Dann kann G so umgefärbt werden, dass an u, v die selbe Farbe fehlt.

→ Aus diesem Lemma lässt sich ein rekursiver Algorithmus ableiten:

Algorithmus 4 FärbeKanten

Input: Graph $G = (V, E)$.
Output: Kantenfärbung mit höchstens $\Delta(G) + 1$ Farben.

if $|E| = \emptyset$ **then return**
else
    Wähle beliebige Kante $\{u, v\} \in E$
    $G' = G \setminus \{u, v\}$
    FärbeKanten(G')   ▷ Höchstens $\Delta(G') + 1$ Farben
    if $\Delta(G') < \Delta(G)$ **then**
        Färbe $\{u, v\}$ mit kleinstmöglicher fehlender Farbe.
    else
        Färbe G' gemäß Lemma so um, dass an u,v die selbe Farbe c fehlt und Färbe $\{u, v\}$ mit c

→ Garantierte absolute Güte: $\boxed{\kappa_{FaerbeKanten} = 1}$.
→ Laufzeit: $\mathcal{O}(|V| \cdot |E|)$
⇒ Entscheidungsproblem, ob Graph mit $\Delta(G)$ Farben kantengefärbt werden kann ist NP-vollständig.

Unmöglichkeitsergebnis für KnapSack

→ Gibt Probleme, wo es keinen polynomiellen Approximationsalgorithmus gibt, der konstante Güte erreichen kann.

Satz 8 (*Allgemeine untere Schranke für KnapSack*)

Falls $P \neq NP$, so gibt es keine Konstante $k \in \mathbb{N}$, so dass es einen polynomiellen Approximationsalgorithmus A für das Rucksackproblem gibt mit

$$\boxed{|A(I) - OPT(I)| \leq k}$$

Beweisidee:
→ Annahme: k und A existieren doch.
→ Zeige, dass sich exakter Algorithmus A' finden lässt, der in Polyzeit das Rucksackproblem löst.
→ Damit könnte man jedoch eine NP-vollständige Entscheidungsvariante des Problems lösen.
⇒ Widerspruch zu $P \neq NP$.

□

→ Problem der absoluten Güte: Bei vielen Problemen kann mit einem einfachen Multiplikationstrick die Lücke zwischen optimaler und zweitbester Lösung beliebig groß gemacht werden.

Approx mit relativer Gütegarantie

→ Statt mit absoluten arbeiten wir hier mit relativen Gütegarantien.
⇒ Rechnen mit prozentualen Abweichungen, da dies die Güte aller Lösungen sichert.

Definition 9 (*Relative Güte*)

Π Optimierungsproblem, A Approximationsalgorithmus.

1. *Relative Güte:*

$$\boxed{\rho_A(I) = \max\{\frac{A(I)}{OPT(I)}, \frac{OPT(I)}{A(I)}\}}$$

→ *Erster Bruch maximal, falls Minimierungsproblem, zweiter bei Maximierungsproblem.*

2. *Relative Worst-Case-Güte:*

$$\boxed{\rho_A^{WC}(n) = \max\{\rho_A(I) \mid I \in D,\ |I| \leq n\}}$$

3. *Garantierte relative Güte $\rho_A(n)$:*

$$\boxed{\rho_A^{WC}(n) \leq \rho_A(n) \qquad \forall n}$$

4. *Relativer Fehler:*

$$\boxed{\epsilon_A(I) = |\frac{A(I)-OPT(I)}{Opt(I)}|}$$

→ *Bei Minimierungsproblem beliebig groß.*
→ *Bei Maximierung nicht einmal 1.*

5. *Garantierter relativer Fehler von höchstens $\epsilon_A(n)$:*

$$\boxed{\epsilon_A(I) \leq \epsilon_A(n) \qquad \forall\ I \in D,\ |I| \leq n}$$

6. *Relative Abweichung von $\rho'_A(n)$:*

$$\boxed{\rho_A^{WC}(n) \geq \rho'_A(n) \qquad \textit{für unendlich viele n}}$$

7. *$\rho'_A(n)$ – Zeugenmenge gegen A:*

$$\boxed{\rho_A(I) \geq \rho'_A(|I|) \qquad \forall I \in D' \subseteq D}$$

⇒ *Einzelne Eingabe wird $\rho'_A(n)$ – Zeuge genannt.*

Bemerkung: Schranken
Π Optimierungsproblem, A Approximationsalgorithmus.

1. Ist Π ein Minimierungsproblem, so gilt:

$$\boxed{OPT(I) \leq A(I) \leq \rho_A(|I|)OPT(I) \textit{ und } \frac{1}{\rho_A(|I|)}A(I) \leq OPT(I) \leq A(I)}$$

2. Ist Π ein Maximierungsproblem, so gilt:

$$\boxed{\frac{1}{\rho_A(|I|)}OPT(I) \leq A(I) \leq OPT(I) \textit{ und } A(I) \leq OPT(I) \leq \rho_A(|I|)A(I)}$$

3. ϵ-Schlauch um die Werte optimaler Lösungen:

$$\boxed{(1-\epsilon_A(|I|))OPT(I) \leq A(I) \leq (1+\epsilon_A(|I|))OPT(I)}$$

Metrisches TSP

→ Änderung zum herkömmlichen TSP: $c(u,v) \leq c(u,w) + c(w,v)$
⇒ Kosten gemäß der Dreiecksungleichung.
→ Keine absolute Gütegarantie für das metrische TSP möglich:
⇒ Wäre es mit absoluter Garantie k approximierbar, kann man die Abstände der Knoten um den Faktor $k+1$ verlängern.
⇒ Additive Lücke zwischen bester und jeder anderen Lösung ist größer als k.

