Approximationsalgorithmen

Vorlesungsskript Sommersemester 2016
Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik, Lehrstuhl 12 (Hardware Software Co-Design)

Aufbauend auf einer Mitschrift der Vorlesung von Prof. Dr. Rolf Wanka im SS 2016 von

Thilo Kratzer (thilo.kratzer@fau.de)

Disclaimer Es werden keine Garantien betreffs der Richtigkeit des Materials und der Präsentationsqualität übernommen; es handelt sich i.w. um eine durch den Veranstalter nicht angepasste studentische Mitschrift. Die Vorlesung basiert auf dem Buch „R. Wanka. Approximationsalgorithmen - Eine Einführung, Teubner, 2006.“ [doi: 10.1007/978-3-8351-9067-2]“.

Inhaltsverzeichnis

Informationen bzgl. Ablauf der Veranstaltung:

Übung: 12:30 Uhr bis 14:00 Uhr (Raum E1.11)

Vorlesung: 14:15 Uhr bis 15:45 Uhr (Raum E1.11)

1 Schnelle Algorithmen und hartnäckige Probleme

1.1 Kombinatorische Optimierungsprobleme

Wir betrachten Eingabe I und haben etwas mit dieser algorithmisch zu tun. Wenn die Aufgabe durch den Algorithmus in Zeit $\mathcal{O}(poly(|I|))$ gelöst wird, ist der Algorithmus schnell. Für NP-vollständige[1] Probleme sind bis heute keine schnellen Algorithmen bekannt.

Beispiel.

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

sind beide NP-vollständig.

Definition 1.1 (Kombinatorisches Optimierungsproblem)**.** Ein kombinatorisches Optimierungsproblem Π ist charakterisiert durch vier Komponenten:

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

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

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

$f(\sigma)$ ist dabei der Wert der zulässigen Lösung σ. Wir schreiben $\text{OPT}(I) = f(\sigma_{\text{OPT}})$.

Beispiel.

(a) Das (volle) Traveling Salesperson Problem (TSP):

- $\mathcal{D} = \{\langle K_n, c\rangle \mid K_n$ ist vollst. Graph auf n Knoten, $c : E \to \mathbb{N}$ Kantengewichtung$\}$
- $\mathcal{S}(\langle K_n, c\rangle) = \{C \mid C = (v_{i_1}, ..., v_{i_n}, v_{i_1})$ ist ein Hamilton-Kreis[2]$\}$

  (jede Permutation der Knoten ist eine zulässige Lösung)
- $f(C) = c(v_{i_n}, v_{i_1}) + \sum_{j=1}^{n-1} c(v_{ij}, v_{ij+1})$
- min

(b) Das Rucksackproblem (RUCKSACK) ist charakterisiert durch:

- $\mathcal{D} = \{\langle W, vol, p, B\rangle \mid W = \{1, ..., n\}$, $vol : W \to \mathbb{N}$, $p : W \to \mathbb{N}$, $B \in \mathbb{N}$ und $\forall w \in W : vol(w) \subseteq B\}$

  (Warenangebot W, vol die Volumen der Waren, p ihre Preise und Rucksack-Volumen B)
- $\mathcal{S}(\langle W, vol, p, B\rangle) = \{A \subseteq W \mid \sum_{w \in A} vol(w) \leq B\}$
- $f(A) = \sum_{w \in A} p(w)$
- max

Bester Algorithmus fürs TSP bis heute hat Laufzeit $\mathcal{O}(n^2 \cdot 2^n)$. Der schnellste Algorithmus für RUCKSACK hat ebenfalls exponentielle Laufzeit.

[1] L NP-vollständig $\Leftrightarrow L \in$ NP und L NP-schwer, dh. $\forall L' \in$ NP gilt $L' \leq_p L$, also polynomiell reduzierbar auf L

[2] Kein Knoten doppelt

1.2 Approximationsalgorithmen: Schnell, aber nicht optimal

Definition 1.2 (Approximationsalgorithmus[3])**.** Sei Π ein komb. Optimierungsproblem. Ein $t(n)$-Zeit-Approximationsalgorithmus A berechnet zu $I \in \mathcal{D}$ in Zeit $t(|I|)$ eine Ausgabe σ_I^A. Wir schreiben $A(I) = f(\sigma_I^A)$[4]. Wir betrachten $t(n) = poly(n)$.

21. April 2016

Wir fordern erst einmal: $t(n)$ ist ein Polynom. Wir wollen außerdem den Unterschied zwischen $A(I)$ und $\textsc{Opt}(I)$ quantifizieren. Quantifizierung durch obere Schranke

- $A(I)$ weicht höchstens „so und so“ von $\textsc{Opt}(I)$ ab

und durch untere Schranken

- Es gibt I, sodass $A(I)$ „so und so“ von $\textsc{Opt}(I)$ tatsächlich abweicht (algorithmusbezogene untere Schranke)
- Es gibt keinen Approximationsalgorithmus, der das Problem Π „so und so gut“ approximiert (problembezogene untere Schranke)

2 Approximation mit absoluter Gütegarantie

Definition 2.1 (Absolute Güte)**.** Π wie gehabt und sei A ein Approximationsalgorithmus für Π.

(a) A hat bei Eingabe I eine absolute Güte[5] von $\kappa_A(I) = |A(I) - \textsc{Opt}(I)|$

(b) Die absolute worst-case-Güte von A ist die Funktion $\kappa_A^{\text{wc}}(n) = \max\{\kappa_A(I) \mid I \in \mathcal{D} \text{ mit } |I| \leq n\}$

(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: $\kappa_A^{\text{wc}}(n) \leq \kappa_A(n)$

2.1 Graphfärbbarkeit

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

$$\triangle(G) = \max\{\deg_G(u) \mid u \in V\}.$$

Definition 2.2 (Graphfärbungen)**.** 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 der benutzten Farben.

Definition 2.3 (Färbungsproblem)**.** Das Knotenfärbungsproblem[6] $\textsc{Col}$ ist charakterisiert durch:

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

[3] Die eigentlich „dumme“ Definition von Approximationsalgorithmus
[4] $A(I)$ ist der Wert der zurückgegebenen Lösung, nicht die Lösung selbst!
[5] auch „individuelle absolute Güte“
[6] analog dazu: Kantenfärbungsproblem

2.1.1 Knotenfärbungen

Knotenfärbungen beliebiger Graphen (mit mindestens einer Kante):

Algorithm 2.1 GreedyCol

```
1: for i = 1 to n do c_V(u) := ∞
2: end for.
3: for i = 1 to n do
4:     c_V(u_i) := min(ℕ \ {c_V(Γ(u_i))})
5: end for.
6: return c_V
```

Satz 2.4. GreedyCol garantiert eine absolute Güte[7] von

$$\kappa_{\text{GreedyCol}}(G) = \text{GreedyCol}(G) - \text{Opt}(G) \leq \triangle(G) + 1 - 2 = \triangle(G) - 1$$

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

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

28. April 2016

2.2 Ein Unmöglichkeitsergebnis für das Rucksackproblem

Satz 2.6. Falls $P \neq NP$, gibt es keine Konstante $k \in \mathbb{N}$, so dass es einen Polynomzeit-Approximations-Algorithmus A für das Rucksackproblem gibt mit

$$|A(I) - \text{Opt}(I)| \leq k.$$

Beweis. Indirekter Beweis, d. h. wir nehmen an, dass A und k doch existieren:
Sei $I = \langle W, vol, p, B\rangle$ eine Instanz von Rucksack (siehe Beispiel 1.3 (b)). Bestimme $I' = \langle W', vol', p', B'\rangle$ mit $W' = W$, $vol' = vol$, $B' = B$ und $p'(w) = (k+1) \cdot p(w)$. Es gilt: $S(I') = S(I)$, da $vol' = vol$.
Sei $\sigma \in S(I') = S(I)$.

