Approximationsalgorithmen – Dr. Ulf-Peter Schröder, Zusammenfassung von Florian Schoppmann
Das Copyright für die dieser Zusammenfassung zugrunde liegenden Vorlesungsunterlagen (Skripte, Folien, etc.) liegt beim Dozenten. Darüber hinaus bin ich, Florian Schoppmann, alleiniger Autor dieses Dokuments und der genannte Dozent ist in keiner Weise verantwortlich. Etwaige Inkorrektheiten sind mit sehr großer Wahrscheinlichkeit erst durch meine Zusammenfassung/Interpretation entstanden.

Zusammenfassung Approx'algorithmen

1. Einleitung

Notation:

- $\mathbb{N} := \{1, 2, \dots\}$, $\mathbb{N}_0 := \{0, 1, 2, \dots\}$
- $[n] := \{1, 2, \dots, n\}$
- $[n]_0 := \{0, 1, 2, \dots, n\}$

1.3. Grundbegriffe

Definition: kombinatorisches Optimierunsproblem Π charakterisiert durch Menge von Probleminstanzen D, für alle $I \in D$: Menge der zu zulässigen Lösungen $S(I)$, Bewertungsfunktion $f : S(I) \to \mathbb{N}$, ziel $\in \{\min, \max\}$.
Zu $I \in D$ gesucht: $\sigma_I^{opt} \in S(I) : f(\sigma_I^{opt}) = \text{ziel}\{f(\sigma) \mid \sigma \in S(I)\}$.

Definition: $t(n)$-Approx'alg. A: Zu Eingabe $I \in D$ Berechnung in Zeit $t(|I|)$ eine Ausgabe $\sigma_I^A \in S(I)$. Schreibeweisen: $\text{OPT}(I) := f(\sigma_I^{opt})$, $A(I) := f(\sigma_I^A)$.

2. Approximation mit absoluter Güte

Definition:
absolute Güte: $\kappa_A(I) := |A(I) - \text{OPT}(I)|$
absolute worst-case Güte: $\kappa_A^{wc}(n) := \max\{\kappa_A(I) \mid I \in D, |I| \le n\}$
Garantie einer absoluten Güte: Funktion in $\mathbb{N}$, die obere Schranke für κ_A^{wc} ist
Absolute Abweichung: Funktion in $\mathbb{N}$, die für unendlich viele $n \in N$ untere Schranke für κ_A^{wc} ist. Weitere Begriffe: Zeugenmenge gegen A, Zeuge.

2.1. Graphfärbbarkeit

Sei $G = (V, E)$ ungerichteter Graph. Menge der Nachbarn von u ist $\Gamma_G(u) = \{v \mid \{u, v\} \in E\}$. Smbol für Grad von G: $\Delta(G) := \max_{u \in V}\{\deg_G(u)\}$

Definition: Knotenfärbung, Kantenfärbung: „Nachbarn haben jeweils unterschiedliche Farben"

Beispiel: Knotenfärbung als Opt'problem: $D = \{G \mid G = (V, E)$ ungerichteter Graph mit ≥ 1 Kanten$\}$, $S(G) = \{c_V \mid c_V$ Knotenfärbung$\}$, $f(c_V) = |c_V(V)|$, ziel = max.

Definition: $\chi(G)$ ist chromatische Zahl von G, $\chi'(G)$ ist chromatischer Index

Algorithmus GreedyCol:
1: **for** $i = 1 \dots n$ **do** $c_v(u_i) := \infty$
2: **for** $i = 1 \dots n$ **do**
3: $\quad c_v(u_i) := \min \mathbb{N} \backslash c_v(\Gamma(u_i))$
4: gib c_v aus

Satz: $\text{GreedyCol}(G) \le \Delta(G) + 1$. Ferner $\kappa_{\text{GreedyCol}}(G) = \text{GreedyCol}(G) - \text{OPT}(G) \le \Delta(G) - 1$.

Bemerkungen: Knotenfärbung planarer Graphen:
i) Mit 6 Farben immer in Polynomzeit mäglich.
ii) 4 Farben wären jedoch immer ausreichend.
iii) Entscheidungsproblem „Mit 3 Farben möglich?" NP-vollständig.
iv) Entscheidungsproblem „Mit 2 Farben möglich?" liegt in P.