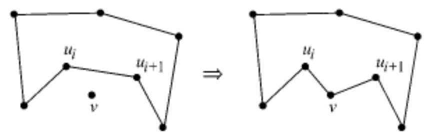
Einfüge-Heuristik

→ Grundidee: Eine bereits bestehende Tour wird durch Einfügen eines neuen Knotens zwischen zwei in der Tour bisher benachbarten Knoten erweitert.

Algorithmus 5 Einfüge

Input: $C = (u_1, \dots, u_k, u_1)$ sei Kreis in K_n, v Knoten, der noch nicht enthalten ist.
Output: Tour mit Knoten v.

Bestimme $i = \min\{c(u_i, v) + c(v, u_{i+1}) - c(u_i, u_i + 1)\}$
▷ Geringster Kostenzuwachs
Gib $(u_1, \dots u_i, v, u_{i+1}, \dots, u_k, u_1)$ aus.



Algorithmus 6 EinfügeHeuristik

Input: Vollständiger Graph $G = (V, E)$.
Output: Tour in G.

$C_1 = (v_1, v_1)$ beliebig.
for j = 2, … n **do**
  Wähle Knoten v_j, der nicht in C_{j-1} ist.
  ▷ Ganze Algorithmenklasse da Knotenwahl offen.
  ▷ Folge $(v_1, \dots, v_n)$ heißt Einfüge-Abfolge.
  C_j = Einfüge(C_{j-1}, v_j)

→ Jede EinfügeHeuristik EH garantiert relative Güte

$$\boxed{EH(K_n, c) \leq (\lceil \log(n) \rceil + 1)OPT(K_n, c)} \qquad (\star)$$

→ Nicht bekannt, ob die obere Schranke $(\star)$ für die relative Güte scharf ist.

Verschiedene Einfügemethoden

- NearestInsertion: Wählt denjenigen Knoten, der am nächsten zu einem in C_{j-1} liegt.
  ⇒ Garantiert eine relative Güte von $2 - \frac{2}{n}$.
  ⇒ Laufzeit von $\mathcal{O}(n^2)$.
- CheapestInsertion: Wählt denjenigen Knoten, der bestehende Tour am wenigsten verlängert.
  ⇒ Garantiert eine relative Güte von $2 - \frac{2}{n}$.
- RandomInsertion: Wählt v zufällig gleichverteilt unter allen übrigen Knoten.
  ⇒ Relative Abweichung von $\Omega(\log\log(n)/\log\log\log(n))$
- FarthestInsertion: Wählt den Knoten mit dem größten Abstand zu einem in C_{j-1}.
  ⇒ In Experimenten für große n beste Auswahlstrategie.

Christofides' Algorithmus

→ Leichtes Matching: Perfektes Matching mit kleinst möglichem Gewicht. (Laufzeit: $\mathcal{O}(n^{2.5}(\log(n))^4)$)
→ Multigraph: Zwei Knoten können durch mehrere Kanten miteinander verbunden sein.
→ Euler-Pfad: Weg, indem jede Kante von G genau einmal vorkommt.

Algorithmus 7 Christofides

Input: Vollständiger Graph $G = (V, E)$.
Output: Hamiltonkreis in G

1: Berechne minimalen Spannbaum T_{CH} von G.
2: $S = \{v \in T_{CH} \mid deg_{T_{CH}}(v) \text{ ist ungerade}\}$ ▷ $|S|$ ist gerade
3: Berechne auf dem durch S induzierten Teilgraphen ein leichtes Matching M_{CH}.
4: Berechne Euler-Tour E auf $T_{CH} \cup M_{CH}$
▷ Knoten haben geraden Grad
5: Wiederholt vorkommende Knoten in E löschen und am Ende E' ausgeben.

→ Garantierte relative Güte: $\boxed{\rho_{CH}(K_n) \leq \frac{3}{2} - \frac{1}{n}}$
→ Laufzeit: $\mathcal{O}(n^{2.5}(\log(n))^4)$

Beweis der relativen Güte:
→ Sei R^* optimale Rundreise für Probleminstanz I.
⇒ $c(R^*) = OPT(I)$.
⇒ Ziel: $CH(I) \leq (\frac{3}{2} - \frac{1}{n})c(R^*)$

Zu Schritt 1:
→ R^* besteht aus n Kanten, da Hamiltonkreis.
⇒ Es gibt mindestens eine Kante $e \in R^* : c(e) \geq \frac{c(R^*)}{n}$.
→ Entfernt man e aus Hamiltonkreis erhält man Spannbaum des Graphen.
⇒ Da T_{CH} minimaler Spannbaum muss gelten:
$c(T_{CH}) \leq c(R^*) - c(e) \leq c(R^*) - \frac{c(R^*)}{n} = (1 - \frac{1}{n})c(R^*)$

Zu Schritt 2:
→ Anzahl Knoten mit ungeradem Grad in jedem Baum gerade.
⇒ Lässt sich induktiv zeigen.

Zu Schritt 3:
→ Wegen Dreiecksungleichung gilt: $c(S) \leq c(R^*)$.
→ Da $|S|$ gerade, kann S in zwei perfekte Matchings M_1, M_2 zerlegt werden.
⇒ OBdA. gilt $c(M_1) \leq c(M_2)$.
⇒ Wegen Minimalität von M_{CH} gilt:
$c(M_{CH}) \leq c(M_1) \leq \frac{1}{2}(c(M_1) + c(M_2)) = \frac{1}{2}c(S) \leq \frac{1}{2}c(R^*)$

Zu Schritt 4:
→ Da jeder Knoten in $T_{CH} \cup M_{CH}$ geradem Grad hat, gibt es eine Euler-Tour E.
⇒ Länge lässt sich aus vorherigen Ungleichungsketten folgern:
$c(E) = c(T_{CH} \cup M_{CH}) \leq (1 - \frac{1}{n})c(R^*) + \frac{1}{2}c(R^*) = (\frac{3}{2} - \frac{1}{n})OPT(I)$

Zu Schritt 5:
→ Löschen von Duplikaten kann wegen Dreiecksungleichung Länge der Tour nur verkürzen.

