Approximationsalgorithmen

Marco Ammon

15. Oktober 2020

Inhaltsverzeichnis

1 Allgemeine Definitionen

1.1 Kombinatorisches Optimierungsproblem Π

$$\begin{aligned}\mathcal{D} &= \text{Menge der Eingaben } I\\ \mathcal{S}(I \in \mathcal{D}) &= \text{Menge der zur Eingabe } I \text{ zulässigen Lösungen}\\ f : \mathcal{S}(I) \mapsto \mathbb{N}^{\neq 0} &= \text{Bewertungs-/Kosten-/Maßfunktion}\\ \text{ziel} &\in \{\min, \max\}\end{aligned}$$

- Beschränkung auf natürliche Zahlen, weil Vergleich reeller Zahlen bislang nicht beweisbar schnell funktioniert.
- Ausschluss der 0 für spätere Definitionen sinnvoll (lässt sich durch Modifikation von f in der Regel trivial erreichen)

Gesucht ist zu $I \in \mathcal{D}$ eine zulässige Lösung $\sigma_{\text{opt}} \in \mathcal{S}(I)$, sodass

$$\text{OPT}(I) = f(\sigma_{\text{opt}}) = \text{ziel}\{f(\sigma) \mid \sigma \in \mathcal{S}(I)\}$$

1.1.1 Das Rucksackproblem Rucksack

$$\begin{aligned}\mathcal{D} &= \{\langle W, \text{vol}, p, B\rangle \mid \{1, \ldots, n\}, \text{vol} : W \mapsto \mathbb{N}, p : W \mapsto \mathbb{N}, B \in \mathbb{N}, \forall w \in W : vol(w) \leq B\}\\ \mathcal{S}(\langle W, \text{vol}, p, B\rangle) &= \{A \subseteq W \mid \sum_{w \in A} \text{vol}(w) \leq B\}\\ f(A) &= \sum_{w \in A} p_w\\ \text{ziel} &= \max\end{aligned}$$

Die Restriktion der verfügbaren Waren auf solche, die in den leeren Rucksack passen, hilft gegen das künstliche Aufblähen der Eingabe.

1.1.2 Das Knotenfärbungsproblem Col

$$\begin{aligned}\mathcal{D} &= \{\langle G\rangle \mid G = (V, E) \text{ ein ungerichteter Graph mit mindestens einer Kante}\}\\ \mathcal{S}(\langle G\rangle) &= \{c_{\text{V}} \mid c_{\text{V}} \text{ ist eine Knotenfärbung von } G\}\\ f(c_{\text{V}}) &= |c_{\text{V}}(V)|\\ \text{ziel} &= \min\end{aligned}$$

Die Größe der kleinsten möglichen Knotenfärbung ist die chromatische Zahl $\chi(G)$.

1.1.3 Das Kantenfärbungsproblem EdgeCol

$$\begin{aligned}\mathcal{D} &= \{\langle G\rangle \mid G = (V, E) \text{ ein ungerichteter Graph mit mindestens einer Kante}\}\\ \mathcal{S}(\langle G\rangle) &= \{c_{\text{E}} \mid c_{\text{E}} \text{ ist eine Kantenfärbung von } G\}\\ f(c_{\text{E}}) &= |c_{\text{E}}(E)|\\ \text{ziel} &= \min\end{aligned}$$

Die Größe der kleinsten möglichen Kantenfärbung ist der chromatische Index $\chi'(G)$.

1.1.4 Das Traveling-Salesperson-Problem TSP

$$\begin{aligned}
\mathcal{D} &= \{\langle K_n, c\rangle \mid K_n \text{ vollständiger Graph auf } n \text{ Knoten}, c : E \mapsto \mathbb{N}, \} \\
\mathcal{S}(\langle K_n, c\rangle) &= \{C \mid C = (v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_i}) \text{ ist ein Hamiltonkreis}\} \\
f(c_{\mathrm{E}}) &= c(v_{i_n}, v_{i_1}) + \sum_{j=1}^{n-1} c(v_{i_j}, v_{i_{j+1}}) \\
\text{ziel} &= \min
\end{aligned}$$

1.1.5 Das metrische Traveling-Salesperson-Problem ΔTSP

$$\begin{aligned}
\mathcal{D} &= \{\langle K_n, c\rangle \mid K_n \text{ vollständiger Graph auf } n \text{ Knoten}, c : E \mapsto \mathbb{N}, \\
&\quad \underbrace{\forall u, v, w \in V : c(u, v) \leq c(u, w) + c(w, v)}_{\text{Dreiecksungleichung}}\} \\
\mathcal{S}(\langle K_n, c\rangle) &= \{C \mid C = (v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_i}) \text{ ist ein Hamiltonkreis}\} \\
f(c_{\mathrm{E}}) &= c(v_{i_n}, v_{i_1}) + \sum_{j=1}^{n-1} c(v_{i_j}, v_{i_{j+1}}) \\
\text{ziel} &= \min
\end{aligned}$$

1.1.6 Das Problem der Unabhängigen Knotenmengen IS

$$\begin{aligned}
\mathcal{D} &= \{\langle G\rangle \mid G = (V, E) \text{ ein Graph}\} \\
\mathcal{S}(\langle G\rangle) &= \{U \mid U \subseteq V, \forall u, v \in U : (u, v) \notin E\} \\
f(U) &= |U| \\
\text{ziel} &= \max
\end{aligned}$$

1.1.7 Das Erfüllbarkeitsproblem Max-SAT

Sei $V = \{x_1, \ldots, x_n\}$ die Menge der Variablen. Als Literal l bezeichnet man eine Variable $x_i \in V$ oder ihre Negation $\overline{x_i}$. Eine Oder-Klausel (kurz Klausel) $C = l_1 \vee \cdots \vee l_k$ ist eine Oder-Verknüpfung von Literalen.

Eine Boolesche (n, m)-Formel $\Phi = C_1 \wedge \cdots \wedge C_m$ in konjunktiver Normalform ist eine Und-Verknüpfung von Oder-Klauseln aus n Variablen aus V.

$$\begin{aligned}
\mathcal{D} &= \{\langle \Phi\rangle \mid \Phi \text{ eine boolesche } (n, m)\text{-Formel in KNF}\} \\
\mathcal{S}(\langle \Phi\rangle) &= \{b \mid b : V \mapsto \{\textsc{False}, \textsc{True}\}\} \\
\text{wahr}(\Phi) &= |\{j \mid C_j \in \Phi, b(C_j) = \textsc{True}\}| \\
\text{ziel} &= \max
\end{aligned}$$

1.1.8 Das Mengenüberdeckungsproblem SetCover

$$\begin{aligned}
\mathcal{D} &= \{S \mid S = \{S_1, \ldots, S_m\}, V = \cup_{i=1}^{m} S_i = \{u_1, \ldots, u_n\}\} \\
\mathcal{S}(S) &= \{S_{\text{cov}} \mid S_{\text{cov}} = \{S_{i_1}, \ldots, S_{i_l}\}, V = \cup_{j=1}^{l} S_{i_j}\} \\
f(S_{\text{cov}}) &= |S_{\text{cov}}| \\
\text{ziel} &= \min
\end{aligned}$$

1.2 $t(n)$-Zeit-Approximationsalgorithmus

Berechnet ein Algorithmus A in Zeit $t(|I|)$ eine Ausgabe $\sigma_I^A \in \mathcal{S}(I)$ für Eingabe $I \in \mathcal{D}$, wird er als $t(n)$-Zeit-Approximationsalgorithmus bezeichnet. Es gilt die Schreibweise $A(I) = f(\sigma_I^A)$.

1.3 Pseudo-polynomieller Algorithmus

Sei Π ein kombinatorisches Optimierungsproblem, sodass in allen Instanzen I alle vorkommenden Zahlen natürliche Zahlen sind. Sei $\text{maxnr}(I)$ die größte in I vorkommende Zahl. Ein Algorithmus wird als pseudo-polynomiell bezeichnet, falls es ein Polynom $\text{poly}(\cdot,\cdot)$ gibt, sodass für alle Instanzen I seine Laufzeit $\text{poly}(|I|, \text{maxnr}(I))$ ist.

Kann also das Problem so eingeschränkt werden, dass für alle Instanzen I die größte vorkommende Zahl durch ein Polynom begrenz wird, also $\text{maxnr}(I) \leq q(|I|)$ mit Polynom q gilt, so ist auch die Laufzeit des Algorithmus polynomiell.

1.4 Starke NP-Vollständigkeit

Ein NP-vollständiges Entscheidungsproblem L wird als stark NP-vollständig bezeichnet, wenn es ein Polynom q gibt, sodass $L_q = \{x \mid x \in L, \text{maxnr}(x) \leq q(|x|)\}$ NP-vollständig ist. Gibt es kein solches Polynom, gilt L als schwach NP-vollständig.

Eine äquivalente Charakterisierung ist: Das NP-vollständige Entscheidungsproblem L ist stark vollständig, falls es keinen pseudo-polynomiellen Algorithmus für L gibt (unter der Annahme $P \neq NP$).

- Hamilton und Clique sind stark NP-vollständig, da bei ihnen das Polynom $q(n) = n$ ausreicht.
- TSP ist stark NP-vollständig, da sogar das ΔTSP mit Gewichten 1 und 2 NP-vollständig ist.
- Rucksack ist schwach NP-vollständig, weil es einen pseudo-polynomiellen Algorithmus für die Optimierungsvariante gibt, der auch für das Entscheidungsproblem verwendet werden kann.

