Average Case Complexity, Smoothed Analysis

Alexander Kuhnle   Jan-Martin Knorr

Proseminar: Die $\mathrm{P} \neq \mathrm{NP}$-Vermutung

Überblick

1. Motivation
2. Levins Theorie ($\mathtt{dist}\mathbf{P}$, $\mathtt{dist}\mathbf{NP}$)
   - Verteilungsprobleme und die Klasse $\mathtt{dist}\mathbf{NP}$
   - Die Klasse $\mathtt{dist}\mathbf{P}$
   - Average-Case Reduktion und $\mathtt{dist}\mathbf{NP}$-Vollständigkeit
   - Satz von Levin
3. Smoothed Analysis
   - Einführung
   - Verallgemeinerte Definition
   - Smoothed Analysis eines SSSP-Algorithmus
4. Zusammenfassung

Laufzeit im Worst-Case

Worst-Case-Laufzeit eines Algorithmus $\mathcal{A}$:
Betrachte für festes $n \in \mathbb{N}$ die Laufzeit einer (für $\mathcal{A}$) "ungünstigsten" Eingabe der Länge n.

$$T_{\mathcal{A}}^{worst}(n) := \max_{x \text{ Eingabe}, |x|=n} T_{\mathcal{A}}(x)$$

Beispiel: Quicksort

- Worst-Case: in jedem Rekursionsschritt sind alle verbleibenden Elemente im gleichen Teilproblem.
- Anzahl der nötigen Vergleiche: $\sum_{i=1}^{n} n - i$
- $\Rightarrow T_{\text{Quicksort}}^{worst}(n) \in \Theta(n^2)$

Laufzeit im Worst-Case

Worst-Case-Laufzeit eines Algorithmus $\mathcal{A}$:
Betrachte für festes $n \in \mathbb{N}$ die Laufzeit einer (für $\mathcal{A}$) "ungünstigsten" Eingabe der Länge n.

$$T_{\mathcal{A}}^{worst}(n) := \max_{x \text{ Eingabe}, |x|=n} T_{\mathcal{A}}(x)$$

Beispiel: Quicksort

- Worst-Case: in jedem Rekursionsschritt sind alle verbleibenden Elemente im gleichen Teilproblem.
- Anzahl der nötigen Vergleiche: $\sum_{i=1}^{n} n - i$
- $\Rightarrow T_{\text{Quicksort}}^{worst}(n) \in \Theta(n^2)$

Probleme einer Worst-Case-Analyse

- Viele Algorithmen benötigen nur für wenige Eingaben die Worst-Case-Laufzeit.
- Worst-Case-Analysen ignorieren, welche Instanzen in der Realität häufig auftauchen.

$\Rightarrow$ Ziel: Laufzeit-Analyse des “durchschnittlichen Falls”.

Verschiedene Betrachtungsweisen eines durchschnittlichen Falls

Durchschnittliche Korrektheit

- Algorithmus hat stets polynomielle Laufzeit, ...
- ... liefert aber nur für bestimmte Instanzen ein Ergebnis

Durchschnittliche Laufzeit

- Algorithmus liefert stets das korrekte Ergebnis, ...
- ... die Laufzeit ist aber nur für bestimmte Instanzen polynomiell beschränkt
- Hauptaugenmerk dieses Vortrags

Verschiedene Betrachtungsweisen eines durchschnittlichen Falls

Durchschnittliche Korrektheit

- Algorithmus hat stets polynomielle Laufzeit, ...
- ... liefert aber nur für bestimmte Instanzen ein Ergebnis

Durchschnittliche Laufzeit

- Algorithmus liefert stets das korrekte Ergebnis, ...
- ... die Laufzeit ist aber nur für bestimmte Instanzen polynomiell beschränkt
- Hauptaugenmerk dieses Vortrags

Laufzeit im Average-Case

Average-Case-Laufzeit eines Algorithmus $\mathcal{A}$:
Betrachte den Erwartungswert der Laufzeit aller Eingaben der Länge $n \in \mathbb{N}$ bezüglich einer Verteilung $\mathcal{D}_n$ mit zugehöriger Wahrscheinlichkeitsfunktion $\mathbb{P}_{\mathcal{D}_n} : \{0,1\}^n \to [0,1]$.

$$T^{average}_{\mathcal{A},\mathcal{D}_n}(n) := \mathbb{E}_{x \in \mathcal{D}_n}[T_{\mathcal{A}}(x)] = \sum_{x \in \{0,1\}^n} T_{\mathcal{A}}(x) \cdot \mathbb{P}_{\mathcal{D}_n}(x)$$

Beispiel: Quicksort

- Hier sei $\mathbb{P}_{\mathcal{D}_n}(x) = 2^{-n}$ (Gleichverteilung).
- Vergleichswahrscheinlichkeit für das i-te und j-te Element ($i < j$) beträgt $p_{i,j} = \frac{2}{j-i+1}$.

Laufzeit im Average-Case

Average-Case-Laufzeit eines Algorithmus $\mathcal{A}$:
Betrachte den Erwartungswert der Laufzeit aller Eingaben der Länge $n \in \mathbb{N}$ bezüglich einer Verteilung $\mathcal{D}_n$ mit zugehöriger Wahrscheinlichkeitsfunktion $\mathbb{P}_{\mathcal{D}_n}\colon \{0,1\}^n \to [0,1]$.

$$T^{average}_{\mathcal{A},\mathcal{D}_n}(n) := \mathbb{E}_{x \in \mathcal{D}_n}[T_{\mathcal{A}}(x)] = \sum_{x \in \{0,1\}^n} T_{\mathcal{A}}(x) \cdot \mathbb{P}_{\mathcal{D}_n}(x)$$

Beispiel: Quicksort

- Hier sei $\mathbb{P}_{\mathcal{D}_n}(x) = 2^{-n}$ (Gleichverteilung).
- Vergleichswahrscheinlichkeit für das i-te und j-te Element ($i < j$) beträgt $p_{i,j} = \frac{2}{j-i+1}$.