$$f_I(\sigma) = \sum_{w \in \sigma} p(w)$$
$$f_{I'}(\sigma) = \sum_{w \in \sigma} (k+1) \cdot p(w) = (k+1) \cdot f_I(\sigma)$$

Eine i-tbeste Lösung zu I ist auch eine i-tbeste Lösung zu I', da durch die monotone Operation der Multiplikation keine Lösung eine andere überholt. Sei σ_1 eine optimale Lösung zu I_1 und σ_2 eine nichtoptimale, zweitbeste Lösung zu I. Dann ist $f_I(\sigma_1) - f_I(\sigma_2) \geq 1$ und $f_{I'}(\sigma_1) - f_{I'}(\sigma_2) \geq k+1$.
Algorithmus zur exakten Lösung von I:

1. berechne I' (beachte: Laufzeit der Multiplikation)
2. bestimme mittels A eine Lösung σ von I'
3. gib σ aus

Dieser Algorithmus läuft in Polynomzeit und kann nur eine optimale Lösung ausgeben, da es in I' keine zweitbeste Lösung mit Abstand $\leq k$ geben kann.
$\Rightarrow P = NP$, also Widerspruch zur Annahme[8] □

[7] Es gibt Graphen, die mindestens $\triangle(G) + 1$ Farben zur Kanten- bzw. Knotenfärbung brauchen.
[8] So etwas nennt man „Selbstreduktion“

3 Approximation mit relativer Gütegarantie

Definition 3.1 (Relative Güte)**.** Sei Π ein kombinatorisches Optimierungsproblem und A ein Approximationsalgorithmus für Π.

(a) A hat bei Eingabe I eine relative Güte von $\varrho_A(I) = \max\{\frac{A(I)}{\text{OPT}}, \frac{\text{OPT}}{A(I)}\} \quad (\geq 1)$

(b) Die relative Worst-Case-Güte von A ist $\varrho_A^{\text{wc}}(n) = \max\{\varrho_A(I) \mid I \in \mathcal{D}, |I| \leq n\}$

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

(d) A macht bei Eingabe I einen relativen Fehler von $\varepsilon_A(I) = \frac{|A(I) - \text{OPT}|}{\text{OPT}(I)} = |\frac{A(I)}{\text{OPT}(I)} - 1|$

ε kann festgesetzt werden wie ϱ.[9]

3.1 Das metrische TSP

$\mathcal{D} = \{\langle K_n, c\rangle \mid K_n$ ist vollständiger Graph auf n Knoten, $c : E \to \mathbb{N}$ Kostenfunktion und $\forall u, v, w \in V : c(u, v) \leq c(u, w) + c(w, v)$[10]$\}$

Das metrische TSP ($\triangle$TSP) ist das volle TSP wie bereits definiert, wobei als Eingabemenge $\mathcal{D}$ die obige Menge benutzt wird.

3.1.1 Christofides' Algorithmus

- Ein Multigraph ist ein Graph, in dem mehr als eine Kante zwischen zwei Knoten vorkommen können.
- Eulertour: Rundreise, die jede Kante genau einmal benutzt[11]. Eulertouren können in Polynomzeit berechnet werden!
- Matching: Teilgraph mit maximalem Grad 1. Hat im Matching jeder Knoten den Grad 1, handelt es sich um perfektes Matching. Im vollständigen Graphen mit Kantengewichten kann ein perfektes leichtestes Matching in Polynomzeit berechnet werden.

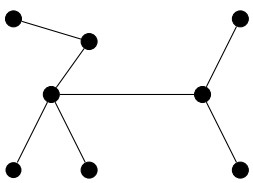
12. Mai 2016

Algorithm 3.1 Christofides' Algorithmus CH (1976)

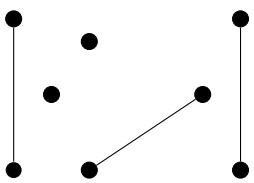
1: berechne einen minimalen Spannbaum T_{CH} von I
2: $S := \{v \in T_{\text{CH}} \mid \deg_{T_{\text{CH}}}(v)$ ist ungerade$\}$; Es gilt: $|S|$ ist gerade
3: berechne auf dem durch S induzierten Teilgraphen von K_n ein leichtestes Matching M_{CH}
4: berechne eine Euler-Tour $E = (u_1, u_2, ..., u_1)$ auf $T_{\text{CH}} \cup M_{\text{CH}}$ (alle Knoten haben geraden Grad)
5: entferne in E Wiederholungen von Knoten bis auf den letzten u_1, so dass man E' erhält
6: gib E' aus.

Beispiel (zu CH)**.**

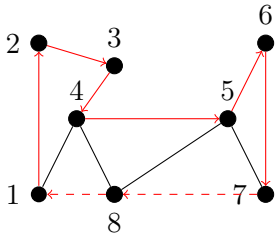
T_{CH}:



M_{CH}:



$T_{\text{CH}} \cup M_{\text{CH}}$ und E':



$\Rightarrow$ alle Knoten in $T_{\text{CH}} \cup M_{\text{CH}}$ haben geraden Grad

[9] Bei Minimierungsproblem beliebig groß, bei Maximierungsproblem nicht einmal 1
[10] Dreiecksungleichung
[11] z. B. das „Haus vom Nikolaus"

Satz 3.2. CH, gestartet mit Eingabe auf n Knoten, garantiert eine relative Güte von $\varrho_{\text{CH}} \leq \frac{3}{2} - \frac{1}{n}$.

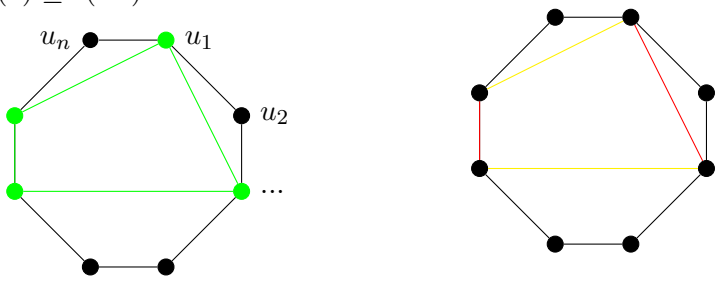
Beweis. Sei R^* eine optimale Rundreise zu $I = \langle K_n, c \rangle$, d. h. $c(R^*) = \text{OPT}(I)$.
Zu zeigen: $\text{CH}(I) \leq (\frac{3}{2} - \frac{1}{n}) \cdot \text{OPT}(I)$.

