Mitschrift zur Vorlesung Approximationsalgorithmen

Dozent: Prof. Dr. Rolf Wanka

Sommersemester 2016

Disclaimer

Diese Mitschrift ist nicht offiziell. Insbesondere erhebe ich keinen Anspruch auf Vollständigkeit oder Korrektheit.

Kapitel 1. Schnelle Algorithmen und hartnäckige Probleme

Wir betrachten Eingaben I und haben etwas mit dieser algorithmisch zu tun. Wenn die Eingabe durch den Algorithmus in Zeit $\mathcal{O}(\text{poly}(I))$ gelöst wird, ist der Algorithmus schnell.

Für NP−vollständige Probleme sind bis heute keine schnellen Algorithmen bekannt.

Bsp. 1.1:

- CLIQUE = $\{\langle G,k\rangle | \, G$ ist ungerichteter Graph, der einen vollständigen Teilgraphen aus mindestens k Knoten enthält$\}$
- IS = $\{\langle G,k\rangle | \, G=(V,E)$ ist ungerichteter Graph, in dem es eine Knotenmenge $U, U \subseteq V$, gibt, sodass kein $u \in U$ durch eine Kante aus Knotenmenge U mit einem anderen $v \in U$ verbunden ist$\}$

sind beide NP−vollständig.

Def. 1.2: Ein kombinatorisches Optimierungsproblem ist charakterisiert durch vier Komponenten:

- $\mathcal{D}$: Die Menge der Probleminstanzen, Eingabe ("domain")
- $S(I)$ für $I \in \mathcal{D}$: die Menge der zu I zulässigen Lösungen
- Die Bewertungsfunktion $f : S(I) \to \mathbb{N}^{\neq 0}$
- ziel $\in \{\min, \max\}$

Gesucht zu $I \in \mathcal{D}$ ist eine zulässige Lösung $\sigma_{\text{opt}} \in S(I)$, sodass $f(\sigma_{\text{opt}}) = \text{ziel}\{f(\sigma) | \, \sigma \in S(I)\}$. $f(\sigma)$ ist der Wert der zulässigen Lösung σ. Wir schreiben $\text{OPT}(I) = f(\sigma_{\text{opt}})$

Bsp. 1.3:

a) Das (volle) Traveling Salesperson Problem (TSP)
   - $\mathcal{D} = (\langle K_n, c\rangle | \, K_n$ vollständiger Graph auf n Knoten, $c = E \to \mathbb{N}$ Kantengewichtung$\}$
   - $S(\langle K_n, c\rangle) = \{c | \, c = (v_{i,1}, ..., v_{i,n})$ ist ein Hamilton-Kreis$\}$ (jede Permutation der Kanten ist eine zulässige Lösung)
   - $f(c) = c(v_{i,n}, v_{i,1}) + \sum_{j=1}^{n-1} c(v_{i,j}, v_{i,j+1})$
   - min

b) Das Rucksackproblem (RUCKSACK) ist charakterisiert durch
   - $\mathcal{D} = \{\langle W, vol, p, B\rangle | \, W = \{1, ..., n\}, vol = W \to \mathbb{N}, p = W \to \mathbb{N}, B \in \mathbb{N}$ und $\forall w \in \mathbb{N} : vol(w) \leq B\}$. W ist das Warenangebot, vol sind die Volumina der Waren, p die Preise und B das Rucksack-Volumen.
   - $S(\langle W, vol, p, B\rangle) = \left\{ A \subseteq W \mid \sum_{w \in A} vol(w) \leq B \right\}$

- $f(A) = \sum_{w \in A} p(w)$
- max

Der bis heute beste Algorithmus für das TSP hat die Laufzeit von $\mathcal{O}^*\left(n^2 \cdot 2^n\right)$. Der schnellste Algorithmus für RUCKSACK hat ebenfalls exponentielle Laufzeit.

1.2. Approximationsalgorithmen: Schnell, aber nicht optimal

Def. 1.4: Sei Π ein kombinatorisches Optimierungsproblem. Ein $t(n)$–Zeit–Approximations-algorithmus A berechnet zu Eingabe $I \in \mathcal{D}$ in Zeit $t(|I|)$ eine Ausgabe $\sigma_I^A \in S(I)$. Wir schreiben $A(I) = f\left(\sigma_i^A\right)$. ("Die eigentlich "dumme" Definition vom Approximationsalgorithmus")

Wir fordern erst einmal, $t(n)$ ist ein Polynom. Wir wollen den Unterschied zwischen $A(I)$ und $\mathrm{OPT}(I)$ quantifizieren:

- durch obere Schranke ("$A(I)$ weicht höchstens "so und so" von $OPT(I)$")
- durch untere Schranke
  - "es gibt I, sodass $A(I)$ "so und so" von $\mathrm{OPT}(I)$ tatsächlich abweicht"
    $\Rightarrow$algorithmusbezogene untere Schranke
  - "es gibt keinen Approximationsalgorihtmus, der das Problem Π "so und so gut" approximiert"
    $\Rightarrow$problembezogene untere Schranke

Kapitel 2. Approximation mit absoluter Gütegarantie

Def. 2.1: Π wie gehabt und sei A ein Approximationsalgorithmus für Π.

a) A hat bei Eingabe I eine absolute Güte ("individuelle absolute Güte") von $\kappa_A(I) = |A(I) - \mathrm{OPT}(I)|$
b) Die absolute worst–case–Güte von A ist die Funktion

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

c) Sei $\kappa_A : \mathbb{N} \to \mathbb{N}$ eine Funktion. A garantiert eine absolute Güte von $\kappa_A(n)$, falls für alle n gilt: $\kappa_A^{\mathrm{WC}}(n) \leq \kappa_A(n)$

2.1. Graphfärbbarkeit

Sei $G = (V, E)$ ein ungerichteter Graph. Für $u \in V$ ist $\Gamma_G(u) = \{v |\ \{u, v\} \in E\}$ die Menge der Nachbarn von u. $\deg_G(u) = |\Gamma_G(u)|$ ist der Grad von u. Der Grad von G ist $\Delta(G) = \max\{\deg_G(u) |\ u \in V\}$.

Def. 2.2: Gegeben sei ein Graph $G = (V, E)$.

a) Eine Abbildung $c_V : V \to \mathbb{N}$ heißt Knotenfärbung von G, falls für alle $\{u, v\} \in E$ gilt: $c_V(u) \neq c_V(v)$.
b) Eine Abbildung $c_E : E \to \mathbb{N}$ heißt Kantenfärbung von G, falls für alle an einem Knoten u aufeinandertreffenden Kanten $\{u, v\}, \{u, w\}$ gilt: $c_E(\{u, v\}) \neq c_E(\{u, w\})$.

$c_V(u)$ und $c_E(\{u, v\})$ heißen Farben. $|c_V(V)|$ bzw. $|c_E(E)|$ ist die Anzahl benutzter Farben.

Def. 2.3: Das Knotenfärbungsproblem ist charakterisiert durch:

- $\mathcal{D} = \{\langle G \rangle |\ G = (V, E)$ ist ungerichteter Graph mit mindestens einer Kante$\}$
- $S(\langle G \rangle) = \{c_V |\ c_V$ ist Knotenfärbung von $G\}$
- $f(c_V) = |c_V(V)|$
- min

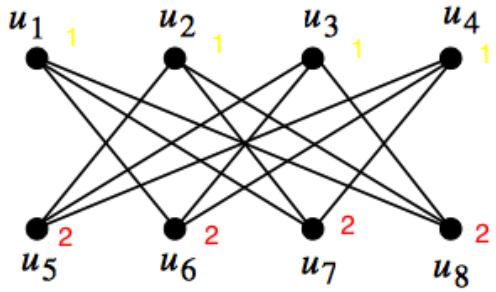
(Analog: Kantenfärbungen)

2.1.1. Knotenfärbungen

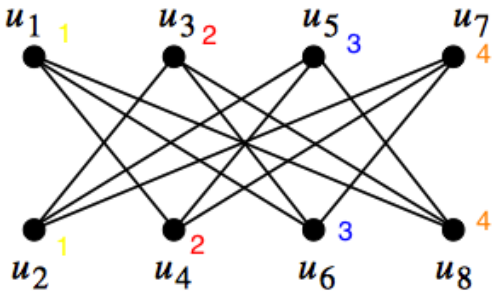
Knotenfärbungen beliebiger Graphen (mit mindestens einer Kante):
$G = (V, E), V = \{u_1, ..., u_n\}$.

