Vorlesung

Kombinatorische Optimierung

(Wintersemester 2014/15)

Kapitel 5: NP-schwierige Probleme

Volker Kaibel

Otto-von-Guericke Universität Magdeburg

(Version vom 13. April 2015)

Rucksack Problem

Problem 5.1 (0/1-Knapsack Problem)

Instanz: *Gewichte* $g \in \mathbb{Q}_+^n$, *Gewichtsschranke* $G \in \mathbb{Q}_+$, *Werte* $w \in \mathbb{Q}_+^n$

Aufgabe: *Finde* $I^\star \subseteq [n]$ *mit* $g(I^\star) \leq G$, *so dass*

$$w(I^\star) = \max\{w(I) \mid I \subseteq [n], g(I) \leq G\}$$

ist.

Problem des Handlungsreisenden

Problem 5.2 (Traveling Salesman Problem (TSP))

Instanz: *Kantenlängen* $c \in \mathbb{Q}^{E_n}$ *des vollständigen Graphen* $K_n = (V_n, E_n)$ *auf* n *Knoten*

Aufgabe: *Finde Hamilton-Kreis* $H^\star \subseteq E_n$ *mit*

$$c(H^\star) = \min\{c(H) \mid H \subseteq E_n \text{ Hamilton-Kreis}\}\,.$$

Maximale Schnitte

Problem 5.3 (Maximum Cut Problem)

Instanz: *Graph* $G = (V, E)$ *mit Kantengewichten* $c \in \mathbb{Q}^E$

Aufgabe: *Finde* $S^\star \subseteq V$ *mit*

$$c(\delta(S^\star)) = \max\{c(\delta(S)) \mid S \subseteq V\}\,.$$

Quadratische 0/1-Optimierung

Problem 5.4 (Quadratische 0/1-Optimierung)

Instanz: *Kostenmatrix* $(c_{ij}) \in \mathbb{Q}^{n \times n}$

Aufgabe: *Finde* 0/1-*Vektor* $x^\star \in \{0,1\}^n$ *mit*

$$\sum_{i,j=1}^{n} c_{ij} x_i^\star x_j^\star = \min\{ \sum_{i,j=1}^{n} c_{ij} x_i x_j \mid x \in \{0,1\}^n \} .$$

Quadratisches Zuordnungsproblem

Problem 5.5 (Quadratic Assignment Problem (QAP))

Instanz: *Zielfunktionskoeffizienten* $d_{ijkl} \in \mathbb{Q}$ *(*$i, j, k, l \in [n]$*)*

Aufgabe: *Finde* $n \times n$ *Permutationsmatrix* $(x^\star_{ij}) \in \{0,1\}^{n \times n}$ *mit*

$$\sum_{i,j,k,l=1}^{n} d_{ijkl} x^\star_{ij} x^\star_{kl}$$

$$= \min \Big\{ \sum_{i,j,k,l=1}^{n} d_{ijkl} x_{ij} x_{kl} \,\Big|\, (x_{ij})^{n \times n} \textit{ Permut.-Matrix} \Big\} .$$

Cliquen und stabile Mengen

Definition 5.6

Sei $G = (V, E)$ ein Graph. Eine **Clique** in G ist eine Knotenteilmenge $K \subseteq V$ mit $\binom{K}{2} \subseteq E$. Eine **stabile Menge** in G ist eine Knotenmenge $S \subseteq V$ mit $\binom{K}{2} \cap E = \emptyset$.

Problem 5.7 (Maximum Clique Problem)

Instanz: *Graph* $G = (V, E)$

Aufgabe: *Finde Clique* $K \subseteq V$ *in* G *maximaler Kardinalität.*

Problem 5.8 (Maximum Stable Set Problem)

Instanz: *Graph* $G = (V, E)$

Aufgabe: *Finde stabile Menge* $S \subseteq V$ *maximaler Kardinalität.*

Bin Packing

Problem 5.9 (Bin-Packing Problem)

Instanz: *Zahlen* $a_1, \ldots, a_n \in \mathbb{Q}_+$

Aufgabe: *Finde eine Abbildung* $f : [n] \to [k]$ *mit*

$$\sum_{i:f(i)=j} a_i \leq 1 \quad \text{für alle } j \in [k]\,,$$

so dass k *minimal ist.*

Färbung von Graphen

Problem 5.10 (Graphenfärbungsproblem)

Instanz: *Graph* $G = (V, E)$

Aufgabe: *Finde eine Abbildung* $f : V \to [k]$ *mit*

$$f(v) \neq f(w) \quad \text{für alle } \{v, w\} \in E\,,$$

so dass k *minimal ist.*

Ausfallsichere Netzwerke

Problem 5.11 (Survivable Network Design Problem)

Instanz: *Graph* $G = (V, E)$, *Kantenkosten* $c \in \mathbb{Q}_+^E$, *Zusammenhangsforderungen* $r_{s,t} \in \mathbb{N}$ *für alle* $s, t \in V$