Laufzeit im Average-Case

Beispiel: Quicksort (Fortsetzung)

- Sei V die Anzahl der Vergleiche bei einer gewissen Eingabe.
- Sei $V_{i,j}$ die Anzahl der Vergleiche des i-ten und j-ten Elements bei einer gewissen Eingabe.
- Beachte: $V_{i,j} \in \{0, 1\}$.

$$\mathbb{E}[V] = \sum_{i<j} \mathbb{E}[V_{i,j}] = \sum_{i<j} p_{i,j} = \sum_{i<j} \frac{2}{j-i+1} = \ldots \leq 2n \ln n$$

- $\Rightarrow T^{average}_{\mathrm{Quicksort},\mathcal{D}_\mathrm{n}}(n) \in \mathcal{O}(n \log n)$

Überblick

1. Motivation
2. Levins Theorie (distP, distNP)
   - Verteilungsprobleme und die Klasse distNP
   - Die Klasse distP
   - Average-Case Reduktion und distNP-Vollständigkeit
   - Satz von Levin
3. Smoothed Analysis
   - Einführung
   - Verallgemeinerte Definition
   - Smoothed Analysis eines SSSP-Algorithmus
4. Zusammenfassung

Verteilungsprobleme

Verteilungsproblem

Ein *Verteilungsproblem* ist ein Paar $\langle L, \mathcal{D} \rangle$ mit:

- L ist ein Entscheidungsproblem,
- $\mathcal{D} = (\mathcal{D}_n)$ ist eine Familie von Verteilungen über L.

Zugehörige Wahrscheinlichkeitsfunktion:

$$\mathbb{P}_{\mathcal{D}_n} : \{0, 1\}^n \to [0, 1].$$

Verteilungsprobleme

Zufallsgraphen

Zufallsgraphen (Erdős-Rényi-Modell)

Gegeben: Knotenanzahl $n \in \mathbb{N}$ und Wahrscheinlichkeit $p \in [0, 1]$.

- Sei $G = (V, E)$ ungerichteter Graph mit $V = \{v_1, \dots, v_n\}$.
- Füge $\{v_i, v_k\}$ mit Wahrscheinlichkeit p zu E hinzu.
- Für $G = (V, E)$ mit $|E| = e$ gilt also: $\mathbb{P}(G) = p^e \cdot (1 - p)^{\binom{n}{2} - e}$.

Beispiel Erwartete Anzahl Dreiecke in $G = (V, E)$ mit $|V| = n$:

$$\mathbb{E}(\text{Dreiecke in } G) = \binom{n}{3} \cdot p^3.$$

Verteilungsprobleme
Zufallsgraphen

Zufallsgraphen (Erdős-Rényi-Modell)

Gegeben: Knotenanzahl $n \in \mathbb{N}$ und Wahrscheinlichkeit $p \in [0, 1]$.

- Sei $G = (V, E)$ ungerichteter Graph mit $V = \{v_1, \ldots, v_n\}$.
- Füge $\{v_i, v_k\}$ mit Wahrscheinlichkeit p zu E hinzu.
- Für $G = (V, E)$ mit $|E| = e$ gilt also: $\mathbb{P}(G) = p^e \cdot (1 - p)^{\binom{n}{2} - e}$.

Beispiel Erwartete Anzahl Dreiecke in $G = (V, E)$ mit $|V| = n$:

$$\mathbb{E}(\text{Dreiecke in } G) = \binom{n}{3} \cdot p^3.$$

Verteilungsprobleme
Polynomiell berechenbare Verteilungen

Polynomiell berechenbare Verteilung

$\mathcal{D} = (\mathcal{D}_n)$ heißt *polynomiell berechenbar*, wenn die *kumulierte Wahrscheinlichkeit*

$$\mu_{\mathcal{D}_n}(x) := \sum_{\substack{y \leq x \\ |y| = n}} \mathbb{P}_{\mathcal{D}_n}(y).$$

für alle $n \in \mathbb{N}$ polynomiell berechenbar ist.

Dann ist auch $\mathbb{P}_{\mathcal{D}_n}$ polynomiell berechenbar:

$$\mathbb{P}_{\mathcal{D}_n}(x) = \mu_{\mathcal{D}_n}(x) - \mu_{\mathcal{D}_n}(x-1).$$

Verteilungsprobleme
Polynomiell berechenbare Verteilungen

Polynomiell berechenbare Verteilung

$\mathcal{D} = (\mathcal{D}_n)$ heißt *polynomiell berechenbar*, wenn die *kumulierte Wahrscheinlichkeit*

$$\mu_{\mathcal{D}_n}(x) := \sum_{\substack{y \leq x \\ |y|=n}} \mathbb{P}_{\mathcal{D}_n}(y).$$

für alle $n \in \mathbb{N}$ polynomiell berechenbar ist.

Dann ist auch $\mathbb{P}_{\mathcal{D}_n}$ polynomiell berechenbar:

$$\mathbb{P}_{\mathcal{D}_n}(x) = \mu_{\mathcal{D}_n}(x) - \mu_{\mathcal{D}_n}(x-1).$$

Die Klasse $\mathtt{dist}\mathrm{NP}$

Die Klasse $\mathtt{dist}\mathrm{NP}$

$\langle L, \mathcal{D} \rangle$ liegt in $\mathtt{dist}\mathrm{NP}$, wenn gilt:

- $L \in \mathrm{NP}$,
- $\mathcal{D}$ ist polynomiell berechenbar.

Es gilt: $\mathtt{dist}\mathrm{NP} \subseteq \mathrm{NP}$.

Die Klasse $\mathtt{dist}\mathrm{NP}$

Die Klasse $\mathtt{dist}\mathrm{NP}$

$\langle L, \mathcal{D} \rangle$ liegt in $\mathtt{dist}\mathrm{NP}$, wenn gilt:

- $L \in \mathrm{NP}$,
- $\mathcal{D}$ ist polynomiell berechenbar.

Es gilt: $\quad \mathtt{dist}\mathrm{NP} \subseteq \mathrm{NP}$.