Algorithmus ColPlan: für planaren Graphen G: Teste ob G 2-färbar, wenn nicht: Färbe mit 6 Farben

Satz: ColPlan garantiert eine absolute Güte von 3.

An einem Knoten eines kantengefärbten Graphen fehlt eine Farbe, wenn alle inzidenten Kanten dort anders gefärbt sind.

Lemma: G kantengefärbt mit $[\Delta(G)+1]$, $u, v \in V$ mit $\{u, v\} \notin E$ und $\deg(u), \deg(v) < \Delta(G)$. Dann kann G umgefärbt werden, so dass an u und v dieselbe Farbe fehlt.

Algorithmus Kantenfärbung:
Eingabe: Graph $G = (V, E)$
1: $E' := \emptyset$, $G' := (V, E')$
2: **while** $E' \subsetneq E$ **do**
3: $\quad$ Wähle eine beliebige Kante $\{u, v\} \in E \setminus E'$
4: $\quad$ *alterGrad* $:= \Delta(G')$
5: $\quad E' := E' \cup \{\{u, v\}\}$
6: $\quad$ **if** $\Delta(G') >$ *alterGrad* **then**
7: $\quad\quad$ Färbe die Kante $\{u, v\}$ mit der kleinstmöglichen an u und v fehlenden Farbe aus $\{1, \dots, \Delta(G) + 1\}$
8: $\quad$ **else**
9: $\quad\quad$ Färbe G' um, so daß an u und v dieselbe Farbe c fehlt
10: $\quad\quad$ färbe $\{u, v\}$ mit c

Das Copyright für die dieser Zusammenfassung zugrunde liegenden Vorlesungsunterlagen (Skripte, Folien, etc.) liegt beim Dozenten. Darüber hinaus bin ich, Florian Schoppmann, alleiniger Autor dieses Dokuments und der genannte Dozent ist in keiner Weise verantwortlich. Etwaige Inkorrektheiten sind mit sehr großer Wahrscheinlichkeit erst durch meine Zusammenfassung/Interpretation entstanden.

Satz: Für jeden Graph G: $\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$.

Folgerung: Kantenfärbung garantiert absolute Güte 1, hat aber auch absolute Abweichung von 1.

2.2. Ein Nicht-Approximationsergebnis

Definition: Rucksackproblem Π:
- $D = \{\langle n, \text{vol}, p, B\rangle \mid n, B \in \mathbb{N}$ und $\text{vol}, p : [n] \to \mathbb{N}$ und $\forall w \in [n] : \text{vol}(w) \leq B\}$
- $S(\langle n, \text{vol}, p, B\rangle) = \{A \subseteq [n] \mid \sum_{w\in A} \text{vol}(w) \leq B\}$
- $f(A) = \sum_{w\in A} p(w)$
- ziel = max.

Wir schreiben $p_j := p(j)$.

Satz: Falls $\mathsf{P} \neq \mathsf{NP}$, gibt es kein $k \in \mathbb{N}$, so dass $\forall I \in D : |A(I) - \text{OPT}(I)| \leq k$.

3. Approximation mit relativer Gütegarantie

Π Optimierungsproblem, A Approx'algorithmus.

Definition:
relative Güte: $\rho_A(I) := \max\{\frac{A(I)}{\text{OPT}(I)}, \frac{\text{OPT}(I)}{A(I)}\}$
relative worst-case Güte: $\rho_A^{wc}(n) := \max\{\rho_A(I) \mid I \in D, |I| \leq n\}$
Garantie einer relativen Güte: Funktion in $\mathbb{N}$, die obere Schranke für ρ_A^{wc} ist
relativer Fehler: $\epsilon_A(I) := \frac{|A(I)-\text{OPT}(I)|}{\text{OPT}(I)}$
Relative Abweichung: Funktion in $\mathbb{N}$, die für unendlich viele $n \in N$ untere Schranke für ρ_A^{wc} ist. Weitere Begriffe: Zeugenmenge gegen A, Zeuge.

Bemerkungen: Bei Max'problem: $\epsilon_A(I) = \rho_A(I) - 1$. Bei Min'problemen: $\epsilon_A(I) = 1 - \frac{1}{\rho_A(I)} \leq \rho_A(I) - 1$.