Aufgabe: *Finde eine Kantenteilmenge* $F^\star \subseteq E$, *für die in* $G[F]$ *für jedes Paar* $s, t \in V$ *wenigstens* $r_{s,t}$ *paarweise kantendisjunkte s-t-Wege existieren, so dass* $c(F^\star)$ *minimal ist.*

Standortplanung

Problem 5.12 (Facility Location Problem)

Instanz: *Endliche Mengen* $\{1, \ldots, K\}$ *von Kunden und* $\{1, \ldots, S\}$ *von möglichen Standorten mit Fixkosten* $f_i \in \mathbb{Q}_+$ *für das Öffnen von Standort* i *und Service-Kosten* $c_{ij} \in \mathbb{Q}_+$ *für das Bedienen von Kunde* j *vom eröffneten Standort* i *aus.*

Aufgabe: *Teilmenge* $X^\star \subseteq [S]$ *von zu öffnenden Standorten und Zuordnung* $\sigma : [K] \to X^\star$*, so das*

$$\sum_{i \in X^\star} f_i + \sum_{j \in [K]} c_{\sigma(j)j}$$

minimal ist.

Maschinenbelegung

Problem 5.13 (Paralleles Maschinen-Scheduling: Makespan-Minimierung)

Instanz: *Dauern* $p \in \mathbb{Q}_+^n$ *von* n *Jobs, Anzahl* $m \in \mathbb{N}$ *gleichartiger Maschinen*

Aufgabe: *Aufteilung* $f : [n] \to [m]$ *der Jobs auf die Maschinen, so dass*

$$\max\{ \sum_{i:f(i)=j} p_i \mid j \in [m]\}$$

möglichst klein ist.

Effiziente Algorithmen?

- Für diese Optimierungsprobleme gilt: Entweder es gibt für jedes von ihnen oder für keines von ihnen einen Algorithmus mit polynomial beschränkter Laufzeit.
- "Polynomial beschränkt" / "polynomial": Es gibt ein Polynom p, so dass die Laufzeit bei Eingabelänge n höchstens $p(n)$ ist.
- "Eingabelänge" bezieht sich auf die Länge einer üblichen Kodierung der Probleminstanzen über einem endlichen Alphabet (z. B. $\{0, 1\}$ oder $\{0, 1, \#\}$); Zahlen werden z.B. binär kodiert.
- Derzeit ist unbekannt, ob es polynomiale Algorithmen für diese Probleme gibt; es wird für sehr unwahrscheinlich gehalten.
- Für (fast) genauso unwahrscheinlich wird gehalten, dass es für eines dieser Probleme ein starkes Dualitätstheorem (MinMax-Theorem) gibt.

$\Longrightarrow$ **Andere Methoden**

Optimierungs- und Entscheidungsprobleme

- **Entscheidungsproblem**: Frage mit JA/NEIN Antwort
- Zu jedem Maximierungs- bzw. Minimierungsproblem gehört ein Entscheidungsproblem:

  "Gibt es eine Lösung mit Zielfunktionswert mindestens bzw. höchstens α?"

  (α ist Teil der Eingabe).
- Beispiel: Gegeben $c \in \mathbb{Q}^{E_n}, \alpha \in \mathbb{Q}$; hat $K_n = (V_n, E_n)$ einen Hamilton-Kreis $H \subseteq E_n$ mit $c(H) \leq \alpha$?
- Kann man ein (kombinatorisches) Optimierungsproblem mit einem Algorithmus A (in polynomialer Zeit) lösen, so kann man auch das zugehörige Entscheidungsproblem mit A (in polynomialer Zeit) lösen.
- Die Umkehrung gilt in der Regel auch (Binärsuche nach Optimalwert).

Entscheidungsprobleme I

Problem 5.14 (Perfektes Matching Problem)

Instanz: *Graph* $G = (V, E)$

Frage: *Gibt es ein perfektes Matching in* G*?*

Problem 5.15 (Hamilton-Kreis Problem)

Instanz: *Graph* $G = (V, E)$

Frage: *Gibt es einen Hamilton-Kreis in* G*?*

Problem 5.16 (Graphenzusammenhangsproblem)

Instanz: *Graph* $G = (V, E)$

Frage: *Ist* G *zusammenhängend?*

Entscheidungsprobleme II

Problem 5.17 (Primzahlproblem)

Instanz: *Zahl* $m \in \mathbb{N}$

Frage: *Ist m eine Primzahl?*

Problem 5.18 (Erfüllbarkeitsproblem (SAT))

Instanz: *Boolesche Formel in konjunktiver Normalform*

Frage: *Gibt es eine Belegung der Variablen mit Werten "wahr" und "falsch", so dass die Formel den Wert "wahr" ergibt?*

(Beispiel: $(x_1 \vee \overline{x}_7 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_8)$)

Entscheidungsprobleme III

Problem 5.19 (*LP*-Zulässigkeit)