Überblick

1 Motivation

2 Levins Theorie (distP, distNP)
- Verteilungsprobleme und die Klasse distNP
- Die Klasse distP
- Average-Case Reduktion und distNP-Vollständigkeit
- Satz von Levin

3 Smoothed Analysis
- Einführung
- Verallgemeinerte Definition
- Smoothed Analysis eines SSSP-Algorithmus

4 Zusammenfassung

Die Klasse $\mathtt{dist}\mathrm{P}$

Durchschnittlich polynomiell berechenbar

Erster Ansatz

$\langle L, \mathcal{D} \rangle$ heißt *durchschnittlich polynomiell berechenbar*, wenn es einen Algorithmus $\mathcal{A}$ für dieses Problem gibt, für den gilt:

$$\mathbb{E}_{x \in \mathcal{D}_n}[T_{\mathcal{A}}(x)] \leq p(|x|) \quad \forall n \in \mathbb{N}.$$

Hierbei:

- $T_{\mathcal{A}}(x)$ Rechenzeit bei Eingabe x mit $|x| = n$,
- $p(\cdot)$ Polynom.

Die Klasse $\mathtt{dist}\mathrm{P}$
Durchschnittlich polynomiell berechenbar

Erster Ansatz nicht robust, betrachte Algorithmus $\mathcal{A}$ mit:

$$T_{\mathcal{A}}(\underbrace{00\ldots00}_{n \text{ Nullen}}) = 2^n, \quad T_{\mathcal{A}}(x) = n \quad \text{sonst.}$$

(Eingabelänge n, $\mathcal{D}_n$ Gleichverteilung $\Rightarrow \mathbb{P}_{\mathcal{D}_n}(x) = 2^{-n}$)

$$\begin{aligned}\mathbb{E}_{x\in\mathcal{D}_n}(T_{\mathcal{A}}(x)) &= \sum_{x\in\{0,1\}^n} \mathbb{P}_{\mathcal{D}_n}(x) * T_{\mathcal{A}}(x) \\ &= (1-2^{-n})\cdot n + 2^{-n}\cdot 2^n \leq n+1.\end{aligned}$$

$$\begin{aligned}\mathbb{E}_{x\in\mathcal{D}_n}(T_{\mathcal{A}}^2(x)) &= \sum_{x\in\{0,1\}^n} \mathbb{P}_{\mathcal{D}_n}(x) * T_{\mathcal{A}}^2(x) \\ &= (1-2^{-n})\cdot n^2 + 2^{-n}\cdot 2^{2n} \geq 2^n.\end{aligned}$$

Die Klasse distP
Durchschnittlich polynomiell berechenbar

Erster Ansatz nicht robust, betrachte Algorithmus $\mathcal{A}$ mit:

$$T_{\mathcal{A}}(\underbrace{00\ldots00}_{n \text{ Nullen}}) = 2^n , \quad T_{\mathcal{A}}(x) = n \quad \text{sonst.}$$

(Eingabelänge n, $\mathcal{D}_n$ Gleichverteilung $\Rightarrow \mathbb{P}_{\mathcal{D}_n}(x) = 2^{-n}$)

$$\begin{aligned} \mathbb{E}_{x\in\mathcal{D}_n}(T_{\mathcal{A}}(x)) &= \sum_{x\in\{0,1\}^n} \mathbb{P}_{\mathcal{D}_n}(x) * T_{\mathcal{A}}(x) \\ &= (1-2^{-n}) \cdot n + 2^{-n} \cdot 2^n \leq n+1 . \end{aligned}$$

$$\begin{aligned} \mathbb{E}_{x\in\mathcal{D}_n}(T^2_{\mathcal{A}}(x)) &= \sum_{x\in\{0,1\}^n} \mathbb{P}_{\mathcal{D}_n}(x) * T^2_{\mathcal{A}}(x) \\ &= (1-2^{-n}) \cdot n^2 + 2^{-n} \cdot 2^{2n} \geq 2^n . \end{aligned}$$

Die Klasse distP
Durchschnittlich polynomiell berechenbar

Erster Ansatz nicht robust, betrachte Algorithmus $\mathcal{A}$ mit:

$$T_{\mathcal{A}}(\underbrace{00\ldots00}_{n \text{ Nullen}}) = 2^n, \quad T_{\mathcal{A}}(x) = n \quad \text{sonst.}$$

(Eingabelänge n, $\mathcal{D}_n$ Gleichverteilung $\Rightarrow \mathbb{P}_{\mathcal{D}_n}(x) = 2^{-n}$)

$$\begin{aligned}
\mathbb{E}_{x \in \mathcal{D}_n}(T_{\mathcal{A}}(x)) &= \sum_{x \in \{0,1\}^n} \mathbb{P}_{\mathcal{D}_n}(x) * T_{\mathcal{A}}(x) \\
&= (1 - 2^{-n}) \cdot n + 2^{-n} \cdot 2^n \leq n + 1.
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_{x \in \mathcal{D}_n}(T^2_{\mathcal{A}}(x)) &= \sum_{x \in \{0,1\}^n} \mathbb{P}_{\mathcal{D}_n}(x) * T^2_{\mathcal{A}}(x) \\
&= (1 - 2^{-n}) \cdot n^2 + 2^{-n} \cdot 2^{2n} \geq 2^n.
\end{aligned}$$

Die Klasse distP

Durchschnittlich polynomiell berechenbar

Robuste Definition

$\langle L, \mathcal{D} \rangle$ heißt *durchschnittlich polynomiell berechenbar*, wenn es einen Algorithmus $\mathcal{A}$ für dieses Problem gibt, für den gilt:

$$\mathbb{E}_{x \in \mathcal{D}_n} \left[\frac{T_{\mathcal{A}}(x)^{\varepsilon}}{|x|} \right] \leq d \quad \text{für } n \in \mathbb{N}.$$

Hierbei:

- $T_{\mathcal{A}}(x)$ Rechenzeit bei Eingabe x mit $|x| = n$,
- $\varepsilon, d > 0$ Konstanten.