3.1 Das metrische TSP

Definition: Als Optimierungsproblem Π:
- $D = \{\langle K_n, c\rangle \mid K_n$ vollständiger Graph auf $n > 1$ Knoten, $c : E \to \mathbb{N}$, $\forall u, v, w \in V : c(u, v) \leq c(u, w) + c(w, v)\}$
- $S(\langle K_n, c\rangle) = \{C \mid C = (u_1, \ldots, u_n, u_1)$ ist Hamiltonkreis$\}$
- $f(C) = \sum_{i=1}^{n} c(u_i, u_{i+1})$, wobei $u_{n+1} := u_1$.
- ziel = min.

Algorithmus Einfügen:
Eingabe: $C = (u_1, \ldots, u_n, u_1)$ wie oben, $v \in E$
1: Bestimme i, so dass $c(u_i, v) + c(v, u_{i+1}) - c(u_i, u_{i+1})$ minimal ist
2: Gib $(u_1, \ldots, u_i, v, u_{i+1}, \ldots, u_n, u_1)$ zurück

Algorithmus EinfügeHeuristik: (Für ΔTSP)
1: $C_1 := (v, v)$ für einen beliebigen Knoten $v \in E$
2: **for** $j := 2, \ldots, n$ **do**
3: $\quad v :=$ Knoten, der nicht in C_{j-1} ist (*)
4: $\quad C_j :=$ Einfügen(C_{j-1}, v)

Satz: Für jede Einfüge-Heuristik A (beachte Wahlmöglichkeit bei (*)) gilt $\rho_A(\langle K_n, c\rangle) \leq \lceil \log n \rceil + 1$.

Satz: Wird bei (*) stets der Knoten gewählt, der am nächsten zu einem Knoten in C_{j-1} liegt, NearIns (nearest insertion) genannt, so gilt $\rho_{\text{NearIns}}(\langle K_n, c\rangle) \leq 2 - \frac{2}{n}$. ($\to$ Spannbaum-Argument)

Definition: Matching M ist Teilgraph mit Grad 1. Gewicht eines Matching ist Summe der vorkommenden Kanten. Ein perfektes Matching enthält alle Knoten. Ein leichtestes Matching ist ein perfektes Matching mit minimalem Gewicht.

Definition: Ein Multigraph erlaubt mehrere Kanten pro Knotenpaar. Eine Euler-Tour ist ein Kreis $(u_1, \ldots u_{|E|}, u_1)$, in dem jede Kante genau einmal vorkommt.

Algorithmus Ch: (Christofides)
Eingabe: $I = \langle K_n, c\rangle$
1: $T_{Ch} :=$ minimaler Spannbaum
2: $S := \{v \in T_{Ch} \mid \deg_{T_{Ch}}(v)$ ist ungerade $\}$ ▷ $|S|$ ist gerade
3: $M_{Ch} :=$ leichtestes Matching auf dem durch S induzierten Teilgraphen von K_n
4: $E := (u_1, u_2, \ldots)$ Euler-Tour auf $T_{Ch} \cup M_{Ch}$
5: $E' := E$ ohne Wiederholungen von Knoten

Satz: $\rho_{\text{Ch}}(\langle K_n, c\rangle) \leq \frac{3}{2} - \frac{1}{n}$.

Definition: Vertex-Cover: Finde zu einem ungerichteten Graphen $G = (V, E)$ eine minimale Knotenüberdeckung: $C \subseteq V$, so dass $\forall\{u, v\} \in E : \{u, v\} \cap C \neq \emptyset$.

Algorithmus GreedyVC:
1: $C := \emptyset$, $E' := E$
2: **while** $E' = \emptyset$ **do**
3: $\quad C := C \cup \{u, v\}$, wobei $\{u, v\} \in E'$ beliebig
4: $\quad E' := E' \setminus \{e \in E' \mid u \in e$ oder $v \in e\}$
5: Gebe C aus

Satz: $\rho_{\text{GreedyVC}}(G) \leq 2$.

Das Copyright für die dieser Zusammenfassung zugrunde liegenden Vorlesungsunterlagen (Skripte, Folien, etc.) liegt beim Dozenten. Darüber hinaus bin ich, Florian Schoppmann, alleiniger Autor dieses Dokuments und der genannte Dozent ist in keiner Weise verantwortlich. Etwaige Inkorrektheiten sind mit sehr großer Wahrscheinlichkeit erst durch meine Zusammenfassung/Interpretation entstanden.