(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 $c(\hat{e}) \geq \frac{c(R^*)}{n}$[12].
Da T_{CH} minimaler Spannbaum ist, gilt:

$$c(T_{\text{CH}}) \leq c(R^*) - c(\hat{e}) \leq c(R^*) - \frac{c(R^*)}{n} = (1 - \tfrac{1}{n}) \cdot \text{OPT}(I)$$

(2) $|S|$ ist gerade: einfache Induktion

(3) Graph R^*: $c(\bullet) \leq c(R^*)$[13]



$S = \bullet$ kann man in zwei perfekte Matchings $\bullet$ und $\bullet$ zerlegt werden, da $|S|$ gerade, und $c(\bullet) + c(\bullet) = c(\bullet) \Rightarrow \min\{c(\bullet), c(\bullet)\} \leq \frac{1}{2} \cdot c(\bullet)$. Wegen der Minimalität von M_{CH} folgt:

$$c(M_{\text{CH}}) \leq \min\{c(\bullet), c(\bullet)\} \leq \tfrac{1}{2} \cdot c(\bullet) \leq \tfrac{1}{2} \cdot c(R^*) = \tfrac{1}{2} \cdot \text{OPT}(I)$$

(4) $c(E) = c(T_{\text{CH}}) + c(M_{\text{CH}}) \leq (1 - \frac{1}{n}) \cdot \text{OPT}(I) + \frac{1}{2} \cdot \text{OPT}(I) = (\frac{3}{2} - \frac{1}{n}) \cdot \text{OPT}(I)$

(5) Streichen von doppelten Kanten in E benutzt Abkürzungen[13] in K_n, also ist

$$c(E') \leq c(E) \leq (\tfrac{3}{2} - \tfrac{1}{n}) \cdot \text{OPT}(I)$$

□

Laufzeit ist polynomiell in $|I|$, wird jedoch dominiert durch die Berechnung des leichtesten Matchings.

Beispiel.
Siehe „R. Wanka. Approximationsalgorithmen - Eine Einführung, Teubner, 2006.“, Seite 54 f.

3.2 Unabhängige Mengen und noch einmal Knotenfärbungen

Definition 3.3 (Das Independent Set Problem)**.** 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.

Algorithm 3.2 GREEDYIS

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: $\quad u_t :=$ ein Knoten in $G^{(t)}$ mit minimalem Grad;
5: $\quad V^{(t+1)} := V^{(t)} \setminus (\{u_t\} \cup \Gamma_{G^{(t)}}(u_t))$; ▷ $\Gamma_{G^{(t)}}(u_t)$: alle Nachbarn von u_t
6: $\quad U := U \cup \{u_t\}$;
7: $\quad t := t + 1$; ▷ t entspricht der chromatischen Zahl χ
8: **done.**
9: gib U aus;

[12] $\forall e : c(R^* \setminus e) \geq c(T_{\text{CH}})$
[13] Anwendung der Dreiecksungleichung

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[14].

Definition 3.4. Chromatische Zahl: $\text{Opt}(G)$, wenn das Knotenfärbungsproblem betrachtet wird.

k sei die chromatische Zahl des Graphen G (jetzt wieder bei IS)

Satz 3.5. Sei $G = (V, E)$ ein Knoten-k-färbbarer Graph. Dann ist $\text{GreedyIS}(G) \geq \lceil \log_k(\frac{|V|}{3}) \rceil$.

Beweis.

Lemma 3.6. Sei $G = (V, E)$ ein Knoten-k-färbbarer Graph. Dann gibt es mindestens ein $u \in V$ mit $\deg_G(u) \leq \lfloor (1 - \frac{1}{k}) \cdot |V| \rfloor$.

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 mit $U_1 \cup U_2 \cup ... \cup U_k = V$. Unter den Mengen $U_1, U_2, ..., U_k$ muss es mindestens eine geben, die mindestens durchschnittlich groß ist. Die durchschnittliche Größe ist $\frac{|V|}{k}$, also $|U| \geq \lceil \frac{|V|}{k} \rceil$. Jeder Knoten in U kann nur mit höchstens allen außerhalb von U liegenden Knoten verbunden sein. Dies sind $|V| - |U| \leq |V| - \lceil \frac{|V|}{k} \rceil = \lfloor (1 - \frac{1}{k}) \cdot |V| \rfloor$.
$\Rightarrow \deg_G(u) \leq \lfloor (1 - \frac{1}{k}) \cdot |V| \rfloor$. □

Sei $|V| = n$ und $|V^{(t)}| = n_t$ und sei $k \geq 2$.
Wegen Lemma 3.8 hat u_t höchstens $\lfloor (1 - \frac{1}{k}) \cdot n_t \rfloor$ Nachbarn.

$$n_0 = n$$

$$n_{t+1} \geq n_t - \lfloor (1 - \frac{1}{k}) \cdot n_t \rfloor - 1 \geq \frac{n_t}{k} - 1$$

und damit $n_t \geq \frac{n}{k^t} - \underbrace{\frac{k}{k-1}(1 - \frac{1}{k^t})}_{\leq 2, \text{ da } k \geq 2} \geq \frac{n}{k^t} - 2.$

Für jede Runde t, für die $n_t \geq 1$ garantiert werden kann, wird ein neuer Knoten nach U gelegt. Das gilt, solange $t \leq \log_k(\frac{n}{3})$ ist $\Rightarrow |U| \geq \lceil \log_k(\frac{n}{3}) \rceil$. □

Ein besserer Knotenfärbungsalgorithmus:

Algorithm 3.3 GreedyCol2

```
1: t := 0; V^(1) := V;
2: while V^(t) ≠ ∅ do
3:     G^(t) := der durch V^(t) induzierte Graph;
4:     U_t := GreedyIS(G^(t));
5:     färbe alle Knoten in U_t mit der Farbe t;
6:     V^(t+1) := V^(t) \ U_t;
7:     t := t + 1;
8: done.
9: gib die Färbung aus;
```

Satz 3.7. Sei $G = (V, E)$ ein Knoten-k-färbbarer Graph. GreedyCol2 gibt eine Färbung mit höchstens $\frac{3n}{\log_k(\frac{n}{16})}$ Farben aus und hat eine relative Gütegarantie von $\mathcal{O}(\frac{n}{\log n})$.

Beweis. $n_t := |V^{(t)}|$. Wegen Satz 3.7 ist $|U_t| \geq \log_k(\frac{n_t}{3})$

$$n_0 = n$$

$$n_{t+1} \leq n_t - \log_k(\frac{n_t}{3})$$

[14] Wichtig für Prüfung! Entspricht kleinstmöglicher Farbenzahl in Col.

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

$$\frac{\textsc{GreedyCol2}(G)}{\textsc{Opt}(G)} \leq \frac{\frac{3n}{\log_k(\frac{n}{3})}}{k} = \frac{3n}{\log(\frac{n}{16})} \cdot \frac{\log k}{k} = \mathcal{O}(\tfrac{n}{\log n})$$

□

19. Mai 2016

3.3 Ein Unmöglichkeitsergebnis für's TSP

Satz 3.8. Wenn es einen polynomiellen Approximationsalgorithmus A mit konstanter, relativer Gütegarantie r für das volle TSP gibt, dann ist $P = NP$.

Beweis. Anmerkung: So einen Algorithmus A gibt es. Ziel ist es, Hamilton mit Hilfe von A in Polynomzeit zu entscheiden.

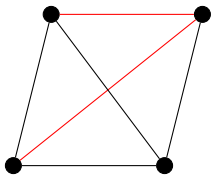
Eingabe: Graph $G = (V, E)$, Frage: hat G einen Hamilton-Kreis?

Mache G vollständig durch Hinzufügen der nicht vorhandenen Kanten und gebe den Kanten folgende Gewichte.

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

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

$$\underbrace{n-1}_{\text{1er-Kanten}} + \underbrace{(r-1) \cdot n + 2}_{\text{lange Kanten}} = r \cdot n + 1 > r \cdot n$$



Eingabe:

> G mit Hamilton-Kreis, dann modifiziert, dann dem Algorithmus A übergeben
> $\Rightarrow$ A muss eine Rundreise ausgeben, die keine langen Kanten enthält!
>
> G ohne Hamilton-Kreis, dann modifiziert, dann dem Algorithmus A übergeben
> $\Rightarrow$ A muss eine lange Kante mit in der Ausgabe haben!