Die Klasse $\mathtt{dist}\mathrm{P}$

Die Klasse $\mathtt{dist}\mathrm{P}$

$\langle L, \mathcal{D} \rangle$ liegt in $\mathtt{dist}\mathrm{P}$, wenn gilt:

- $\langle L, \mathcal{D} \rangle \in \mathtt{dist}\mathrm{NP}$,
- $\langle L, \mathcal{D} \rangle$ ist durchschnittlich polynomiell berechenbar.

Es gilt: $\mathrm{P} \subseteq \mathtt{dist}\mathrm{P} \overset{?}{\approx} \mathtt{dist}\mathrm{NP} \subseteq \mathrm{NP}$.

Die Klasse $\mathtt{dist}\mathrm{P}$

Die Klasse $\mathtt{dist}\mathrm{P}$

$\langle L, \mathcal{D} \rangle$ liegt in $\mathtt{dist}\mathrm{P}$, wenn gilt:

- $\langle L, \mathcal{D} \rangle \in \mathtt{dist}\mathrm{NP}$,
- $\langle L, \mathcal{D} \rangle$ ist durchschnittlich polynomiell berechenbar.

Es gilt: $\mathrm{P} \subseteq \mathtt{dist}\mathrm{P} \overset{?}{\approx} \mathtt{dist}\mathrm{NP} \subseteq \mathrm{NP}.$

Überblick

1 Motivation

2 Levins Theorie (distP, distNP)
- Verteilungsprobleme und die Klasse distNP
- Die Klasse distP
- Average-Case Reduktion und distNP-Vollständigkeit
- Satz von Levin

3 Smoothed Analysis
- Einführung
- Verallgemeinerte Definition
- Smoothed Analysis eines SSSP-Algorithmus

4 Zusammenfassung

Average-Case Reduktion
Transformationen

Transformation

- Verteilungsproblem $\langle L_1, \mathcal{D}_1 \rangle$, Sprache L_2,
- Abbildung $f\colon L_1 \to L_2$, also $x \in L_1 \Leftrightarrow f(x) \in L_2$.

Dann heißt f *Transformation* und es gilt:

- $\langle L_2, f \circ \mathcal{D} \rangle$ ist ein Verteilungsproblem,
- $\mathbb{P}_{f \circ \mathcal{D}}(y) := \sum_{x \in f^{-1}(y)} \mathbb{P}_{\mathcal{D}}(x)$.

Average-Case Reduktion
Definition

Average-Case Reduktion

- Verteilungsprobleme $\langle L_1, \mathcal{D}_1 \rangle$, $\langle L_2, \mathcal{D}_2 \rangle$,
- Transformation $f\colon L_1 \to L_2$.

f ist eine *Average-Case Reduktion* und $\langle L_1, \mathcal{D}_1 \rangle \leq \langle L_2, \mathcal{D}_2 \rangle$, wenn es Polynome $p(\cdot)$ und $q(\cdot)$ gibt, so dass:

- f ist polynomiell berechenbar,
- *Längentreue*: $|f(x)| = p(|x|)$,
- *Dominanz*: $\mathbb{P}_{f \circ \mathcal{D}_1}(f(x)) \leq q(|x|) \cdot \mathbb{P}_{\mathcal{D}_2}(f(x))$.

$\mathtt{dist}\mathrm{NP}$-Vollständigkeit

Satz

$$\left.\begin{array}{l}\langle L_1, \mathcal{D}_1\rangle \leq \langle L_2, \mathcal{D}_2\rangle \\ \langle L_2, \mathcal{D}_2\rangle \in \mathtt{dist}\mathrm{P}\end{array}\right\} \Rightarrow \langle L_1, \mathcal{D}_1\rangle \in \mathtt{dist}\mathrm{P}.$$

$\mathtt{dist}\mathrm{NP}$-Vollständigkeit

$\langle L_1, \mathcal{D}_1\rangle \in \mathtt{dist}\mathrm{NP}$ ist $\mathtt{dist}\mathrm{NP}$-*vollständig*, wenn gilt:

Für alle $\langle L_2, \mathcal{D}_2\rangle \in \mathtt{dist}\mathrm{NP}$ gilt: $\langle L_2, \mathcal{D}_2\rangle \leq \langle L_1, \mathcal{D}_1\rangle$.

$\mathtt{dist}\mathrm{NP}$-Vollständigkeit

Satz

$$\left.\begin{array}{l}\langle L_1, \mathcal{D}_1\rangle \le \langle L_2, \mathcal{D}_2\rangle \\ \langle L_2, \mathcal{D}_2\rangle \in \mathtt{dist}\mathrm{P}\end{array}\right\} \Rightarrow \langle L_1, \mathcal{D}_1\rangle \in \mathtt{dist}\mathrm{P}.$$

$\mathtt{dist}\mathrm{NP}$-Vollständigkeit

$\langle L_1, \mathcal{D}_1\rangle \in \mathtt{dist}\mathrm{NP}$ ist $\mathtt{dist}\mathrm{NP}$-*vollständig*, wenn gilt:

Für alle $\langle L_2, \mathcal{D}_2\rangle \in \mathtt{dist}\mathrm{NP}$ gilt: $\langle L_2, \mathcal{D}_2\rangle \le \langle L_1, \mathcal{D}_1\rangle$.

Überblick

1. Motivation
2. Levins Theorie (distP, distNP)
   - Verteilungsprobleme und die Klasse distNP
   - Die Klasse distP
   - Average-Case Reduktion und distNP-Vollständigkeit
   - Satz von Levin
3. Smoothed Analysis
   - Einführung
   - Verallgemeinerte Definition
   - Smoothed Analysis eines SSSP-Algorithmus
4. Zusammenfassung

Satz von Levin

Existenz eines distNP-vollständigen Problems

Das universelle Verteilungsproblem $\langle U, \mathcal{U}_n \rangle$ ist distNP-vollständig.

- U ist die Sprache aller Tupel $\langle M, x, 1^t \rangle$ mit NDTM M, welche die Eingabe x nach höchstens t Schritten akzeptiert.
- In $\mathcal{U}_n$ wird ein Tupel $\langle M, x, 1^t \rangle$ der Länge n wie folgt zufällig gewählt:
  1. Kodierung von M aus allen Wörtern der Länge $\leq \log n$
  2. $t \in \{0, \ldots, n - |M|\}$ binär kodiert
  3. $x \in \{0, 1\}^{n-t-|M|}$

Satz von Levin

Vorbemerkungen(1)

$U \in \mathrm{NP}$, denn:

- Rate die nichtdeterministischen Zustandsübergänge von M und simuliere t Schritte der Abarbeitung von x.