Algorithmus GREEDYCOL:
for $i := 1$ to n do $c_V(u) := \infty$;
for $i := 1$ to n do
    $c_V(u_i) := \min\{\mathbb{N} \setminus \{c_V(\Gamma(u_i))\}\}$
gib c_V aus

Ein Beispiel:



GREEDYCOL färbt den obigen Graphen mit 2 Farben. Allerdings steigt die Farbenanzahl mit einer anderen Nummerierung:



Satz 2.6: GREEDYCOL garantiert eine absolute Güte von

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

Satz 2.5: Sei $G = (V, E)$ ein Graph. GREEDYCOL berechnet in Zeit $\mathcal{O}(|V| + |E|)$ eine Knotenfärbung aus höchstens $\Delta(G) + 1$ Farben. (d. h., $\text{GREEDYCOL}(G) \leq \Delta(G) + 1$)

Beweis: Korrektheit und Laufzeit ✓.
Wenn GREEDYCOL bei u_i angekommen ist, können nicht alle Farben aus $\{1, ..., \deg_G(u_i) + 1\}$ an die Nachbarn von u_i vergeben sein. Also werden insgesamt nie mehr als $\Delta(G) + 1$ Farben vergeben. □

2.2. Ein Unmöglichkeitsergebnis für das Rucksackproblem

Problembezogene untere Schranke

Satz 2.12: Falls P≠NP, gibt es keine Konstante $k \in \mathbb{N}$, sodass es einen polynomiellen Approximationsalgorithmus A für das Rucksackproblem gibt mit $|A(I) - \text{OPT}(I)| \leq k$.

Beweis: Annahme: es gibt einen derartigen Approximationsalgorithmus A.
Ziel: Polynomzeitalgorithmus, der das Rucksackproblem mit Hilfe von A exakt löst.
Sei $I = \langle W, vol, p, B\rangle$ eine Eingabe zum Rucksackproblem. Wir korrespondieren $I' = \langle W', vol', p', B'\rangle$ mit $W' = W$, $vol' = vol$, $B' = B$ und $p'(w) = (k+1) \cdot p(w)$. Zulässige Lösungen ändern sich nicht, $S(I') = S(I)$. Da die Multiplikation monoton steigend ist, ändert sich die Bewertungsreihenfolge der zulässigen Lösungen nicht.

$$\begin{aligned} \sigma \in S(I) &= S(I') \\ f_I(\sigma) &= \sum_{w\in\sigma} p(w) \\ f_{I'}(\sigma) &= \sum_{w\in\sigma} (k+1)\, p(w) = (k+1) \sum_{w\in\sigma} p(w) = (k+1)\, f_I(\sigma) \end{aligned}$$

Exakter Algorithmus für RUCKSACK: Berechne I', löse I', gib die Lösung aus.
Laufzeit: Multiplikation mit $k+1$ geht in Polyzeit in $|I|$, bzw. auch insgesamt in Polyzeit.
Sei σ eine optimale Lösung für I und $\hat{\sigma}$ eine zweitbeste, nicht optimale Lösung für $\hat{\sigma}$. $|f_I(\sigma) - f_I(\hat{\sigma})| \geq 1$. Also $|f_{I'}(\sigma) - f_{I'}(\hat{\sigma})| \geq k+1$. Im "Umkreis" von k um σ gibt es bei I' keine zweitbeste Lösung mehr. $\Rightarrow$ der exakte Algorithmus gibt eine Lösung aus, die für I' optimal ist, aber auch für I. □

Gap Amplification
Rucksack exakt gelöst mittels Approximationsalgorithmus für Rucksack ("Selbstreduktion").

Kapitel 3. Approximation mit relativer Gütegarantie

Def. 3.1: Sei Π ein kombinatorisches Optimierungsproblem und sei A ein Approximationsalgorithmus für Π.

a) A hat bei Eingabe I eine relative Güte von

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

wobei der linke Term einem Maximierungsproblem und der rechte Term einem Minimierungsproblem entspricht.

b) Die relative worst-case-Güte von A ist die Funktion

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

c) Sei $\rho_A : \mathbb{N} \to \mathbb{N}$ eine Funktion. A garantiert eine relative Güte von $\rho_A(n)$, falls für alle n gilt: $\rho_A^{WC}(n) \leq \rho_A(n)$

d) A macht bei Eingabe I einen relativen Fehler von

$$\epsilon_A(I) = \frac{|A(I) - \mathrm{OPT}(I)|}{\mathrm{OPT}(I)} = \left|\frac{A(I)}{OPT(I)} - 1\right|.$$

Gleiche Verallgemeinerung zu $\epsilon_A(n)$ wie bei $\rho_A(n)$ oben.

e) Relative Abweichung und Zeugen gegen A werden definiert wie bei absoluter Gütegarantie.

3.1. Das metrische Traveling Salesperson Problem

Wir modifizieren das volle TSP zu:

$$\mathcal{D} = \{\langle K_n, c\rangle \,|\, K_n \text{ vollständiger Graph auf } n \text{ Knoten}, c : E \to \mathbb{N}, \forall u, v, w \in V : c(u,v) \leq c(u,w) + c(w,v)\}$$

(Metrisches TSP, ΔTSP).

3.1.1. Einfügeheuristiken

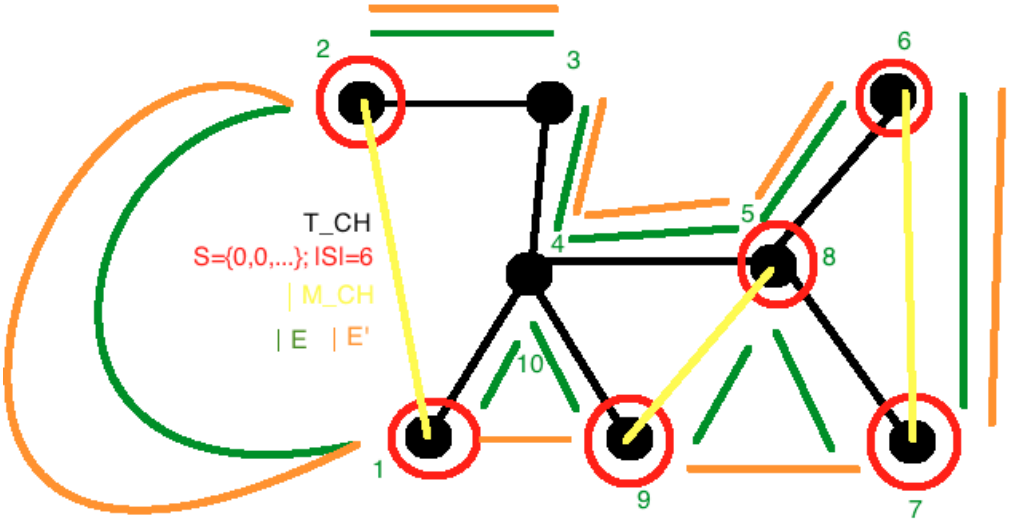
s. Buch

3.1.2. Christofides' Algorithmus

Ein Matching M eines kantengewichteten Graphen G ist ein Teilgraph von G, in dem jeder Knoten maximal Grad 1 hat. Das Gewicht von M ist die Summe der Gewichte der in M vorkommenden Kanten. Ist G ein vollständiger Graph mit gerader Anzahl n an Knoten, so gibt es perfekte Matchings, d. h., Matchings, in denen jeder der n Knoten genau Grad 1 hat.
Die Berechnung eines leichtesten Matchings ist in Zeit $\mathcal{O}\left(n^{2,5}\,(\log n)^4\right)$ möglich.
Ein Euler-Pfad in einem zusammenhängendem Graphen ist ein Weg, in dem jede Kante einmal vorkommt.
Ein Euler-Kreis ist ein geschlossener Eulerpfad. Genau dann hat G einen Euler-Kreis, wenn alle Knoten geraden Grad haben.
Zudem: Euler-Kreis kann in Zeit $\mathcal{O}\,(|V|+|E|)$ berechnet werden.

Algorithmus CH

1) berechne einen minimalen Spannbaum T_{CH} zu $T = \langle K_n, c\rangle$
2) $S := \left\{v \in T_{CH} \,\middle|\, \deg_{T_{CH}}(v) \text{ ist ungerade}\right\}$ (Es gilt: $|S|$ ist gerade)
3) berechne auf dem durch S induzierten Teilgraphen von K_n ein leichtestes Matching M_{CH}
4) berechne eine Euler-Tour $E = (u_1, u_2, \ldots)$ auf $T_{CH} \dot\cup M_{CH}$ (disjunkte Vereinigung $\Rightarrow$ kann ein Multigraph sein wegen möglicher Verdopplung von Kanten)
   (alle Knoten haben geraden Grad)
5) entferne in E Wiederholungen von Knoten, sodass man E' erhält, und gib E' aus

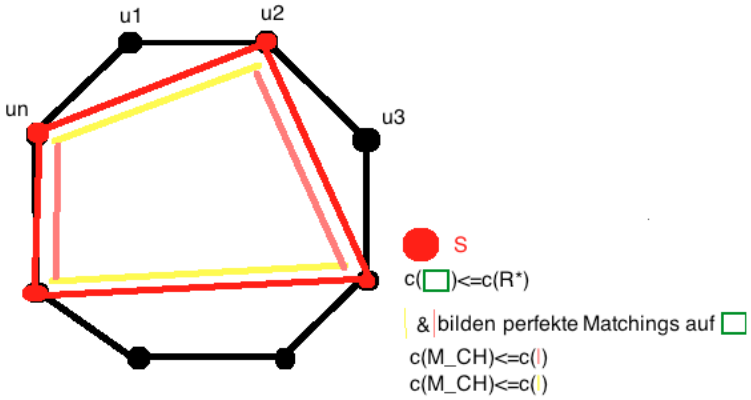


Satz 3.8: CH, gestartet mit einer Eingabe auf n Knoten, garantiert eine relative Güte von $\rho_{CH} \le \frac{3}{2} - \frac{1}{n}$.