2 Absolute Gütegarantie

2.1 Definition

1. A hat bei Eingabe I absolute Güte von
$$\kappa_A(I) = |A(I) - \text{OPT}(I)|$$
2. Die absolute Worst-Case-Güte von A abhängig von der Eingabelänge $n = |I|$ ist die Funktion
$$\kappa_A^{\text{wc}}(n) = \max\{\kappa_A(I) \mid I \in \mathcal{D}, |I| <= n\}$$
3. A garantiert eine absolute Güte von $\kappa_A : \mathbb{N} \mapsto \mathbb{N}$, falls für alle $n \in \mathbb{N}$ gilt:
$$\kappa_A^{\text{wc}}(n) \leq \kappa_A(n)$$
4. A hat eine absolute Abweichung von $\kappa'_A : \mathbb{N} \mapsto \mathbb{N}$, falls für unendlich viele n gilt
$$\kappa'_A(n) \leq \kappa_A^{wc}(n)$$
Eine unendlich große Menge $\mathcal{D}' \subseteq \mathcal{D}$ heißt $\kappa'_A(n)$-Zeugenmenge gegen A, wenn für alle $I \in \mathcal{D}'$ gilt:
$$\kappa_A(I) \geq \kappa'_A(|I|)$$

2.2 Unmöglichkeitsergebnis für das Rucksackproblem

Satz. *Falls $P \neq NP$, dann gibt es keine Konstante $n \in \mathbb{N}$, sodass es einen polynomiellen Approximationsalgorithmus A für* Rucksack *gibt mit*

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

Widerspruchsbeweis. Unter der Annahme, dass A und k existieren, kann Rucksackin Polynomzeit exakt gelöst werden, was $P = NP$ zur Folge hat:

Konstruiere aus einer Instanz $I = \langle W, \mathrm{vol}, p, B\rangle$ eine neue Probleminstanz $I' = \langle W, \mathrm{vol}, p', B\rangle$ mit $p'(w) = (k+1) \cdot p(w)$. Eine zulässige Lösung σ für I ist auch eine zulässige Lösung für I'. Gleiches gilt aufgrund der Monotonie der Multiplikation auch für optimale Lösungen. Durch die Multiplikation aller Preise mit $k+1$ beträgt die „Lücke“ zwischen den optimalen und der ersten nicht-optimalen Lösung für I' mindestens $k+1$.

Da A eine absolute Güte von k garantiert und in Polynomzeit terminiert, kann es nur eine optimale Lösung für I', welche auf optimal für I ist, zurückgeben. Damit ist das NP-vollständige Rucksack in Polynomzeit exakt lösbar. □

Die hierbei verwendete Vorgehensweise einer Selbstreduktion sowie das „Aufblasen“ des Problems („Scaling“, „Gap Amplification“) lässt sich auch auf viele andere Probleme wie etwa SetCover anwenden. Folglich kann eine konstante Gütegarantie nur für vergleichsweise wenig Probleme erreicht werden.

3 Relative Gütegarantie

3.1 Definition

1. A hat bei Eingabe I eine relative Güte von

$$\rho_A(I) = \max\left\{\frac{A(I)}{\mathrm{OPT}(I)}, \frac{\mathrm{OPT}(I)}{A(i)}\right\} \geq 1$$

2. Die relative Worst-Case-Güte von A ist die Funktion

$$\rho_A^{\mathrm{wc}}(n) = \max\{\rho_A(I) \mid I \in \mathcal{D}, |i| \leq n\}$$

3. A garantiert eine relative Güte von $\rho_A : \mathbb{N} \mapsto \mathbb{N}$, falls für alle $n \in \mathbb{N}$ gilt

$$\rho_A^{\mathrm{wc}}(n) \leq \rho_A(n)$$

4. A macht für die Eingabe $I \in \mathcal{D}$ einen relativen Fehler von

$$\varepsilon_A(I) = \frac{|A(I) - \mathrm{OPT}(I)|}{\mathrm{OPT}(I)} = |\frac{A(I)}{\mathrm{OPT}(I)} - 1|$$

5. A garantiert einen relativen Fehler von $\varepsilon_A(n)$, falls für alle $\{I \mid I \in \mathcal{D}, |I| \leq n\}$ gilt

$$\varepsilon_A(I) \leq \varepsilon_A(n)$$

6. A hat eine relative Abweichung von $\rho'_A : \mathbb{N} \mapsto \mathbb{N}$, falls für unendlich viele n gilt

$$\rho_A^{\mathrm{wc}}(n) \geq \rho'_A(n)$$

   Eine unendlich große Menge $\mathcal{D}' \subseteq \mathcal{D}$ heißt $\rho'_A(n)$-Zeugenmenge gegen A, wenn für alle $I \in \mathcal{D}'$ gilt

$$\rho_A(I) \geq \rho'_A(|I|)$$

Es folgen daraus direkt, dass

1. bei einem Minimierungsproblem $1 + \varepsilon_A(n) = \rho_A(n)$ ist.
2. bei einem Maximierungsproblem $1 - \varepsilon_A(n) = \frac{1}{\rho_A(n)}$ ist.
3. für alle Probleme $\varepsilon_A(n) \leq \rho_A(n) - 1$ ist.

Weiter lassen sich damit obere bzw. untere Schranken der Optimallösung aus einer approximierten Lösung angeben. Es folgt, dass

1. bei einem Minimierungsproblem gilt
$$\frac{1}{\rho_A(|I|)} \cdot A(I) \leq \mathrm{OPT}(I) \leq A(I) \leq \rho_A(|I|) \cdot \mathrm{OPT}(I)$$
2. bei einem Maximierungsproblem gilt
$$\frac{1}{\rho_A(|I|)} \cdot \mathrm{OPT}(I) \leq A(I) \leq \mathrm{OPT}(I) \leq \rho_A(|I|) \cdot A(I)$$
3. bei beiden Problemtypen mit der Beziehung
$$|A(I) - \mathrm{OPT}(I)| \leq \varepsilon_A(|I|) \cdot \mathrm{OPT}(I)$$
gilt
$$(1 - \varepsilon_A(|I|)) \cdot \mathrm{OPT}(I) \leq A(I) \leq (1 + \varepsilon_A(|I|)) \cdot \mathrm{OPT}(I)$$

3.2 Unmöglichkeitsergebnis für das allgemeine (volle) TSP

Satz. *Wenn es einen polynomiellen Approximationsalgorithmus A mit konstanter relativer Gütegarantie r für das volle TSP gibt, dann gilt $P = NP$.*

Beweis durch Reduktion. Durch Benutzung von A mit beliebiger konstanter relativer Gütegarantie $r \in \mathbb{N}$ kann HAMILTON auf das volle TSPreduziert werden. Es wird also HAMILTON $\leq$ pTSP$[r]$ für alle r gezeigt:

Sei der Graph $G = (V, E)$ mit $n = |V|$, gegeben. Dazu wird nun passend eine Probleminstanz $I_G = \langle K_n, c \rangle$ für TSPerzeugt. c wird wie folgt konstruiert:

$$c(u, v) = \begin{cases} 1 & \text{falls } \{u, v\} \in E \text{ („kurze“ Kante)} \\ (r-1) \cdot n + 2 & \text{sonst („lange“ Kante)} \end{cases}$$

I_G kann in Polynomzeit aus G berechnet werden. Es gilt weiter:

- $G \in$ HAMILTON $\Rightarrow$ kürzeste Rundreise in I_G hat die Länge n
- $G \notin$ HAMILTON $\Rightarrow$ kürzeste Rundreise in I_G nimmt mindestens eine der langen Kanten und hat damit eine Länge von mindestens
$$(r-1) \cdot n + 2 + n - 1 = r \cdot n + 1 > r \cdot n$$
- I_G besitzt keine zulässige Lösung σ mit $n + 1 \leq c(\sigma) \leq r \cdot n$.

Durch den folgenden Algorithmus kann also HAMILTON entschieden werden:

1: konstruiere I_G
2: approximiere mit A eine kürzeste Rundreise $A(I_G)$
3: **if** $A(I_G) > r \cdot |V|$ **then**

4: **return** $G \notin$ HAMILTON
5: **else**
6: **return** $G \in$ HAMILTON
7: **end if**

□

Der Ansatz der Konstruktion von Probleminstanzen anderer NP-schwerer Probleme und der anschließenden Verwendung eines Scaling-Arguments kann auch für weitere Probleme verwendet werden. Ebenfalls können damit bestimmte Bereiche für mögliche konstante relative Gütegarantien ausgeschlossen werden, etwa $\rho < \frac{3}{2}$ bei BINPACKING.

4 Approximationsschemata

Während die Ergebnisse von Approximationsalgorithmen mit absoluter oder relativer Gütegarantie nur durch eine Modifikation oder Wechsel des Algorithmus verbessert werden können, ist es manchmal gewünscht, im Gegenzug für eine verlängerte Laufzeit eine bessere Güte zu erreichen. Dafür sind sogenannte Approximationsschemata geeignet.

4.1 Definition

Sei Π ein Optimierungsproblem und A ein Approximationsalgorithmus für Π, der eine Probleminstanz I von Π und ein $0 < \varepsilon < 1$ bekommt.

1. A ist ein polynomielles Approximationsschema (PAS) für Π, wenn A zu jedem I und für jedes ε in Zeit $\mathcal{O}(\text{poly}(|I|))$ eine zulässige Lösung zu I mit relativem Fehler $\varepsilon_A(I, \varepsilon) \leq \varepsilon$ berechnet.

2. A ist ein streng polynomielles Approximationsschema (FPAS), wenn A ein PAS mit Laufzeit $\mathcal{O}(poly\left(|I|, \frac{1}{\varepsilon}\right))$ ist.

