Die Komplexitätsklassen P und NP unter besonderer Berücksichtigung des Traveling Salesman Problems.

Bevor die Komplexitätsklassen P und NP, welche eine Klassifizierung von Algorithmen bezüglich ihrer Schwierigkeit darstellen, eingeführt werden, soll zunächst die Frage beantwortet werden, ob es überhaupt Probleme gibt, die nicht allgemein durch Algorithmen gelöst werden können. Diese Frage beantworteten im Jahre 1936 A. M. Turing und A. Church, die jeweils Probleme fanden, für die es keinen noch so ineffizienten Lösungsalgorithmus geben kann. Derartige Probleme werden als unentscheidbar bezeichnet. [1] Im Gegensatz dazu nennt man Probleme entscheidbar, wenn es einen Lösungsalgorithmus gibt. Im folgenden werden nur noch entscheidbare Probleme betrachtet, insbesondere das Traveling Salesman Problem.

Die Komplexitätstheorie beschäftigt sich nun mit dem Zeitaufwand, den Algorithmen in Abhängigkeit der Inputlänge benötigen. Dabei wird zumeist der worst case betrachtet, d.h. es wird auf den maximalen Zeitaufwand abgezielt. [2] Man kann zunächst eine Unterscheidung zwischen polynomial-zeitbeschränkten und exponentiell-zeitbeschränkten Algorithmen vornehmen. Dabei heißt ein Algorithmus mit Input-Länge L polynomial-zeitbeschränkt (kürzer polynomial, effizient oder gut), wenn seine Laufzeit O(p(L)) für ein Polynom p ist. [3] Andernfalls nennt man einen Algorithmus exponentiell-zeitbeschränkt (kürzer exponentiell, ineffizient oder schlecht).

Entsprechend spricht man auch von effizient bzw. nicht effizient lösbaren Problemen, je nachdem ob ein effizienter Algorithmus existiert oder nicht. Folgende Tabelle verdeutlicht, wie sich der Rechenaufwand von Algorithmen unterschiedlicher Komplexität entwickelt. [4] Hierbei wurden 1 Million Rechenoperationen pro Sekunde unterstellt.

Zeit-Komplexitäts-Funktion	Inputlänge n=10	Inputlänge n=20	Inputlänge n=50
n	0.00001 s	0.00002 s	0.00005 s
n^2	0.0001 s	0.0004 s	0.0025 s
n^3	0.001 s	0.008 s	0.125 s
n^5	0.1 s	3.2 s	5.2 min
2^n	0.001 s	1 s	35.7 Jahre
3^n	0.059 s	58 min	20 Mrd. Jahre

Selbst eine Steigerung der Rechengeschwindigkeit um den Faktor 1000 würde bei einer Komplexität von 3^n lediglich bewirken, daß in gleicher Zeit eine um 6.29 er-

[1] Turing zeigte, daß das allgemeine Halteproblem für Turing-Maschinen unentscheidbar ist. Church bewies gleiches für die Prädikatenlogik.

[2] Zunehmend häufiger werden auch average-case-Analysen durchgeführt, die den durchschnittlich zu erwartenden Lösungsaufwand untersuchen.

[3] Eine Funktion f(n) heißt O(g(n)), falls es eine Konstante K gibt mit $f(n) \le K \cdot g(n) \ \forall\ n \ge 0$.

[4] Vgl. M. R. Garey, D. S. Johnson: Computers and Intractability (1979), S. 7.

höhte Inputlänge "verarbeitet" werden kann. Demgegenüber könnte man mit einem Algorithmus der Laufzeit n^3 bei Vertausendfachung der Geschwindigkeit in gleicher Zeit ein Problem mit 10-facher Inputlänge lösen.

Die dargestellten Erkenntnisse verleiten zur Einführung einer Bezeichnung für effizient lösbare Probleme:

Definition: Die Klasse der Probleme, für die ein polynomialer Algorithmus existiert, wird mit P bezeichnet.

Ein bekanntes Beispiel für ein P -komplexes Problem ist die lineare Optimierung. Hier trifft man auf die Besonderheit, daß es zwar polynomiale Algorithmen gibt [5], der in der Praxis gängige Simplexalgorithmus jedoch exponentielle Laufzeit aufweist. Desweiteren sind bspw. das Knapsack Problem und das Cutting Stock Problem P -komplex.

Bevor die Klasse NP definiert wird, soll ein Verfahren vorgestellt werden, das es erlaubt, Probleme hinsichtlich ihrer Komplexität zu vergleichen. Hierbei handelt es sich um die polynomiale Reduktion, die formal wie folgt definiert wird:

Definition: Ein Problem A heißt polynomialzeit-reduzierbar auf ein Problem B, falls es einen Algorithmus für A gibt, der als Unterprogramm einen Algorithmus für B aufruft, und folgendes gilt:
Wird jeder Aufruf des Algorithmus für B als ein einziger Rechenschritt aufgefaßt, so hat der Algorithmus für A polynomiale Laufzeit.