Beweis: Sei R^* eine optimale Rundreise zu $I = \langle K_n, c\rangle$, d. h. $c(R^*) = OPT(I)$. Zu zeigen: $\mathrm{CH}(I) < \left(\frac{3}{2} - \frac{1}{n}\right) \cdot OPT(I)$. Für alle e gilt: $c(R^* - e) \ge c(T_{CH})$
zu 1): Da R^* aus n Kanten besteht, gibt es mindestens eine Kante $\hat e$ in R^*, die mindestens die durchschnittliche Länge $\frac{c(R^*)}{n}$ hat, also $c(\hat e) \ge \frac{c(R^*)}{n}$. Da T_{CH} minimaler Spannbaum ist, gilt:

$$c(T_{CH}) \le c(R^*) - c(\hat e) \le c(R^*) - \frac{c(R^*)}{n} = \left(1 - \frac{1}{n}\right) \cdot OPT(I).$$

zu 2): $|S|$ ist gerade: einfache Induktion
zu 3):



$$c(\text{gelb}) + c(\text{rot}) = c(\text{grün}) \Rightarrow \min(c(\text{gelb}), c(\text{rot})\} \leq \frac{1}{2} \cdot c(\text{grün}) = \frac{1}{2} OPT(I)$$

Für den Beweis der Ungleichung zwischen Kreis und R wird die ΔUG benötigt.
zu 4): $c(E) = c(T_{CH}) + c(M_{CH}) \leq \left(1 - \frac{1}{n}\right) \cdot OPT(I) + \frac{1}{2} \cdot OPT(I) = \left(\frac{3}{2} - \frac{1}{n}\right) \cdot OPT(I)$
zu 5): Streichen von doppelten Knoten in E benutzt Abkürzungen (Anwendung der ΔUG) in K_n, also ist

$$c(E') \leq c(E) \leq \left(\frac{3}{2} - \frac{1}{n}\right) \cdot OPT(I).$$

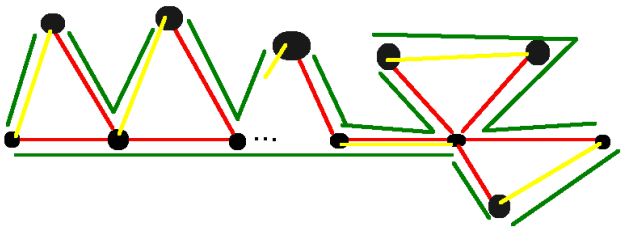
□

Laufzeit ist polynomiell in $|I|$.

Betrachten wir nun den Graphen $G_{\text{Bad}}(n)$ (s. Abb. 3.4 auf S. 54 im Buch). Eingezeichnete Kanten haben dabei die Länge 1; nicht eingezeichnete Kanten haben Länge eines kürzesten Weges zwischen den Knoten.

$$OPT(G_{\text{Bad}}(n)) = n; \; R^* = (1, 2, ..., n, 1)$$

Möglicher minimaler Spannbaum:



$$c(T_{CH}) = n - 1; c(M_{CH}) = \frac{n}{2}; c(T_{CH} \dot{\cup} M_{CH}) = \frac{3}{2}n - 1$$

Jedes Streichen von doppelt vorkommenden Knoten verändert nichts. Länge der Ausgabe: $\frac{3}{2}n - 1$, d. h., $\frac{\text{CH}(G_{\text{Bad}}(n))}{OPT(G_{\text{Bad}}(n))} = \frac{\frac{3}{2}n-1}{n} = \frac{3}{2} - \frac{1}{n}$.

3.2. Unabhängige Mengen und noch einmal Knotenfärbungen

Das Independent Set Problem

Def. 3.10 Sei $G = (V, E)$ ein Graph und sei $U \subseteq V$ eine Knotenmenge. U wird unabhängig genannt, wenn für alle Knotenpaare $u, v \in U$ gilt: $\{u, v\} \notin E$.
Das Independent Set Problem IS ist das Optimierungsproblem, zum Eingabegraphen eine möglichst große unabhängige Knotenmenge zu bestimmen.
Die Entscheidungsvariante ist NP−vollständig.

Algorithmus GREEDYIS
$U := \emptyset$; $t := 0$; $V^{(0)} := V$
while $V^{(t)} \neq \emptyset$ do {Rundo t}
    $G^{(t)}$:= der durch $V^{(t)}$ induzierte Graph
    u_t := ein Knoten in $G^{(t)}$ mit minimalem Grad
    $v^{(t+1)} := V^{(t)} \setminus (\{u_t\} \cup \Gamma_{G^{(t)}}(u_t))$
    $t := t + 1$
done
gib U aus.

GREEDYIS berechnet in Polynomzeit eine nicht erweiterbare unabhängige Menge von G, d. h. eine zulässige Lösung für IS.

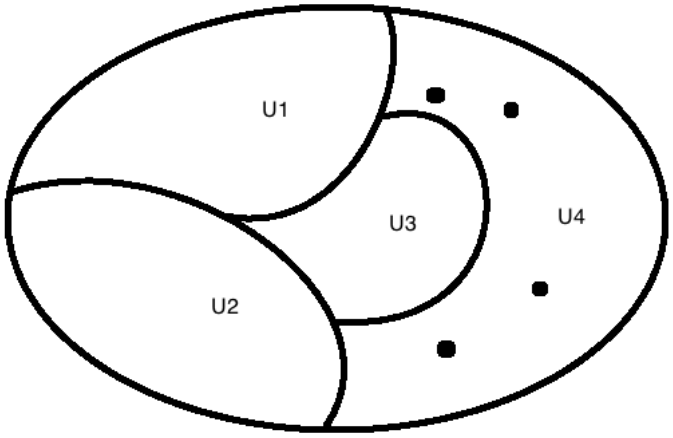
Analyse von GREEDYIS in Abhängigkeit von der chromatischen Zahl
chromatische Zahl: $OPT(G)$, wenn das Knotenfärbungsproblem betrachtet wird
k sei die chromatische Zahl des Graphen G (jetzt wieder bei IS)

Satz 3.13: Sei $G = (V, E)$ ein knoten-k-färbbarer Graph. Dann ist $\text{GREEDYIS}(G) \geq \left\lceil \log_k \left(\frac{|V|}{3} \right) \right\rceil$.

Beweis:

Lemma 3.14: Sei $G = (V, E)$ ein knoten-k-färbbarer Graph. Dann gibt es mindestens ein $u \in V$ mit $\deg_G(u) \leq \left\lfloor \left(1 - \frac{1}{k}\right) \cdot |V| \right\rfloor$.

Beweis: Sei G mit k Farben gefärbt und bezeichne u_i die Menge der Knoten, die die Farbe i bekommen haben.



Die U_i sind unabhängige Mengen. Unter den Mengen $U_1, ..., U_k$ muss es mindestens eine Menge U geben, die mindestens durchschnittlich groß ist. Die Durchschnittsgröße ist $\frac{|V|}{k}$, also $|U| \geq \left\lceil \frac{|V|}{k} \right\rceil$. Jeder Knoten in U kann nur mit höchstens allen außerhalb von U liegenden Knoten verbunden sein. Dies sind $|V| - |U| \leq |V| - \left\lceil \frac{|V|}{k} \right\rceil = \left\lfloor \left(1 - \frac{1}{k}\right) |V| \right\rfloor$. D. h., $\deg_G(u) \leq \left\lfloor \left(1 - \frac{1}{k}\right) |V| \right\rfloor$. □ (Lemma 3.14)

Sei $|V| = n$, $\left|V^{(t)}\right| = n_t$ und sei $k \geq 2$. Wegen Lemma 3.14 hat u_t höchstens $\left\lfloor \left(1 - \frac{1}{k}\right) \cdot n_t \right\rfloor$ Nachbarn.

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

und damit $n_t \geq \frac{n}{k^t} - \frac{k}{k-1}\left(1 - \frac{1}{k^t}\right)$. Da $k \geq 2$, ist $\frac{k}{k-1}\left(1 - \frac{1}{k^t}\right) \leq 2$, also ist $n_t \geq \frac{n}{k^t} - 2$.
Für jede Runde t, für die $n_t \geq 1$ garantiert werden kann, wird ein neuer Knoten nach U gelegt.