Instanz: $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m$

Frage: *Gibt es* $x \in \mathbb{Q}^n$ *mit* $Ax \leq b$*?*

Problem 5.20 (*IP*-Zulässigkeit)

Instanz: $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m$

Frage: *Gibt es* $x \in \mathbb{Z}^n$ *mit* $Ax \leq b$*?*

Die Komplexitätsklassen EXP und NP

- Die Entscheidungsprobleme 5.14 bis 5.20 sowie die zu den Optimierungsproblemen 5.1 bis 5.13 gehörenden Entscheidungsprobleme sind in der Klasse EXP aller Entscheidungsprobleme, für die es eine Konstante $k \in \mathbb{N}$ gibt, so dass man sie in Zeit 2^{n^k} (bei Eingabelänge n) lösen kann.
- Alle hier aufgelisteten Entscheidungsprobleme haben aber eine Besonderheit: Man kann, falls die Antwort "JA" ist, dies in polynomialer Zeit mit Hilfe eines geeigneten Zertifikats beweisen. Solche Entscheidungsprobleme bilden die Klasse NP ("nondeterministic polynomial time").
- Beispiele:
  - Mit Hilfe eines Hamilton-Kreises $H \subseteq E$ in $G = (V, E)$ (als Zertifikat) kann man in polynomialer Zeit nachweisen, dass G einen Hamilton-Kreis hat.
  - Mit Hilfe eines Hamilton-Kreises $H \subseteq E_n$ in $K_n = (V_n, E_n)$ vom Gewicht $w(H) \leq \alpha$ (als Zertifikat) kann man in polynomialer Zeit nachweisen, dass K_n einen Hamilton-Kreis vom w-Gewicht höchstens α hat.

Die Komplexitätsklasse coNP

- Die Klasse coNP enthält alle Entscheidungsprobleme, für die man, falls die Antwort "NEIN" ist, diese in polynomialer Zeit mit Hilfe eines geeigneten Zertifikats beweisen kann.

- Das Primzahlproblem ist offensichtlich in coNP: Ist $m = pq$ (mit $p, q \in \mathbb{N}$), so kann man mit Hilfe des Zertifikats (p, q) in polynomialer Zeit (in $\log(m)$) nachweisen, dass m keine Primzahl ist. (Dass das Primzahlproblem auch in NP und sogar in P ist, ist nicht so offensichtlich.)

Die Komplexitätsklasse NP ∩ coNP

- NP ∩ coNP ist die Klasse der Probleme, die "gute Charakterisierungen" im Sinne von Edmonds haben.
- Z. B. ist das Entscheidungsproblem

  "Hat ein gegebener bipartiter Graph $G = (V \uplus W, E)$ mit $|V| = |W|$ ein perfektes Matching?"

  in NP ∩ coNP (es ist sogar in P):
  - Ist die Antwort "JA", so kann man das mit Hilfe eines perfekten Matchings $M \subseteq E$ als Zertifikat in polynomialer Zeit nachweisen.
  - Ist die Antwort "NEIN", so kann man das mit Hilfe einer Menge $X \subseteq V$ mit $|N(X)| < |X|$ als Zertifikat in polynomialer Zeit nachweisen.

Die Komplexitätsklasse P

- Die Klasse P enthält alle Entscheidungsprobleme, die in polynomialer Zeit gelöst werden können.
- Es gilt: $\mathsf{P} \subseteq \mathsf{NP} \cap \mathsf{coNP}$ (Wenn man die Antwort in polynomialer Zeit finden kann, braucht man nicht einmal ein Zertifikat.)
- Die Entscheidungsprobleme "Perfektes Matching", "Graphenzusammenhang", "Primalzahltest", "*LP*-Zulässigkeit" sind in P.

NP-Vollständigkeit

- Seien π_1 und π_2 zwei Entscheidungsprobleme. π_1 ist **Karp-reduzierbar** auf π_2, wenn es einen polynomialen Algorithmus gibt, der aus jeder Instanz I_1 für π_1 eine Instanz I_2 für π_2 berechnet, so dass die Antwort auf I_1 gleich der Antwort auf I_2 ist.
- Ein Entscheidungsproblem π ist **NP-vollständig**, wenn $\pi \in \mathsf{NP}$ ist und jedes Entscheidungsproblem aus NP auf π Karp-reduzierbar ist. Die Klasse der NP-vollständigen Entscheidungsprobleme sei NPC.
- Es gilt:

$$\mathsf{NPC} \cap \mathsf{P} \neq \emptyset \quad \Rightarrow \quad \mathsf{P} = \mathsf{NP}$$

$$\mathsf{NPC} \cap \mathsf{coNP} \neq \emptyset \quad \Rightarrow \quad \mathsf{NP} = \mathsf{coNP}$$

(Beides wird für sehr unwahrscheinlich gehalten.)