An dieser Stelle ist es sinnvoll, zwischen zwei Problemtypen in der Optimierung zu differenzieren. Zum einen handelt es sich um Optimierungsprobleme im eigentlichen Sinne, die durch eine Vielzahl potentieller Lösungen gekennzeichnet sind. Außer den oben als P -komplex eingestuften Problemen kann z.B. das Traveling Salesman Problem genannt werden. Die für die späteren Betrachtungen wesentliche Problemklasse stellen jedoch die Entscheidungsprobleme dar. [6] Diese sind dadurch charakterisiert, daß es nur zwei potentielle Lösungen gibt (ja - nein, 0 - 1, etc.). [7] Die Frage nach der Existenz eines Hamiltonschen Kreises in einem gegebenen Graph kann hier beispielhaft angeführt werden. [8]

Nun stellt sich die Frage, inwiefern auch Optimierungsprobleme von der auf Entscheidungsprobleme ausgerichteten Komplexitätstheorie erfaßt werden können. Man überlegt sich leicht, daß jedes Optimierungsproblem durch Hinzufügen einer

---

[5] Z. B. die Ellipsoidmethode von L. G. Khachiyan (1979) oder der Algorithmus von N. Karmarkar (1984).

[6] Dies liegt in der engen Beziehung zwischen Entscheidungsproblemen und formalen Sprachen begründet. So läßt sich die Eigenschaft der NP -Vollständigkeit nur für Entscheidungsprobleme nachweisen, da nur für deren Analyse gesichert ist, daß das Konzept der nondeterministischen Turing-Maschinen den Lösungsaufwand korrekt wiedergibt (vgl. S. Zelewski: Komplexitätstheorie (1989), S. 5-10).

[7] Anhand des Begriffs des Entscheidungsproblems erklären sich nun auch die Bezeichnungen "entscheidbares" bzw. "unentscheidbares Problem" von Seite 1.

[8] Ein Hamiltonscher Kreis ist eine geschlossene Kantenfolge in einem Graph, welche jede Ecke genau einmal enthält.

Schranke in ein Entscheidungsproblem transformiert werden kann. So könnte beim Traveling Salesman Problem gefragt werden, ob es eine Tour der Länge $\leq S$ gibt, wobei S eine vorgegebene Schranke ist. Faßt man als Lösung eines Optimierungsproblems den Zielfunktionswert auf, so ist klar, daß ein Optimierungsproblem mindestens so schwer wie ein daraus abgeleitetes Entscheidungsproblem ist. [9] Bezüglich der Komplexität (polynomial - exponentiell) gilt für viele Optimierungsprobleme sogar die Umkehrung, d.h. die Existenz eines polynomialen Algorithmus für das zugehörige Entscheidungsproblem impliziert, daß damit auch das Optimierungsproblem selbst effizient lösbar ist. Für das Traveling Salesman Problems soll dies nun mittels polynomialer Reduktion bewiesen werden. Betrachte dazu den folgenden in Pseudocode dargestellten Algorithmus:

Input: Eine natürliche Zahl n und eine $n \times n - \text{Matrix } C = (c_{ij})$ mit nichtnegativen, ganzzahligen Einträgen. [10]

Setze a:=0 und b:= $n \cdot \max_{1 \leq i,j \leq n} c_{ij}$.

Solange $a \neq b$:

Falls TSP?(n,C, $\left\lfloor \frac{a+b}{2} \right\rfloor$)='ja', setze b:= $\left\lfloor \frac{a+b}{2} \right\rfloor$.

Andernfalls setze a:= $\left\lfloor \frac{a+b}{2} \right\rfloor$+1.

Setze opt:=a.

Für i,j=1,...,n :

Setze zwerg:= c_{ij} und $c_{ij} := n \cdot \max_{1 \leq i,j \leq n} c_{ij} + 1$.

Falls TSP?(n,C,opt)='nein', setze c_{ij} :=zwerg.

Output: Eine $n \times n - \text{Matrix } C$ mit $n^2 - n$ Einträgen $= n \cdot \max_{1 \leq i,j \leq n} c_{ij} + 1$ und n unveränderten Einträgen, welche eine optimale Tour repräsentieren.

Hierbei ist TSP?(n,C,S) ein Unterprogramm, das das durch die Schranke S induzierte Entscheidungsproblem löst, also

$$\text{TSP?}(n,C,S) := \begin{cases} \text{'ja'}, & \text{falls es eine Tour mit Entfernung } \leq S \text{ gibt} \\ \text{'nein'}, & \text{falls die optimale Tour Entfernung } > S \text{ hat} \end{cases}.$$

Die Bisektion (erste Schleife) berechnet die Länge einer optimalen Tour und erfordert maximal $\left\lceil \log_2 \left(n \cdot \max_{1 \leq i,j \leq n} c_{ij} \right) \right\rceil$ Iterationsschritte. Die Schleife im zweiten Teil

---

[9] Faßt man als Lösung eines Optimierungsproblems nicht den Zielfunktionswert auf, so könnte sich eine Schwierigkeit daraus ergeben, daß das Problem zwar gelöst werden kann, die Ermittlung des Zielfunktionswertes aus dieser Lösung jedoch vergleichsweise großen Aufwand erfordert. Dies würde dazu führen, daß das zugehörige Entscheidungsproblem evtl. schwerer als das eigentliche Optimierungsproblem ist. Vgl auch S. Zelewski, a. a. O.