□

→ Die Güte-Analyse kann nicht verschärft werden

Alternative:
→ Ersetze Schritt 2 + 3 im Algorithmus durch:

2': Verdoppele die Kanten des minimalen Spannbaums.

→ Garantierte relative Güte: $\boxed{\rho_{CH_{Alt}}(K_n) \leq 2 - \frac{2}{n}}$
→ Laufzeit: $\mathcal{O}(n^2)$

Unmöglichkeitsergebnis für das TSP

Satz 10 (*Unmöglichkeitsergebnis TSP*)

Wenn es einen polynomiellen Approximationsalgorithmus A mit konstanter relativer Güte r für das volle TSP gibt, dann ist $P = NP$.

Beweis:
→ Ziel: Zeige, dass Hamiltonkreis-Problem dann in P liegt.
→ Annahme: Es gibt A für TSP mit relativer Güte $r \in \mathbb{N}$.
⇒ Reduziere $Hamilton \leq_p TSP[r]$.
→ Hat Eingabegraph Hamiltonkreis, $n = |V|$, dann hat I kosten

$$c(u,v) = \begin{cases} 1 & \text{falls } \{u,v\} \in E \\ (r-1)n + 2 & \text{falls } \{u,v\} \notin E \end{cases}$$

⇒ Frage ob G einen Hamiltonkreis hat kann so offensichtlich in Polyzeit beantwortet werden.
⇒ Hat G Hamiltonkreis, so hat kürzeste Rundreise Länge n.
⇒ Hat G keinen Hamiltonkreis, so hat kürzeste Rundreise mindestens Länge $(r-1)n + 2 + n - 1 > rn$.
⇒ Es kann keine zulässige Lösung der Länge $[n+1, \ldots, rn]$ geben.
⇒ Wenn G Hamiltonkreis hat, muss $A(I) \leq rn$ sein.
→ Da $A(I)$ nach vorheriger Beobachtung Länge n haben muss, haben wir optimale Rundreise gefunden.
⇒ Widerspruch, Hamiltonkreis wäre polynomiell lösbar.

□

Independent Set und Knotenfärbung

Independet Set

→ Independet Set: Knotenmenge wird unabhänig genannt, wenn kein Knoten innerhalb dieser Menge durch eine Kante mit einem anderen verbunden ist.
⇒ Entscheidungsvariante zu IS ist NP-vollständig.
→ Approximationsalgorithmus legt immer Knoten mit möglichst kleinem Grad in unabhängige Menge:

Algorithmus 8 GreedyIS

Input: Graph $G = (V, E)$.
Output: Unabhängige Menge $U \subseteq V$.

1: $U = \emptyset,\ t = 0,\ V^{(0)} = V$.
2: **while** $V^{(t)} \neq \emptyset$ **do**
3: Sei $G^{(t)}$ durch $V^{(t)}$ induzierter Graph.
4: Sei u_t Knoten mit minimalem Grad in $G^{(t)}$.
5: $V^{(t+1)} = V^{(t)} - (\{u_t\} \cup \Gamma_{G^{(t)}}(u_t))$
▷ Löscht u_t und Nachbarn
6: $U = U \cup \{u_t\}$.
7: $t = t + 1$.
return U

→ Garantierte relative Güte k-färbbarer Graph:

$$\boxed{GreedyIS(G) \geq \lceil \log_k(|V|/3) \rceil}$$

→ Laufzeit: $\mathcal{O}(|V| + |E|)$

→ Beziehung zwischen Anzahl Farben und Grad eines Knoten:
⇒ Sei G ein knoten-k-färbbarer Graph. Dann gibt es $u \in V$ mit $deg_G(u) \leq \lfloor (1 - \frac{1}{k})|V| \rfloor$. (Durchschnittsargument)
⇒ Es können nicht alle Knoten großen Grad haben!

Knotenfärbung

→ Für GreedyCol gilt bzgl. relativer Güte: $\rho \cdot \mathcal{O}(n)$.
→ Algorithmus wird besser, wenn auf einmal so viele Knoten mit einer Farbe gefärbt werden wie erlaubt.
⇒ Starker Zusammenhang zum Independent Set.
→ Achtung: Ist ein Graph mit möglichst wenig Farben gefärbt, so ist die größte Menge von Knoten gleicher Farbe nicht zwangsläufig einer maximale unabhängige Menge!
⇒ Färbungsproblem ist Suche nach kleinstmöglichem System unabhängiger Mengen.