$\Rightarrow$ Man kann Hamilton in Polynomzeit entscheiden[15]!

Algorithm 3.4 EntscheideHamilton

```
1: berechne G' und approximiere mit A eine kürzeste Rundreise in G'
2: if A(G') > r · |V| then
3:     gib aus: „G hat keinen Hamilton-Kreis“
4: else
5:     gib aus: „G hat einen Hamilton-Kreis“
6: end
```

□

[15] Algorithmus nicht Bestandteil der Vorlesung, aber zum besseren Verständnis aus Buch übernommen.

4 Approximationsschemata

Definition 4.1. Seit Π ein kombinatorisches Optimierungsproblem. Sei A ein Approximationsalgorithmus für Π, der als Eingabe eine Instanz $I \in \mathcal{D}$ bekommt und ein ε mit $0 < \varepsilon < 1$ bekommt.

(a) A heißt polynomielles Approximationsschema PAS[16] für Π, wenn A zu jeder Instanz I und zu jedem $\varepsilon \in]0,1[$ in Zeit $\mathcal{O}(poly(|I|))$ eine zulässige Lösung mit relativem Fehler $\varepsilon_A(I,\varepsilon) \leq \varepsilon$ berechnet.[17]

(b) A heißt streng polynomielles Approxiamationsschema FPAS[18], wenn A ein PAS mit Laufzeit $\mathcal{O}(\ poly(|I|, \frac{1}{\varepsilon})\ )$ ist.[19]

Satz 4.2. Sei Π ein komb. Optimierungsproblem, A ein (F)PAS, und zu Eingabe I sei $Z(I)$ eine obere Schranke für $\textsc{Opt}(I)$, $\textsc{Opt}(I) \leq Z(I)$.
Sei $\varepsilon^* = \dfrac{1}{Z(I)+1}$. Dann ist $A(I,\varepsilon^*) = \textsc{Opt}(I)$. Ist A ein FPAS, dann ist die Laufzeit $\mathcal{O}(\ poly(|I|, Z(I))\ )$.

Beweis. Starte A mit I und ε^*. Für die gefundene zulässige Lösung gilt folgende Fehlerschranke:

$$\varepsilon_A(I,\varepsilon^*) = \frac{|\textsc{Opt}(I) - A(I,\varepsilon^*)|}{\textsc{Opt}(I)} \leq \varepsilon^*$$
$$\Rightarrow |\textsc{Opt}(I) - A(I,\varepsilon^*)| \leq \varepsilon^* \cdot \textsc{Opt}(I) = \frac{\textsc{Opt}(I)}{Z(I)+1} < 1$$

$\Rightarrow |\textsc{Opt}(I) - A(I,\varepsilon^*)| = 0$[20]
$\Rightarrow A(I,\varepsilon^*) = \textsc{Opt}(I)$ □

02. Juni 2016

4.1 Ein pseudopolynomieller exakter Algorithmus für das Rucksackproblem

- $W = \{1,\dots,n\}$, p_i: Preis der Ware i, $vol(i)$: Volumen der Ware i, B: Rucksackkapazität
- $P_{\max}$[21] $\leq \textsc{Opt}(I) \leq n \cdot P_{\max}$, da es n Waren gibt und jede in den Rucksack passt.
- Der exakte Algorithmus ist ein typischer Vertreter der dynamischen Programmierung
- $F_j(\alpha)$ für $\alpha \in \mathbb{Z}$ und $j \in \{0,1,2,\dots,n\}$ 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.
  $F_j(\alpha) = \min\{vol(R) \mid R \subseteq \{1,\dots,j\}, p(R) \geq \alpha\}$

Lemma 4.3.

(1) $\alpha \leq 0, j \in \{0,\dots,n\} : F_j(\alpha) = 0$

(2) $\alpha \geq 1, j = 0 : F_j(\alpha) = \infty$

(3) $\alpha \geq 1, j \in \{1,\dots,n\} : F_j(\alpha) = \min\{F_{j-1}(\alpha), F_{j-1}(\alpha - p_j) + vol(j)\}$

Zu (3) – *Ballmansche Optimalitätsgleichung*: Wenn das kleinste benötigte Rucksackvolumen mit den Waren $1,\dots,j-1$ für den Gesamtwert α berechnet ist und nun Ware j mit Volumen $vol(j)$ und Preis p_j dazukommt, ändert das entweder nichts, oder man kann eine Rucksackfüllung mit kleinerem Volumen erzeugen, indem man mit Waren $1,\dots,j-1$ den Wert $\alpha - p_j$ erzielt und Ware j mit in den Rucksack legt.

[16] manchmal auch PTAS: time
[17] zulässig z. B. Zeit $\mathcal{O}(n^{\frac{1}{\varepsilon}})$, lediglich $|I|$ ist nur in der Basis erlaubt!
[18] oder FPTAS mit fully (engl.)
[19] zulässig z. B. Zeit $\mathcal{O}(n^7 \cdot (\frac{1}{\varepsilon})^{13}))$, $|I|$ und $\frac{1}{\varepsilon}$ sind nur in der Basis erlaubt!
[20] da Differenz nat. Zahl sein muss!
[21] Preis der wertvollsten Ware

Was brauchen wir? Wir suchen $\max\{\alpha \mid F_n(\alpha) \leq B\}$ → Algorithmus DynRucksack

Algorithm 4.1 DynRucksack

1: $\alpha := 0$;
2: **do**
3: $\quad \alpha := \alpha + 1$
4: $\quad$ **for** j $= 1$ **to** n **do**
5: $\quad\quad F_j(\alpha) := \min\{F_{j-1}(\alpha), F_{j-1}(\alpha - p_j) + vol(j)\}$
6: $\quad$ **end for.**
7: **while** $B < F_n(\alpha)$
8: gib $\alpha - 1$ aus;

	0	1	...	j	...	n
0	0	0	...	0	...	0
1	∞					
2	∞					
⋮	⋮					
$\alpha - p_j$	⋮		•			
⋮	⋮					
α	∞		•	$F_j(\alpha)$		
⋮	⋮					
Opt(I)					•	
Opt$(I)+1$						

Satz 4.4. DynRucksack berechnet zu I den Wert Opt(I) in Zeit $\mathcal{O}(n \cdot \text{Opt}(I)) = \mathcal{O}(n^2 \cdot P_{\max})$.[22]

Definition 4.5. Sei Π ein kombinatorisches Optimierungsproblem, so dass für alle Instanzen I gilt, dass alle in I vorkommenden Zahlen natürliche Zahlen sind. Sei maxnr(I) die größte in I vorkommende Zahl. Ein Algorithmus für Π heißt pseudopolynomiell, falls es ein Polynom $poly(\cdot,\cdot)$ gibt, so dass die Laufzeit für alle I höchstens $poly(|I|, \text{maxnr}(I))$ 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 $\lfloor \frac{p_j}{k} \rfloor$ ersetzen. Diese Eingabe nennen wir I_{red}. das „richtige“ k wird von I und dem vorgegebenen ε abhängen.