Satz (Umwandlung eines (F)PAS in einen exakten Algorithmus)**.** *Sei A ein (F)PAS und zu jeder Eingabe I $Z(I)$ eine obere Schranke. Sei $\varepsilon^* = \frac{1}{Z(I)+1}$, dann ist $A(I, \varepsilon^*) = \text{OPT}(I)$. Sofern A ein FPAS ist, liegt die Laufzeit in $\mathcal{O}(\text{poly}(|I|, Z(I)))$.*

Beweis. Starte A mit Eingabe I und ε^*. Es wird eine zulässige Lösung zu I gefunden, für ihren relativen Fehler gilt

$$\varepsilon_a(I, \varepsilon^*) = \frac{|\text{OPT}(I) - A(I, \varepsilon^*)|}{\text{OPT}(I)} \leq \varepsilon^*$$

Weil $\text{OPT}(I) \leq Z(I)$ beschränkt ist, folgt für die Abweichung

$$|\text{OPT}(I) - A(I, \varepsilon^*)| \leq \varepsilon^* \cdot \text{OPT}(I) = \frac{\text{OPT}(I)}{Z(I) + 1} < 1$$

Da die Werte zulässiger Lösungen immer ganzzahlig sind, folgt $|\text{OPT}(I) - A(I, \varepsilon^*)| = 0$, damit also die Optimalität von $A(I, \varepsilon^*)$. □

4.2 Unmöglichkeitsergebnisse für Approximationschemata

Satz. *Sei Π ein Optimierungsproblem. Wenn es ein Polynom $q(\cdot, \cdot)$ gibt, sodass $\forall I \in \mathcal{D}$: $\text{OPT}(I) \leq q(|I|, \text{maxnr}(I))$ gilt, dann folgt aus der Existenz eines FPAS für Π, dass ein pseudo-polynomieller exakter Algorithmus für Π existiert.*

Satz. *Wenn es für eine Optimierungsvariante eines stark NP-vollständigen Problems ein FPAS gibt, dann folgt $P = NP$.*

5 GreedyIS für IS

1: $U = \emptyset, t = 0, V^{(0)} = V$
2: **while** $V^{(t)} \neq \emptyset$ **do**
3: $\quad G^{(t)} =$ der durch $V^{(t)}$ induzierte Graph
4: $\quad u_l =$ ein Knoten mit minimalem Grad in $G^{(t)}$
5: $\quad V^{(t+1)} = V^{(t)} - (\{u_l\} \cup \Gamma_{G^{(t)}}(u_l))$
6: $\quad U = U \cup \{u_l\}$
7: $\quad t = t + 1$
8: **end while**
9: **return** U

Satz. *Sei G ein knoten-k-färbbarer Graph, dann ist*

$$\textsc{GreedyIS}(G) \geq \left\lceil \log_k \left(\frac{|V|}{3} \right) \right\rceil$$

Beweis. Mit dem folgenden Hilfslemma kann eine Beziehung zwischen der Anzahl der notwendigen Farben und dem minimalen Grad des Graphs hergestellt werden.

Lemma. *Sei G ein knoten-k-färbbarer Graph, dann gilt:*

$$\exists u \in V : \deg_G(u) \leq \left\lfloor \left(1 - \frac{1}{k}\right) \cdot |V| \right\rfloor$$

Beweis. Da G mit k Farben gefärbt ist, gibt es k Mengen U_i an Knoten, die jeweils mit der gleichen Farbe i gefärbt sind. Es muss nach einem Durchschnitsargument eine Menge U_i mit $|U_i| \geq \left\lceil \frac{1}{k} \cdot |V| \right\rceil$ geben. Jeder der Knoten u in U_i kann maximal mit allen Knoten aus $V \setminus U_i$ verbunden sein. Es folgt also

$$\deg_G(u) \leq |V| - |U_i| \leq |V| - \left\lceil \frac{1}{k} \cdot |V| \right\rceil = \left\lfloor \left(1 - \frac{1}{k}\right) \cdot |V| \right\rfloor$$

□

Zur Vereinfachung gelte $n = |V|$ und $n_t = |V^{(t)}|$. Es kann $k \geq 2$ angenommen werden. Mit dem Hilfslemma ergibt sich für die Anzahl der Knoten folgende Rekursion:

$$\begin{aligned} n_0 &= n \\ n_{t+1} &\geq n_t - \left\lfloor \left(1 - \frac{1}{k}\right) \cdot n_t \right\rfloor - 1 \geq \frac{n_t}{k} - 1 \end{aligned}$$

Sie kann zur Ungleichung

$$n_t \geq \frac{n}{k^t} - \underbrace{\frac{k}{k-1} \cdot \left(1 - \frac{1}{k^t}\right)}_{\leq 2 \text{ für } k \geq 2} \geq \frac{n}{k^t} - 2$$

aufgelöst werden. Solange $n_t \geq 1$ gilt, wird ein neuer Knoten nach U gelegt. Durch Umformen obiger Ungleichung lässt sich dies für $t \geq \log_k \left(\frac{n}{3}\right)$ garantieren. Es folgt also $|U| \geq \left\lceil \log_k \left(\frac{n}{3}\right) \right\rceil$. □

6 Knotenfärbungsalgorithmen

6.1 GreedyCol

1: **for all** $u_i \in V$ **do**
2: $\quad c_V(u_i) = \infty$

3: **end for**
4: **for all** $i \in \{1, \ldots, |V|\}$ **do**
5: $\quad c_V(u_i) = \min\{\mathbb{N} \setminus \{c_V(\Gamma(u_i))\}\}$
6: **end for**
7: **return** c_V

Satz. *GreedyCol berechnet in Zeit $\mathcal{O}(|V| + |E|)$ eine Knotenfärbung aus höchstens $\Delta(G) + 1$ Farben.*

Beweis. Da ein Knoten u maximal $\Delta(G)$ viele Nachbarn haben kann, muss in $[1, \ldots, \Delta(G) + 1]$ noch mindestens eine Farbe frei sein. □

Satz. *GreedyCol garantiert eine absolute Güte von*

$$\kappa_{GreedyCol}(G) = GreedyCol(G) - \mathrm{OPT}(G) \leq \Delta(G) + 1 - 2 = \Delta(G) - 1$$

, weil die untere Schranke $\mathrm{OPT}(G) \geq 2$ *für Graphen mit* $|V| \geq 2$ *gilt.*

Zeuge. $\Delta(G) - 1$*-Zeuge gegen GreedyCol: TODO (Abbildung 2.1)*

6.2 GreedyCol2

1: $t = 1, V^{(1)} = V$
2: **while** $V^{(t)} \neq \emptyset$ **do**
3: $\quad G^{(t)} =$ der durch $V^{(t)}$ induzierte Graph
4: $\quad U_t = \textsc{GreedyIS}(G^{(t)})$
5: $\quad$ färbe alle Knoten in U_t mit Farbe t
6: $\quad V^{(t+1)} = V^{(t)} - U_t$
7: $\quad t = t + 1$
8: **end while**
9: **return** berechnete Färbung

Satz. *Für einen knoten-k-färbbaren Graph $G = (V, E)$ mit $n = |V|$ gibt GreedyCol2 eine Färbung mit höchstens $\frac{3n}{\log_k\left(\frac{n}{16}\right)}$. Die relative Gütegarantie liegt als in $\mathcal{O}\left(\frac{n}{\log n}\right)$.*

Beweis. Zur Vereinfachung bezeichne $n_t = |V^{(t)}|$. Aus der Analyse von GreedyIS folgt $|U_t| \geq \log_k\left(\frac{n_t}{3}\right)$. Es ergibt sich die Rekursion

$$n_1 = n$$
$$n_{t+1} \leq n_t - \log_k\left(\frac{n_t}{3}\right)$$

Nun wird bestimmt, für welches t $n_t < 1$ eintritt, denn dann bricht der Algorithmus ab. Behelfsmäßig sei $n_t \geq \frac{n}{\log_k\left(\frac{n}{16}\right)}$. Mit der Beziehung $\frac{n}{\log_k n} \geq \frac{3}{4} \cdot \sqrt{n}$ ergibt sich

$$\log_k\left(\frac{n_t}{3}\right) \geq \log_k\left(\frac{n}{3 \cdot \log_k n}\right) \geq \log_k\left(\sqrt{\frac{n}{16}}\right) = \frac{1}{2} \cdot \log_k\left(\frac{n}{16}\right)$$

Solange $n_t \geq \frac{n}{\log_k\left(\frac{n}{16}\right)}$ also gilt, werden pro Runde mindestens $\frac{1}{2} \cdot \log_k\left(\frac{n}{16}\right)$ Knoten pro Runde gefärbt. Nach höchstens $t \leq \frac{2n}{\log_k\left(\frac{n}{16}\right)}$ gilt die Ungleichung nicht mehr. Färbt man jetzt alle verbliebenen Knoten mit jeweils einer eigenen Farbe, werden insgesamt maximal $\frac{n}{\log_k\left(\frac{n}{16}\right)} + t \leq \frac{3n}{\log_k\left(\frac{n}{16}\right)}$ vergeben.

Mit $k = \chi_G = \mathrm{OPT}(G)$ ergibt sich für die relative Gütegarantie:

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

□

7 Kantenfärbungsalgorithmen

TODO: Übung

8 Christofides' Algorithmus CH für ΔTSP

Ein Matching M eines kantengewichteten Graphen G ist ein Teilgraph von G mit $\Delta(G) \leq 1$. Ist G ein vollständiger Graph mit $|V|$ gerade, dann gibt es perfekte Matchings. In einem perfekten Matching haben alle Knoten genau den Grad 1. Ein perfektes Matching mit kleinstmöglichem Gewicht wird als leichtestes Matching bezeichnet. Ein solches leichtestes Matching kann in $\mathcal{O}(n^{2.5} \cdot (\log n)^4)$ berechnet werden.