Algorithmus 9 GreedyCol2

Input: Graph $G = (V, E)$.
Output: Färbung von G

```
t = 1, V^(1) = V.
while V^(t) ≠ ∅ do
    Sei G^(t) durch V^(t) induzierter Graph.
    Sei U_t = GreedyIS(G^(t)).
    Färbe alle Knoten in U_t mit Farbe t.
    V^(t+1) = V^(t) − U_t.
    t = t + 1
return Berechnete Färbung.
```

→ Garantierte relative Güte:

$$\boxed{\rho_{GreedyCol2}(G) \leq \frac{3n}{\log(n/16)} \frac{\log(k)}{k} = \mathcal{O}(\frac{n}{\log(n)})}$$

→ Färbung mit höchstens $\frac{3n}{\log_k(n/16)}$ Farben.
Beweis der garantierten relativen Güte:
→ $n_t = |V^{(t)}|$.
→ Aus Analyse zu GreedyIS folgt: $|U_t| \geq \log_k(n_t/3)$
⇒ Rekursive Beziehung: $n_1 = n,\ n_{t+1} \leq n_t - \log_k(n_t/3)$
→ GreedyCol2 bricht ab, wenn $n_t < 1$.
⇒ Nach $t \leq \frac{3n}{\log_k(n/16)}$ vergebenen Farben ist dies der Fall.
→ Da $k = OPT(G)$ lässt sich relative Güte berechnen:

$$\frac{GreedyCol2}{OPT(G)} \leq \frac{\frac{3n}{\log_k(n/16)}}{k} = \frac{3n}{\log(n/16)} \frac{\log(k)}{k} = \mathcal{O}(\frac{n}{\log(n)})$$

□

Approximationsschemata

→ Bisher: Abweichung von der optimalen Lösung war fest.
⇒ In Anlehnung an Konvergenzidee: Genauigkeit der Lösung steigern durch Eingabe eines relativen Fehlers im Algorithmus.

Definition 11 (*Approximationsschema*)

Sei Π Optimierungsproblem, A Approximationsalgorithmus mit Eingabe der Probleminstanz I und $0 < \epsilon < 1$.

1. *Polynomielles Approximationsschema (PAS): A berechnet zu jedem I und für jedes ϵ in $\mathcal{O}(poly(|I|))$ eine zulässige Lösung mit relativem Fehler $\epsilon_A(I, \epsilon) \leq \epsilon$.*
   → Einfluss von ϵ egal, wird als konstante behandelt.
2. *Streng polynomielles Approximationsschema (FPAS): A ist PAS mit Laufzeit $\mathcal{O}(poly(|I|,\ \frac{1}{\epsilon}))$.*
   → FPAS stärker als PAS.
   → FPAS skaliert, d.h. bei Verdoppelung der Rechengeschwindigkeit des Computers kann die Eingabegröße um einen festen Faktor größer werden bei gleicher tatsächlicher Zeit.

→ Lässt sich für Π ein (F)PAS angeben, so kann man es auch immer mit konstanter Güte approximieren.

→ Annäherung kann bis zu einem exakten Algorithmus fortgeführt werden.
→ Dieser ist jedoch i.A. nicht polynomiell.
→ Grund hierfür ist meist die Diskretheit kombinatorischer Optimierungsprobleme.
⇒ Theoretisch: Wunschfehler kann man so klein wählen, dass Fehler zwischen bester und zweitbester Lösung zu groß ist.

Satz 12 (*Fehler für exakten Algorithmus*)

Sei Π kombinatorisches Optimierungsproblem, A ein (F)PAS und zu I gilt $OPT(I) \leq k(I)$. Wenn $\epsilon^* = \frac{1}{k(I)+1}$ ist, dann gilt $A(I, \epsilon^*) = OPT(I)$.
Ist A ein FPAS, so ist die Laufzeit $\mathcal{O}(poly(|I|, k(I)))$.

→ Bemerke: Zur Laufzeit kann keine Aussage getroffen werden, wenn A PAS ist.
Beweis:
→ Gemäß Definition 9 wird eine zulässige Lösung zu I gefunden mit relativem Fehler

$$\epsilon_A(I, \epsilon^*) = \frac{|OPT(I) - A(I, \epsilon^*)|}{OPT(I)} \leq \epsilon^*$$

$$\Leftrightarrow |OPT(I) - A(I, \epsilon^*)| \leq \epsilon^* OPT(I) = \frac{OPT(I)}{k(I)+1} \overset{OPT(I) \leq k(I)}{<} 1$$

→ Wert aller zulässigen Lösungen ist ganzzahlig.
⇒ $A(I, \epsilon^*) = OPT(I)$.

□

→ Zwei Arten von Approximationsschemata:

- In Abhängigkeit von ϵ werden nur wenige Lösungen bestimmt, aus denen die Beste ausgewählt wird.
- Nachdem zulässige Lösung bestimmt ist wir diese so lange verbessert, bis Laufzeit abgelaufen ist.

Pseudopoly exakter Alg. für KnapSack

→ Instanz eines Rucksackproblems: Waren $W = \{1, \dots, n\}$, Volumen der Waren vol, Preise der Waren p, Rucksackvolumen B, Teuerste Ware $p_{max} = \max\{p_j \mid j \in \{1, \dots, n\}\}$.
→ $F_j(a)$, $a \in \mathbb{Z}$ sei kleinstes benötigtes Rucksackvolumen, mit dem Wert von min a mit den ersten j Waren erzielt wird.
⇒ $F_j(a) = \infty$: a ist nicht zu erreichen.

Lemma 13 (*Rekursion $F_j(a)$*)

Für $F_j(a)$ gilt folgende Rekursion:

1. $a \leq 0,\ j \in \{0, \dots, n\} : F_j(a) = 0$
2. $a \geq 1,\ j = 0 : F_0(a) = \infty$
3. $a \geq 1,\ j \in \{1, \dots, n\} :$
   $F_j(a) = \min\{F_{j-1}(a - p_j) + vol(j),\ F_{j-1}(a)\}$

Beweis:
→ (1) besagt, dass ungefüllter Rucksack bereits Wert 0 hat.
→ (2) besagt, dass man keinen von 0 verschiedenen Wert mit leerem Rucksack erzielen kann.
→ (3) nennt man Bellmannsche Optimalitätsgleichung.
⇒ Wenn kleinstes benötigtes Rucksackvolumen mit den Waren $1, \dots, j-1$ berechnet für den Gesamtwert a berechnet ist und nun die Ware j mit Volumen $vol(j)$ und Wert p_j dazu kommt, gibt es zwei Möglichkeiten:
  ⇒ Es ändert sich nichts an der Sache.
  ⇒ Man kann Rucksackfüllung mit kleinerem Volumen erzeugen, indem man mit den Waren $1, \dots, j-1$ den Wert $a - p_j$ erzielt und Ware j mit in den Rucksack legt.
→ Gibt keine Rucksackfüllung aus erlaubten Waren mit Mindestwert a mit noch geringerem Volumen.