Algorithm 4.2 AR_k

1: reduziere die Preise zu $\lfloor \frac{p_j}{k} \rfloor$
2: berechne mittels DynRucksack eine optimale Lösung R_k auf I_{red}
3: gibt R_k aus

Satz 4.6. AR_k macht bei Eingabe I einen relativen Fehler von $\varepsilon_{\text{AR}_k}(I) \leq \frac{k \cdot n}{P_{\max}}$ bei Laufzeit von $\mathcal{O}(n^2 \cdot \frac{P_{\max}}{k})$.

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

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

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

$$\Rightarrow |A_{R_k}(I) - \text{Opt}(I)| \leq k \cdot n \text{ und } \varepsilon_{\text{AR}_k}(I) \leq \frac{k \cdot n}{\text{Opt}(I)} \leq \frac{k \cdot n}{P_{\max}}$$

□

09. Juni 2016

Jetzt: Eingabe Instanz I und Fehlerschranke ε

[22] Exponentielle Laufzeit, da $P_{\max}$ zur Kodierung nur $\mathcal{O}(\log P_{\max})$ viele Bits benötigt, $P_{\max} = \mathcal{O}(2^{|I|})$!

Algorithm 4.3 FPASRucksack(I, ε)

1: bestimme aus I die Parameter n und $P_{\max}$
2: $k := \varepsilon \cdot \dfrac{P_{\max}}{n}$
3: starte AR_k mit I und gib dessen Lösung aus

Satz 4.7. FPASRucksack ist ein FPAS[23] für das Rucksackproblem mit Laufzeit $\mathcal{O}(n^3 \cdot \frac{1}{\varepsilon}))$.

Beweis.

$$\varepsilon_{\mathrm{AR}_k} \leq \frac{k \cdot n}{P_{\max}} = \frac{\varepsilon \cdot \frac{P_{\max}}{n} \cdot n}{P_{\max}} = \varepsilon \text{ und } \mathcal{O}(n^2 \cdot \frac{P_{\max}}{k}) = \mathcal{O}(n^2 \cdot \frac{P_{\max}}{\varepsilon \cdot \frac{P_{\max}}{n}}) = \mathcal{O}(n^3 \cdot \frac{1}{\varepsilon})$$

□

4.3 Unmöglichkeitsergebnisse für Approximationsschemata

Definition 4.8. Ein NP-vollständiges Entscheidungsproblem L heißt stark NP-vollständig, wenn es ein Polynom q gibt, so dass $L_q = \{x \mid x \in L \text{ und } \mathrm{maxnr}(x) \leq q(|x|)\}$ NP-vollständig ist. Gibt es kein solches Polynom, so ist L schwach NP-vollständig.

Beispiel.

- Hamilton und Clique$_{ent}$ 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 3.10 kann auch $r = 1$ eingesetzt werden. Dann sind die Kantenlängen alle nur 1 oder 2, d. h. $q(n) = n$ ist wieder das Polynom, so dass die Einschränkung NP-vollständig bleibt. Ja, sogar: wenn alle Kantenlängen nur 1 oder 2 sind, gilt im vollständigen Graphen die Dreiecksungleichung, also ist auch $\triangle$TSP stark NP-vollständig.
- Rucksack ist schwach NP-vollständig, da die Optimierungsvariante einen pseudopolynomiellen Algorithmus hat.

Satz 4.9. Sei Π ein kombinatorisches Optimierungsproblem. Wenn es ein Polynom $q(x_1, x_2)$ gibt, so dass für alle Instanzen I gilt, dass $\mathrm{Opt}(I) \leq q(|I|, \mathrm{maxnr}(I))$ ist, dann folgt aus der Existenz eines FPAS für Π, dass es einen pseudopolynomiellen Algorithmus für Π gibt.

Korollar 4.10. Wenn es für die Optimierungsvariante eines stark NP-vollständigen Problems ein FPAS gibt, ist $P \neq NP$.

6 Techniken für randomisierte Approximationsalgorithmen

6.1 Die probabilistische Methode

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

Definition 6.2. Eine Instanz Φ von MaxSAT ist eine Bool'sche (n, m)-Formel in KNF[24]. Eine zulässige Lösung zu Φ ist eine Belegung $b : V \to \{\mathrm{True}, \mathrm{False}\}$ der Variablen. Die Bool'schen Wahrheitswerte für Literale, Klauseln und Formeln ergeben sich kanonisch[25]. Die Bewertungsfunktion ist

$$\mathrm{wahr}(b, \Phi) = |\{j \mid C_j \in \Phi, b(C_j) = \mathrm{True}\}|$$

[23] Algorithmus A ist FPAS $\Leftrightarrow$ A ist PAS ($\Leftrightarrow \varepsilon_{\mathrm{A}}(I, \varepsilon) \leq \varepsilon$) und hat Laufzeit $\mathcal{O}(poly(|I|, \frac{1}{\varepsilon})$.
[24] n ist die Anzahl der Variablen, m ist die Anzahl der Klauseln
[25] wissenschaftlich für „so wie man es denkt“

wahr(b, Φ) ist die Anzahl der Klauseln, die unter b erfüllt werden.
Ziel: finde b^* mit wahr(b^*, Φ) ist maximal.

Beispiel.
$\Phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$ mit $\text{Opt}(\Phi) = 3$
Wir zeigen: $\text{Opt}(\Phi) \geq \frac{m}{2}$ für alle Φ.

Satz 6.3. Sei Φ eine Bool'sche (n, m)-Formel in KNF. Dann ist

$$\max\{ \begin{array}{c} \text{wahr}((\text{False}, \text{False}, \ldots, \text{False}), \Phi) \\ \text{wahr}((\text{True}, \text{True}, \ldots, \text{True}), \Phi) \end{array} \} \geq \frac{m}{2}$$

Beweis. Da $\text{Opt}(\Phi) \leq m$, liefert dies bereits ein Approximationsverfahren der relativen Güte 2. □

16. Juni 2016

Algorithm 6.1 A

1: **for** i = 1 **to** n **do**
2: $\begin{cases} \text{mit Wahrscheinlichkeit } \frac{1}{2}: & x_i \leftarrow \text{True} \\ \text{mit Wahrscheinlichkeit } \frac{1}{2}: & x_i \leftarrow \text{False} \end{cases}$
3: **end for.**
4: gib die Belegung $b_A = (x_1, \ldots, x_n)$ aus

$\Pr[x_i = \text{True}] = \Pr[x_i = \text{False}] = \frac{1}{2}$[26]

Lemma 6.4. Sei k_j die Anzahl der Literale in Klausel C_j. Es gilt:

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

Beweis. Die Wahrscheinlichkeit, dass ein einzelnes Literal zu False wird, ist $\frac{1}{2}$. Wahrscheinlichkeit, dass alle k_j Literale zu False werden, ist $(\frac{1}{2})^{k_j}$ wegen der Unabhängigkeit.