Optimierungsprobleme und NP/coNP

- Die Entscheidungsprobleme zu den Optimierungsproblemen 5.1 bis 5.13 sowie die Entscheidungsprobleme 5.15, 5.18 und 5.20 sind in NPC (Satz von Cook (1971): SAT ist in NPC)

- Insbesondere würde ein polynomialer Algorithmus für eines von ihnen $P = NP$ und die polynomiale Lösbarkeit aller in diesem Kapitel angesprochenen Probleme implizieren.

- Außerdem: Gibt es für eins der Optimierungsprobleme 5.1 bis 5.13 für jede Instanz ein polynomial überprüfbares Optimalitätszertifikat (z.B. auf Grund eines MinMax-Theorems), so ist $NP = coNP$.

Verhältnis von Komplexitätsklassen

- Falls $P \neq NP$ ist, dann gilt $NP \setminus NPC \neq \emptyset$.
- Es gilt

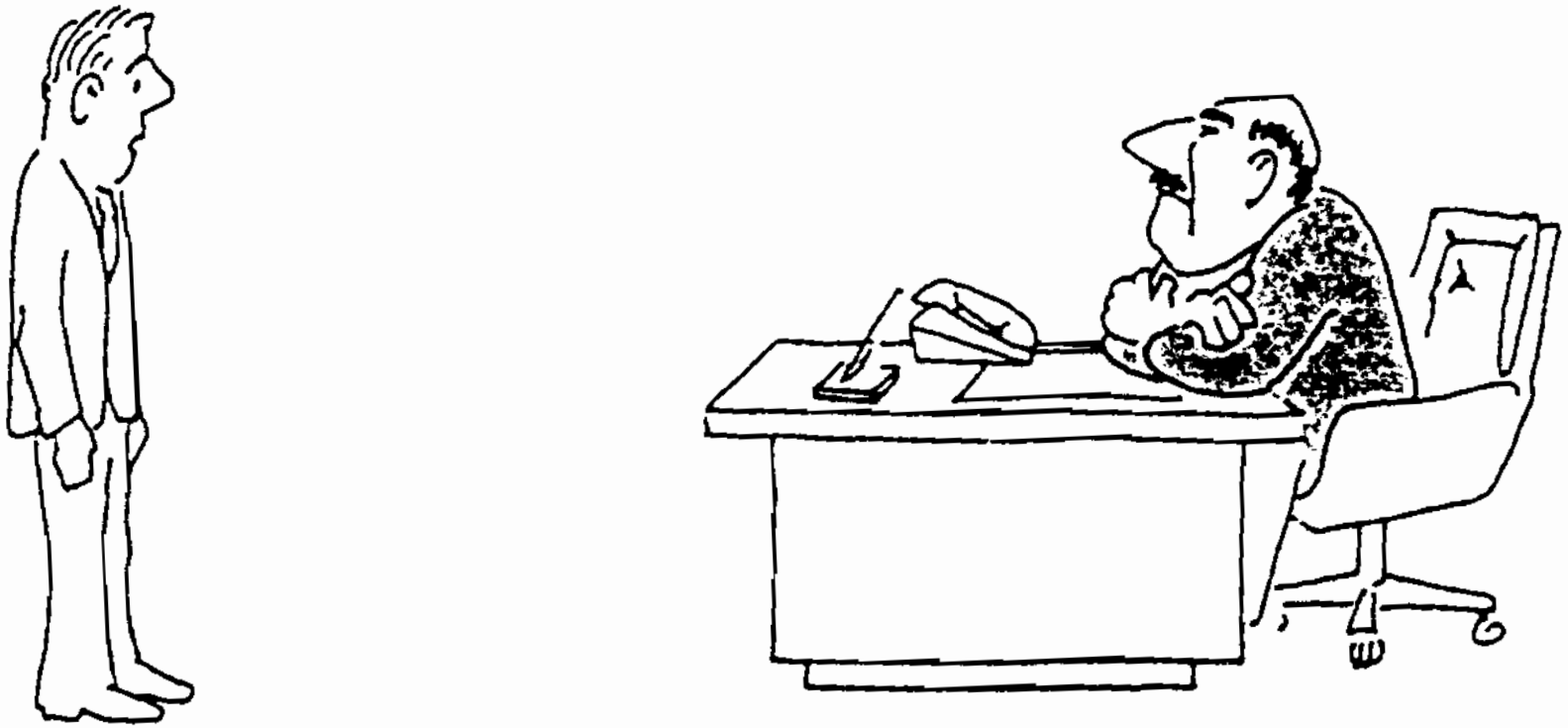
  $$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME \subseteq NEXPSPACE$$

  mit

  $$\begin{array}{lcl} P & \subsetneq & EXPTIME \\ NP & \subsetneq & NEXPTIME \\ PSPACE & \subsetneq & NEXPSPACE \end{array}$$

  Insbesondere gibt es Entscheidungsprobleme, die man in exponenzieller Zeit, aber nicht in polynomialer Zeit lösen kann; aber kein solches Problem in NP ist bekannt.