□

⇒ Gesucht ist im Rucksackproblem also das größte a, so dass $F_n(a)$ noch in den Rucksack der Kapazität B passt.

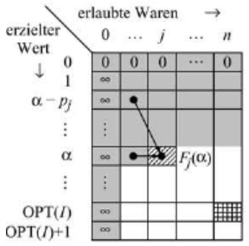
Algorithmus 10 DynRucksack

Input: Instanz I des Rucksackproblems.
Output: Größtmöglicher Wert a.

$a = 0$, F ist Matrix.
while $B > F_n(a)$ **do**
  $a = a + 1$
  for $j = 1, \dots, n$ **do**
    $F_j(a) = \min\{F_{j-1}(a - p_j) + vol(j),\ F_{j-1}(a)\}$
return $a - 1$

→ Laufzeit: $\mathcal{O}(n \cdot OPT(I)) \overset{OPT(I) \leq n \cdot p_{max}}{=} \mathcal{O}(n^2 \cdot p_{max})$.
→ Wort-Case-Laufzeit: $\mathcal{O}(n \cdot 2^{|I|})$
⇒ DynRucksack ist jedoch polynomiell, wenn p_{max} polynomiell in der Eingabelänge ist.

Ordnet man $F_j(a)$ in Tabellenform an, wird die Tabelle zeilenweise von oben nach unten jeweils von links nach rechts gefüllt:



⇒ Wesentlich hierbei: Warenwerte sind natürliche Zahlen.

Definition 14 (*Pseudopolynomielle Algorithmen*)

Sei Π kombinatorisches Optimierungsproblem, so dass in allen Instanzen I nur natürliche Zahlen vorkommen. Sei größte vorkommene Zahl $maxnr(I)$. Algorithmus heißt pseudopolynomiell, falls es ein Polynom gibt, so dass für alle I die Laufzeit $poly(|I|,\ maxnr(I))$ ist.

FPAS für KnapSack

→ Mit Hilfe von DynRucksack wird FPAS angegeben, das in Abhängigkeit von ϵ nur eine zulässige Lösung bestimmt.
⇒ Verringerung der Laufzeit, indem man alle Preise um einen Faktor $\frac{1}{k}$ verkleinert.

Algorithmus 11 AR_k

Input: Instanz des Rucksacksproblems, Reduzierer k.
Output: Rucksackfüllung R_k.

Reduziere Preise um k: $p_j \to \lfloor \frac{p_j}{k} \rfloor$.
Berechne mit DynRucksack optimale Rucksackfüllung R_k auf Eingabe mit reduzierten Preisen.
return R_k

→ Garantierter relativer Fehler: $\boxed{\epsilon_{AR_k}(I) \leq \frac{kn}{p_{max}}}$
→ Laufzeit: $\mathcal{O}(n^2 \frac{p_{max}}{k})$
→ Gaußklammern verursachen relativen Fehler kleiner 1. (max)
→ Um FPAS für Rucksackproblem zu bekommen müssen wir zeigen, dass jedes $\epsilon \in]0,1[$ als relativer Fehler erreichbar ist.
⇒ Erreichbar durch konkrete Wahl von k.

Algorithmus 12 FPASRucksack

Input: Instanz des Rucksackproblems, Fehler ϵ.
Output: Rucksackfüllung.

Setze $k = \epsilon \frac{p_{max}}{n}$.
Setze R_k als Lösung von AR_k.
return R_k

→ Laufzeit: $\mathcal{O}(n^3 \frac{1}{\epsilon})$

Unmöglichkeit für Approximationsschemata

Satz 15 (*Existenz von FPAS*)

Sei Π Optimierungsproblem und sei k fest, so dass Frage $OPT(I) \leq (\geq)\ k$ NP-vollständig ist. Gibt es (F)PAS, dann ist $P = NP$

Definition 16 (*Schwach/stark NP-vollständig*)

NP-vollständiges Entscheidungsproblem L heißt stark NP-vollständig, wenn es Polynom q gibt, so dass $L_q = \{x \mid x \in L,\ maxnr(x) \leq q(|x|)\}$ NP-vollständig. Gibt es kein solches Polynom heißt L schwach NP-vollständig.

→ Äquivalent: NP-vollständige Entscheidungsproblem L ist stark NP-vollständig, falls es keinen pseudopolynomiellen Algorithmus für L gibt, es sei denn, $P = NP$.
⇒ Wenn es für eine Optimierungsvariante eines stark NP-vollständigen Problems FPAS gibt, dann ist $P = NP$.
→ Rucksack schwach NP-vollständig.
→ TSP, Hamilton, Clique stark NP-vollständig.

Techniken für randomisierte Approxalg

Probabilistische Methode

→ SAT- Problem in KNF:

$$\boxed{\Phi(x_1, \dots, x_n) = \underbrace{\bar{x}_1}_{Literal} \wedge \underbrace{(x_1 \vee \dots \vee x_k)}_{Klausel\ C_j} \wedge \dots}$$

Definition 17 (*MaxSAT*)

Probleminstanz Φ ist Boolesche (n,m)-Formel in KNF. Zulässige Lösung ist Belegung $b : V \rightarrowtail \{False, True\}$. Bewertungsfunktion: $wahr(b, \Phi) = |\{j \mid C_j \in \Phi,\ b(C_j) = True\}|$. Ziel ist es maximales b^* zu finden.