Als Multi-Graph wird ein Graph bezeichnet, der um mehrere Kanten zwischen den gleichen Knoten erweitert wurde.

Wird in einem Pfad jede Kante des (Multi-)Graph genau einmal besucht, so spricht man von einem Euler-Pfad. Bildet der Pfad einen Kreis, so nennt man ihn Euler-Kreis oder Euler-Tour. Haben alle Knoten von G geraden Grad, so existiert eine Euler-Tour in G. Diese lässt sich in $\mathcal{O}(|V| + |E|)$ berechnen.

Der Algorithmus von Christofides (CH) geht wie folgt vor:

1: berechne einen minimalen Spannbaum T_{CH} von $I = \langle K_n, c \rangle$
2: $S = \{v \in T_{\text{CH}} \mid \deg_{T_{\text{CH}}}(v) \text{ ungerade}\}$ ▷ $|S|$ ist gerade
3: berechne auf dem durch S induzierten Teilgraphen des K_n ein leichtestes Matching M_{CH}
4: berechne eine Euler-Tour $E = (u_1, u_2, \dots)$ auf $T_{\text{CH}} \cup M_{\text{CH}}$ ▷ $T_{\text{CH}} \cup M_{\text{CH}}$ kann Multi-Graph sein, alle Knoten haben geraden Grad
5: entferne Wiederholungen von Knoten in E, sodass man E' erhält
6: **return** E'

Satz. *CH, gestartet mit einer Eingabe auf n Knoten, garantiert eine relative Güte von* $\rho_{\text{CH}} \leq \frac{3}{2} - \frac{1}{n}$ *in einer Laufzeit von* $\mathcal{O}(n^{2.5} \cdot (\log n)^4)$.

Beweis. Sei R^* eine optimale Rundreise für I, d.h. $c(R^*) = \text{OPT}(I)$. Es gilt $\text{CH}(I) = c(E') \leq (\frac{3}{2} - \frac{1}{n}) \cdot c(R^*)$ zu zeigen.

1. Da R^* aus n Kanten besteht, muss durch ein Durchschnittsargument mindestens eine Kante e mit $c(e) \geq \frac{c(R^*)}{n}$ existieren. Wird diese aus R^* entfernt, so enthält man einen Spannbaum des K_n. Da T_{CH} minimal ist, gilt

$$c(T_{\text{CH}}) \leq c(R^*) - \frac{c(R^*)}{n} = \left(1 - \frac{1}{n}\right) \cdot c(R^*)$$

2. In beliebigen Bäumen ist die Anzahl der Knoten mit ungeradem Grad gerade.

3. Zur Vereinfachung werden die Knoten so umbenannt, dass $R^* = (u_1, u_2, \dots, u_n, u_1)$ ist. S kann dann als $S = \{u_{i_1}, \dots, u_{i_{|S|}}\}$ mit $i_1 < \dots < i_{|S|}$ geschrieben werden.

   Aus S kann ein Kreis $H = (u_{i_1}, \dots, u_{i_{|S|}}, u_{i_1})$ gebildet werden. Durch die Dreiecksungleichung ($|H| \leq n$ und jede „Abkürzung“ ist maximal gleich lang wie der Weg in R^*) gilt $c(H) \leq c(R^*)$.

   Es können zwei perfekte Matching M_1 und M_2 auf H berechnet werden, denn $|S|$ ist gerade. Weil M_{CH} minimal ist, folgt o.B.d.A. mit $c(M_1) \leq c(M_2)$ die Aussage

$$c(M_{\text{CH}}) \leq c(M_1) \leq \frac{1}{2} \cdot (c(M_1) + c(M_2)) = \frac{1}{2} \cdot c(H) \leq \frac{1}{2} \cdot c(R^*)$$

4. Da jeder Knoten in $T_{\text{CH}} \cup M_{\text{CH}}$ geraden Grad hat, kann eine Euler-Tour E berechnet werden. Weil diese nur Kanten aus $T_{\text{CH}} \cup M_{\text{CH}}$ benutzt, kann ihre Länge mit den vorherigen Ergebnissen wie folgt beschränkt werden:

$$c(E) = c(T_{\text{CH}} \cup M_{\text{CH}}) \leq \left(1 - \frac{1}{n}\right) \cdot c(R^*) + \frac{1}{n} \cdot c(R^*) = \left(\frac{3}{2} - \frac{1}{n}\right) \cdot \text{OPT}(I)$$

5. Durch die Dreiecksungleichung kann E' nicht länger als E werden.

□

Zeuge. *TODO*

9 Approximationsschema für Rucksack

9.1 DynRucksack zur exakten Lösung von Rucksack

Für eine Instanz $I = \langle W, \text{vol}, p, B \rangle$ kann direkt eine obere und eine untere Grenze für den maximalen Wert der Füllung angegeben werden:

$$P_{\max} \leq \text{OPT}(I) \leq n \cdot P_{\max}$$

Sei $F_j(\alpha)$, wobei $j \in \{0, 1, \ldots, n\}$ und $\alpha \in \mathbb{Z}$ gilt, das kleinste benötigte Rucksackvolumen, mit dem man einen Wert von mindestens α erreichen kann, wenn man die ersten j Waren einpacken darf. Die formale Definition

$$F_j(\alpha) = \min\{\text{vol}(R) \mid R \subseteq \{1, \ldots, j\}, p(R) \geq \alpha\}$$

lässt sich durch die folgende Rekursion ausdrücken:

$$f_j(\alpha) = \begin{cases} 0 & \text{falls } \alpha \leq 0 \\ \infty & \text{falls } \alpha \geq 1, j = 0 \\ \min\{F_{j-1}(\alpha - p_j) + \text{vol}(j), F_{j-1}(\alpha)\} & \text{sonst (sog. Bellmannsche Optimalitätsgl.)} \end{cases}$$

Der Algorithmus DynRucksack setzt diese durch dynamische Programmierung um:

```
1: α = 0
2: repeat
3:     α = α + 1
4:     for i ∈ {1, ..., n} do
5:         F_j(α) = min{F_{j-1}(α − p_j) + vol(j), F_{j-1}(α)}
6:     end for
7: until B < F_n(α)
8: return α − 1
```

Satz. *DynRucksack berechnet zur Eingabe I den Wert $\text{OPT}(I)$ in Zeit $\mathcal{O}(n \cdot \text{OPT}(I)) = \mathcal{O}(n^2 \cdot P_{\max})$.*

9.2 AR_k zur Approximation mit konstantem relativen Fehler

Der Algorithmus AR_k rechnet mit um den Faktor k reduzierten, gerundeten Preisen:

```
1: p_red(w) = ⌊p(w)/k⌋
2: I_red = ⟨W, vol, p_red, B⟩
3: R_k = DynRucksack(I_red)
4: return R_k
```

Satz. *AR_k macht bei Eingabe I einen relativen Fehler von $\varepsilon_{AR_k} \leq \frac{k \cdot n}{P_{\max}}$ und hat eine Laufzeit von $\mathcal{O}(n^2 \cdot \frac{P_{\max}}{k})$.*

Beweis. Sei R^* die Indexmenge einer optimalen Rucksackfüllung für I und R_k die berechnete Indexmenge der Lösung des um k reduzierten Problems I_{red}.

Da R_k eine optimale Lösung für I_{red} ist, gilt $\text{OPT}(I_{\text{red}}) \geq \sum_{j \in R^*} \lfloor \frac{p_j}{k} \rfloor$. Weiterhin ist R_k eine zulässige Lösung für I. Es gilt dann:

$$\begin{aligned}
\text{AR}_k(I) = p(R_k) &\geq k \cdot \sum_{j \in R_k} \left\lfloor \frac{p_j}{k} \right\rfloor = k \cdot \text{OPT}(I_{\text{red}}) \\
&\geq k \cdot \sum_{j \in R^*} \left\lfloor \frac{p_j}{k} \right\rfloor \geq k \cdot \sum_{j \in R^*} \left(\frac{p_j}{k} - 1 \right) = \sum_{j \in R^*} (p_j - k) = p(R^*) - k \cdot |R^*| \\
&= \text{OPT}(I) - k \cdot |R^*| \geq \text{OPT}(I) - k \cdot n
\end{aligned}$$

Damit folgt für den relativen Fehler die zu zeigende Aussage

$$\varepsilon_{\text{AR}_k} = \frac{|\text{AR}_k(I) - \text{OPT}(I)|}{\text{OPT}(I)} \leq \frac{k \cdot n}{\text{OPT}(I)} \leq \frac{k \cdot n}{P_{\max}}$$

□

9.3 FPASRucksack zur Umwandlung in ein streng polynomielles Approximationsschema

Um ein FPAS zu erreichen, muss gezeigt werden, dass jedes $\varepsilon \in]0, 1[$ als relativer Fehler erreichbar ist. Der Algorithmus FPASRucksack verwendet dazu AR_k und konstruiert ein passendes k:

1: bestimme n und $P_{\max}$ aus der Eingabe I
2: $k = \varepsilon \cdot \frac{P_{\max}}{n}$
3: **return** $\text{AR}_k(I)$

Satz. *FPASRucksack ist ein FPAS für Rucksack mit einer Laufzeit von $\mathcal{O}\left(n \cdot \log \frac{1}{\varepsilon} + \frac{1}{\varepsilon^4}\right)$.*