Das gilt, solange $t \leq \log_k\left(\frac{n}{3}\right)$ ist. D. h., $|U| \geq \left\lceil \log_k\left(\frac{n}{3}\right)\right\rceil$. □

Ein besserer Knotenfärbungsalgorithmus

Algorithmus GREEDYCOL2
$t := 1$; $V^{(1)} := V$
while $V^{(t)} \neq \varnothing$ do
    $G^{(t)}$:= der durch $V^{(t)}$ induzierte Graph
    $U_t := \text{GREEDYIS}\left(G^{(t)}\right)$
    färbe alle Knoten in U_t mit der Farbe t
    $V^{(t+1)} := V^{(t)} \backslash U_t$
    $t := t + 1$
done
gib die Färbung aus

Satz 3.15: Sei $G = (V, E)$ ein knoten-k-färbbarer Graph. GREEDYCOL gibt eine Färbung mit höchstens $\frac{3n}{\log_k\left(\frac{n}{16}\right)}$ Farben aus und hat eine relative Gütegarantie von $\mathcal{O}\left(\frac{n}{\log n}\right)$.

Beweis: $n_t := \left|V^{(t)}\right|$. Wegen Satz 3.13 ist $|U_t| \geq \log_k\left(\frac{n_t}{3}\right)$.

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

Der Algorithmus ist fertig, wenn $n_t < 1$ ist, und hat t Farben vergeben. Das dauert $t \approx \frac{3n}{\log_k\left(\frac{n}{3}\right)}$ Runden.

Zur relativen Güte: $\frac{\text{GREEDYCOL2}(G)}{OPT(G)} \leq \frac{\frac{3n}{\log_k\left(\frac{n}{3}\right)}}{k} = \frac{3n}{\log\left(\frac{n}{16}\right)} \cdot \frac{\log k}{k} = \mathcal{O}\left(\frac{n}{\log n}\right)$. □

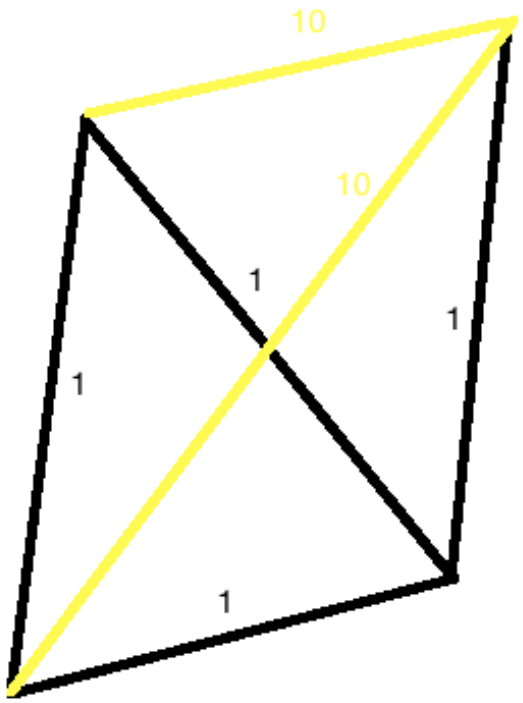
3.3. Ein Unmöglichkeitsergebnis fürs TSP

Satz 3.16: Wenn es einen polynomiellen Approximationsalgorithmus A mit konstanter relativer Gütegarantie r für das volle TSP gibt, dann ist P=NP.

Beweis: Annahme: So einen Algorithmus A gibt es. Ziel: HAMILTON in Polyzeit zu entscheiden mit Hilfe von A.
Eingabe: Graph $G = (V, E)$, Frage: Hat G einen Hamilton-Kreis?

Mache G vollständig durch Hinzufügen der nicht vorhandenen Kanten und gib den Kanten folgende Gewichte ($n = |V|$):

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



"Ich kann das TSP mit $r = 3$ lösen $\Rightarrow n = 4$

- Hat G einen Hamilton-Kreis, dann ist die Länge einer kürzesten Rundreise durch den modifizierten Graphen n.
- Hat G keinen Hamilton-Kreis, dann hat die kürzeste Rundreise durch den modifizierten Graphen mindestens folgende Länge, da mindestens eine lange Kante benutzt werden muss:

$$n - 1 + (r - 1) \cdot n + 2 = r \cdot n + 1 > r \cdot n$$

(linker Term: kurze Kanten; rechter Term: lange Kante)
D. h., besitzt G einen Hamilton-Kreis, darf A keine neue Kante benutzen. D. h., A muss einen Hamilton-Kreis in G finden. Gibt A eine Rundreise aus, die eine neue Kante benutzt, kann G keinen Hamilton-Kreis enthalten.

Kapitel 4. Approximationsschemata

Def. 4.1: Sei Π ein kombinatorisches Optimierungsproblem. Sei A ein Approximationsalgorithmus, der als Eingabe eine Instanz I von Π und ein ϵ, $0 < \epsilon < 1$, bekommt.

a) A ist ein polynomielles Approximationsschema (P(T)AS, engl. Polynomial (Time) Approximation Scheme) für Π, wenn A zu jeder Instanz I und für jedes $\epsilon \in]0, 1[$ in Zeit poly $(|I|)$ eine zulässige Lösung zu I mit relativem Fehler $\epsilon_A(I, \epsilon) \leq \epsilon$ berechnet.
b) A ist ein streng polynomielles Approximationsschema (FP(T)AS, engl. fully PAS), wenn A ein PAS mit Laufzeit poly $\left(|I|, \frac{1}{\epsilon}\right)$ (erlaubte Laufzeit: z. B. $n^3 \cdot \left(\frac{1}{\epsilon}\right)^7$).

Satz 4.2: Sei Π ein kombinatorisches Optimierungsproblem, A ein (F)PAS und zu Eingabe I sei $Z(I)$ eine obere Schranke für $OPT(I)$, d. h., $OPT(I) \leq Z(I)$. Sei $\epsilon^* = \frac{1}{Z(I)+1}$. Dann ist $A(I, \epsilon^*) = OPT(I)$. Ist A ein FPAS, so ist die Laufzeit poly $(|I|, Z(I))$.

Beweis: Starte A mit I und ϵ^*.

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

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

$\Rightarrow OPT(I) = A(I,\epsilon^*)$. □

4.1. Ein pseudopolynomieller exakter Algorithmus für das Rucksackproblem

$W = \{1,...,n\}$, p_i: Preis der Ware i, $\text{vol}(i)$: Volumen der Ware i, B: Rucksackkapazität.
$P_{\max} \leq OPT(I) \leq n \cdot P_{\max}$ (der Preis der wertvollsten Ware).
Der exakte Algorithmus ist ein typischer Vertreter der dynamischen Programmierung.
$F_j(\alpha)$ für $\alpha \in \mathbb{Z}$ und $j \in \{0,1,...,n\}$ sei das kleinste benötigte Rucksackvolumen, mit dem man mindestens den Wert α erzielen kann, wenn man die ersten j Waren einpacken darf.
$j = 0$: Man darf nichts einpacken.
$F_j(\alpha) = \infty$, wenn man den gewünschten Wert nicht erreichen kann.

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

Lemma 4.3:

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

Was brauchen wir? Wir suchen das größte α, sodass $F_n(\alpha)$ noch in unseren Rucksack der Kapazität B passt. D. h., $\max\{\alpha \,|F_n(\alpha) \leq B\} = OPT(I)$.

	0	1	...	j-1	j	...	n
0	0				0		0
1	∞						
2	∞						
...							
$\alpha - p_j$				$F_{j-1}(\alpha - p_j)$			
...							
α				$F_{j-1}(\alpha)$	$F_j(\alpha)$		
...							
$OPT(I)$							

Algorithmus DYNRUCKSACK
$\alpha := 0$;
repeat $\alpha := \alpha + 1$; // Beim Heruntersetzen muss dieser Schritt geändert werden.
  for $j := 1$ to n do
    $F_j(\alpha) = \min\{F_{j-1}(\alpha), F_{j-1}(\alpha - p_j) + \text{vol}(j)\}$
until $B < F_n(\alpha)$
gib $\alpha - 1$ aus.

Satz 4.4: DYNRUCKSACK berechnet zu I den Wert $OPT(I)$ in Zeit $\mathcal{O}(n \cdot OPT(I)) = \mathcal{O}(n^2 \cdot P_{\max})$. (Exponentielle Laufzeit, da $P_{\max}$ zur Kodierung nur $\mathcal{O}(\log P_{\max})$ viele Bits benötigt.