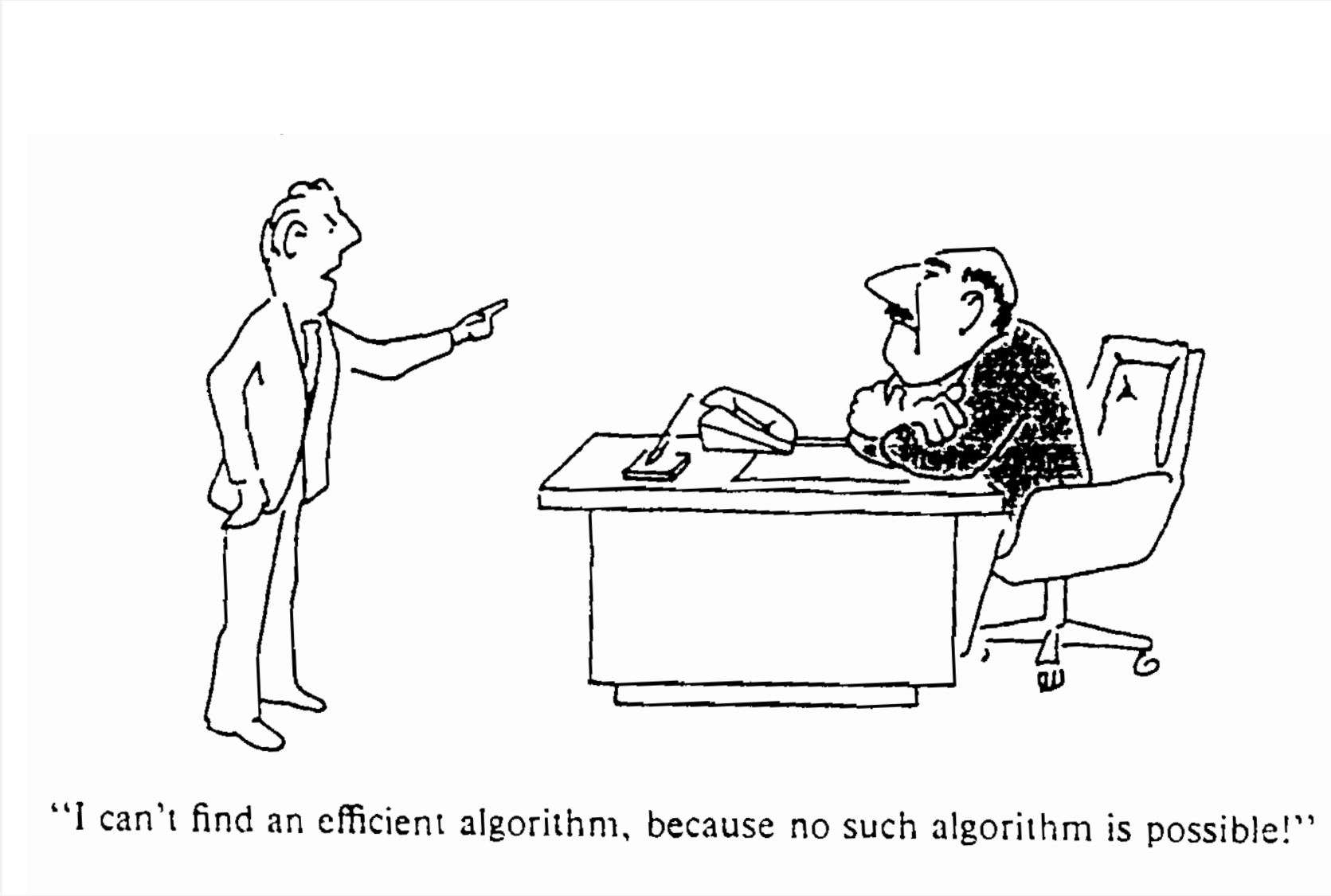
Die Welt ohne Komplexitätstheorie



"I can't find an efficient algorithm, I guess I'm just too dumb."

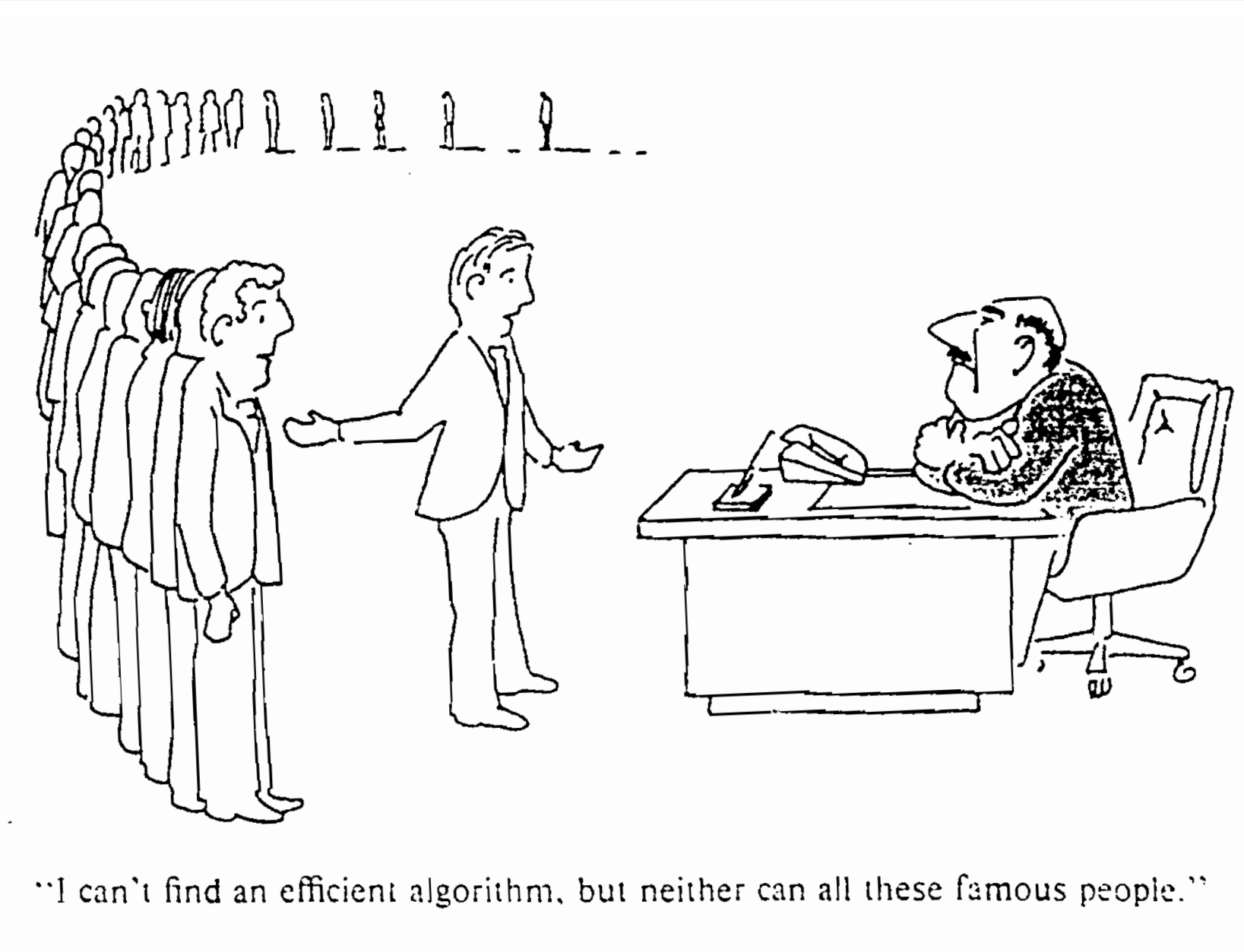
Aus: Garey, Johnson: *Computers and Intractibility*, Wiley 1979.

Die Welt im Fall P $\neq$ NP



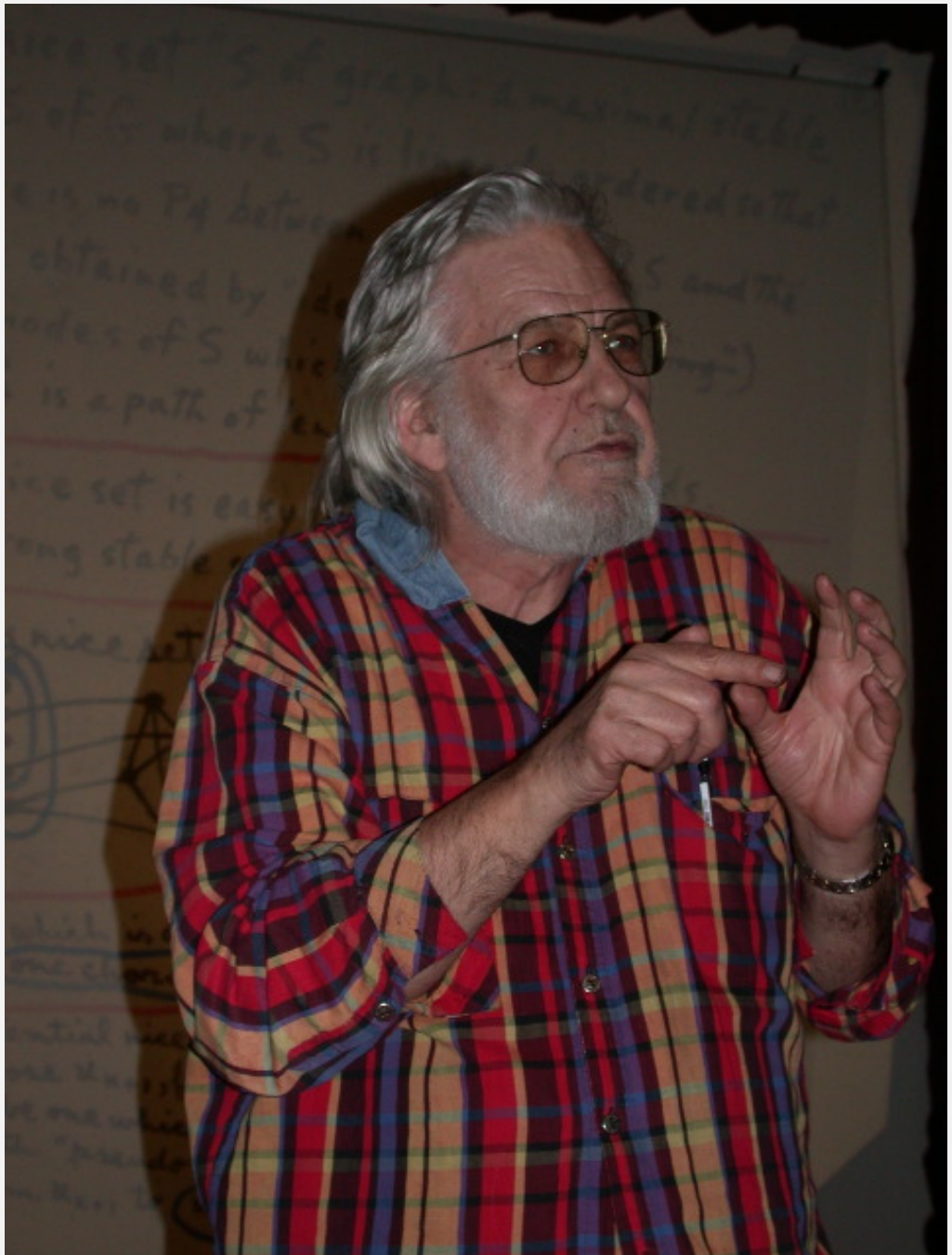
Aus: Garey, Johnson: *Computers and Intractibility*, Wiley 1979.

Die heutige Welt



Aus: Garey, Johnson: *Computers and Intractibility*, Wiley 1979.

Jack Edmonds



Exponenziell versus polynomial

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

(aus : J. Edmonds, ´Paths, trees, and flowers´, Canad. J. Mathematics 17 (1965),

I remember my obsessions and my talking at full tilt —, the biggest reaction I got is: "Well, it's kind of silly to expect such a thing, and let us see, it doesn't have any real meaning, oh, and so what, if it were n to the 28th, you know, that doesn't . . .", and all that kind of stuff. I just collected these kinds of reactions that weren't questioning it as a natural phenomenon.

(aus : J. Edmonds, ´A Glimpse of Heaven´)

Effiziente Algorithmen als mathematisches Problem

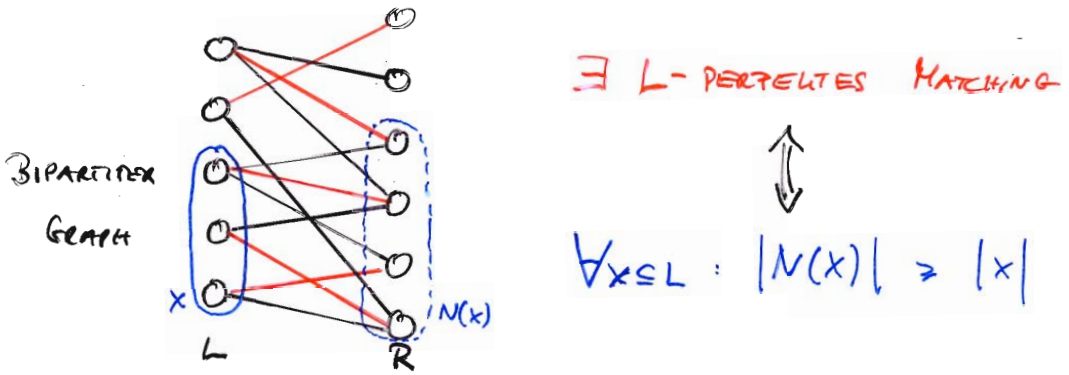
There is the polynomial – I think I said algebraic – time algorithm for the optimum assignment problem. Is there or is there not a polynomial-time algorithm for obviously finite bounded integer programming? I mean, this is my sermon: Is there or is there not? And it is one thing to have an algorithm and announce bounds on it. I don't know why it's a deal to discuss 'or is there not', the idea of proving if it is not, but it took some doing to sell Knuth on this, and other people. Isn't there an algorithm?

(aus : J. Edmonds, ´A Glimpse of Heaven´)

Zertifikate / Gute Charakterisierungen

And then the next part of the sermon — and that was really important to me — was this certificate as opposed to an algorithm.

This marriage theorem here, god, I spent hours to understand it, reading, maybe it was in Ryser's book or a book by Marshall Hall.