U ist NP-schwer, denn:

- Sei $L \in \mathrm{NP}$ und M eine NDTM mit $L(M) = L$, deren Laufzeit von einem Polynom p beschränkt ist.
- Betrachte für Eingabe x die Reduktion $f(x) = \langle M, x, 1^{p(|x|)} \rangle$

Obige Reduktion zeigt $U \in \mathrm{NPC}$, betrachtet jedoch keine Verteilung und ist somit keine Average-Case-Reduktion.

Satz von Levin

Vorbemerkungen(2)

- Dominanz-Bedingung:
  $\forall n \in \mathbb{N} \quad x \in \{0,1\}^n : \mathbb{P}_{f \circ \mathcal{D}_n}(f(x)) \leq q(n)\mathbb{P}_{\mathcal{U}_{p(n)}}(f(x))$
- $\mathbb{P}_{\mathcal{U}_n}(f(x)) \leq 2^{-n}$ gemäß Definition
- Bei Eingabe x könnte es einen Spitzenwert von $\mathbb{P}_{\mathcal{D}_n}$ geben, sodass $\mathbb{P}_{f \circ \mathcal{D}_n}(f(x)) >> 2^{-n}$
- $\Rightarrow$ Ziel: Spitzenwerte einer Verteilung eliminieren

Satz von Levin
Beweis(1)

Lemma: Spitzenwert-Eliminierung

Sei $\mathcal{D}_n$ eine polynomiell berechenbare Verteilung. Dann existiert eine polynomiell berechenbare Funktion $g\colon \{0,1\}^* \to \{0,1\}^*$ mit:

1. g ist injektiv: $g(x) = g(y) \Leftrightarrow x = y$,
2. $\forall x \in \{0,1\}^* : |g(x)| \leq |x| + 1$,
3. $\forall x \in \{0,1\}^n : \mathbb{P}_{g \circ \mathcal{D}_n}(x) \leq 2^{-n+1}$.

Beweisidee:

- Für $x \in \{0,1\}^*$ sei $h(x)$ das längste gemeinsame Präfix der Binärrepräsentationen von $\mu_{\mathcal{D}_n}(x)$ und $\mu_{\mathcal{D}_n}(x-1)$.
- Definiere $g(x) := \begin{cases} 0x & \text{wenn } \mathbb{P}_{\mathcal{D}_n}(x) \leq 2^{-|x|} \\ 1h(x) & \text{sonst} \end{cases}$

Satz von Levin
Beweis(1)

Lemma: Spitzenwert-Eliminierung

Sei $\mathcal{D}_n$ eine polynomiell berechenbare Verteilung. Dann existiert eine polynomiell berechenbare Funktion $g\colon \{0,1\}^* \to \{0,1\}^*$ mit:

1. g ist injektiv: $g(x) = g(y) \Leftrightarrow x = y$,
2. $\forall x \in \{0,1\}^* : |g(x)| \leq |x| + 1$,
3. $\forall x \in \{0,1\}^n : \mathbb{P}_{g \circ \mathcal{D}_n}(x) \leq 2^{-n+1}$.

Beweisidee:

- Für $x \in \{0,1\}^*$ sei $h(x)$ das längste gemeinsame Präfix der Binärrepräsentationen von $\mu_{\mathcal{D}_n}(x)$ und $\mu_{\mathcal{D}_n}(x-1)$.
- Definiere $g(x) := \begin{cases} 0x & \text{wenn } \mathbb{P}_{\mathcal{D}_n}(x) \leq 2^{-|x|} \\ 1h(x) & \text{sonst} \end{cases}$

Satz von Levin

Beweis(2)

Konstruktion

- Sei $\langle L, \mathcal{D}_n \rangle \in \mathtt{dist}\mathrm{NP}$ und M eine NDTM mit $L(M) = L$ und $T_M(n) \leq t(n)$ für ein Polynom t.
- $f : L \to U, \quad x \mapsto \langle M', g(x), 1^k \rangle$
  liefert eine Average-case-Reduktion, wobei:
  1. Bei Eingabe y rät und überprüft M' x, sodass $y = g(x)$ und simuliert dann M auf Eingabe x
  2. g wie im Lemma
  3. $k = t(n) + n + 1 - |g(x)|$

Satz von Levin

Beweis(3)

Korrektheit

z.z.: $\forall x \in \{0,1\}^* : x \in L \Leftrightarrow f(x) \in U$

$x \in L$
$\Leftrightarrow M$ akzeptiert x
$\Leftrightarrow M'$ akzeptiert $g(x)$ in höchstens t(|x|) Schritten, mit g injektiv
$\Leftrightarrow \langle M', g(x), 1^k \rangle \in U$, wobei $k \geq t(|x|)$
$\Leftrightarrow f(x) \in U$

Satz von Levin

Beweis(4)

Längentreue

z.z.: $\forall x \in \{0,1\}^n : |f(x)| = p(n)$

$$
\begin{aligned}
|f(x)| &= |\langle M', g(x), 1^k \rangle| = |M'| + |g(x)| + k \\
&= |M'| + |g(x)| + t(n) + n + 1 - |g(x)| \\
&= t(n) + n + 1 + |M'|
\end{aligned}
$$

Satz von Levin

Beweis(5)