Das heißt $\Pr[\text{alle Literale False}] = (\frac{1}{2})^{k_j}$
$\Rightarrow \Pr[\text{mind. 1 Literal True}] = 1 - (\frac{1}{2})^{k_j} = \Pr[C_j \text{ wird erfüllt}]$

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

Zufallsvariable 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 Bool'sche (n, m)-Formel Φ in KNF, in der jede Klausel mindestens k Literale hat, gilt:

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

Beweis. Für $j \in \{1, \ldots, m\}$ sei Z_j die Zufallsvariable[27] mit

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

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

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

Korollar 6.7. Algorithmus A hat für jede Bool'sche (n, m)-Formel Φ in KNF, in der jede Klausel mindestens die Länge k hat, eine erwartete relative Güte von

$$E[\varrho_{\text{A}}(\Phi)] = \frac{\text{Opt}(\Phi)}{E[\text{A}(\Phi)]} \leq \frac{m}{m \cdot (1 - \frac{1}{2^k})} \leq \frac{1}{1 - \frac{1}{2^k}}$$

[26] Diese Experimente sind unabhängig!
[27] Indikatorvariable, d. h. $\text{E}[Z_j] = 0 \cdot \Pr[b_{\text{A}}(C_j) = \text{False}] + 1 \cdot \Pr[b_{\text{A}}(C_j) = \text{True}] = \Pr[C_j \text{ wird durch A erfüllt}]$

6.2 Randomized Rounding

6.2.1 Arithmetisierung von MaxSAT

$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ ist Bool'sche (n, m)-Formel in KNF.

$$\begin{aligned} \textsc{False} &\equiv 0 \\ \textsc{True} &\equiv 1 \\ x_i \equiv \hat{x}_i &\in \{0,1\} \\ C_j \equiv \hat{Z}_j &\in \{0,1\} \end{aligned}$$

Anzahl der erfüllten Klauseln: $\sum_{j=1}^{m} \hat{Z}_j$

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

$S_j^{\ominus}$: Menge der Variablen, die in C_j negiert vorkommen.
$S_j^{\oplus}$: Menge der Variablen, die in C_j positiv vorkommen.

Ganzzahliges Lineares Programm B für Max-SAT

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

In C_j haben wir die Nebenbedingung: $(1 - \hat{x}_1) + x_3 + (1 - \hat{x}_7) + x_9 \geq \hat{Z}_j$

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}:

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

Das relaxierte Programm kann in Polynomzeit gelöst werden.

Der Lösungsraum wird größer, also kann sein

$$\textsc{Opt}(\Phi) = \textsc{Opt}(B) \leq \textsc{Opt}(B_{rel})$$

Beispiel. $\Phi = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_2)$ und $x_1 = \top, x_2 = \top$

$$\begin{aligned} \max \quad & \hat{Z}_1 + \hat{Z}_2 + \hat{Z}_3 + \hat{Z}_4 \\ & \hat{x}_1 + \hat{x}_2 \geq \hat{Z}_1 \\ & (1 - \hat{x}_1) + \hat{x}_2 \geq \hat{Z}_2 \\ & \hat{x}_1 + (1 - \hat{x}_2) \geq \hat{Z}_3 \\ & (1 - \hat{x}_1) + (1 - \hat{x}_2) \geq \hat{Z}_4 \\ & \hat{x}_i, \hat{Z}_j \in \{0,1\} \\ \Rightarrow \; & \textsc{Opt}(B) = 3 \end{aligned}$$

Setze $\hat{x}_1 = \hat{x}_2 = \frac{1}{2}$ und $0 \leq \hat{x}_i, \hat{Z}_j \leq 1 \Rightarrow \textsc{Opt}(B_{rel}) = 4$
$\Rightarrow \textsc{Opt}(B) \leq \textsc{Opt}(B_{rel})$[28]

23. Juni 2016

B_{rel} kann in Polynomzeit gelöst werden, aber die Lösung ist nicht unbedingt eine zulässige Lösung des ursprünglichen Problems.

[28] „Superoptimalität“ und $\frac{\textsc{Opt}(B_{rel})}{\textsc{Opt}(B)}$ heißt „Ganzzahligkeitslücke“

6.2.2 Von der rationalen zur ganzzahligen Lösung

Sei $\pi : [0,1] \to [0,1]$ eine stochastische Funktion. Hier: $\pi = id$, also $\pi(x) = x$.

Algorithm 6.2 RandomizedRounding$[\pi]$

1: **for** i = 1 **to** n **do**
2: $\begin{cases} \text{mit Wahrscheinlichkeit } \pi(\hat{x}_i): & x_i \leftarrow \text{True} \\ \text{mit Wahrscheinlichkeit } 1-\pi(\hat{x}_i): & x_i \leftarrow \text{False} \end{cases}$
3: **end for.**

$\Pr[x_i = \text{True}] = \hat{x}_i$ und $\Pr[x_i = \text{False}] = 1 - \hat{x}_i$

Algorithm 6.3 B

1: löse das relaxierte LP B_{rel} zu Φ
2: ermittle durch RandomizedRounding$[id]$ eine Belegung der (Bool'schen) Variablen

Lemma 6.8. Sei k_j die Anzahl der Literale in C_j.

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

Beweis.

$$C_j = \left[\bigvee_{x_i \in S_j^{\oplus}} x_i\right] \vee \left[\bigvee_{x_i \in S_j^{\ominus}} \overline{x}_i\right]$$

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

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

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

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

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

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

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

[29] [30]

[29] $\prod_{i=1}^{k} a_i \leq \left(\frac{1}{k}\sum_{i=1}^{k} a_i\right)^k \Leftrightarrow 1 - \prod_{i=1}^{k} a_i \geq 1 - \left(\frac{1}{k}\sum_{i=1}^{k} a_i\right)^k$

[30] 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 6.9. Für jede Bool'sche Formel Φ in KNF, in der jede Klausel höchstens k Literale enthält, ist

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

Beweis. Sei $\Phi = C_1 \vee \cdots \vee C_m$ und k die Länge der längsten Klausel.

$$\begin{aligned} E[\mathrm{B}(\Phi)] &= \sum_{i=1}^{m} \Pr[C_j \text{ wird durch B erfüllt}] \\ &\geq \sum_{j=1}^{n} \left(1 - \left(1 - \frac{1}{k_j}\right)_j^k\right) \cdot \hat{Z}_j \\ &\geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \textsc{Opt}(B_{rel}) \\ &\geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \textsc{Opt}(\Phi) \end{aligned}$$

□

Da $1 - (1 - \frac{1}{k})^k \geq 1 - \frac{1}{e}$ für alle $k \in \mathbb{N}$, erreicht Algorithmus B immer:

- mindestens $(1 - \frac{1}{e}) \cdot \textsc{Opt}(\Phi)$ erfüllte Klauseln[31]
- eine erwartete relative Güte von $E[\varrho_{\mathrm{B}}(\Phi)] \leq \dfrac{1}{1 - \frac{1}{e}} \approx 1,582$

6.3 Der hybride Ansatz

Max-EkSAT[32]: alle Klauseln haben genau die Länge k.

	A	B	
k	$1 - \frac{1}{2^k}$	$1 - (1 - \frac{1}{k})^k$	Erwartungswert
1	0.5	1	Worst-Case von A
2	0.75	0.75	gemeinsamer Worst-Case
3	0.875	0.704	
4	0.938	0.684	
5	0.969	0.672	
⋮	⋮	⋮	
∞	1	0.632	Worst-Case von B

30. Juni 2016

Algorithm 6.4 C_{p_A}

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

Algorithm 6.5 C_{alle}

1: führe beide Algorithmen aus und
2: gib das bessere Ergebnis aus

$E[\mathrm{C}_{alle}(\Phi)] \geq E[\mathrm{C}_{p_A}(\Phi)] = p_A \cdot E[\mathrm{A}(\Phi)] + (1 - p_A) \cdot E[\mathrm{B}(\Phi)]$
Für MaxSAT: $p_A = \frac{1}{2}$

[31] Eulersche Zahl $e \approx 2,7182$
[32] E für exakt

Satz 6.10. Algorithmus $\mathrm{C}_{\frac{1}{2}}$ hat erwartete relative Güte $\frac{4}{3}$