⇒ Es gilt:

$$\boxed{\max\{wahr((False, \dots, False), \Phi), wahr((True, \dots, True), \Phi)\} \geq \frac{1}{2}m}$$

Algorithmus 13 A

Input: Instanz des SAT-Problems Φ
Output: Gültige Belegung. ▷ Vollständig stoch. unab.

for $1, \dots, n$ **do**
  Setze $\begin{cases} mit\ Wahrscheinlichkeit\ \frac{1}{2} : x_i = True \\ mit\ Wahrscheinlichkeit\ \frac{1}{2} : x_i = False \end{cases}$
return $b_A = (x_1, \dots, x_n)$

→ Annahme $OPT(\Phi) \leq m$: A hat erwartete relative Güte:

$$\boxed{E[\rho_A(I)] = \frac{OPT(\Phi))}{E[A(\Phi)]} \leq \frac{1}{1 - \frac{1}{2^k}}}$$

→ Begründung: Es gibt b mit $wahr(b, \Phi) \geq (1 - \frac{1}{2^k})m$.
⇒ Für $2^k > m$ ist jedes Φ, in dem jede Klausel mindestens Länge k hat, erfüllbar.

Lemma 18 (*Erfüllungswsk einzelner Klauseln durch A*)

Sei k_j die Anzahl der Literale in C_j. Es gilt:

$$\boxed{Pr[C_j \text{ wird durch A erfüllt}] = 1 - \frac{1}{2^{k_j}}}$$

Beweis:
→ Bemerke: Nur eine Variable muss True werden!
→ Anzahl möglicher Belegungen für True: $2^{k_j} - 1$
⇒ $Pr[C_j$ durch A erfüllt$] = \frac{2^{k_j} - 1}{2^{k_j}} = 1 - \frac{1}{2^{k_j}}$ □

Satz 19 (***Erwartungswert Boolesche Formel***)

Hat jede Klausel in Φ min. k Literale, so gilt:

$$\boxed{E[A(\Phi)] \geq (1 - \tfrac{1}{2^k})m}$$

Beweis:

$\rightarrow$ Für $j \in \{1, \ldots, m\}$ sei Z_j ZV mit

$$Z_j = \begin{cases} 1 & falls\ b_A(C_j) = True \\ 0 & sonst. \end{cases}$$

$\rightarrow$ Es gilt: $E[Z_j] = 0 \cdot Pr[b_A(C_j) = False] + 1 \cdot Pr[b_A(C_j) = True]$

$\Rightarrow$ Wähle $A(\Phi) = \sum Z_j$.

$$E[A(\Phi)] = E[\sum Z_j] = \sum E[Z_j] \overset{Le.17}{=} \sum(1 - \tfrac{1}{2^{k_j}}) \geq (1 - \tfrac{1}{2^k})m$$

$\square$

Randomized Rounding

$\rightarrow$ Modellieren von MaxSAT durch ganzzahliges LP:

- $\Rightarrow$ S_j^+ Menge der nicht negierten Variablen in C_j.
- $\Rightarrow$ S_j^- Menge der negierten Variablen in C_j.
- $\Rightarrow$ Führe für jede boolesche Variable 0-1-Variable $\hat{x}_i$ ein.
- $\Rightarrow$ Führe für jede Klausel 0-1-Variable $\hat{Z}_j$ ein.

Ganzzahliges LP B für MaxSAT:

$$\begin{aligned} & maximiere \quad \sum \hat{Z}_j \\ & \quad s.t. \quad \sum_{x_i \in S_j^+} \hat{x}_i + \sum_{x_i \in S_j^-} (1 - \hat{x}_i) \geq \hat{Z}_j \qquad \forall j \\ & \qquad\qquad \hat{x}_i,\ \hat{Z}_j \in \{0, 1\} \qquad \forall i, j \end{aligned}$$

$\rightarrow$ Lösung von IPs nur in exponentieller Laufzeit möglich, daher Relaxierung der Variablen.

$\Rightarrow$ Es gilt: $OPT(\Phi) = OPT(B) \overset{\text{Superoptimalität}}{\leq} OPT(B_{rel})$

$\rightarrow$ Ganzzahligkeitslücke: $\frac{OPT(B_{rel})}{OPT(B)}$

Von rationaler zur ganzzahliger Lösung

$\rightarrow$ Wähle stochastische Funktion $\pi : [0, 1] \rightarrowtail [0, 1]$.

$\rightarrow$ Wir wählen $\pi(x) = x$

Algorithmus 14 RandomizedRounding

Input: Funktion $pi(x)$ und Instanz von SAT

Output: Gültige Belegung.

```
for i = 1,...n do
    Setze { mit Wahrscheinlichkeit π(x̂_i) : x_i = True
          { mit Wahrscheinlichkeit 1 − π(x̂_i) : x_i = False
return b_B = (x_1,...,x_n)
```

Algorithmus 15 B

Input: Instanz von SAT.