Dominanz

z.z.: $\forall n \in \mathbb{N} \quad x \in \{0,1\}^n : \mathbb{P}_{f \circ \mathcal{D}_n}(f(x)) \leq q(n)\mathbb{P}_{\mathcal{U}_{p(n)}}(f(x))$

$$\mathbb{P}_{f \circ \mathcal{D}_n}(f(x)) = \mathbb{P}_{f \circ \mathcal{D}_n}(\langle M', g(x), 1^k \rangle) = \mathbb{P}_{g \circ \mathcal{D}_n}(y) \leq 2^{-|y|+1}$$

$$\mathbb{P}_{\mathcal{U}_{p(n)}}(f(x)) = \mathbb{P}_{\mathcal{U}_m}(\langle M', y, 1^k \rangle) \geq \frac{1}{2^{\log m}} \cdot \frac{1}{2^{|y|}} \cdot \frac{1}{m} = \frac{1}{m^2} \cdot \frac{1}{2^{|y|}}$$

$$2^{-|y|+1} = \frac{2}{2^{|y|}} \leq q(n) \cdot \frac{1}{m^2} \cdot \frac{1}{2^{|y|}} \Leftrightarrow 2 \leq q(n)\frac{1}{m^2} \Leftrightarrow 2m^2 \leq q(n)$$

Überblick

1. Motivation
2. Levins Theorie (distP, distNP)
   - Verteilungsprobleme und die Klasse distNP
   - Die Klasse distP
   - Average-Case Reduktion und distNP-Vollständigkeit
   - Satz von Levin
3. Smoothed Analysis
   - Einführung
   - Verallgemeinerte Definition
   - Smoothed Analysis eines SSSP-Algorithmus
4. Zusammenfassung

Was ist Smoothed Analysis?

Motivation

- Mischung aus Worst-Case- und Average-Case-Analyse.
- Worst-Case-Analyse oft zu pessimistisch.
- Average-Case-Analyse nicht realistisch, wenn nicht die richtige Verteilung angenommen wird.
- Smoothed Analysis wurde 2001 von D. A. Spielman und S. Teng vorgeschlagen.
- Spielman und Teng erhielten 2008 den Gödel-Preis für die Entwicklung der Smoothed Analysis.

Was ist Smoothed Analysis?

Idee

- Betrachte für jede Eingabe den Erwartungswert, wenn diese Eingabe einem gewissen “Rauschen” unterliegt.
- Maximiere über alle diese Erwartungswerte.
- Formal wird für jede Eingabe *x* eine Wahrscheinlichkeitsverteilung $\mathbb{P}_\sigma$ definiert, deren Wahrscheinlichkeit an der Stelle *x* maximal wird.
- $\mathbb{P}_\sigma$ durch ein $\sigma \in \mathbb{R}$ parametrisiert, sodass:
  - $\sigma \to 0 \Rightarrow$ Smoothed Analysis $\to$ Worst Case Analysis.
  - $\sigma \to \infty \Rightarrow$ Smoothed Analysis $\to$ Average Case Analysis.

Was ist Smoothed Analysis?
Ansatz für LINEAR PROGRAMMING(1)

Ein Algorithmus $\mathcal{A}(A, b, c)$ für LINEAR PROGRAMMING soll mittels Smoothed Analysis analysiert werden.
Dazu wird die Eingabematrix A verrauscht.

LINEAR PROGRAMMING:

1. gegeben: $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n, c \in \mathbb{R}^m$
2. gefordert: $Ax \leq b, x \geq 0$
3. gesucht: $x \in \mathbb{R}^m$, sodass $c^T x$ maximiert wird.

Was ist Smoothed Analysis?
Ansatz für LINEAR PROGRAMMING(2)

Die zufällig verrauschten Instanzen sind gegeben durch:

1. gegeben: $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n, c \in \mathbb{R}^m$
2. gefordert: $(A + \sigma G)x \leq b, x \geq 0$
3. gesucht: $x \in \mathbb{R}^m$, sodass $c^T x$ maximiert wird.

Hierbei ist $G \in \mathbb{R}^{n \times m}$ eine Matrix deren Einträge $g_{i,j}$ unabhängig voneinander gewählt sind. Jedes $g_{i,j}$ ist Φ-verteilt, also standard-normalverteilt (d.h. Gaußverteilung mit Erwartungswert 0 und Varianz 1).

Für festes b und c ist die smoothed analysierte Laufzeit:

$$T^{smoothed}_{\mathcal{A},\sigma}(n, m) := \max_{A \in \mathbb{R}^{n \times m}} \{\mathbb{E}_{G \in_\Phi \mathbb{R}^{n \times m}}[T_{\mathcal{A}}(A + \sigma G, b, c)]\}.$$

Smoothed Analysis des Simplex-Algorithmus

Simplex-Algorithmus

- In der Praxis effizienter Algorithmus zur Lösung von LINEAR PROGRAMMING.
- (Shadow vertex) Simplex ist der Algorithmus, mit dem Smoothed Analysis eingeführt wurde.
- $T^{smoothed}_{\mathrm{Simplex},\sigma}(n, m)$ ist polynomiell in n, m und $1/\sigma$.

Überblick

1. Motivation
2. Levins Theorie (distP, distNP)
   - Verteilungsprobleme und die Klasse distNP
   - Die Klasse distP
   - Average-Case Reduktion und distNP-Vollständigkeit
   - Satz von Levin
3. Smoothed Analysis
   - Einführung
   - Verallgemeinerte Definition
   - Smoothed Analysis eines SSSP-Algorithmus
4. Zusammenfassung

Verallgemeinerte Definition(1)

Verallgemeinerte Definition

- Raum aller Eingaben $X = \bigcup_{n\in\mathbb{N}} X_n$.
- System von Umgebungen S^x_δ, $\delta \in [0,\infty)$, von $x \in X$ mit:
  1. $S^x_0 = \{x\}$
  2. S^x_δ vergrößert sich monoton: $\delta < \delta' \;\Rightarrow\; S^x_\delta \subseteq S^x_{\delta'}$
- Wahrscheinlichkeitsmaß μ^x_σ: $X \to [0,1]$, $\sigma \in [0,\infty)$, mit:
  1. μ^x_σ ist abfallend: $y \in S^x_\delta, z \notin S^x_\delta \;\Rightarrow\; \mu^x_\sigma(y) > \mu^x_\sigma(z)$,
  2. μ^x_σ konzentriert sich um x: $\sigma < \sigma' \;\Rightarrow\; \mu^x_\sigma(S^x_\delta) > \mu^x_{\sigma'}(S^x_\delta)$,
  3. $\lim_{\sigma\to 0} \mu^x_\sigma(S^x_\delta) = 1$.
- Die **Smoothed Analysis Laufzeit** ist dann

$$T^{smoothed}_{\mathcal{A},\sigma}(n) := \max_{x\in X_n} \mathbb{E}_{y\in\mu^x_\sigma}[T_{\mathcal{A}}(y)].$$

Verallgemeinerte Definition(1)

Verallgemeinerte Definition

- Raum aller Eingaben $X = \bigcup_{n \in \mathbb{N}} X_n$.
- System von Umgebungen S_δ^x, $\delta \in [0, \infty)$, von $x \in X$ mit:
  1. $S_0^x = \{x\}$
  2. S_δ^x vergrößert sich monoton: $\delta < \delta' \;\Rightarrow\; S_\delta^x \;\subseteq\; S_{\delta'}^x$
- Wahrscheinlichkeitsmaß $\mu_\sigma^x \colon X \to [0, 1]$, $\sigma \in [0, \infty)$, mit:
  1. μ_σ^x ist abfallend: $y \in S_\delta^x, z \notin S_\delta^x \;\Rightarrow\; \mu_\sigma^x(y) \;>\; \mu_\sigma^x(z)$,
  2. μ_σ^x konzentriert sich um x: $\sigma < \sigma' \;\Rightarrow\; \mu_\sigma^x(S_\delta^x) \;>\; \mu_{\sigma'}^x(S_\delta^x)$,
  3. $\lim_{\sigma \to 0} \mu_\sigma^x(S_\delta^x) = 1$.
- Die **Smoothed Analysis Laufzeit** ist dann

$$T_{\mathcal{A},\sigma}^{smoothed}(n) := \max_{x \in X_n} \mathbb{E}_{y \in \mu_\sigma^x}[T_{\mathcal{A}}(y)].$$

Verallgemeinerte Definition(1)

Verallgemeinerte Definition

- Raum aller Eingaben $X = \bigcup_{n \in \mathbb{N}} X_n$.
- System von Umgebungen S_δ^x, $\delta \in [0, \infty)$, von $x \in X$ mit:
  1. $S_0^x = \{x\}$
  2. S_δ^x vergrößert sich monoton: $\delta < \delta' \Rightarrow S_\delta^x \subseteq S_{\delta'}^x$
- Wahrscheinlichkeitsmaß $\mu_\sigma^x \colon X \to [0, 1]$, $\sigma \in [0, \infty)$, mit:
  1. μ_σ^x ist abfallend: $y \in S_\delta^x, z \notin S_\delta^x \Rightarrow \mu_\sigma^x(y) > \mu_\sigma^x(z)$,
  2. μ_σ^x konzentriert sich um x: $\sigma < \sigma' \Rightarrow \mu_\sigma^x(S_\delta^x) > \mu_{\sigma'}^x(S_\delta^x)$,
  3. $\lim_{\sigma \to 0} \mu_\sigma^x(S_\delta^x) = 1$.
- Die **Smoothed Analysis Laufzeit** ist dann

$$T_{\mathcal{A},\sigma}^{smoothed}(n) := \max_{x \in X_n} \mathbb{E}_{y \in \mu_\sigma^x}[T_{\mathcal{A}}(y)].$$

Überblick

1. Motivation
2. Levins Theorie (distP, distNP)
   - Verteilungsprobleme und die Klasse distNP
   - Die Klasse distP
   - Average-Case Reduktion und distNP-Vollständigkeit
   - Satz von Levin
3. Smoothed Analysis
   - Einführung
   - Verallgemeinerte Definition
   - Smoothed Analysis eines SSSP-Algorithmus
4. Zusammenfassung

Goldberg's Algorithmus
Das Problem

Das Problem: Single Source Shortest Path (SSSP)

- $G = (V, E)$ ist ein beliebiger Graph.
- $c\colon E \to [0, 2^N - 1]$ ist eine Kostenfunktion (N-Bit-Integer).
- Gesucht ist der kürzeste Weg von einem gegebenen Knoten zu allen anderen.

Goldberg's Algorithmus
Schon gezeigt

Andrew Goldberg's Algorithmus

- Allgemein:

$$\mathcal{O}\Big(|E| + |V| + \sum_{v \in V} (N - \log \mathrm{minincost}(v) + 1)\Big).$$

- Worst-Case:

$$\mathcal{O}(|E| + N \cdot |V|).$$

- Average-Case:

$$\mathcal{O}(|E| + |V|).$$

Goldberg's Algorithmus

Die Smoothed-Variante

Die Smoothed-Variante

- Randomisiere die letzten $k \leq N$ Bits der Kantengewichte:

$$c_k : E \to [0, 2^N - 1].$$

- Randomisierter Teil der Kantengewichte:

$$\hat{c}_k : E \to [0, 2^k - 1].$$

Beispiel Für $c(e) = 100110110$ gilt:

$c_3(e) = 100110X_1X_2X_3$ und $\hat{c}_3(e) = X_1X_2X_3$.

Goldberg's Algorithmus
Die Smoothed-Variante

Die Smoothed-Variante

- Randomisiere die letzten $k \leq N$ Bits der Kantengewichte:

$$c_k : E \to [0, 2^N - 1].$$

- Randomisierter Teil der Kantengewichte:

$$\hat{c}_k : E \to [0, 2^k - 1].$$

Beispiel Für $c(e) = 100110110$ gilt:

$$c_3(e) = 100110X_1X_2X_3 \quad \text{und} \quad \hat{c}_3(e) = X_1X_2X_3.$$

Vorbetrachtungen
Bezeichnungen

Definitionen

- $c_k : E \to [0, 2^N - 1]$: Kostenfunktion, die letzten k Bits randomisiert.
- $\hat{c}_k : E \to [0, 2^k - 1]$: Nur die randomisierten k Bits.
- $\mathrm{minincost}(v)$: Kleinstes Gewicht einer eingehenden Kante.
- $\mathrm{inedges}(v)$: Menge aller eingehenden Kanten.
- $\mathrm{indeg}(v)$: Anzahl eingehender Kanten.
- $\mathrm{zeroes}(a)$: Anzahl führender Nullen.

Lemma

- (1) $\log(c_k(e)) \geq k - 1 - \mathrm{zeroes}(\hat{c}_k(e))$.
- (2) $\mathbb{E}[\mathrm{zeroes}(\hat{c}_k(e))] \leq 2$.

Vorbetrachtungen

Vorbetrachtung (1)

Abschätzung der Kostenfunktion

Sei $v \in V$ beliebig und $e \in E$ eine zu v inzidente Kante. Dann gilt:

$$\begin{aligned}\log(c_k(e)) &\geq \log(\hat{c}_k(e)) \\ &\geq \log(2^{k-1-\mathrm{zeroes}(\hat{c}_k(e))}) \\ &= k-1-\mathrm{zeroes}(\hat{c}_k(e)).\end{aligned}$$

Wir haben also: (1) $\log(c_k(e)) \geq k-1-\mathrm{zeroes}(\hat{c}_k(e))$.

Vorbetrachtungen
Vorbetrachtung (2)

Abschätzung der erwarteten Nullen

Die Wahrscheinlichkeit für $0 \leq n \leq k$ Nullen ist:

$$\mathbb{P}(\text{zeroes}(\hat{c}_k(e)) \geq n) = \tfrac{1}{2} \cdot \dots \cdot \tfrac{1}{2} = \left(\tfrac{1}{2}\right)^n.$$

Damit ergibt sich die erwartete Anzahl:

$$\begin{aligned} \mathbb{E}[\text{zeroes}(\hat{c}_k(e))] &= \sum_{n=0}^{k} \mathbb{P}(\text{zeroes}(\hat{c}_k(e)) \geq n) \\ &= \sum_{n=0}^{k} \left(\tfrac{1}{2}\right)^n \leq \sum_{n=0}^{\infty} \left(\tfrac{1}{2}\right)^n \\ &= \frac{1}{1-\frac{1}{2}} = 2. \end{aligned}$$

Wir haben also: (2) $\mathbb{E}[\text{zeroes}(\hat{c}_k(e))] \leq 2.$

Smoothed Analysis des Algorithmus

Smoothed Analysis

Wir beginnen mit folgender Abschätzung:

$$\begin{aligned} N - \log(\text{minincost}(v)) &\overset{(1)}{\leq} N - 1 - k + \max\{\text{zeroes}(\hat{c}_k(e)) : e \in \text{inedges}(v)\} \\ &\leq N - 1 - k + \sum_{e \in \text{inedges}(v)} \text{zeroes}(\hat{c}_k(e)). \end{aligned}$$

Damit ergibt sich der Erwartungswert:

$$\begin{aligned} (*) \quad \mathbb{E}[N - \log \text{minincost}(v)] &= N - 1 - k + \sum_{e \in \text{inedges}(v)} \mathbb{E}[\text{zeroes}(\hat{c}_k(e))] \\ &\overset{(2)}{\leq} N - 1 - k + \sum_{e \in \text{inedges}(v)} 2 \\ &= N - 1 - k + 2 \cdot \text{indeg}(v). \end{aligned}$$

Smoothed Analysis des Algorithmus

Smoothed Analysis

Schließlich haben wir dann:

$$\begin{aligned}
& \mathbb{E}\left[|E| + |V| + \sum_{v \in V}(N - \log \operatorname{minincost}(v) + 1)\right] \\
= \quad & |E| + |V| + \sum_{v \in V}(\mathbb{E}[N - \log \operatorname{minincost}(v)] + 1) \\
\overset{(*)}{\leq} \quad & |E| + |V| + \sum_{v \in V}(N - 1 - k + 2\operatorname{indeg}(v) + 1) \\
= \quad & |E| + |V| + \sum_{v \in V}(N - k) + \sum_{v \in V} 2\operatorname{indeg}(v) \\
= \quad & |E| + |V| + |V| \cdot (N - k) + 2 \cdot |E| \\
= \quad & 3 \cdot |E| + |V| \cdot (N - k + 1) \\
\in \quad & \mathcal{O}(|E| + |V| \cdot (N - k)).
\end{aligned}$$

Das Ergebnis der Analyse
Überblick

- Allgemein:
$$\mathcal{O}\Big(|E| + |V| + \sum_{v \in V} (N - \log \mathrm{minincost}(v) + 1)\Big).$$
- Smoothed Analysis:
$$\mathcal{O}(|E| + |V| \cdot (N - k)).$$
- Für $k \to 0$ ergibt sich der Worst-Case:
$$\mathcal{O}(|E| + N \cdot |V|).$$
- Für $k \to N$ ergibt sich der Average-Case:
$$\mathcal{O}(|E| + |V|).$$

Das Ergebnis der Analyse

Überblick

- Allgemein:

$$\mathcal{O}\Big(|E| + |V| + \sum_{v \in V} (N - \log \operatorname{minincost}(v) + 1)\Big).$$

- Smoothed Analysis:

$$\mathcal{O}(|E| + |V| \cdot (N - k)).$$

- Für $k \to 0$ ergibt sich der Worst-Case:

$$\mathcal{O}(|E| + N \cdot |V|).$$

- Für $k \to N$ ergibt sich der Average-Case:

$$\mathcal{O}(|E| + |V|).$$

Zusammenfassung

Zusammenfassung

- Klassifikation von durchschnittlich polynomiell lösbaren Problemen: $\mathtt{dist}\mathrm{P}$, $\mathtt{dist}\mathrm{NP}$.
- Levin's Theorem: Es gibt ein $\mathtt{dist}\mathrm{NP}$-vollständiges Problem.
- Smoothed Analysis zur realitätsnäheren Laufzeitanalyse zwischen Worst-Case- und Average-Case-Analyse.
- Simplex-Algorithmus (für LINEAR PROGRAMMING) hat polynomielle Laufzeit mit Smoothed Analysis.

Quellenangaben I

Sanjeev Arora und Boaz Barak
Computational Complexity: A Modern Approach.
Cambridge University Press, 2009.