Def. 4.5: Sei Π ein kombinatorisches Optimierungsproblem, sodass für alle Instanzen I gilt, dass alle in I vorkommenden Zahlen natürliche Zahlen sind. Sei $\text{maxnr}(I)$ die größte in I vorkommende Zahl. Ein Algorithmus für Π heißt pseudopolynomiell, falls es ein Polynom $\text{poly}(\cdot,\cdot)$ gibt, sodass die Laufzeit für alle I höchstens $\text{poly}(|I|, \text{maxnr}(I))$ ist.

4.2. Ein streng polynomielles Approximationsschema für das Rucksackproblem

Den Algorithmus DYNRUCKSACK bekommen wir polynomiell, indem wir die Preise um das "richtige" k reduzieren, also p_j durch $\left\lfloor \frac{p_j}{k} \right\rfloor$ ersetzen. Diese Eingabe nennen wir I_{red}.
Das richtige k wird von I und dem vorgegebenen ϵ abhängen.

Algorithmus AR_k
1) reduziere die Preise zu $\left\lfloor \frac{p_j}{k} \right\rfloor$
2) berechne mittels DYNRUCKSACK eine optimale Lösung R_k auf I_{red}
3) gib R_k aus

Satz 4.6: AR_k macht bei Eingabe I einen relativen Fehler von $\epsilon_{AR_k}(I) \leq \frac{k \cdot n}{P_{\max}}$ bei Laufzeit von $\mathcal{O}\left(n^2 \cdot \frac{P_{\max}}{k}\right)$.

Beweis: Sei R^* eine optimale Rucksackfüllung zu I und sei R_k die berechnete optimale Lösung zu I_{red}.

$$OPT(I_{\text{red}}) \geq \sum_{j \in R^*} \left\lfloor \frac{p_j}{k} \right\rfloor$$

$$AR_k(I) = p(R_k) = \sum_{j \in R_k} p_j \geq k \cdot \sum_{j \in R_k} \left\lfloor \frac{p_j}{k} \right\rfloor = k \cdot OPT(I_{\text{red}}) \geq k \cdot \sum_{j \in R^*} \left\lfloor \frac{p_j}{k} \right\rfloor \geq k \cdot \sum_{j \in R^*} \left(\frac{p_j}{k} - 1 \right)$$

$$= \sum_{j \in R^*} (p_j - k) = OPT(I) - k \cdot |R^*| \geq OPT(I) - k \cdot n.$$

$\Rightarrow |AR_k(I) - OPT(I)| \leq k \cdot n \Rightarrow \epsilon_{AR_k} = \frac{|AR_k(I) - OPT(I)|}{OPT(I)} \leq \frac{k \cdot n}{OPT(I)} \leq \frac{k \cdot n}{P_{\max}}$. □

Jetzt: Eingabe Instanz I und Fehlerschranke ϵ

Algorithmus FPASRUCKSACK(I, ϵ)
bestimme aus I die Parameter n und $P_{\max}$
$k := \epsilon \cdot \frac{P_{\max}}{n}$
starte AR_k mit I und gib dessen Lösung aus

Satz 4.7: FPASRUCKSACK ist ein FPAS für das Rucksackproblem mit Laufzeit $\mathcal{O}\left(n^3 \cdot \frac{1}{\epsilon}\right)$

$$\epsilon_{AR_k} \leq \frac{k \cdot n}{P_{\max}} = \frac{\epsilon \cdot \frac{P_{\max}}{n} \cdot n}{P_{\max}} = \epsilon; \mathcal{O}\left(n^2 \cdot \frac{P_{\max}}{k}\right) = \mathcal{O}\left(n^2 \cdot \frac{P_{\max}}{\epsilon \cdot \frac{P_{\max}}{n}}\right) = \mathcal{O}\left(n^3 \cdot \frac{1}{\epsilon}\right)$$

4.3. Unmöglichkeitsergebnisse für Approximationsschemata

Def. 4.9: Ein NP−vollständiges Entscheidungsproblem L heißt stark NP−vollständig, wenn es ein Polynom q gibt, sodass $L_q = \{x \mid x \in L \text{ und } \text{maxnr}(x) \leq q(|x|)\}$ NP−vollständig ist. Gibt es kein solches Polynom, so ist L schwach NP−vollständig.

Bsp. 4.10:

- HAMILTON und $\text{CLIQUE}_{\text{ent}}$ sind stark NP−vollständig, da in ihnen keine großen Zahlen vorkommen und das Polynom $q(n) = n$ ausreicht.
- TSP ist stark NP−vollständig. In Satz 3.16 kann auch $r = 1$ eingesetzt werden. Dann sind die Kantenlängen alle nur die 1 oder 2. $q(n) = n$ ist wieder das Polynom, sodass die Einschränkung NP−vollständig bleibt. Ja, sogar: Wenn alle Kantenlängen nur 1 oder 2 sind, gilt im vollständigen Graphen die Dreiecksungleichung, also ist auch ΔTSP stark NP−vollständig.
- RUCKSACK ist schwach NP−vollständig, da die Optimierungsvariante einen pseudopolynomiellen Algorithmus hat.

Satz 4.11: Sei Π ein kombinatorisches Optimierungsproblem. Wenn es ein Polynom $q(x_1, x_2)$ gibt, sodass für alle Instanzen I gilt, dass $OPT(I) \leq q(|I|, \text{maxnr}(I))$ ist, dann folgt aus der Existenz eines FPAS für Π, dass es einen pseudopolynomiellen Algorithmus für Π gibt.

Beweis: Die Aussage folgt wie im Satz 4.2.

Korollar 4.12: Wenn es für die Optimierungsvariante eines stark NP−vollständigen Problems ein FPAS gibt, ist P=NP.

Teil II: Techniken

Kapitel 6: Techniken für randomisierte Approximationsalgorithmen

6.1. Die Probabilistische Methode

Def. 6.1: Sei $V = \{x_1, ..., x_n\}$ die Menge der Booleschen Variablen. Ein Literal ist eine Variable $x_i \in V$ oder ihre Negation $\overline{x_i}$. Eine Klausel $C = l_1 \vee ... \vee l_k$ ist eine Oder-Verknüpfung von Literalen. Eine Boolesche (n, m) −Formel $\Phi = C_1 \wedge ... \wedge C_m$ in konjunktiver Normalform (KNF) ist eine Und-Verknüpfung von Klauseln aus Variablen aus V. Wir schreiben auch $C_j \in \Phi$.

Def. 6.2: Eine Instanz Φ von MaxSAT ist eine Boolesche (n, m) −Formel in KNF. Eine zulässige Lösung zu Φ ist eine Belegung $b : V \to \{\text{TRUE}, \text{FALSE}\}$ der Variablen. Die Booleschen Wahrheitswerte für Literale, Klauseln und Formeln ergeben sich kanonisch.
Die Bewertungsfunktion ist

$$wahr(b, \Phi) = |\{j \,|C_j \in \Phi, b(C_j) = \text{TRUE}\}|.$$

$wahr(b, \Phi)$ ist die Anzahl der Klauseln, die unter b erfüllt werden.
Ziel: finde b^* mit $wahr(b^*, \Phi)$ ist maximal.
Bsp: $\Phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2}) \Rightarrow OPT(\Phi) = 3$.
Wir zeigen: $OPT(\Phi) \geq \frac{m}{2}$ für alle Φ.

Satz 6.3: Sei Φ eine Boolesche (n, m) −Formel in KNF. Dann ist

$$\max\{wahr((\text{FALSE}, ..., \text{FALSE}), \Phi), wahr((\text{TRUE}, ..., \text{TRUE}), \Phi)\} \geq \frac{m}{2}$$

□

Da $OPT(\Phi) \leq m$, liefert dies bereits ein Approximationsverfahren der relativen Gütegarantie 2.

Algorithmus A for $i := 1$ to n do
$$\begin{cases} \text{mit Wahrscheinlichkeit } \frac{1}{2}: & x_i := \text{TRUE} \\ \text{mit Wahrscheinlichkeit } \frac{1}{2}: & x_i := \text{FALSE} \end{cases}$$ diese Experimente sind unabhängig!
done
gib die Belegung $b_A = (x_1, ..., x_n)$ aus

$\Pr[x_i = \text{TRUE}] = \Pr[x_i = \text{FALSE}] = \frac{1}{2}$.

Lemma 6.4: Sei k_j die Anzahl der Literale in Klausel C_j. Es gilt: $\Pr[C_j \text{ wird durch } A \text{ erfüllt}] = 1 - \frac{1}{2^{k_j}}$.