10 Randomisierte Approximationsalgorithmen

10.1 Die probabilistische Methode mit Algorithmus A

Satz. *Sei Φ eine boolesche (n, m)-Formel in KNF, dann gilt*

$$\max\{\text{wahr}\left(\textsc{False}, \ldots, \textsc{False}, \Phi\right), \text{wahr}\left(\textsc{True}, \ldots, \textsc{True}, \Phi\right)\} \geq \frac{1}{2} \cdot m$$

Beweis. TODO □

Der folgende Algorithmus A nutzt die sogenannte probabilistische Methode aus, in dem eine Belegung für jede Variable stochastisch unabhängig durchgeführt wird:

1: **for** $i \in \{1, \ldots, n\}$ **do**
2: $\quad x_i = \begin{cases} \textsc{True} & \text{mit Wahrscheinlichkeit } \frac{1}{2} \\ \textsc{False} & \text{mit Wahrscheinlichkeit } \frac{1}{2} \end{cases}$
3: **end for**
4: **return** $b_A = (x_1, \ldots, x_n)$

Satz. *Sei k_j die Anzahl der Literale in C_j. Es gilt*

$$P\left[C_j \text{ wird durch den Algorithmus } A \text{ erfüllt}\right] = 1 - \frac{1}{2^{k_j}}$$

Beweis. Beweis über Gegenwahrscheinlichkeit, dass alle Literale nicht erfüllt sind (stochastische Unabhängigkeit). □

Satz. *Für jede boolesche (n, m)-Formel Φ in KNF, in der jede Klausel mindestens k Literale hat, gilt:*

$$E\left[A(\Phi)\right] \geq \left(1 - \frac{1}{2^k}\right) \cdot m$$

Des Weiteren existiert eine Belegung b mit $\text{wahr}(b, \Phi) \geq \left(1 - \frac{1}{2^k}\right) \cdot m$.

Beweis. Für jede Klausel C_j wird eine Indikator-Variable Z_j mit

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

eingeführt. Da ihr Erwartungswert gleich der Wahrscheinlichkeit, dass sie 1 annimmt, ist, folgt

$$E\left[A(\Phi)\right] = E\left[\text{wahr}(b_A, \Phi)\right] = E\left[\sum_{j=1}^{m} Z_j\right] = \sum_{j=1}^{m} \left(1 - \frac{1}{2^{k_j}}\right) \geq \left(1 - \frac{1}{2^k}\right) \cdot m$$

Die Existenz einer solchen Belegung folgt aus einem klassischen Durchschnittsargument. □

Satz. *Sei $2^k > m$, dann ist jede boolesche (n, m)-Formel Φ in KNF, in der jede Klausel mindestens k Literale hat, erfüllbar.*

Beweis. Eingesetzt in die vorherige Aussage, erhält man dann $\left(1 - \frac{1}{2^k}\right) \cdot m = m - \overbrace{\frac{m}{2^k}}^{<1}$. Aus einem Durchschnittsargument folgt auch hier, dass es eine Belegung b mit $m - 1 < \text{wahr}(b, \Phi) \leq m$ geben muss, wegen der Ganzzahligkeit von wahr werden also alle m Klauseln erfüllt. □

Satz. *Algorithmus A hat für jede (n, m)-Formel Φ in KNF, in der jede Klausel mindestens k Literale hat, eine erwartete relative Güte von*

$$E\left[\rho_A(\Phi)\right] = \frac{\text{OPT}(\Phi)}{E\left[A(\Phi)\right]} \leq \frac{m}{\left(1 - \frac{1}{2^k}\right) \cdot m} = \frac{1}{1 - \frac{1}{2^k}}$$

. Die Laufzeit des Algorithmus ist $\mathcal{O}(n)$. Wenn die kürzeste Klausel mindestens 2 Literale hat, hat A die erwartete relative Güte $\frac{4}{3}$.

Satz. *Algorithmus A garantiert für jede boolesche (n, m)-Formel Φ in KNF sogar eine erwartete relative Güte von $\frac{3}{2}$.*

10.2 Arithmetisierung und Randomized Rounding mit Algorithmus B

Max-SATkann auch als Lineares Programm (LP) ausgedrückt werden. Es bezeichnen im Folgenden $S_j^{\oplus}$ die Menge der Variablen, die in C_j nicht negiert vorkommen, und $S_j^{\ominus}$ die Menge der Variablen, die in C_j negiert vorkommen. Für jede boolesche Variable x_i wird eine 0-1-Variable $\hat{x}_i$ eingeführt; für jede Klausel C_j eine 0-1-Variable $\hat{Z}_j$. Es ergibt sich also das folgende ganzahlige lineare Programm (ILP) B für Max-SAT:

$$\begin{aligned}
\text{maximiere} \quad & \sum_{j=1}^{m} \hat{Z}_j & \\
\text{gemäß} \quad & \sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i) \geq \hat{Z}_j & \forall j \\
& \hat{x}_i \in \{0, 1\} & \forall i \\
& \hat{Z}_j \in \{0, 1\} & \forall j
\end{aligned}$$

Relaxiert man die Bedingungen, dass $\hat{x}_i, \hat{Z}_j \in \{0,1\}$ gelten muss, und stellt fordert stattdessen $\forall i,j : 0 \leq \hat{x}_i, \hat{Z}_j \leq 1$, kann das resultierende LP B_{rel} in Polynomzeit gelöst werden. Das LP selbst hat ebenfalls eine polynomiell beschränkte Anzahl an Bedingungen, sodass die Gesammtlaufzeit auch polynomiell beschränkt ist.

Durch die Relaxierung entsteht die folgende Beziehung (exemplarisch für ein Maximierungsproblem), die als Superoptimalität bezeichnet wird:

$$\text{OPT}(B_{\text{rel}}) \geq \text{OPT}(B) = \text{OPT}(\Phi)$$

Im Allgemeinen ist die rationale Lösung σ_{rel} von B_{rel} also keine zulässige Lösung für B. Hierzu müssen die $\hat{x}_i$ auf 0 oder 1 (bzw. die x auf FALSE oder TRUE) gerundet müssen. Eine Möglichkeit hierzu stellt der durch die stochastische Funktion $\phi : [0,1] \mapsto [0,1]$ parametrisierte Algorithmus RANDOMIZEDROUNDING$[\pi]$ dar:

1: **for** $i \in \{1, \ldots, n\}$ **do**
2: $\quad x_i = \begin{cases} \text{TRUE} & \text{mit Wahrscheinlichkeit } \pi(\hat{x}_i) \\ \text{FALSE} & \text{mit Wahrscheinlichkeit } 1 - \pi(\hat{x}_i) \end{cases}$
3: **end for**
4: **return** $b_B = (x_1, \ldots, x_n)$

Damit kann dann der Algorithmus B gebildet werden, der MAX-SAT approximiert:

1: konstruiere das LP B_{rel} zur Eingabe Φ
2: ermittle die rationale Lösung b_{rel} von B_{rel}
3: **return** RANDOMIZEDROUNDING$[\pi(x) = x](b_{\text{rel}})$

Satz. *Sei k_j die Anzahl der Literale in C_j, dann gilt:*

$$P\left[C_j \textit{ wird durch den Algorithmus } B \textit{ erfüllt}\right] \geq \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j$$

Beweis. Erneut wird über die Gegenwahrscheinlichkeit argumentiert:

$$P\left[C_j \text{ wird durch den Algorithmus } B \text{ nicht erfüllt}\right] = \left[\Pi_{x_i \in S_j^{\oplus}} (1 - \hat{x}_i)\right] \cdot \left[\Pi_{x_i \in S_j^{\ominus}} \hat{x}_i\right]$$

Es folgt

$$\begin{aligned}
P\left[C_j \text{ wird durch den Algorithmus } B \text{ erfüllt}\right] &= 1 - \left[\Pi_{x_i \in S_j^{\oplus}} (1 - \hat{x}_i)\right] \cdot \left[\Pi_{x_i \in S_j^{\ominus}} \hat{x}_i\right] \\
&\geq 1 - \left(\frac{\sum_{x_i \in S_j^{\oplus}} (1 - \hat{x}_i) + \sum_{x_i \in S_j^{\ominus}} \hat{x}_i}{k_j}\right)^{k_j} \\
&= 1 - \left(\frac{|S_j^{\oplus}| - \sum_{x_i \in S_j^{\oplus}} \hat{x}_i + |S_j^{\ominus}| - \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i)}{k_j}\right)^{k_j} \\
&= 1 - \left(\frac{k_j - \left(\sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i)\right)}{k_j}\right)^{k_j} \\
&\overset{\text{NB von } B}{\geq} 1 - \left(1 - \frac{\hat{Z}_j}{k_j}\right)^{k_j} \overset{\text{Konkavität}}{\geq} \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j
\end{aligned}$$

□

Satz. *Für jede boolesche Formel Φ in KNF, in der jede Klausel höchstens k Literale hat, gilt*

$$E\left[B(\Phi)\right] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \mathrm{OPT}(\Phi)$$

Satz. *Mit der bekannten Abschätzung $1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}$ für alle $k \in \mathbb{N}$ beträgt die erwartete Anzahl an erfüllten Klauseln von Algorithmus B bei einer Eingabe Φ in KNF mindestens $\left(1 - \frac{1}{e}\right) \cdot \mathrm{OPT}(\Phi) \approx 0.632 \cdot \mathrm{OPT}(\Phi)$.*

Algorithmus B hat also eine erwartete relative Güte von

$$E\left[\rho_B(\Phi)\right] \leq \frac{1}{1 - \frac{1}{e}} \approx 1.582$$

Satz. *Mit $\pi(x) = \frac{1}{2} \cdot x + \frac{1}{4}$ oder $1 - \frac{1}{4^x} \leq \pi(x) \leq 4^{x-1}$ kann Algorithmus B sogar eine erwartete relative Güte von $\frac{4}{3}$ erreichen.*

10.3 Hybrider Ansatz durch Kombination mehrerer Verfahren

Aus den bestimmten Gütegarantien lässt sich ablesen, dass Algorithmus A besonders gut für Formeln geeignet ist, die keine Klauseln aus nur einem Literal enthalten. Algorithmus B eignet sich allgemein gut, wenn die auftretenden Klauseln kurz sind.