Output: Gültige Belegung.

```
Löse relaxiertes LP B_rel zu Φ.
Ermittle Belegung b_B mit RandomizedRounding(π(x) = x)
```

$\Rightarrow$ Garantierte relative Güte für Klauseln mit genau k Literalen

$$\boxed{E[B(\Phi)] \geq (1 - (1 - \tfrac{1}{k})^k) OPT(\Phi)}$$

Beweis durch Nachrechnen:

$$\begin{aligned} E[B(\Phi)] &= \sum Pr[C_j \text{ wird durch B erfüllt}] \\ &\geq \sum (1 - (1 - \frac{1}{k_j})^{k_j}) \hat{Z}_j \\ &\geq (1 - (1 - \frac{1}{k})^k) OPT(B_{rel}) \\ &\geq (1 - (1 - \frac{1}{k})^k) OPT(\Phi) \quad \text{wegen Superoptimalität} \end{aligned}$$

$\square$

Lemma 20 (***Erfüllungswsk einzelner Klauseln durch B***)

Sei k_j die Anzahl der Literale in C_j. Es gilt:

$$\boxed{Pr[C_j \text{ wird durch B erfüllt}] \geq (1 - (1 - \tfrac{1}{k_j})^{k_j}) \hat{Z}_j}$$

Beweis:

$\rightarrow$ Betrachte $\hat{Z}_j \leq \sum_{x_i \in S_j^+} \hat{x}_i + \sum_{x_i \in S_j^-} (1 - \hat{x}_i)$.

$\rightarrow$ Klausel bleibt unerfüllt, wenn alle $x_i \in S_j^+$ auf False gesetzt werden und alle $x_i \in S_j^-$ auf True.

$\Rightarrow$ $Pr[C_j$ durch B nicht erfüllt$] = \prod_{x_i \in S_j^+} (1 - \hat{x}_i) \prod_{x_i \in S_j^-} \hat{x}_i$

$\rightarrow$ Produkt besteht aus k_j Faktoren.

$\Rightarrow$ $Pr[C_j$ durch B erfüllt$] \geq (1 - (1 - \frac{1}{k_j})^{k_j}) \hat{Z}_j$

$\square$

Hybrider Ansatz

$\rightarrow$ Tabelle zeigt erwartete relative Güte, wenn alle Klauseln genau Länge k haben:

k	$\frac{1}{E[\rho_A]} = 1 - \frac{1}{2^k}$	$\frac{1}{E[\rho_B]} \geq 1 - \left(1 - \frac{1}{k}\right)^k$
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672
$\vdots$	$\vdots$	$\vdots$

$\rightarrow$ A: Güte steigt mit steigendem k, B genau anders herum.

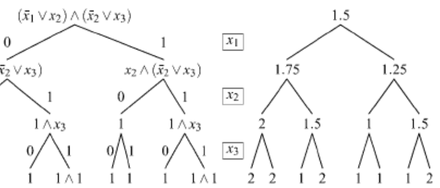
$\rightarrow$ Ideen für zwei Algorithmen:

- $\Rightarrow$ C_{p_A}: Starte mit Wsk p_A A, mit Wsk $1 - p_A$ B.
- $\Rightarrow$ C_{alle}: Starte Beide und gibt besseres Ergebnis aus.
  - $\Rightarrow$ $E[C_{alle}\Phi] \geq p_A E[A(\Phi)] + (1 - p_A) E[B(\Phi)]$

$\Rightarrow$ Für $p_A = \frac{1}{2}$ so hat Alg. $C_{\frac{1}{2}}$ erwartete relative Güte von $\frac{3}{4}$.

Derandomisierung

$\rightarrow$ Methode der bedingten Erwartungswerte.



Algorithmus 16 Derand.A

Input: Instanz von SAT.

Output: Gültige Belegung.

```
for i = 1,...,n do
    W_0 = E[A(I) | x_1,...,x_{i-1}; x_i = 0]
    W_1 = E[A(I) | x_1,...,x_{i-1}; x_i = 1]
    if thenW_0 ≤ W_1
        x_i = 1
    else
        x_i = 0
return x_1,...,x_n
```

Lineare Opt. und Approx-Alg.

Ganzzahligkeitslücke und ihre Beziehung zur relativen Güte

Vorgehen beim Rundungsansatz für Maximierungsprobleme :

1. Bilde Arithmetisierung X von Instanz I, dann gilt $OPT(I) = OPT(X)$.
2. Relaxiere X zu Polyzeit lösbarem X_{rel}.
   Superoptimalität: $OPT(X) \leq OPT(X_{rel})$
3. Approximationsalgorithmus A rundet geschickt fraktionale Lösung zu zulässiger Lösung von I.
4. Zeige, dass $A(I) \geq \frac{1}{\rho} OPT(X_{rel})$. Hier ist $\frac{1}{\rho}$ relative Güte von A.
5. Wegen Superoptimalität ist $A(I) \geq \frac{1}{\rho} OPT(I)$.

⇒ Für Minimierungsproblem müssen alle Relationszeichen umgedreht werden.

Definition 21 (*Ganzzahligkeitslücke*)

Sei Π kombinatorisches Optimierungsproblem und X äquivalentes IP mit X_{rel} relaxiertem Programm. Dann ist die Ganzzahligkeitslücke der Relaxierung

$$\gamma = \max\{\frac{OPT(X_{rel})}{OPT(X)} \mid I \in D\} \qquad \textit{für Max}$$

$$\gamma = \max\{\frac{OPT(X)}{OPT(X_{rel})} \mid I \in D\} \qquad \textit{für Min}$$

SetCover und seine Arithmetisierung

Definition 22 (*SetCover*)

Eine Instanz besteht aus einer Sammlung $S = \{S_1, \dots, S_m\}$ verschiedener endlicher Mengen von Objekten, sei V Menge der Objekte. Eine Teilsammlung S_{Cov} heißt Überdeckung von V, wenn $V = S_{i_1} \cup \dots \cup S_{i_l}$ gilt. Aufgabe des SetCover ist es, eine möglichst kleine Überdeckung zu finden.