Definition: Scheduling: n Jobs mit Lasten $w_1, \ldots, w_n$, m Maschinen. Gesucht: Zuordnung $(s_1, \ldots, s_n) \in [m]^n$, so dass $\max_{k\in[m]} \sum_{i\in[n]|s_i=k} w_i$ minimal wird.

Satz: Die Graham-Heuristik, die Jobs greedy sequentiell auf die jeweils am wenigsten ausgelastete Maschine zu verteilen, hat für jede Eingabe relative Güte $2 - \frac{1}{m}$.

Satz: LPT (least processing time first), der vor der Graham-Heuristik die Jobs nach aufsteigendem Gewicht sortiert, hat relative Güte $\frac{4}{3} - \frac{1}{3m}$.

Definition: Set-Cover-Problem: Gegeben endliche Menge X und $\mathcal{F} \subseteq \mathcal{P}(X)$ mit $\bigcup_{S\in\mathcal{F}} S = X$. Finde kleinstes $\mathcal{C} \subseteq \mathcal{F}$, das X überdeckt.

Algorithmus GreedySetCover:
1: $U := X$, $\mathcal{C} := \emptyset$
2: **while** $U \neq \emptyset$ **do**
3: wähle $S \in \mathcal{F}$, so dass $|S \cap U|$ maximal
4: $U := U \setminus S$
5: $\mathcal{C} := \mathcal{C} \cup \{S\}$

Satz: $\rho_{\text{GreedySetCover}}(\langle X, \mathcal{F}\rangle) \leq H(\max_{S\in\mathcal{F}} |S|)$

Definition: Maximum Set-Cover-Problem: Gegeben endliche Menge X, Gewichtsfunktion $w : X \to \mathbb{R}$, $\mathcal{F} \subseteq \mathcal{P}(X)$, $k \in \mathbb{N}$. Gesucht: $\mathcal{C} \subseteq \mathcal{F}$ mit $|\mathcal{C}| = k$, so dass $w(\mathcal{C}) := w(\bigcup_{S\in\mathcal{C}} S)$ maximiert wird. (w sei dazu kanonisch auf $\mathcal{P}(X)$ fortgesetzt.)

Algorithmus MSC:
1: $U := X$, $\mathcal{C} := \emptyset$
2: **for** $i := 1, \ldots, k$ **do**
3: wähle $S \in \mathcal{F}$, so dass $w(S \cap U)$ maximal
4: $U := U \setminus S$
5: $\mathcal{C} := \mathcal{C} \cup \{S\}$

Satz: $\rho_{\text{MSC}}(\langle X, w, \mathcal{F}, k\rangle) \leq (1 - \frac{1}{e})^{-1} \approx 1{,}59$.

Definition: $G = (V, E)$ Graph, Knotenmenge $U \subseteq V$ heißt unabhängig, wenn $\forall u, v \in U : \{u, v\} \notin E$. Beim Independent-Set-Problem ist größtmögliche unabhängige Menge zu finden.

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

Satz: $\rho_{\text{GreedyIS}}(G) \leq \frac{|E|}{|V|} + 1$. Die relative Abweichung ist mindestens $\frac{1}{4}(|V| - 1)$.

Satz: Sei $G = (V, E)$ mit k Farben knotenfärbbar. Dann: $\text{GreedyIS}(G) \geq \lceil \log_k(\frac{|V|}{3}) \rceil$.

Ein verbesserter Greedy-Knotenfärbungsalgorithmus:
Algorithmus GreedyCol2:
1: $t := 1$, $V^{(1)} := V$
2: **while** $V^{(t)} \neq \emptyset$ **do**
3: $G^{(t)} :=$ der durch $V^{(t)}$ induzierte Graph
4: $U^{(t)} := \text{GreedyIS}(G^{(t)})$
5: Färbe alle Knoten in $U^{(t)}$ mit der Farbe t
6: $V^{(t+1)} := V^{(t)} - U^{(t)}$
7: $t := t + 1$
8: Gib Färbung aus

Satz: Sei $G = (V, E)$ mit k Farben knotenfärbbar, $n := |V|$. Dann $\text{GreedyCol2}(G) \leq \frac{3n}{\log_k \frac{n}{16}}$. Ferner $\rho_{\text{GreedyCol2}}(G) = \mathcal{O}(n/\log n)$.