Beide Verfahren können nun automatisch kombiniert werden, um eine insgesamt bessere Gütegarantie zu erreichen. Dier erste Möglichkeit hierzu ist der Algorithmus C_{p_A}:

1: **return** $\begin{cases} A(\Phi) & \text{mit Wahrscheinlichkeit } p_A \\ B(\Phi) & \text{mit Wahrscheinlichkeit } 1 - p_A \end{cases}$

Alternativ startet der Algorithmus C_{alle} immer beide Algorithmen:

1: $b_A = A(\Phi)$
2: $b_B = B(\Phi)$
3: **return** $\max\{b_A, b_B\}$

Es gilt offensichtlich

$$E\left[C_{\text{alle}}(\Phi)\right] \geq E\left[C_{p_A}(\Phi)\right] = p_A \cdot E\left[A(\Phi)\right] + (1 - p_A) \cdot E\left[B(\Phi)\right]$$

für alle $p_A \in [0, 1]$.

Satz. *Der Algorithmus $C_{\frac{1}{2}}$ hat eine erwartete relative Güte von $\frac{4}{3}$*

Beweis. Es sind folgende Erwartungswerte bekannt:

$$E\left[A(\Phi)\right] = \sum_{k=1}^{n} \sum_{C_j \text{ hat } k \text{ Literale}} \left(1 - \frac{1}{2^k}\right) \overset{0 \leq \hat{Z}_j \leq 1}{\geq} \sum_{k=1}^{n} \sum_{C_j \text{ hat } k \text{ Literale}} \left(1 - \frac{1}{2^k}\right) \cdot \hat{Z}_j$$

$$E\left[B(\Phi)\right] \geq \sum_{k=1}^{n} \sum_{C_j \text{ hat } k \text{ Literale}} \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \hat{Z}_j$$

Es folgt also für $C_{\frac{1}{2}}$:

$$E\left[C_{\frac{1}{2}}(\Phi)\right] \geq \frac{1}{2} \cdot \sum_{k=1}^{n} \sum_{C_j \text{ hat } k \text{ Literale}} \underbrace{\left(1 - \frac{1}{2^k} + 1 - \left(1 - \frac{1}{k}\right)^k\right)}_{\geq \frac{3}{2}} \cdot \hat{Z}_j$$

$$\geq \frac{1}{2} \cdot \frac{3}{2} \cdot \sum_{k=1}^{n} \sum_{C_j \text{ hat } k \text{ Literale}} \hat{Z}_j$$

$$= \frac{3}{4} \sum_{j=1}^{m} \hat{Z}_j \geq \frac{3}{4} \cdot \mathrm{OPT}(\Phi)$$

□

10.4 Derandomisierung durch die Methode der bedingten Erwartungswerte

Die Ansätze mancher randomisierten Algorithmen können so modifiziert werden, dass sie deterministisch ein Ergebnis zurückliefern, was mindestens so gut wie der Erwartungswert des nichtdeterministischen Algorithmus ist. Man macht sich hierbei unter anderem die Eigenschaft des randomisierten Algorithmus zu Nutze, dass der Erwartungswert auch ohne wirkliche Ausführung des Algorithmus berechenbar ist.

Der Algorithmus DERAND_A benutzt den Erwartungswert für Algorithmus A um nach und nach alle Variablen x_i zu belegen. Für jede Variable wird einmal TRUE und FALSE eingesetzt, dann mit der resultierenden Formel mit höherem Erwartungswert fortgefahren:

1: **for** $i \in \{1, \ldots, n(I)\}$ **do**
2: $\quad W_{\text{FALSE}} = E\left[A(I) \mid x_1, \ldots, x_{i-1}, x_i = \text{FALSE}\right]$
3: $\quad W_{\text{TRUE}} = E\left[A(I) \mid x_1, \ldots, x_{i-1}, x_i = \text{TRUE}\right]$
4: $\quad$ **if** $W_{\text{FALSE}} < W_{\text{TRUE}}$ **then**
5: $\quad\quad x_i = \text{TRUE}$
6: $\quad$ **else**
7: $\quad\quad x_i = \text{FALSE}$
8: $\quad$ **end if**
9: **end for**
10: **return** $b = (x_1, \ldots, x_{n(I)})$

Satz. *Der Algorithmus* DERAND_A *approximiert* MAX-SAT*mit relativer Worst-Case-Güte* $\rho^{\text{wc}}_{\text{DERAND\_}A} = \frac{1}{1-\frac{1}{2^k}}$ *in Polynomzeit.*

11 Approximation durch Lineare Optimierung

Die zum Lösen eines LP mit k Variablen benötigte Zeit wird im Folgenden als $L\left(|\text{LP}|, k\right)$ bezeichnet. Sie liegt in der Größenordnung $\mathcal{O}(k^4 \cdot |\langle \text{LP} \rangle|^2)$.

11.1 Ganzzahligkeitslücke

Der für MAX-SAT gewählte Ansatz lässt sich im Prinzip auch auf verschiedenste andere kombinatorische Optimierungsprobleme Π anwenden. Verallgemeinert bezeichnet man ihn als Rundungsansatz (exemplarisch für Maximierungsprobleme):

1. Beschreibung der Instanz I von Π durch ein ganzzahliges Lineares Programm X, was als Arithmetisierung bezeichnet wird. Es gilt damit $\text{OPT}(I) = \text{OPT}(X)$.
2. Fallenlassen der Ganzzahligkeitsbedingung (Relaxierung), sodass das entstandene Lineare Programm X_{rel} in Polynomzeit lösbar ist. Durch die Superoptimalität gilt $\text{OPT}(X_{\text{rel}}) \geq \text{OPT}(X)$.
3. Der Approximationsalgorithmus A löst X_{rel} und rundet die rationalen Lösungen geschickt zu einer zulässigen Lösung $\sigma \in \mathcal{S}(I)$.
4. Beweis, dass $A(I) \geq \frac{1}{\rho} \cdot \text{OPT}(X_{\text{rel}})$ gilt.
5. Aus der Superoptimalität folgt $A(I) \geq \frac{1}{p} \cdot \text{OPT}(I)$.

Für ein kombinatorisches Maximierungsproblem Π mit Eingaben $I \in \mathcal{D}$ sei X jeweils das äquivalente ILP und X_{rel} jeweils das relaxierte LP. Dann bezeichnet

$$\gamma = \max\left\{ \frac{\text{OPT}(X_{\text{rel}})}{\text{OPT}(X)} \mid I \in \mathcal{D} \right\}$$

die Ganzzahligkeitslücke („integrality gap“) der Relaxierung.

Für das LP B_{rel} für MAX-SAT bestimmt sich die Ganzzahligkeitslücke aus der booleschen $(2,4)$-Formel $\Phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$: $\text{OPT}(\phi) = 3$, aber $\text{OPT}(B_{\text{rel}}) = 4$. Damit folgt $\gamma \leq \frac{4}{3}$.

Allgemein folgen mit $\gamma \geq \frac{\text{OPT}(X_{\text{rel}})}{\text{OPT}(X)}$ die beiden Aussagen

$$A(I) \geq \frac{\gamma}{\rho} \cdot \text{OPT}(I)$$

$$\rho\gamma$$

Daraus kann geschlossen werden, dass der Rundungsansatz die Ganzzahligkeitslücke nicht überwinden kann und Modellierung in LP mit geringer Ganzzahligkeitslücke wichtig ist.

11.2 Arithmetisierung von SetCover

Sei $X = \{S_{i_1}, \ldots, S_{i_l}\}$ eine Sammlung von Gruppen, dann bezeichne die Notation $V(X) = S_{i_1} \cup \cdots \cup S_{i_l}$ die Menge der von X überdeckten Objekte. Weiter bezeichne $G_S = \max\{|S_i| \mid 1 \leq i \leq m\}$ die Mächtigkeit der größten Gruppe, $\deg_S(u) = |\{S_i \mid u \in S_i \in S\}|$ den Grad des Objekts $u \in V$ und $\Delta_S = \max\{\deg_S(u) \mid u \in V\}$ den Grad von S.

Bei der Arithmetisierung wird pro Gruppe S_i eine 0-1-Variable x_i eingeführt, die genau dann 1 ist, wenn $S_i \in S_{\text{cov}}$. Damit lautet das ILP X für SETCOVER:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} x_i & \\ \text{gemäß} \quad & \sum_{i:u \in S_i} x_1 \geq 1 & \forall u \in V \\ & x_i \in \{0,1\} & \forall i \in \{1, \ldots, m\} \end{aligned}$$

Satz. *Für die kanonische Relaxierung X_{rel} von X gilt für die Ganzzahligkeitslücke*

$$\gamma \geq \frac{1}{2} \cdot \log n$$

11.3 Deterministisches Runden mit DetRoundSC

```
1: (x_1, ..., x_m) = löse X_rel
2: S_cov = ∅
3: for i ∈ {1, ..., m} do
4:     if x_i ≥ 1/Δ_S then
5:         S_cov = S_cov ∪ {S_i}
6:     end if
7: end for
8: return S_cov
```

Satz. *DETROUNDSC berechnet für eine Instanz S von SETCOVER in Zeit $\mathcal{O}(m + L(nm, m))$ eine Überdeckung. Ferner gilt $\text{DETROUNDSC}(S) \leq \Delta_S \cdot \text{OPT}(S)$.*

Beweis. Zu einem beliebigen Objekt $u \in V$ gehört die Nebenbedingung $\sum_{i:u \in S_i} x_i \geq 1$ mit $\deg_S(u)$ Summanden. Mit einem Durchschnittsargument kann jetzt also gefolgert werden, dass für einen der Summanden x_i gilt:

$$x_i \geq \frac{1}{\deg_S(u)} \geq \frac{1}{\Delta_S}$$

Damit wird für jedes u mindestens ein überdeckendes S_i aufgenommen, also stellt die Ausgabe eine Überdeckung dar.

Weil für jede Gruppe $S_i \in S_{\text{cov}}$ $\Delta_S \cdot x_i \geq 1$ gilt, folgt für die Qualität der Ausgabe

$$\textsc{DetRoundSC}(S) = |S_{\text{cov}}| \leq \sum_{i=1}^{m} \Delta_S \cdot x_i = \Delta_S \cdot \sum_{i=1}^{m} x_i = \Delta_S \cdot \text{OPT}(X_{\text{rel}}) \leq \Delta_S \cdot \text{OPT}(S)$$

□

11.4 Unzuverlässiges Randomized Rounding mit RandRoundSC$[r]$

1: $(x_1, \ldots, x_m) =$ löse X_{rel}
2: $\chi = \emptyset$
3: **for** $i \in \{1, \ldots, m\}$ **do**
4: mit Wahrscheinlichkeit $1 - e^{-r \cdot x_i}$: $\chi = \chi \cup \{S_i\}$
5: **end for**
6: **return** χ

RandRoundSC[r] ist ein sogenannter Monte-Carlo-Algorithmus, weil er auch nicht zulässige Lösungen zurückliefern kann.

Satz. *Für eine Eingabe S von SetCover sei χ die Ausgabe von RandRoundSC$[r]$. Dann gelten*

1. $P\left[\chi \text{ ist eine Überdeckung}\right] \geq 1 - n \cdot e^{-r}$
2. $E\left[|\chi|\right] \leq r \cdot \text{OPT}(S)$

Beweis. 1. Sei $u \in V$, dann gilt:

$$\begin{aligned} P\left[u \in V(\chi)\right] &= P\left[\forall i \,\text{mit}\, u \in S_i : S_i \notin \chi\right] \\ &= \Pi_{i:u \in S_i} e^{-r \cdot x_i} = \left(\Pi_{i:u \in S_i} e^{-x_i}\right)^r = \left(e^{-\sum_{i:u \in S_i} x_i}\right)^r \\ &\leq e^{-r} \end{aligned}$$

Die Wahrscheinlichkeit, dass es ein nicht überdecktes Objekt gibt, ist folglich

$$P\left[\exists U \in V : u \notin V(\chi)\right] \leq n \cdot e^{-r}$$

2. Für $|\chi|$ folgt

$$E\left[|\chi|\right] = \sum_{i=1}^{m} P\left[S_i \in \chi\right] = \sum_{i=1}^{m} \left(1 - e^{-r \cdot x_i}\right) \leq r \cdot \sum_{i=1}^{m} x_i = r \cdot \text{OPT}(X_{\text{rel}}) \leq r \cdot \text{OPT}(S)$$

□

11.5 Las-Vegas-Algorithmus LasVegasSC$[r]$ für zuverlässig zulässige Lösungen

1: löse das LP X_{rel}
2: $\tau = 0$
3: **repeat**
4: $\chi = \textsc{RandRoundSC}[r](S)$
5: $\tau = \tau + 1$
6: **until** $V(\chi) = V$
7: **return** $S_{\text{cov}} = \chi$

Das LP X_{rel} wird dabei nur einmal und nicht erneut in RandRoundSC$[r]$ gelöst. LasVegasSC$[r]$ ist ein sogenannter Las-Vegas-Algorithmus, weil er immer eine zulässige Lösung zurückliefert, seine Laufzeit allerdings zufällig ist.

Satz. *Sei S eine Eingabe von SetCover und gelte $r > \ln n$. Für LasVegasSC[r] gelten dann*

1. *S_{cov} ist eine Überdeckung mit erwarteter Größe von höchstens $r \cdot \text{OPT}(S)$.*
2. *Die erwartete Anzahl der Iterationen der Repeat-Schleife ist höchstens $\frac{e^{2r}}{(n-e^r)^2}$.*

Beweis. 1. Das Ergebnis folgt direkt aus der Analyse von RandRoundSC[r].

2.

$$\begin{aligned} E[\tau] &= \sum_{t=1}^{\infty} t \cdot P\left[\text{erst die } t. \text{ Wiederholdung ist eine Überdeckung}\right] \\ &\leq \sum_{t=1}^{\infty} t \cdot P\left[\ t-1 \text{ Wiederholdungen sind keine Überdeckung}\right] \\ &\leq \sum_{t=1}^{\infty} t \cdot \left(n \cdot e^{-r}\right)^{t-1} \overset{\text{Konvergenz durch } r>\ln n}{=} \frac{e^{2r}}{\left(n-e^r\right)^2} \end{aligned}$$

□

Satz. *LasVegasSC[$\ln n + 1$] garantiert eine relative erwartete Güte von $\ln n + 1$. Der Erwartungswert für die Anzahl der Iterationen der Schleife ist $\left(\frac{e}{e-1}\right)^2 < 2.503$.*

12 Ausnutzen der Dualität von Linearen Programmen

12.1 Herleitung der Dualität

Ein LP für ein o.B.d.A. Minimierungsproblem kann wie folgt geschrieben werden:

$$\begin{aligned} \min \quad & z(\vec{x}) = \vec{c}^{\mathrm{T}} \cdot \vec{x} \\ \text{gemäß} \quad & A \cdot \vec{x} \geq \vec{b} \\ & \vec{x} \geq \vec{0} \end{aligned}$$

Es bezeichnen $\text{row}_i[A]$ die i. Zeile der Matrix A und $\text{row}_i[A] \cdot \vec{x} \geq b_i$ die i. Nebenbedingung des LP.

Mit $\vec{y} \geq \vec{0}$ können folgende neue Nebenbedingungen erzeugt werden, die ebenfalls von jeder zulässigen Lösung erfüllt werden:

$$\vec{y}^{\mathrm{T}} \cdot A \cdot \vec{x} \geq \vec{y}^{\mathrm{T}} \cdot \vec{b}$$

Wählt man die y_j so, dass die aus ihnen berechneten Koeffizienten der x_i kleiner als die jeweiligen c_i, gilt für diese $\vec{y}$: $z(\vec{x}) \geq \vec{y}^{\mathrm{T}} \cdot A \cdot \vec{x} \geq \vec{y}^{\mathrm{T}} \cdot \vec{b}$ für alle zulässigen Lösungen $\vec{x}$. Damit bildet $\vec{y}^{\mathrm{T}} \cdot \vec{b}$ eine untere Schranke für den Wert einer optimalen Lösung. Diese untere Schranke kann folglich mit dem sogenannten dualen LP maximiert werden:

$$\begin{aligned} \max \quad & \zeta(\vec{y}) = \vec{b}^{\mathrm{T}} \cdot \vec{y} \\ \text{gemäß} \quad & A^{\mathrm{T}} \cdot \vec{y} \leq \vec{c} \\ & \vec{y} \geq \vec{0} \end{aligned}$$

Satz. *Jede zulässige Lösung $\vec{y}$ liefert eine untere Schranke $\zeta(\vec{y})$ für die Zielfunktion $z(x)$ für alle zulässigen Lösungen des Primals. Damit gilt die als schwache Dualität bezeichnete Beziehung*

$$\zeta(\vec{y}) \leq z(\vec{x})$$

Satz. *Für optimale Lösungen $\vec{x}_{\text{opt}}$ des Primals beziehungsweise $\vec{y}_{\text{opt}}$ des Duals gilt $z(\vec{x}_{\text{opt}}) = \zeta(\vec{y}_{\text{opt}})$.*

Man bezeichnet die Differenz zwischen dem Wert einer Nebenbedingung und ihrer Grenze als Schlupf. Der primale Schlupf der j. Nebenbedingung des Duals ist also $p_j = \text{row}_j[A] \cdot \vec{x} - b_j$; der duale Schlupf der i. Nebenbedingung des Duals $s_i = c_i - \text{row}_i\left[A^{\mathrm{T}}\right] \cdot \vec{y}$ Beträgt der Schlupf 0, so ist die Nebenbedingung scharf.

Satz (Satz vom komplementären Schlupf)**.** *Seien $\vec{x}$ und $\vec{y}$ zulässige Lösungen des Primals beziehungsweise Duals. Sie sind genau dann optimale Lösungen, wenn für alle i und j $y_j \cdot (\text{row}_j[A] \cdot \vec{y} - b_j) = 0$ und $\left(c_i - \text{row}_i\left[A^{\mathrm{T}}\right] \cdot \vec{y}\right) \cdot x_i = 0$ gilt. Die Bedingung können auch als*

$$y_j > 0 \Rightarrow \text{row}_j[A] \cdot \vec{x} = b_j \tag{1}$$

$$x_i > 0 \Rightarrow \text{row}_i\left[A^{\mathrm{T}}\right] \cdot \vec{y} = c_i \tag{2}$$

formuliert werden. Für Approximationsalgorithmen ist vor allem die zweite Aussage wichtig.