→ Entscheidungsproblem ist stark NP-vollständig.
→ Mächtigkeit größte Gruppe $G_S = \max\{|S_i| \mid 1 \leq i \leq m\}$
→ Grad eines Objekts: $deg_S(u) = |\{S_i \mid u \in S_i\}|$
⇒ Grad von S: $\Delta_S = \max\{deg_S(u) \mid u \in V\}$

Ganzzahliges LP X für SetCover:

$$\begin{aligned} &\textit{minimiere} && \sum x_i \\ &\textit{s.t.} && \sum_{i:u \in S_i} x_i \geq 1 && \forall u \in V \\ &&& x_i \in \{0,1\} && \forall i \end{aligned}$$

→ Größe des ILP ist $\mathcal{O}(nm)$.
→ Summe der Nebenbedingungen ist höchstens Δ_S.
→ Um zu bestimmen, wie gut ein Rundungsansatz sein kann, berechne untere Schranke für die Ganzzahligkeitslücke.

Satz 23 (*Untere Schranke Ganzzahligkeitslücke*)

Für die Ganzzahligkeitslücke γ der Relaxierung X_{rel} gilt:

$$\boxed{\gamma \geq \frac{1}{2}\log(n)}$$

→ Bemerke: SetCover ist Minimierungsproblem, daher Superoptimalität: $OPT(X_{rel}) \leq OPT(S)$

Deterministisches Runden

→ X_{rel} wird in Polyzeit $L(nm, m)$ gelöst.

Algorithmus 17 DetRoundSC

Input: Instanz des SetCover.
Output: Überdeckung S_{Cov}.

```
Löse Relaxierung X_rel für SetCover.
S_Cov = ∅.
for i = 1, ..., m do
    if x_i ≥ 1/Δ_S then
        S_Cov = S_Cov ∪ {S_i}.
return S_Cov
```

→ Garantierte relative Güte:

$$\boxed{DetRoundSC(S) = |S_{Cov}| \leq \sum \Delta_S x_i \leq \Delta_S OPT(S)}$$

→ Laufzeit: $\mathcal{O}(m + L(nm, m)) = \mathcal{O}(m^6 n^2)$

→ S_{Cov} ist Überdeckung wegen Durschnittsargument:
⇒ Summe der NB besteht aus $deg_S(u)$ Summanden.
⇒ Mindestens einer der Summanden x_i hat mindesten Wert $\frac{1}{deg_S(u)} \geq \frac{1}{\Delta_S}$, da $\sum x_i \geq 1$.
⇒ Zur Not wird für mindestens dieses u S_i aufgenommen.

Dualität

→ Ziel: Untere Schranke für Minimierungsproblem herleiten.

Primal:		Dual:	
minimiere	$z(x) = c^T x$	*maximiere*	$\xi(y) = b^T y$
s.t.	$Ax \geq b$	*s.t.*	$A^T y \leq c$
	$x \geq 0$		$y \geq 0$

→ Jede zulässige Lösung y liefert eine untere Schranke für die ZF des Primalen: $\xi(y) \leq z(x)$
⇒ Beziehung nennt man schwache Dualität.

Satz 24 (*Dualitätssatz*)

Seien x_{opt}, y_{opt} optimale Lösungen von primal bzw. dual. Dann gilt $z(x_{opt}) = \xi(y_{opt})$.

Satz 25 (*Satz vom komplementären Schlupf*)

Seien x und y zulässige Lösungen des Primals bzw. des Duals, dann sind x und y genau dann optimale Lösungen, wenn für alle i und j gilt:

$$y_j > 0 \Rightarrow row_j[A]x = b_j$$
$$row_j[A]x - b_j < 0 \Rightarrow y_j = 0$$
$$x_i > 0 \Rightarrow row_i[A^T]y = c_i$$
$$row_i[A^T]x - c_i < 0 \Rightarrow x_i = 0$$

→ Zum Approximieren: Ignoriere erste Zeile und erfülle zweite Zwangsweise.
→ Da x_i oft binär, setze die $x_i = 1$, wo Schlupf 0 ist.

Dualität und ApproxAlg

→ Mit Y_{rel} relaxiertes Duales gilt für Minimierungsprobleme folgender Zusammenhang:

$$\xi(y) \leq OPT(Y_{rel}) = OPT(X_{rel}) \leq OPT(X) = OPT(I) \leq z(x)$$

Primales SetCover:	Duales SetCover:
$\begin{array}{lll} min & \sum x_i & \\ s.t. & \sum_{i:u_j \in S_i} x_i \geq 1 & \forall u_j \in V \\ & x \in 0,1 & \forall i \end{array}$	$\begin{array}{lll} max & \sum y_j & \\ s.t. & \sum_{j:u_j \in S_i} y_j \leq 1 & \forall S_i \in S \\ & 0 \leq y_j \leq 1 & \forall j \end{array}$

→ $H(n)$ ist Harmonische Reihe.

Algorithmus 18 PrimalDualSC2

Input: Instanz von SetCover.
Output: Gültige Überdeckung.

for $i = 1, \ldots, m$ **do**
  $x_i = 0$
$C = \emptyset$ ▷ schon überdeckte Objekte
while $C \neq V$ **do**
  Bestimme Index i mit $|S_i \setminus C|$ maximal.
  $x_i = 1$
  for $\forall u_j \in S_i \setminus C$ **do** ▷ nur für Analyse
    $preis(u_j) = \frac{1}{|S_i \setminus C|}$
    $y_j = \frac{1}{H(G_S)} preis(u_j)$
  $C = C \cup S_i$
return $(x_1, \ldots, x_m)$

→ Garantierte relative Güte: $\boxed{H(G_S) \leq \ln(n) + 1}$
→ Laufzeit: $\mathcal{O}(nm)$

Hinweise aus den Übungen: