Algorithmen für NP-harte Probleme

Tim Baumann, http://timbaumann.info/uni-spicker

Dies ist eine Zusammenfassung zur gleichnamigen Vorlesung von Professor Dr. Torben Hagerup im Sommersemester 2017.

Def. Ein **Optimierungsproblem** ist ein Tupel $(\mathcal{X}, \mathcal{F}, Z)$ wobei

- $\mathcal{X}$ eine Menge von **Instanzen**,
- $\mathcal{F}$ eine Abbildung ist, welche jeder Instanz x eine Menge $\mathcal{F}(x)$ von **möglichen Lösungen** zuordnet und
- Z eine reellwertige Abbildung (die **Zielfunktion**) ist, die jedem $x \in \mathcal{X}$ und $y \in \mathcal{F}(x)$ einen *Zielwert* zuordnet.

Def. Eine **optimale Lösung** eines Optimierungsproblems $(\mathcal{X}, \mathcal{F}, Z)$ zu einer Instanz $x \in \mathcal{X}$ ist ein $y \in \mathcal{F}(x)$ mit

$$Z(x, y) = \min_{y \in \mathcal{F}(x)} Z(x, y) =: \mathrm{Opt}(x).$$

Def. Ein Algorithmus **löst** ein Optimierungsproblem $(\mathcal{X}, \mathcal{F}, Z)$, falls er für jedes $x \in \mathcal{X}$

- eine optimale Lösung $y \in \mathcal{F}(x)$ berechnet, falls solch eine existiert,
- „unmöglich“ ausgibt, falls keine Lösung existiert oder
- „möglich, aber keine optimale Lösung“ sonst.

Def. NPO ist die Klasse aller Optimierungsprobleme $(\mathcal{X}, \mathcal{F}, Z)$ mit

- $\mathcal{X} \in P$
- Es gibt ein Polynom p, sodass für alle $x \in X$
  - $|y| \leq p(|x|)$ für alle $y \in \mathcal{F}(x)$ und
  - für alle Wörter w der Länge $|w| \leq p(|x|)$ in polynomieller Zeit (in $|x|$) entscheidbar ist, ob $w \in \mathcal{F}(x)$.
- Die Funktion Z ist in polynomieller Zeit berechenbar.

Def. PO $\subseteq$ NPO ist die Subklasse für die ein Lösungsalgorithmus existiert, der in Polynomialzeit läuft.

Beob. PO = NPO $\implies$ P = NP

Def. Sei $\mathcal{P} = (\mathcal{X}, \mathcal{F}, Z)$ ein Optimierungsproblem.

- Das zugeh. **Auswertungsproblem** $\mathcal{P}_E$ ist: Gegeben $x \in \mathcal{X}$,
  - berechne $\mathrm{Opt}(x)$, falls x eine optimale Lösung besitzt,
  - berechne $\inf \mathcal{F}(x) \in \mathbb{R} \cup \{-\infty\}$, falls es Lösungen gibt, aber keine optimale
  - oder gib „unmöglich“ aus, falls keine Lösung existiert.
- Das zugeh. **Entscheidungsproblem** $\mathcal{P}_D$ ist: Gegeben $x \in \mathcal{X}$ und $k \in \mathbb{Q}$, gibt es eine Lösung $y \in \mathcal{F}(x)$ mit $Z(x, y) \leq k$?

Def. $\mathcal{P} \in \mathrm{NPO} \implies \mathcal{P}_D \in \mathrm{NP}$

Def.
- Ein Entscheidungsproblem $\mathcal{P}_1$ ist (in Polynomialzeit) auf ein Entscheidungsproblem $\mathcal{P}_2$ **many-to-one-reduzierbar** (notiert $\mathcal{P}_1 \leq_m \mathcal{P}_2$) falls eine (in Polynomialzeit) berechenbare Funktion $f : \{\text{Instanzen von } \mathcal{P}_1\} \to \{\text{Instanzen von } \mathcal{P}_2\}$ existiert, sodass die Antwort auf eine Instanz x von $\mathcal{P}_1$ gleich der Antwort auf die Instanz $f(x)$ von $\mathcal{P}_2$ ist.
- Ein Problem $\mathcal{P}_1$ ist (in Polynomialzeit) auf ein Problem $\mathcal{P}_2$ **Turing-reduzierbar** (notiert $\mathcal{P}_1 \leq_T \mathcal{P}_2$) falls ein Algorithmus existiert, der unter Verwendung eines Orakels für $\mathcal{P}_2$ das Problem $\mathcal{P}_1$ (in Polynomialzeit) löst.

Beob. $\mathcal{P}_1 \leq_m \mathcal{P}_2 \implies \mathcal{P}_1 \leq_T \mathcal{P}_2$

Beob. Für $\mathcal{P} \in \mathrm{NPO}$ gilt $\mathcal{P}_D \leq_T \mathcal{P}_E \leq_T \mathcal{P}$.

Satz. Habe $\mathcal{P} = (\mathcal{X}, \mathcal{F}, Z) \in \mathrm{NPO}$ eine Zielfunktion mit Werten in den ganzen Zahlen.

- Es gilt $\mathcal{P}_D \equiv_T \mathcal{P}_E$.
- Angenommen, $\mathcal{P}_D$ ist NP-vollständig. Dann gilt $\mathcal{P} \equiv_T \mathcal{P}_D$.

Def. Ein Optimierungsproblem $\mathcal{P}$ heißt **NP-hart**, falls $\mathcal{P}' \leq_T \mathcal{P}$ für jedes Entscheidungsproblem $\mathcal{P}'$ in NP.

Beob. $\mathcal{P} \in \mathrm{NPO}$, $\mathcal{P}$ NP-vollständig $\implies$ $\mathcal{P}$ NP-hart

Die Gierige Strategie

Problem (Cabin Manager's Problem)**.** MIS auf Intervallgraphen

Algorithmus (**Greedy MIS für Intervallgraphen**)**.** Beginne mit $C := \emptyset$, füge dann wiederholt gierig das vom aktuellen C unabhängige Intervall mit dem kleinsten Endpunkt zu C hinzu, bis es kein solches Intervall mehr gibt.

Satz. Dieser Algorithmus berechnet tatsächlich ein MIS.

Algorithmus (**Greedy Minimum Makespan Scheduling**)**.** Gehe die Jobs in nach Dauer absteigender Reihenfolge durch, weise jeden Job dem Arbeiter zu, der bisher am wenigsten ausgelastet ist.

Probleme

Problem (**Maximum Independent Set**, MIS)**.** Geg. einen unger. Graphen $G = (V, E)$, berechne eine *unabhängige Menge* $M \subseteq V$, d. h.

$$\forall\, v \in M \,:\, \forall\, w \in V \,:\, (v, w) \in E \implies w \notin M,$$

die maximale Größe $|M|$ unter allen unabhängigen Mengen besitzt.

Problem (**Minimum Vertex Cover**, MVC)**.** Geg. einen unger. Graphen $G = (V, E)$, berechne eine *Knotenüberdeckung* C, d. h.

$$\forall\, v, w \in V \,:\, (v, w) \in E \implies v \in C \vee w \in C,$$

die minimale Größe $|C|$ unter allen Knotenüberdeckungen besitzt.

Def. Ein **Intervallmodell** eines Graphen $G = (V, E)$ ist eine Abbildung $\phi : E \to \{[a, b] \mid a, b \in \mathbb{Q}\}$, sodass

$$\forall\, v \neq w \in V \,:\, (v, w) \in E \iff \phi(v) \cap \phi(w) \neq \emptyset.$$

Ein Graph heißt **Intervallgraph**, falls er ein Intervallmodell besitzt.

Problem (**Minimum Makespan Scheduling**)**.** Seien $p, n \in \mathbb{N}$ und $l_1, \ldots, l_n \in \mathbb{R}_{>0}$ gegeben. Für $f : \{1, \ldots, n\} \to \{1, \ldots, p\}$ setze

$$t(f) := \max_{1 \leq i \leq p} \sum_{j \in f^{-1}(i) l_j}.$$

Berechne das f, für das $t(f)$ minimal wird!

Interpretation. p ist die Anzahl von *Arbeitern*, $l_1, \ldots, l_n$ sind die Längen von zu erledigenden *Jobs* und $t(f)$ ist die *Gesamtdauer* bei der durch f gegebenen Verteilung der Jobs auf die Arbeiter an.