Satz: Graphen G mit $\chi(G) \leq 3$ können in polynomieller Zeit mit $\mathcal{O}(\sqrt{|V|})$ Farben gefärbt werden:
Algorithmus Spezialfärbung:
1: **while** $\Delta(G) \geq \sqrt{n}$ **do**
2: Finde $v \in G$ mit $\deg_G(v) \geq \sqrt{n}$
3: Färbe durch $\{v\} \cup \Gamma_G(v)$ induzierten Teilgraph mit 3 Farben
4: $G := G$ ohne $\{v\} \cup \Gamma_G(v)$
5: Färbe G mit GreedyCol-Algorithmus mit $\Delta(G) + 1$ Farben

3.3. Ein Nicht-Approximierbarkeitsergebnis für relative Gütegarantie

Satz: Wenn TSP mit konstanter relativer Güte approximiert werden kann, gilt $\mathsf{P} = \mathsf{NP}$.

Satz: Sei $L \subseteq \Sigma^*$ sei NP-vollständig, Π ein Minimierungsproblem. Gibt es zwei in Polynomzeit berechenbare Funktionen $f : \Sigma^* \to D$ und $c : \Sigma^* \to \mathbb{N}$ und eine Konstante $\gamma > 0$, so dass

$$\forall x \in \Sigma^* : \text{OPT}(f(x)) = \begin{cases} \leq c(x) & x \in L \\ \geq c(x) \cdot (1 + \gamma) & x \notin L, \end{cases}$$

dann gibt es keinen polynomiellen Approximationsalgorithmus der relativen Güte $r < 1+\gamma$, es sei denn $\mathsf{P} = \mathsf{NP}$.

Das Copyright für die dieser Zusammenfassung zugrunde liegenden Vorlesungsunterlagen (Skripte, Folien, etc.) liegt beim Dozenten. Darüber hinaus bin ich, Florian Schoppmann, alleiniger Autor dieses Dokuments und der genannte Dozent ist in keiner Weise verantwortlich. Etwaige Inkorrektheiten sind mit sehr großer Wahrscheinlichkeit erst durch meine Zusammenfassung/Interpretation entstanden.

4. Approximationsschemata

Definition: Π Opt'problem, A, Approx'algorithmus für Π mit Eingabe bestehend aus Instanz I und $\epsilon \in (0,1)$.

PTAS: A ist polynomielles Approx'schema, wenn für alle $I \in D$, $\epsilon \in (0,1)$ in Zeit $\mathcal{O}(\text{poly}(|I|))$ eine zulässige Lösung zu I berechnet wird mit $\epsilon_A(I,\epsilon) \leq \epsilon$.

FPTAS: A ist streng polynomielles Approx'schema, wenn A ein PTAS mit Laufzeit $\mathcal{O}(\text{poly}(|I|, \frac{1}{\epsilon}))$ ist.

Satz: A (F)PTAS. Bei Eingabe I sei $z(I)$ obere Schranke für $\text{OPT}(I)$. $\epsilon^* := \frac{1}{z(I)+1}$. Dann ist $A(I,\epsilon^*) = \text{OPT}(I)$. Ist A ein FPTAS, so ist die Laufzeit $\mathcal{O}(\text{poly}(|I|, z(I)))$.

4.1. Ein pseudopolynomieller exakter Algorithmus für das 0-1 Rucksackverfahren

Sei $I = \langle n, \text{vol}, p, B\rangle$ Instanz des Rucksackproblems. $p_{\max} := \max_{i\in[n]} p_j$. Prädikat $F_j(\alpha) := \inf\{\text{vol}(R) \mid R \subseteq [j] \text{ and } p(R) \geq \alpha\}$.

Algorithmus DynRucksack:
1: $\alpha := 0$
2: **repeat**
3: $\quad\alpha := \alpha + 1$
4: $\quad$**for** j:=1, …, n **do**
5: $\quad\quad F_j(\alpha) := \min\{F_{j-1}(\alpha), F_{j-1}(\alpha - p_j) + \text{vol}(p_j)\}$
6: **until** $B < F_n(\alpha)$
7: Gib $\alpha - 1$ aus

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

Definition: Π Optimierungsproblem mit Instanzen aus ausschließlich natürlichen Zahlen. $\text{maxnr}(I)$ sei größte in I vorkommende Zahl. Algorithmus A heißt pseudopolynomiell, wenn für alle $I \in D$ seine Laufzeit $\text{poly}(|I|, \text{maxnr}(I))$ ist.