Is it easy to tell when you can pull a distinct element from each set? Is it easy to look at *every* bloody subfamily, you know, and, I mean, that is the way I think most people look at that theorem: if and only if for *every* subfamily. And the light hit me! This is not about *every* subfamily having this property! This theorem is telling me, either you can pull a distinct element from each set, or, *or*, there exists *one* subfamily that makes it easy for me to see that you cannot. And, boy, I mean, you know, this certificate of this one subfamily. So that was the only way I could appreciate that theorem, before whether it is easy or hard to prove.

(aus : J. Edmonds, ´A Glimpse of Heaven´)

Mathematischer Aufbau der Komplexitätstheorie

- Für $\Sigma = \{0, 1\}$ sei $\Sigma^\star$ die Menge aller endlichen Folgen (**Wörter**) $x = (x_1, \ldots, x_\ell)$ ($\ell \in \mathbb{N}$) von Elementen aus Σ. Die Länge des Wortes $x = (x_1, \ldots, x_\ell)$ ist $|x| := \ell$.
- Ein **Entscheidungsproblem** (**Sprache**) ist eine Teilmenge $\Pi \subseteq \Sigma^\star$.
- Beispiel Hamilton-Kreis Problem:
  - Bilde die Menge aller Graphen mit einer Abbildung (**Kodierung**) κ injektiv in $\Sigma^\star$ ab (z.B.: Beschränkung auf Graphen $G = ([n], E)$, Kodierung mittels Binärdarstellung von Adjazenzlisten).
  - Das Hamilton-Kreis Problem ist dann modelliert durch

    $$\Pi_{\text{Hamilton}} := \{\kappa(G) \mid G = ([n], E) \text{ für ein } n, G \text{ hat Hamilton-Kreis}\}.$$

Algorithmen I

- Klassisches Modell: Turing-Maschine; hier etwas anderes Modell (für Komplexitätstheorie in unserem Sinne äquivalent zur Turing-Maschine).
- Ein Algorithmus ist eine endliche Menge

$$A = \{(u_1, u_1'), \ldots, (u_r, u_r')\} \subseteq \Sigma^\star \times \Sigma^\star$$

  von (geordneten) Paaren von von Wörtern.
- Für $w \in \Sigma^\star$ sei

$$i(w) := \min(\{i \in [r] \mid u_i \text{ Teilwort von } w\} \cup \{0\}) .$$

- Falls $i(w) \neq 0$, sei w' das Wort, das aus w entsteht, wenn man das erste Teilwort $u_{i(w)}$ in w durch $u'_{i(w)}$ ersetzt.
- Für gegebenes w_0 sei $w_0, w_1, \ldots$ die Folge mit $w_{k+1} = w_k'$, die bei w_k abbricht, wenn $i(w_k) = 0$ ist. Der Algorithmus **akzeptiert** w_0, wenn $w_0, w_1, \ldots$ endlich ist (also irgendwann abbricht).

Algorithmen II

- Die vom Algorithmus A akzeptierte Sprache ist

$$L_A = \{w \in \Sigma^\star \mid A \text{ akzeptiert } w\}.$$

- Der Algorithmus A **löst** das Entscheidungsproblem $\Pi \subseteq \Sigma^\star$ in **polynomialer Zeit**, wenn $L_A = \Pi$ ist und Konstanten $C, k \in \{1, 2, \dots\}$ existieren, so dass für jedes $w \in \Pi$ die von A mit $w_0 = w$ erzeugte Folge höchstens $C \cdot |w|^k$ Glieder hat.
- Man kann argumentieren, dass dieses Algorithmenmodell (bis auf polynomiale Laufzeitfaktoren) genau so mächtig ist wie "reale Computer" (mit beliebig großem Speicher).

Komplexitätsklassen

- Die Klasse P ist die Menge alle $\Pi \subseteq \Sigma^\star$, für die ein Algorithmus A existiert, der Π in polynomialer Zeit löst.
- Die Klasse NP ist die Menge aller $\Pi \subseteq \Sigma^\star$, für die ein $\Pi' \in \mathrm{P}$ und Konstanten $C, k \in \{1, 2, \dots\}$ existieren, so dass für jedes $w \in \Sigma^\star$ gilt:

$$w \in \Pi \Longleftrightarrow \exists z \in \Sigma^\star : |z| = C \cdot |w|^k, wz \in \Pi' .$$

- Die Klasse coNP ist die Menge aller $\Pi \subseteq \Sigma^\star$, für die ein $\Pi' \in \mathrm{P}$ und Konstanten $C, k \in \{1, 2, \dots\}$ existieren, so dass für jedes $w \in \Sigma^\star$ gilt:

$$w \notin \Pi \Longleftrightarrow \exists z \in \Sigma^\star : |z| = C \cdot |w|^k, wz \in \Pi'$$

(d.h. $\Sigma^\star \setminus \Pi \in \mathrm{NP}$).