Beweis: Wahrscheinlichkeit, dass ein einzelnes Literal zu FALSE wird, ist $\frac{1}{2}$.
Wahrscheinlichkeit, dass alle k_j Literale zu FALSE werden, ist $\left(\frac{1}{2}\right)^{k_j}$ wegen der Unabhängigkeit.
Das heißt, $\Pr[\text{alle Literale FALSE}] = \left(\frac{1}{2}\right)^{k_j} \Rightarrow \Pr[\text{mindestens 1 Literal TRUE}] = 1 - \left(\frac{1}{2}\right)^{k_j} = \Pr[C_j \text{ wird erfüllt}]$
□

Es gibt 2^{k_j} Belegungen, $2^{k_j} - 1$ Belegungen davon sind erfüllend.
$\Rightarrow \Pr[C_j \text{ wird erfüllt}] = \frac{2^{k_j}-1}{2^{k_j}} = 1 - \frac{1}{2^{k_j}}$

Zuvallsvariable X, die nur 0 oder 1 annehmen kann. (Indikatorvariable)

$$E[X] = 0 \cdot \Pr[X = 0] + 1 \cdot \Pr[X = 1] = \Pr[X = 1]$$

Satz 6.5: Für jede Boolesche (n, m) −Formel in KNF, in der jede Klausel mindestens k Literale hat, gilt:

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

Beweis: Für $j \in \{1, ..., m\}$ sei Z_j die Zuvallsvariable mit

$$Z_j = \begin{cases} 1 & \text{falls } b_A(C_j) = \text{TRUE (falls } C_j \text{ durch } A \text{ erfüllt wurde)} \\ 0 & \text{sonst} \end{cases}$$

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

□

Korollar 6.6: Für jede Boolesche (n, m)–Formel Φ in KNF, in der jede Klausel mindestens k Literale hat, gibt es eine Belegung b der Variablen mit wahr $(b, \Phi) \geq \left(1 - \frac{1}{2^k}\right) \cdot m$.

Satz 6.8: Algorithmus A hat für jede boolesche (n, m)–Formel Φ in KNF, in der jede Klausel mindestens die Länge k hat, eine erwartete relative Güte von

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

□

6.2. Randomized Rounding

6.2.1. Arithmetisierung von Max-SAT

$\Phi = C_1 \wedge C_2 \wedge ... \wedge C_m$ Boolesche (n, m)–Formel in KNF
FALSE$\widehat{=}$0
TRUE$\widehat{=}$1
$x_i \widehat{=} \hat{x}_i \in \{0, 1\}$
$C_j \widehat{=} \hat{Z}_j \in \{0, 1\}$
Anzahl der erfassten Klauseln: $\sum_{j=1}^{m} \hat{Z}_j$
$S_j^{\ominus}$: Menge der Variablen, die in C_j negiert vorkommen.
$S_j^{\oplus}$: Menge der Variablen, die in C_j positiv vorkommen.

Bsp:

$$C_j = \overline{x}_1 \vee x_3 \vee \overline{x}_7 \vee x_9 \Rightarrow S_j^{\ominus} = \{x_1, x_7\}; S_j^{\oplus} = \{x_3, x_9\}$$

Ganzzahliges Lineares Programm B für Max-SAT:

$$\begin{aligned} &\max \sum_{j=1}^{m} \hat{Z}_j \\ &\text{s. t.} \sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i) \geq \hat{Z}_j \text{ für alle } C_j \\ &\hat{x}_i, \hat{Z}_j \in \{0, 1\} \end{aligned}$$

c_j haben wir die Nebenbedingungen: $(1 - \hat{x}_1) + x_3 + (1 - \hat{x}_7) + x_9 \geq \hat{z}_j$

Relaxierung: Streiche $\hat{x}_i, \hat{Z}_j \in \{0, 1\}$. Setze ein: $0 \leq \hat{x}_i \leq 1$, $0 \leq \hat{Z}_j \leq 1$.
Das relaxierte Lineare Programm B_{rel}:

$$\begin{aligned} &\max \sum_{j=1}^{m} \hat{Z}_j \\ &\text{s. t.} \sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i) \geq \hat{Z}_j \text{ für jede Klausel } C_j \\ &0 \leq \hat{x}_i, \hat{Z}_j \leq 1 \end{aligned}$$

Das relaxierte Programm kann in Polynomzeit gelöst werden. Der Lösungsraum wird größer, also kann sein

$$\mathrm{OPT}(\Phi) = \mathrm{OPT}(B) \leq \mathrm{OPT}(B_{rel})$$

$\Phi = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2) \vee (x_1 \vee \overline{x_2})\,(\overline{x_1} \vee \overline{x_2})$
$\max \hat{Z}_1 + \hat{Z}_2 + \hat{Z}_3 + \hat{Z}_4$
$\hat{x}_1 + \hat{x}_2 \geq \hat{Z}_1$
$(1 - \hat{x}_1) + \hat{x}_2 \geq \hat{Z}_2$
$\hat{x}_1 + (1 - \hat{x}_2) \geq \hat{Z}_3$
$(1 - \hat{x}_1) + (1 - \hat{x}_2) \geq \hat{Z}_4$
$0 \leq \hat{x}_i, \hat{Z}_j \leq 1$
$\hat{x}_1 = \hat{x}_2 = \frac{1}{2} \Rightarrow \mathrm{OPT}(B) = 3; \mathrm{OPT}(B_{rel}) = 4$

Bei Maximierungsproblemen nennt man $\mathrm{OPT}(B) \leq \mathrm{OPT}(B_{rel})$ Superoptimalität und $\frac{\mathrm{OPT}(B_{\mathrm{rel}})}{\mathrm{OPT}(B)}$ die Ganzzahligkeitslücke. B_{rel} kann in Polynomzeit gelöst werden, aber die Lösung ist nicht unbedingt eine zulässige Lösung des ursprünglichen Problems.

6.2.2. Von der rationalen zur ganzzahligen Lösung

Sei $\pi : [0,1] \to [0,1]$ eine stochastische Funktion. Hier: $\pi = \mathrm{id}$.

RANDOMIZED_ROUNDING[π]
for $i := 1$ to n do
$\begin{cases} \text{mit Wahrscheinlichkeit } \pi(\hat{x}_i) & x_i := \text{TRUE} \\ \text{mit Wahrscheinlichkeit } \pi(\hat{x}_i) & x_i := \text{FALSE} \end{cases}$ unabhängig!

$\Pr[x_i = \text{TRUE}] = \hat{x}_i$; $\Pr[x_i = \text{FALSE}] = 1 - \hat{x}_i$.

Algorithmus B

1. löse das relaxierte LP B_{rel}
2. ermittle durch RANDOMIZED_ROUNDING[id] eine Belegung der (Booleschen) Variablen

Lemma 6.11: Sei k_j die Anzahl der Literale in C_j. $\Pr[C_j \text{ wird durch } B_j \text{ erfüllt}] \geq \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j$

Beweis: $C_j = \left(\bigvee_{x_i \in S_j^{\oplus}} x_i\right) \vee \left(\bigvee_{x_i \in S_j^{\ominus}} \overline{x_i}\right)$. $\hat{Z}_j \leq \sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i)$. $\left|S_j^{\oplus} \cup S_j^{\ominus}\right| = k_j$

$$\Pr[C_j \text{ wird durch } B \underline{\text{NICHT}} \text{ erfüllt}] = \left(\prod_{x_i \in S_j^{\oplus}} (1 - \hat{x}_i)\right) \cdot \left(\prod_{x_i \in S_j^{\ominus}} \hat{x}_i\right)$$

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

$$\prod_{i=1}^{k} a_i \leq \left(\frac{1}{k} \sum_{i=1}^{k} a_i\right)^k \Rightarrow \Pr[C_j \text{ wird durch } B \text{ erfüllt}] \geq 1 - \left(\frac{\sum_{x_i \in S_j^{\oplus}} (1 - \hat{x}_i) + \sum_{x_i \in S_j^{\ominus}} \hat{x}_j}{k}\right)^k$$

$$= 1 - \left(\frac{\left|S_j^{\oplus}\right| - \sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \left|S_j^{\ominus}\right| - \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i)}{k_j}\right)^{k_j} = 1 - \left(\frac{k_j - \left(\sum_{x_i \in S_j^{\oplus}} \hat{x}_i + \sum_{x_i \in S_j^{\ominus}} (1 - \hat{x}_i)\right)}{k_j}\right)^{k_j}$$

$$\geq 1 - \left(1 - \frac{\hat{Z}_j}{k_j}\right)^{k_j} \geq \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j$$

Ist $f(x)$ auf $[a,b]$ konkav und gilt $f(a) \geq m \cdot a + n$ und $f(b) \geq m \cdot b + n$, dann ist für alle $x \in [a,b]$: $f(x) \geq m \cdot x + n$.
□

Satz 6.12: Für jede Boolesche Formel Φ in KNF, in der jede Klausel höchstens k Literale enthält, ist

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

Beweis: $\Phi = C_1 \wedge ... \wedge C_m$.

$$E[B(\Phi)] = \sum_{j=1}^{m} \Pr[C_j \text{ wird durch } B \text{ erfüllt}] \geq \sum_{j=1}^{m} \left(1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right) \cdot \hat{Z}_j \geq \overbrace{\left(1 - \left(1 - \frac{1}{k}\right)^k\right)}^{\text{monoton fallend}} \cdot \sum_{j=1}^{m} \hat{Z}_j$$
$$\geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \mathrm{OPT}(\Phi)$$

□

Der Algorithmus B erreicht immer $\left(1 - \frac{1}{e}\right) \cdot \mathrm{OPT}(\Phi)$.