4.2. Ein streng polynomielles Approximationsschema für das 0-1 Rucksackproblem

Algorithmus AR_k:
Eingabe: Instanz $I = \langle n, \text{vol}, p, B\rangle$
1: Ersetze p_j durch $\lfloor \frac{p_j}{k} \rfloor$
2: Berechne mittels DynRucksack eine optimale Rucksackfüllung R_k auf der Eingabe I_{red}
3: Gib R_k aus

Satz: $\epsilon_{\text{AR}_k}(\langle n, \text{vol}, p, B\rangle) \leq \frac{k \cdot n}{p_{\max}}$. Die Laufzeit ist $\mathcal{O}(n^2 \cdot \frac{p_{\max}}{k})$.

Algorithmus FPTASRucksack:
Eingabe: Instanz $I = \langle n, \text{vol}, p, B\rangle$, Fehler ϵ
1: $k := \epsilon \cdot \frac{p_{\max}}{n}$
2: Rufe $\text{AR}_k(I)$ auf

Satz: FPTASRucksack ist ein FPTAS für das Rucksackproblem mit Laufzeit $\mathcal{O}(n^3 \cdot \frac{1}{\epsilon})$.

4.3. „Negativergebnisse" für Approximationsschemata

Satz: Π Optimierungsproblem, $k \in \mathbb{N}$ fest. Falls Π Minimierungsproblem, sei die Frage "Ist zur Eingabe I von Π der Wert $\text{OPT}(I) \leq k$?" bzw., falls Π Maximierungsproblem ist, die Frage "Ist zur Eingabe I von Π der Wert $\text{OPT}(I) \geq k$?" NP-vollständig. Gibt es ein FPTAS(PTAS) für I, dann ist P = NP.

Definition: L NP-vollständig. Dann heißt L stark NP-vollständig, wenn es ein Polynom q gibt, so dass $L_q := \{x \in \Sigma^* \mid x \in L \text{ und } \text{maxnr}(x) \leq q(|x|)\}$ NP-vollständig ist. Gibt es kein solches q, so heißt L schwach NP-vollständig.
D. h., für eine NP-vollständige Sprache L gilt, falls P ≠ NP: L stark NP-vollständig ⇔ es gibt keinen pseudopolynomiellen Algorithmus für L

Satz: Π Optimierungsproblem. Gibt es ein Polynom $q(x_1, x_2)$ mit $\text{OPT}(I) \leq q(|I|, \text{maxnr}(I))$ für alle $I \in D$, dann folgt aus der Existenz eines FPTAS für Π die Existenz eines pseudopolynomiellen exakten Algorithmus.

Folgerung: Gibt es für eine Optimierungsvariante eines stark NP-vollständigen Problems ein FPTAS, so folgt P = NP.

5. Klassen und Hierarchien

Definition: Optimierungsproblem Π liegt in NPO, wenn:
i) Gültige Instanzen I werden in $\text{poly}(|I|)$ Zeit erkannt.
ii) Es gibt ein Polynom p, so dass $\forall \sigma \in S(I) : |\sigma| \leq p(|I|)$. Ferner $\forall y \in \Sigma^*, |y| \leq p(|I|) : y$ kann in Polynomialzeit in $|I|$ als Lösung verifiziert werden.

Das Copyright für die dieser Zusammenfassung zugrunde liegenden Vorlesungsunterlagen (Skripte, Folien, etc.) liegt beim Dozenten. Darüber hinaus bin ich, Florian Schoppmann, alleiniger Autor dieses Dokuments und der genannte Dozent ist in keiner Weise verantwortlich. Etwaige Inkorrektheiten sind mit sehr großer Wahrscheinlichkeit erst durch meine Zusammenfassung/Interpretation entstanden.

iii) Für alle $\sigma \in S(I), I \in D$ kann $f(\sigma)$ in Polynomialzeit in $|I|$ berechnet werden.