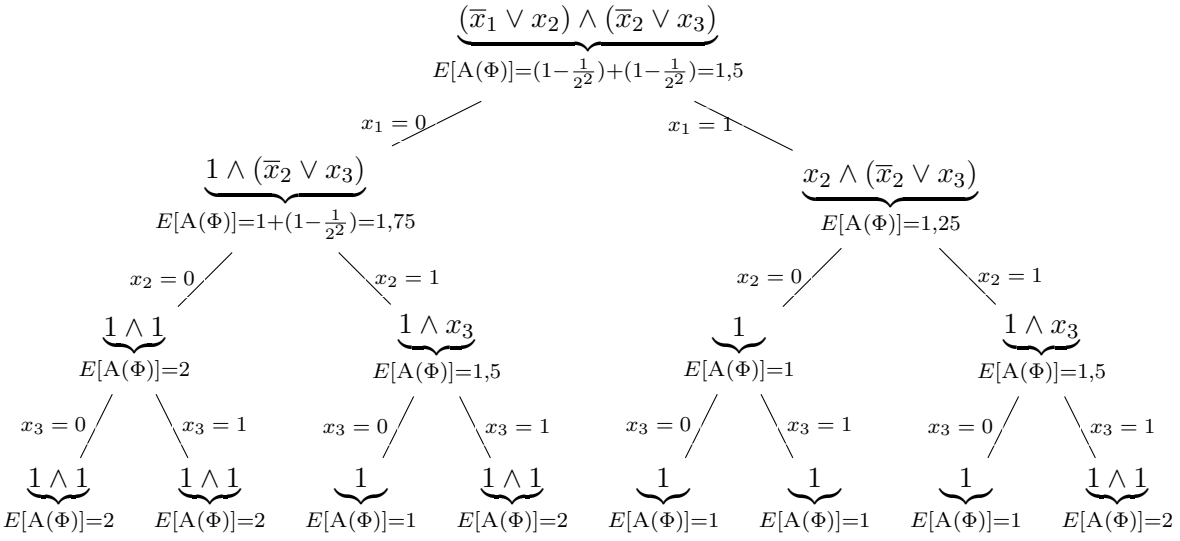
Beweis. Wir suchen $E[\mathrm{C}_{\frac{1}{2}}(\Phi)] = \frac{1}{2} \cdot E[\mathrm{A}(\Phi)] + \frac{1}{2} \cdot E[\mathrm{B}(\Phi)]$, die Lemmata 6.4 und 6.8 ergeben:

$$
\begin{aligned}
E[\mathrm{A}(\Phi)] &= \sum_{k=1}^{n} \sum_{\substack{C_j \text{ hat die} \\ \text{Länge } k}} \left(1 - \frac{1}{2^k}\right) \geq \sum \sum \left(1 - \frac{1}{2^k}\right) \cdot \hat{Z}_j \\
E[\mathrm{B}(\Phi)] &\geq \sum_{k=1}^{n} \sum_{\substack{C_j \text{ hat die} \\ \text{Länge } k}} \left(1 - (1 - \frac{1}{k})^k\right) \cdot \hat{Z}_j \\
E[\mathrm{C}(\Phi)] &\geq \sum_{k=1}^{n} \sum_{\substack{C_j \text{ hat die} \\ \text{Länge } k}} \tfrac{1}{2} \cdot \underbrace{\left(1 - \frac{1}{2^k} + 1 - (1 - \frac{1}{k})^k\right)}_{\geq \frac{3}{2} \text{ für } k \in \mathbb{N}^+} \cdot \hat{Z}_j \\
&\geq \frac{3}{4} \cdot \underbrace{\sum_{j=1}^{m} \hat{Z}_j}_{= \textsc{Opt}(B_{rel})} \\
&\geq \frac{3}{4} \cdot \textsc{Opt}(\Phi) \qquad \Rightarrow \varrho_{\mathrm{C}}(\Phi) = \frac{\textsc{Opt}(\Phi)}{E[\mathrm{C}(\Phi)]} \leq \frac{4}{3}
\end{aligned}
$$

□

6.4 Derandomisierung: Die Methode der bedingten Erwartungswerte

Beispiel.



7. Juli 2016

Wenn wir also einem Pfad zu Blatt β folgen, so dass von jedem Knoten u zu v mit $E[\mathrm{A}(u)] \leq E[\mathrm{A}(v)]$ weitergegangen wird, so heißt das, dass $\mathrm{A}(\beta) = E[\mathrm{A}(\beta)] \geq E[\mathrm{A}(I)]$ ist und die Kantenbeschriftung des Pfades einer Belegung der Variablen $x_1, \ldots, x_n$, also einer zulässigen Lösung zu I entspricht. Einen solchen Pfad liefert beispielsweise der folgende Algorithmus $\textsc{Derand}_A$:

Algorithm 6.6 $\textsc{Derand}_A$

```
1: for i = 1 to n do                          ▷ Anzahl der Literale x_{i∈[1,...,n]}
2:     w_0 = E[A(Φ) | x_1 ... x_{i-1} belegt und x_i = 0]
3:     w_1 = E[A(Φ) | x_1 ... x_{i-1} belegt und x_i = 1]
4:     if w_0 ≤ w_1 then
5:         x_i = 1
6:     else
7:         x_i = 0
8:     end
9: end for.
```

7 Lineare Optimierung und Approximationsalgorithmen

7.1 Die Ganzzahligkeitslücke und ihre Beziehung zur realtiven Güte

Sei Π ein kombinatorisches Maximierungsproblem und I eine Instanz. Der Rundungsansatz arbeitet wie folgt:

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

Bei Minimierungsproblem: $\textsc{Opt}(X) \geq \textsc{Opt}(X_{rel})$ und $A(I) \leq \varrho \cdot \textsc{Opt}(X_{rel})$.

Definition 7.1. Sei Π ein komb. Maximierungsproblem. Für $I \in \mathcal{D}$ sei X ein äquivalentes ganzzahliges lineares Programm und sei X_{rel} das relaxierte Programm. Dann ist

$$\gamma = \max\{\frac{\textsc{Opt}(X_{rel})}{\textsc{Opt}(X)} \mid I \in \mathcal{D}\}$$

die Ganzzahligkeitslücke (engl.: integrality gap) der Relaxierung. Ist Π ein Minimierungsproblem, so wird der Kehrwert genommen.

Beispiel.

MaxSAT: $\gamma = \frac{4}{3}$

VertexCover: $\gamma = 2 - \frac{2}{n}$

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

$$A(I) \geq \frac{1}{\varrho} \cdot \textsc{Opt}(X_{rel}) = \frac{1}{\varrho} \cdot \frac{\textsc{Opt}(X_{rel})}{\textsc{Opt}(X)} \cdot \textsc{Opt}(X) = \frac{\gamma}{\varrho} \cdot \textsc{Opt}(X)$$

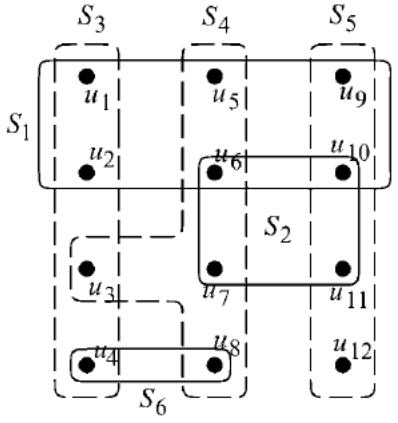
Da Π ein Maximierungsproblem ist, gilt $\frac{\gamma}{\varrho} \leq 1$, d. h. $\gamma \leq \varrho$.