6.3. Der hybride Ansatz

Max-EkSAT (E steht für exakt): alle Klauseln haben genau die Länge k.

k	$A: 1 - \frac{1}{2^k}$	$B: 1 - \left(1 - \frac{1}{k}\right)^k$
1	0.5	1
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672
∞	1	0.632

$\Rightarrow$ gemeinsamer Worst Case ist $k = 2$ und besser als der Worst Case von A ($k = 1$) und der von B ($k = \infty$)

Algorithmus C_{p_A}

$$\begin{cases} \text{mit Wahrscheinlichkeit } p_A: & \text{starte } A \\ \text{mit Wahrscheinlichkeit } 1 - p_A: & \text{starte } B \end{cases}$$

Algorithmus C_{alle}.
– lass alle Algorithmen (d. h. A und B) laufen.
– gib das beste Ergebnis aus

$E[C_{\text{alle}}(\Phi)] \geq E[C_{p_A}(\Phi)] = p_A \cdot E[A(\Phi)] + (1 - p_A) \cdot E[B(\Phi)]$
Für jetzt: $p_A = \frac{1}{2}$.

Satz 6.13: Für jede Boolesche (n, m)–Formel in KNF gilt: $E[B(\Phi)] \geq \left(1 - \frac{1}{e}\right) \cdot \mathrm{OPT}(\Phi)$

Satz 6.14: Der Algorithmus $C_{\frac{1}{2}}$ hat erwartete relative Güte von $\frac{4}{3}$.

Beweis: Wir suchen $E\left[C_{\frac{1}{2}}(\Phi)\right] = \frac{1}{2} \cdot E[A(\Phi)] + \frac{1}{2} \cdot E[B(\Phi)]$.

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

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

$$E\left[C_{\frac{1}{2}}(\Phi)\right] = \frac{1}{2} \cdot E[A(\Phi)] + \frac{1}{2} \cdot E[B(\Phi)] \geq \frac{1}{2} \sum_{k=1}^{n} \sum_{C_j \text{ hat } k \text{ Literale}} \underbrace{\left(1 - \frac{1}{2^k} + 1 - \left(1 - \frac{1}{k}\right)^k\right)}_{\geq \frac{3}{2} \text{ für } k \in \mathbb{N}^+} \cdot \hat{Z}_j \geq \frac{3}{4} \sum_{j=1}^{m} \hat{Z}_j \geq \frac{3}{4} \cdot \mathrm{OPT}(\Phi)$$

□

6.5. Derandomisierung: Die Methode der bedingten Erwartungswerte

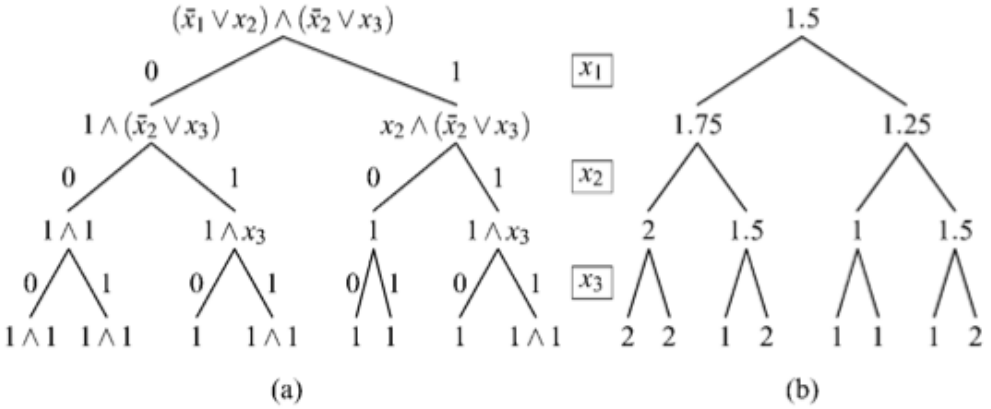


Abbildung 6.3: (a) Der Berechnungsbaum zu $\Phi = (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3)$ (0 steht für FALSE und 1 für TRUE) und (b) die entsprechenden Erwartungswerte für die Ausgabe von A für die Anzahl der erfüllten Klauseln.

Algorithmus DERAND_A
Eingabe: Boolesche (n, m) –Formel in KNF
for $i := 1$ to n do
  $W_0 := E[A(\Phi) | x_1, ..., x_{i-1}, x_i = 0]$ (können wir exakt ausrechnen)
  $W_1 := E[A(\Phi) | x_1, ..., x_{i-1}, x_i = 1]$ (können wir exakt ausrechnen)
  if $W_0 \leq W_1$
    then $x_i := 1$
    else $x_i := 0$
  end
done
gib x_1, ..., x_m aus

Satz 6.23: DERAND_A approximiert MaxSAT deterministisch mit absoluter Gütegarantie wie A in polynomieller Zeit.

Kapitel 7. Lineare Optimierung und Approximationsalgorithmen

7.1. Die Ganzzahligkeitslücke und die Beziehung zur relativen Güte

Rundungssansatz für ein kombinatorisches Optimierungsproblem π (o. B. d. A.: Maximierungsproblem)

1. Beschreibe die Instanz I durch ein ganzzahliges lineares Programm (Arithmetisierung). Es gilt: $\text{OPT}(I) = \text{OPT}(X)$.
2. Lass die Ganzzahligkeitsbedingung fallen (Relaxierung), sodass man das in Polynomzeit lösbare relaxierte Programm X_{rel} erhält. Es gilt $\text{OPT}(X) \leq \text{OPT}(X_{\text{rel}})$ (Superoptimalität).
3. Der Approximationsalgorithmus A löst X_{rel} und rundet geschickt die gebrochen-rationale Lösung zu einer zulässigen Lösung von I.
4. Zeige, dass $A(I) \geq \frac{1}{\rho} \cdot \text{OPT}(X_{\text{rel}})$.
5. Wegen der Superoptimalität ist $A(I) \geq \frac{1}{\rho} \cdot \text{OPT}(I)$.

Def. 7.1: Sei Π ein kombinatorisches Optimierungsproblem. Für $I \in \mathcal{D}$ sei X ein approximiertes ganzzahliges lineares Programm und sei X_{rel} das zugehörige lineare Programm. Dann ist $\gamma = \max\left\{\frac{\text{OPT}(X_{\text{rel}})}{\text{OPT}(X)} | I \in \mathcal{D}\right\}$ die Ganzzahligkeitslücke (engl. Integrality Gap) der Relaxierung.

Sei I eine Instanz mit $\gamma = \frac{\mathrm{OPT}(X_{\mathrm{rel}})}{\mathrm{OPT}(X)}$. Nach 4:

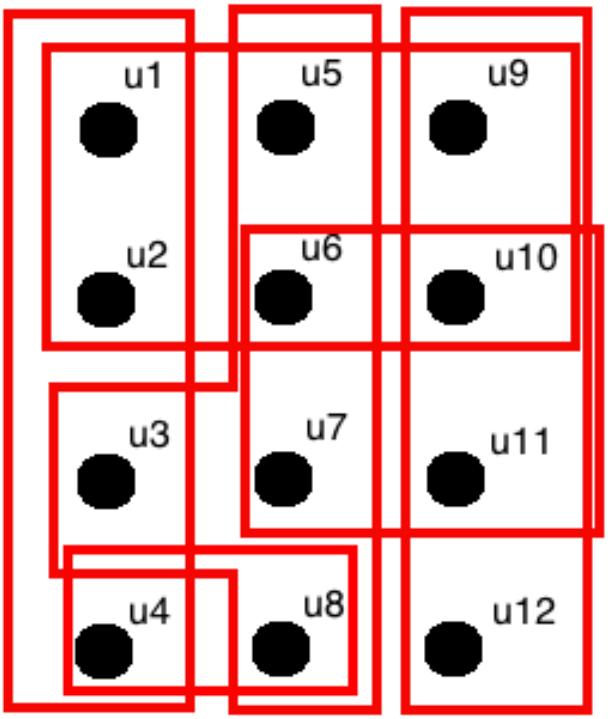
$$\underline{A(I) \geq \frac{1}{\rho} \cdot \mathrm{OPT}(X_{\mathrm{rel}}) = \frac{1}{\rho} \cdot \frac{\mathrm{OPT}(X_{\mathrm{rel}})}{\mathrm{OPT}(X)} \cdot \mathrm{OPT}(X) = \frac{\gamma}{\rho} \cdot \mathrm{OPT}(X)}$$

$$\Rightarrow (\text{da Maximierungsproblem}) \ \frac{\gamma}{\rho} \leq 1 \Rightarrow \gamma \leq \rho.$$

7.2. Das SETCOVER-Problem und seine Arithmetisierung

Def. 7.2: Die Instanz von SETCOVER ist eine Sammlung $S = \{S_1, ..., S_n\}$ verschiedener endlicher Mengen von Objekten. Die Mengen werden auch Gruppen genannt. Sei $V = S_1 \cup ... \cup S_m$ die Menge der Objekte. Eine Teilsammlung $S_{\mathrm{cov}} = \{S_{i_1}, ..., S_{i_l}\}$ ist eine Überdeckung von V, wenn $V = S_{i_1} \cup ... \cup S_{i_l}$. l ist die Größe der Überdeckung. Die Aufgabe von SETCOVER: Finde möglichst kleine Überdeckung.