Alternativ: $\exists \Pi' \in \mathsf{NPO} : \Pi \leq_p \Pi'$.

Definition: Zugehöriges Entscheidungsproblem Π_{ent}.

Satz: $\Pi \in \mathsf{NPO} \Rightarrow \Pi \in \mathsf{NP}$

Definition: Sprachklasse $\mathsf{PO} \subseteq \mathsf{NPO}$, wobei zu jeder Instanz I in Polynomialzeit in $|I|$ eine optimale Lösung berechnet werden kann.

Definition: Sei F Menge von Funktionen $\mathbb{N} \to \mathbb{R}_{\geq 0}$. Dann liegt $\Pi \in \mathsf{NPO}$ in der Sprachklasse F-APX, wenn es einen polynomiellen Approx'algorithmus A gibt mit $\rho_A \in F$. Ist F die Menge der konstanten Funktionen, so heißt diese Klasse nur APX.

Definition: $\Pi \in \mathsf{NPO}$ liegt in der Sprachklasse PAS bzw. FPAS, wenn es für Π ein (F)PTAS gibt.

Satz: $\mathsf{PO} \subseteq \mathsf{FPAS} \subseteq \mathsf{PAS} \subseteq \mathsf{APX} \subseteq \mathsf{NPO}$

Definition: ABS ist Klasse der mit konstant absoluter Güte approximierbaren Optimierungsprobleme.
Bemerkungen: $\mathsf{PO} \subseteq \mathsf{ABS} \subseteq \mathsf{APX}$

6. Techniken für randomisierte Approximationsalgorithmen

6.1. Die probabilistische Methode

Definition: $V := \{x_1, \ldots, x_n\}$ Menge von Variablen. Literal ist Variable $x_i \in V$ oder Negation $\overline{x_i}$. Klausel $C = l_1 \vee \cdots \vee l_k$ ist oder-Verknüpfung von Literalen. Boole'sche (n, m)-Formel in konjunktiver Normalform ist $\Phi = C_1 \wedge \cdots \wedge C_m$.

Definition: Probleminstanz von MaxSAT ist solches Φ. Bewertungsfunktion: $\text{wahr}(b, \Phi) = |\{C_j \in \Phi \mid b(C_j) = \text{wahr}\}|$, wobei $b : V \to \{\text{true,false}\}$ Belegung ist.

Algorithmus A:
1: **for** $i := 1, \ldots, n$ **do**
2: $x_i := \begin{cases} \text{true} & \text{mit Wkt. } \frac{1}{2} \\ \text{false} & \text{mit Wkt. } \frac{1}{2} \end{cases}$
3: Gib $b_\text{A} = (x_1, \ldots, x_n)$ aus

Satz: Für jede Boole'sche (n, m) Formel Φ in KNF mit mindestens k Literalen pro Klausel gilt: $\text{E}[A(\Phi)] \geq (1 - \frac{1}{2^k}) \cdot m$.

Definition: Erwartete relative Güte

Satz: $\text{E}[\rho_\text{A}(\Phi)] \leq \frac{2^k}{2^k - 1}$.

6.2. „Randomized Rounding“

Ganzzahliges lineares Programm B für MaxSAT:

$$\begin{aligned}
\text{max.} \quad & \sum_{i=1}^{m} \widehat{C}_j \\
\text{gem.} \quad & \sum_{x_k \in S_i^{\oplus}} \widehat{x}_k + \sum_{x_k \in S_j^{\ominus}} (1 - \widehat{x}_k) \geq \widehat{C}_i \quad \forall i \in [m] \\
& \widehat{x}_i \in \{0, 1\} \quad \forall i \in [n] \\
& \widehat{C}_i \in \{0, 1\} \quad \forall i \in [m]
\end{aligned}$$

Relaxierung B_{rel} durch Ersetzung der Nebenbedingungen.

Algorithmus B:
1: Löse relaxiertes Problem B_{rel} zu Φ
2: Ermittle Belegung b_B mittels „randomized rounding“ für eine Funktion $\pi : [0, 1] \to [0, 1]$ (z. B.: $\pi(x) = x$)

Satz: Für alle Φ gilt $\text{E}[B(\Phi)] \geq (1 - (1 - 1/k)^k) \cdot \text{OPT}(\Phi)$.