Satz. *Sei Π ein Minimierungsproblem, I eine Instanz von Π, X das zu I gehörige ILP, X_{rel} dessen Relaxierung und Y_{rel} das Dual zu X_{rel} sowie $\vec{x}$ und $\vec{y}$ beliebige zulässige Lösungen von X bzw. Y_{rel}, dann gilt:*

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

12.2 Lineare Programme für SetCover

12.2.1 Primal X

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} x_i & \\ \text{gemäß} \quad & \sum_{i:u_j \in S_i} x_i \geq 1 & \forall u_j \in V \\ & x_i \in \{0,1\} & \forall i \in \{1,\ldots,m\} \end{aligned}$$

12.2.2 Relaxiertes Dual Y_{rel}

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} y_j & \\ \text{gemäß} \quad & \sum_{j:u_j \in S_i} y_j \leq 1 & \forall S_i \in S \\ & 0 \leq y_j \leq 1 & \forall j \in \{1,\ldots,n\} \end{aligned}$$

12.3 Entwurf neuer Algorithmen für SetCover mittels Dual Fitting

Beim Dual-Fitting wird allgemein versucht, Gleichung (2) zu erfüllen. Zunächst wird dazu eine zulässige Lösung $\vec{y}$ des Duals bestimmt, bei der einige Nebenbedingungen scharf sind. Die den scharfen Nebenbedingungen entsprechenden Variablen x_i des Primals werden dann als Approximation auf 1 gesetzt, die anderen auf 0. Das so entstandene $\vec{x}$ gibt mit $z(\vec{x})$ eine obere Schranke an. Die Qualität der Lösung hängt dabei stark vom ursprünglich bestimmten $\vec{y}$ ab.

12.3.1 DualPurSC

DualPurSC löst das relaxierte Dual optimal und baut damit eine zulässige 0-1-Lösung für X:

1: bestimme optimale Lösung $\vec{y}$ des relaxierten Duals Y_{rel}
2: **for** $i \in \{1,\ldots,m\}$ **do**

3: **if** i. Nebenbedingung von Y_{rel} ist scharf **then**
4: $x_i = 1$ ▷ mit erfüllter Nebenbedingung folgt $\sum_{j:u_j \in S_i} y_j = x_i$
5: **else**
6: $x_i = 0$
7: **end if**
8: **end for**
9: **return** $(x_1, \ldots, x_m)$

Satz. *DualPurSC liefert eine Überdeckung der relativen Güte Δ_S in Zeit $\mathcal{O}(m + L(mn, n))$.*

Beweis. Die Ausgabe ist eine Überdeckung, weil bei einer Nichtüberdeckung eines Objekts u_i alle x_i für $S_i \in S$ mit $u_j \in S_i$ gleich 0 wären. Dann wäre auch der duale Schlupf der zugehörigen i. Nebenbedingung nicht 0 und y_j könnte vergrößert werden. Damit ist die ursprüngliche Lösung $\vec{y}$ aber nicht optimal gewesen.

Da für jedes i mit $x_i = 1$ gilt $\sum_{j:u_j \in S_i} y_j = 1$, folgt für die Qualität der Lösung:

$$\begin{aligned}\text{DualPurSC}(S) = \sum_{i=1}^{m} x_i = \sum_{i:x_i=1} 1 = \sum_{i:x_i=1} \sum_{j:u_j \in S_i} y_j \leq \Delta_S \cdot \sum_{j=1}^{n} y_j \\ = \Delta_S \cdot \text{OPT}(Y_{\text{rel}}) \leq \Delta_S \cdot \text{OPT}(S)\end{aligned}$$

□

12.3.2 PrimalDualSC_1

PrimalDualSC_1 berechnet jetzt keine optimale Lösung von Y_{rel} mehr:

1: **for all** y_j **do**
2: $y_j = 1$
3: **end for**
4: **for** $i \in \{1, \ldots, m\}$ **do**
5: $x_i = 0$
6: dualer_schlupf$[i] = 1$
7: **end for**
8: **while** es ein nicht überdecktes Objekt u_j gibt **do**
9: bestimme eine Gruppe S_i mit $u_j \in S_i$ mit minimalem dualer_schlupf$[i]$
10: $x_i = 1$
11: $y_j =$ dualer_schlupf$[i]$ ▷ i. Nebenbedingung scharf: $x_i = \sum_{j:u_j \in S_i} y_j$
12: **for all** i' mit $u_j \in S_{i'}$ **do**
13: dualer_schlupf$[i'] =$ dualer_schlupf$[i'] - y_j$ ▷ Aktualisieren des dualen Schlupfs
14: **end for**
15: **end while**
16: **return** $(x_1, \ldots, x_m)$

Satz. *PrimalDualSC_1 gibt eine Überdeckung der relativen Güte Δ_S nach Zeit $\mathcal{O}(n \cdot m)$ aus.*

Beweis. Die While-Schleife kann bis zu n mal durchlaufen werden, wobei für die Bestimmung der Gruppe mit minimalem dualen Schlupf jeweils $\mathcal{O}(m)$ Schritte benötigt werden.

$\vec{y}$ ist zu Beginn eine zulässige Lösung und jeder Eintrag wird nur maximal einmal verändert, also ist auch $\vec{y}$ auch am Ende eine zulässige Lösung. Für die Qualität kann die Rechnung für DualPurSC erneut verwendet werden. □

12.4 Analyse bestehender Algorithmen für SetCover

Ein bestehender Algorithmus, der eine 0-1-Lösung des ursprünglichen ILPs X berechnet, wird so erweitert, dass er gleichzeitig eine zulässige Lösung $\vec{y}$ für Y_{rel} konstruiert. Da $\vec{x}$ und $\vec{y}$ von

einander abhängen, hängt auch $z(\vec{x})$ von $\vec{y}$ ab. Kann daraus der Wert $\zeta(\vec{y})$ isoliert werden, kann eine obere Schranke des Wertes der Lösung zu I berechnet werden.

12.4.1 PrimalDualSC_2

Ein bestehender gieriger Algorithmus, der immer die Gruppe wählt, die am meisten noch nicht überdeckte Objekte überdeckt, wird so erweitert, dass man die Dualität der LP zur Analyse verwenden kann. Es sei $\mathcal{H}(n) = \sum_{i=1}^{n} \frac{1}{i}$ die n. harmonische Zahl. Für sie gilt die Ungleichung $\mathcal{H}(n) \leq \ln n + 1$.

1: **for all** $i \in \{1, \ldots, m\}$ **do**
2: $\quad x_i = 0$
3: **end for**
4: $C = \emptyset$ ▷ C enthält die schon überdeckten Objekte
5: **while** $C \neq V$ **do**
6: $\quad$ bestimme einen Index i mit maximalem $|S_i \setminus C|$
7: $\quad x_i = 1$
8: $\quad$ **for all** $u_j \in S_i \setminus C$ **do**
9: $\quad\quad p[u_j] = \frac{1}{|S_i \setminus C|}$ ▷ $x_i = 1$ und $\sum_{u_j \in S_i \setminus C} p[u_j] = 1$
10: $\quad\quad y_j = \frac{1}{\mathcal{H}(G_S)} \cdot p[u_j]$ ▷ $\mathcal{H}(G_S) \cdot y_j = p[u_j]$
11: $\quad$ **end for**
12: $\quad C = C \cup S_i$
13: **end while**
14: **return** $(x_1, \ldots, x_m)$

Satz. *PrimalDualSC_2 liefert eine Überdeckung der relativen Güte $\mathcal{H}(G_S)$ in Zeit $\mathcal{O}(n \cdot m)$.*

Beweis. Erneut kann die While-Schleife maximal n Mal durchlaufen werden, und die Bestimmung des Index i kann $\mathcal{O}(m)$ Schritte dauern.

Zunächst ist zu zeigen, dass $\vec{y}$ eine zulässige Lösung von Y_{rel} bildet: Betrachte die i. Nebenbedingung und die zugehörige Gruppe $S_i = \{u_{j_1}, \ldots, u_{j_k}\}$, wobei die Objekte in o.B.d.A. genau dieser Reihenfolge durch den Algorithmus überdeckt worden sein sollen. Es gilt $k \leq G_S$. Nun wird die Runde betrachtet, in der u_{j_l} überdeckt wurde. Wäre S_i in dieser Runde gewählt worden, wären neben u_{j_1} noch $k-l$ weitere Objekte unüberdeckt. Die tatsächlich gewählte Gruppe trifft also noch mindestens $k - l + 1$ Objekte inklusive u_{j_l}. Damit gilt $p[u_{j_l}] \leq \frac{1}{k-l+1}$ und

$$y_{j_l} \leq \frac{1}{\mathcal{H}(G_S)} \cdot \frac{1}{k - l + 1}$$

Folglich ist die i. Nebenbedingung des Duals erfüllt mit

$$\sum_{l=1}^{k} y_{j_l} \leq \frac{1}{\mathcal{H}(G_S)} \cdot \sum_{l=1}^{k} \frac{1}{k - l + 1} = \frac{\mathcal{H}(k)}{\mathcal{H}(G_S)} \leq 1$$

Für die ausgegebene Überdeckung gilt

$$\textsc{PrimalDualSC\_2}(S) = \sum_{i=1}^{m} x_i = \sum_{j=1}^{n} p[u_j] = \mathcal{H}(G_S) \cdot \sum_{j=1}^{n} y_j \leq \mathcal{H}(G_S) \cdot \text{OPT}(S)$$

□

Mit $G_S \leq n$ folgt unmittelbar, die relative Gütegarantie von $\ln n + 1$ von PrimalDualSC_2.