7.2 SetCover und seine Arithmetisierung

Definition 7.2. Bei SetCover ist Sammlung $\mathcal{S} = \{S_1, \ldots, S_m\}$ verschiedener Mengen von Objekten gegeben. Die $\mathcal{S}_i$ heißen auch Gruppen.

$$V = \bigcup_{i=1}^{m} S_i = \{u_1, \ldots, u_n\}$$

Eine Teilsammlung $\mathcal{S}_{\text{Cov}} = \{S_{i_1}, \ldots, S_{i_l}\}$ von $\mathcal{S}$ heißt Überdeckung von V, falls $V = S_{i_1} \cup \cdots \cup S_{i_l}$. l ist die Größe der Überdeckung, Ziel ist es, l zu minimieren.



$\mathcal{S}_{\text{Cov}} = \{S_3, S_4, S_5\}$ ist eine minimale Knotenüberdeckung

- u_6 ist in S_1, S_2 und S_4, d. h. die NB ist $x_1 + x_2 + x_4 \geq 1$
- Nebenbedingung zu u_7: $x_2 + x_4 \geq 1$
- Nebenbedingung zu u_{12}: $x_5 \geq 1$

Ganzzahliges Lineares Programm X für SetCover:

$$\min \sum_{i=1}^{m} x_i$$
$$s.t. \sum_{i:u\in S_i} x_i \geq 1 \text{ für alle } u \in V$$

Für jedes S_i haben wir $x_i \in \{0,1\}$ mit $x_i = 1 \Leftrightarrow S_i$ in der Überdeckung.

Satz 7.3. Die Ganzzahligkeitslücke γ der Relaxierung X_{rel} ist mindestens $\frac{1}{2} \cdot \log n$.

- Für $u \in V$ ist $\deg_{\mathcal{S}}(u) = |\{S_i \mid u \in S_i \text{ mit } S_i \in \mathcal{S}\}|$ der Grad von u und $\triangle_{\mathcal{S}} = \max\{\deg_{\mathcal{S}}(u) \mid u \in V\}$ ist der Grad von $\mathcal{S}$.
- $\mathcal{G}_{\mathcal{S}} = \max\{|S_i| \mid 1 \leq i \leq n\}$ ist die Mächtigkeit der größten Gruppen.

7.3 Randomized Rounding: von Monte Carlo nach Las Vegas

Algorithm 7.1 RandomizedRoundingSC[r]

1: löse das relaxierte LP X_{rel} für SetCover
2: $\mathcal{X} := \emptyset$
3: **for** $i = 1$ **to** n **do**
4: mit Wahrscheinlichkeit $1 - e^{-r\cdot x_i}$: $\mathcal{X} = \mathcal{X} \cup \{S_i\}$ ▷ viele Einzelläufe mit W'keit x_i
5: **end for.**
6: gibt $\mathcal{X}$ aus

$\Pr[S_i \notin \mathcal{X}] = 1 - (1 - e^{-r\cdot x_i}) = e^{-r\cdot x_i}$

Satz 7.4. Sei $\mathcal{S}$ eine Eingabe von SetCover, $\mathcal{X}$ die Ausgabe von RandomizedRoundingSC[r]. Dann ist:

(a) $\Pr[\mathcal{X} \text{ ist eine Überdeckung}] \geq 1 - n \cdot e^{-r}$

(b) $E[|\mathcal{X}|] \leq r \cdot \text{Opt}(\mathcal{S})$

14. Juli 2016

Beweis.

(a) Sei $u \in V$ ein beliebiges Objekt und $V(\mathcal{X}) = \bigcup_{S_i \in \mathcal{X}} S_i$

$$\Pr[u \notin V(\mathcal{X})] = \Pr[\forall i \text{ mit } u \in S_i : S_i \notin \mathcal{X}] = \prod_{i:u\in S_i} \Pr[S_i \notin \mathcal{X}] = \prod_{i:u\in S_i} e^{-r\cdot x_i} =$$
$$\left(\prod_{i:u\in S_i} e^{-x_i}\right)^r = \left(e^{-\sum_{i:u\in S_i} x_i}\right)^r \leq \left(e^{-1}\right)^r = \left(\frac{1}{e}\right)^r = e^{-r}$$

$$\Rightarrow \Pr[\exists u \in V : u \notin V(\mathcal{X})] \leq n \cdot e^{-r}$$

(b) $\mathrm{E}[|\mathcal{X}|] = \sum_{i=1}^{m} \Pr[S_i \in \mathcal{X}] = \sum_{i=1}^{m} (1 - e^{-r\cdot x_i}) \leq r \cdot \sum_{i=1}^{m} x_i = r \cdot \text{Opt}(X_{rel}) \leq r \cdot \text{Opt}(\mathcal{S})$ [33]

□

[33] $e^{-r\cdot x_i} = \sum_{i=0}^{\infty} \frac{(-r\cdot x)^i}{i!} \geq 1 - rx$ für $r \geq 0$

Algorithm 7.2 LasVegasSetCover$[r]$

1: löse das relaxierte LP X_{rel} für SetCover
2: $\tau := 0$ ▷ zählt Durchläufe der Schleife für spätere Analyse
3: **do** $\tau = \tau + 1$
4: $\quad \mathcal{X} =$ RandomizedRoundingSC$[r](\mathcal{S})$
5: **while** $V(\mathcal{X}) = V$
6: gib $\mathcal{X}$ aus.

Satz 7.5. Sei $\mathcal{S}$ eine Eingabe von SetCover und sei $r > \ln n$. Für LasVegasSetCover$[r]$ gilt:

(a) $\mathcal{S}_{cov}$ ist eine Überdeckung mit erwarteter Größe höchstens $r \cdot \text{Opt}(\mathcal{S})$

(b) Die erwartete Anzahl der Durchgänge durch die do-while-Schleife ist höchstens $\dfrac{e^{2r}}{(n-e^r)^2}$

Beweis.

(a) wegen Satz 7.5

(b) Erwartungswert:

$$\begin{aligned} \mathrm{E}[\tau] &= \sum_{t=1}^{\infty} t \cdot \Pr[\text{erst die } t\text{-te Wiederholung ist eine Überdeckung}] \\ &\leq \sum_{t=1}^{\infty} t \cdot \Pr[t-1 \text{ Wiederholungen liefern } \underline{\text{keine}} \text{ Überdeckung}] \\ &\leq \sum_{t=1}^{\infty} t \cdot \left(n \cdot e^{-r}\right)^{t-1} = \frac{e^{2r}}{(n-e^r)^2} \end{aligned}$$

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

□

Korollar 7.6. LasVegasSetCover$[\ln n + 1]$ garantiert eine erwartete relative Güte von $\ln n + 1$. Der Erwartungswert für die Anzahl der Durchgänge durch die Schleife ist $\leq \dfrac{e^2}{(e-1)^2} < 2,503$.

Notizen für mögliche Fragen in der Klausur:

- Was tun, um optimale Lösung zu finden?
  → alles ausprobieren, leider meist in $\mathcal{O}(2^n)$
- Bei Binärbaum:
  vollständig durchsuchen in $\mathcal{O}(2^n)$, bei „folge größerem Erwartungswert“ nur noch $\mathcal{O}(2n)$!
  (pro Stufe müssen nur zwei Knoten betrachtet werden, alle anderen werden verworfen)