Bsp. 7.3: $V = \{u_1, ..., u_{12}\}$, $n = 12$.
S besteht aus $m = 6$ Gruppen: $S_1 = \{u_1, u_2, u_5, u_6, u_9, u_{10}\}$, $S_2 = \{u_6, u_7, u_{10}, u_{11}\}$, $S_3 = \{u_1, u_2, u_3, u_4\}$, $S_4 = \{u_3, u_5, u_6, u_7, u_8\}$, $S_5 = \{u_9, u_{10}, u_{11}, u_{12}\}$, $S_6 = \{u_4, u_8\}$.



$S_{\mathrm{cov}} = \{S_3, S_4, S_5\}$ ist eine Überdeckung von S und sie ist auch minimal.

- $G_S = \max\{|S_i| \, | \, 1 \leq i \leq m\}$ ist die Mächtigkeit der größten Gruppe. Hier ist $G_S = 6$.
- Für das Objekt $u \in V$ ist $\deg_S(u) = |\{S_i \, | \, u \in S_i \in S\}|$ der Grad von u. $\Delta_S = \max\{\deg_S(u) \, | \, u \in V\}$ der Grad von S. Hier $\Delta_S = 3$.

Für jede Gruppe S_i definieren wir eine $0-1$–Variable x_i, $x_i = 1 \Leftrightarrow S_i \in S_{\mathrm{cov}}$. Ein Objekt u ist überdeckt, falls mindestens eine Gruppe, die u enthält, in S_{cov} ist.

$$\begin{aligned} \min \ & \sum_{i=1}^{m} x_i \\ s.t. \ & \sum_{i:u \in S_i} x_i \geq 1, u \in V \\ & x_i \in \{0,1\} \ \forall i = 1, ..., m \end{aligned}$$

In unserem Beispiel:

$$\begin{aligned} \min\ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ s.t.\ & x_1 + x_3 \geq 1 \\ & x_1 + x_3 \geq 1 \\ & x_3 + x_4 \geq 1 \\ & \ldots \\ & x_1 + x_2 + x_3 + x_4 \geq 1 \\ & \ldots \\ & x_1, \ldots, x_6 \in [0,1] \end{aligned}$$

Satz 7.4: Für die Ganzzahligkeitslücke γ der Relaxierung gilt: $\gamma \geq \frac{1}{2} \cdot \log n$.

ILP X für SetCover:

$$\begin{aligned} \min\ & \sum_{i=1}^{m} x_i \\ s.t.\ & \sum_{i, u \in S_i} x_i \geq 1 \text{ für alle } u \in V \\ & x_i \in \{0,1\} \text{ für alle } i \in \{1, \ldots, m\} \\ & 0 \leq x_i \leq 1 \end{aligned}$$

7.3. Deterministisches Runden

$\sum_{i=1}^{m} x_i = \mathrm{OPT}(X_{\mathrm{rel}}) \leq \mathrm{OPT}(S)$

Algorithmus DetRoundingSC; Δ_S ist der Grad der Eingabe
löse das relaxierte LP X_{rel} für SetCover;
$S_{\mathrm{cov}} = \varnothing$;
for $i := 1$ to m do
  if $v_i \geq \frac{1}{\Delta_S}$ then $S_{\mathrm{cov}} = S_{\mathrm{cov}} \cup S_i$
gib S_{cov} aus

Satz 7.5: Sei S eine Instanz von SetCover. Dann berechnet DetRoundingSC in polynomieller Zeit eine Überdeckung und es ist $\mathrm{DetRoundingSC}(S) \leq \Delta_S \cdot \mathrm{OPT}(S)$

Beweis: Da für jedes Objekt u in der zugehörigen Nebenbedingung mindestens ein x_i mindestens $\frac{1}{\Delta_S}$ ist, wird zu u mindestens eine Gruppe, die u enthält, in die Ausgabe übernommen; also ist jedes u überdeckt und S_{cov} ist tatsächlich eine Überdeckung.
$\mathrm{DetRoundSC}(S) = |S_{\mathrm{cov}}| \leq \sum_{i=1}^{m} \Delta_S \cdot x_i = \Delta_S \cdot \sum_{i=1}^{m} x_i = \Delta_S \cdot \mathrm{OPT}(X_{\mathrm{rel}}) \leq \Delta_S \cdot \mathrm{OPT}(S)$ □

7.4. Randomized Rounding: Von Monte Carlo nach Las Vegas

Algorithmus RandRoundingSC[r]
löse das relaxierte LP X_{rel} für SetCover
$X = \varnothing$;
for $i := 1$ to m do
  mit Wahrscheinlichkeit $1 - e^{-r \cdot x_i}$: $X = X \cup \{S_i\}$
gib X aus.

$\Pr[S_i \in X] = e^{-r \cdot x_i}$

Satz 7.6: Sei S eine Instanz von SetCover. Sei X die Ausgabe von RandRoundingSC[r]. Dann gilt:

a) $\Pr[X \text{ ist eine Überdeckung}] \geq 1 - n \cdot e^{-r}$
b) $E[|X\|] \leq r \cdot \mathrm{OPT}(S)$

Beweis:

a) Sei $u \in V$ ein beliebiges Objekt

$$\Pr[u \notin V(X)] = \Pr[\forall i \text{ mit } u \in S_i : S_i \notin X] = \prod_{i:u \in S(i)} e^{-r \cdot x_i} = \left(\prod_{i:u \in S(i)} e^{-\cdot x_i} \right)^r = \left(e^{-\sum_{i:u \in S(i)} x_i} \right)^r \leq e^{-r}$$

$$\Pr\left[X \text{ ist keine Überdeckung}\right] = \Pr[\exists u \in V_i : u \notin V(X)]$$

b) Für $|X|$ haben wir:

$$E[|X|] = \sum_{i=1}^{m} \Pr[S_i \in X] = \sum_{i=1}^{m} (1 - e^{-r \cdot x_i})$$

$$e^{-r \cdot x} = \sum_{i=0}^{\infty} \frac{(-r \cdot x)^i}{i!} \geq 1 - r \cdot x \Rightarrow E[|X|] \leq r \cdot \sum_{i=1}^{m} x_i = r \cdot \mathrm{OPT}(X_{\mathrm{rel}}) \leq r \cdot \mathrm{OPT}(S). \qquad \square$$

Algorithmus LasVegasSetCover$[r]$
löse das relaxierte LP X_{rel} für SetCover
$\tau := 0$;
repeat $\tau := \tau + 1$
    $X :=$ RandRoundSC$[r](S)$
until $V(X) = V$
$S_{\mathrm{cov}} := X$
gib S_{cov} aus

Satz 7.7: Sei S eine Eingabe von SetCover und sei $r > \ln n$. Für LasVegasSetCover$[r]$ gilt:

a) S_{cov} ist eine Überdeckung mit erwarteter Größe höchstens $r \cdot \mathrm{OPT}(S)$.
b) Die erwartete Anzahl der Durchgänge durch die repeat-Schleife ist höchstens $\frac{e^{2 \cdot r}}{(n - e^r)^2}$.

Beweis:

a) wegen Satz 7.6
b)

$$E[\tau] = \sum_{t=1}^{\infty} t \cdot \Pr\left[\text{erst die } t-\text{te Wiederholung ist eine Überdeckung}\right]$$

$$\leq \sum_{t=1}^{\infty} t \cdot \Pr\left[t-1 \text{ Wiederholungen liefern keine Überdeckung}\right] = \sum_{t=1}^{\infty} t \cdot (n \cdot e^{-r})^{t-1} = \frac{e^{2 \cdot r}}{(n - e^r)^2}.$$

Die Reihe konvergiert nur, wenn $n \cdot e^{-r} < 1$, also $r > \ln n$.

Korollar 7.8: LasVegasSetCover$[\ln n + 1]$ garantiert eine erwartete relative Güte von $\ln n + 1$. Der Erwartungswert für die Anzahl der Durchgänge durch die repeat-Schleife ist $\leq \frac{e^2}{(e-1)^2} < 2,503$.