Satz: $\text{E}[\rho_\text{B}(\Phi)] \leq \frac{1}{1 - 1/e} \approx 1{,}582$.

6.3. Hybrider Ansatz: Eine Kombination mehrerer Verfahren

Satz: Für einen Algorithmus C, der jeweils mit W'keit $\frac{1}{2}$ A oder B startet, gilt $\text{E}[\rho_\text{C}] \leq \frac{4}{3}$.

6.4 Semidefinite Optimierung

Definition: Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt positiv semidefinit, wenn gilt: $\forall x \in \mathbb{R}^n : x^T \cdot A \cdot x \geq 0$. Ein semidefinites Programm ist ein lineares Programm, bei dem die Lösungen für die Variablen jeweils als symmetrisch und positiv semidefinite Matrix geschrieben werden können.

Definition: Max-Cut-Problem: Gegeben ungerichteter Graph $G = (V, E)$. Gesucht: $U \subseteq V$, so dass die Schnittgröße $|\{\{u, v\} \in E \mid u \in U \text{ und } v \in V \setminus U\}|$ maximal.

Satz: Durch einen zusammenhängenden Graphen gibt es immer einen Schnitt der Größe $\geq \frac{1}{2} \cdot |E|$.

Das Copyright für die dieser Zusammenfassung zugrunde liegenden Vorlesungsunterlagen (Skripte, Folien, etc.) liegt beim Dozenten. Darüber hinaus bin ich, Florian Schoppmann, alleiniger Autor dieses Dokuments und der genannte Dozent ist in keiner Weise verantwortlich. Etwaige Inkorrektheiten sind mit sehr großer Wahrscheinlichkeit erst durch meine Zusammenfassung/Interpretation entstanden.

Sei $n := |V|$ und $A_G = (a_{ij})$ Adjazenzmatrix von G. Für jeden Knoten $v_i \in V$ Variable $x_i \in \{-1, 1\}$. Quadratisches Programm für Max-Cut:

$$\begin{aligned} \text{max.} \quad & \frac{1}{2} \cdot \sum_{i<j} a_{ij} \cdot (1 - x_i \cdot x_j) \\ \text{gem.} \quad & x_i \in \{-1, 1\} \qquad \forall i \in [n] \end{aligned}$$

Relaxiertes quadratisches Programm:

$$\begin{aligned} \text{max.} \quad & \frac{1}{2} \cdot \sum_{i<j} a_{ij} \cdot (1 - \langle u_i, u_j \rangle) \\ \text{gem.} \quad & u_i \in \mathbb{R}^n, \|u_i\|_2 = 1 \qquad \forall i \in [n] \end{aligned}$$

Umformulierung in semidefinites Optimierungsproblem: Sei $Y \in \mathbb{R}^{n \times n}$, $Y = (y_{ij})$ mit $y_{ij} := \langle u_i, u_j \rangle$. Dann ist Y symmetrisch und positiv semidefinit. (Mit $B := (u_1, \ldots, u_n)$ gilt $Y = B^T \cdot B$.)

$$\begin{aligned} \text{max.} \quad & \frac{1}{2} \cdot \sum_{i<j} a_{ij} \cdot (1 - y_{ij}) \\ \text{gem.} \quad & Y = (y_{ij}) \text{ symm., pos. semidef.} \\ & y_{ii} = 1 \qquad \forall i \in [n] \end{aligned}$$

Algorithmus MaxCut:

1: $\epsilon := 0,0005$
2: Löse das semidefinite Programm für Max-Cut mit absoluter Güte ϵ und erhalte $Y^{(\epsilon)}$
3: Führe Cholesky-Zerlegung $Y^{(\epsilon)} = B^T \cdot B$ und damit die Berechnung der u_i durch
4: Würfle gleichverteilt einen zufälligen n-dimensionalen Vektor r aus
5: **for** i := 1, ..., n **do**
6: **if** $\langle u_i, r \rangle \geq 0$ **then**
7: $x_i := 1$
8: **else**
9: $x_i := -1$

Satz: $\mathrm{E}[\text{MaxCut}(G)] \geq 0{,}878 \cdot \mathrm{OPT}(G)$ bzw. $\mathrm{E}[\rho_{\text{MaxCut}}(G)] \leq \frac{1}{0{,}878} \approx 1{,}139$ für alle Graphen G