[10] Der triviale Fall $C \equiv 0$ sei hier ausgeschlossen.

wird genau n^2-mal durchlaufen, wobei hier eine Tour mit der als optimal berechneten Länge bestimmt wird. In jedem Iterationsschritt und in jedem Durchlauf der zweiten Schleife wird das Unterprogramm TSP? genau einmal aufgerufen, d.h. insgesamt ergeben sich maximal $\left\lceil \log_2\left(n \cdot \max_{1\le i,j\le n} c_{ij} \right) \right\rceil + n^2$ Aufrufe von TSP?.

Setzt man $c_{max} := \max_{1\le i,j\le n} c_{ij}$, so kann die Input-Länge des Traveling Salesman Problems nach oben durch $L := n^2 \cdot \lceil \log_2(c_{max}) \rceil$ abgeschätzt werden. Nun gilt aber :

$$\lceil \log_2(n \cdot c_{max}) \rceil + n^2 = \lceil \log_2(n) + \log_2(c_{max}) \rceil + n^2 \le n + \log_2(c_{max}) + n^2$$
$$\le n^2 \cdot \lceil \log_2(c_{max}) \rceil + n^2 \cdot \lceil \log_2(c_{max}) \rceil + n^4 \cdot \lceil \log_2(c_{max}) \rceil^2 = 2L + L^2 .$$

Damit ist also die Anzahl der Aufrufe von TSP? durch ein quadratisches Polynom in L beschränkt. Da vor bzw. nach jedem Aufruf von TSP? nur eine feste Anzahl an Zuordnungen erfolgt, hat der obige Algorithmus somit polynomiale Laufzeit, sofern jeder Aufruf von TSP? als ein einziger Rechenschritt gezählt wird. Es liegt also eine polynomiale Reduktion des Traveling Salesman Problems auf das zugeordnete Entscheidungsproblem TSP? vor. Berücksichtigt man nun, daß das Produkt zweier Polynome wieder ein Polynom ist, so ist hiermit der folgende Satz bewiesen.

Satz: Es gibt genau dann einen polynomialen Algorithmus für das Traveling Salesman Problem, wenn es einen solchen für das durch Hinzufügen einer Schranke induzierte Entscheidungsproblem gibt. Anders formuliert gilt: $TSP \in P \Leftrightarrow TSP? \in P$.[11]

Wie bereits oben erwähnt, läßt sich für eine Vielzahl von Optimierungsproblemen eine derartige Beziehung zum jeweils zugehörigen Entscheidungsproblem beweisen. Losgelöst vom Traveling Salesman Problem ergibt sich nun allgemeiner der

Satz: Wenn es eine polynomiale Reduktion von A auf B gibt und ein polynomialer Algorithmus für B existiert, so existiert ein polynomialer Algorithmus für A.

Das heißt insbesondere, daß ein Problem B als schwer eingestuft werden kann, wenn es eine polynomiale Reduktion eines als schwer vermuteten Problems A auf B gibt. Nutzt man wieder aus, daß das Produkt zweier Polynome ein Polynom ist, so kann als weiteres Resultat die Transitivität der polynomialen Reduktion festgehalten werden.

Satz: Existieren polynomiale Reduktionen von A auf B und von B auf C, dann läßt sich A polynomial auf C reduzieren.

---

[11] Bisher ist für das Traveling Salesman Problem kein polynomialer Algorithmus bekannt. Das zur Zeit beste polynomiale Näherungsverfahren für das metrische Traveling Salesman Problem stellt der Algorithmus von Christofides dar, der eine Tour liefert, dessen Länge um maximal 50 Prozent von der Länge einer optimalen Tour abweicht. Vgl. hierzu C. H. Papadimitriou, K. Steiglitz: Combinatorial Optimization (1982), S. 410-419.

Ein Spezialfall der polynomialen Reduktion ist die polynomiale Transformation. Hierunter versteht man die Reduktion eines Entscheidungsproblems A auf ein Entscheidungsproblem B, wobei das Unterprogramm für B nur genau einmal aufgerufen wird und die übergebene Lösung von B schon die Lösung von A ist. Dies soll am Beispiel der Transformation von (HK) auf (ILP) [12] dargestellt werden:

(HK) Gegeben ein Graph G=(V,E) mit $V=\{v_1,\dots,v_N\}$.
Frage: Existiert ein Kreis (geschlossene Kantenfolge) in G, der jede Ecke aus V genau einmal durchläuft ?

(ILP) Gegeben eine ganzzahlige $m \times n$ – Matrix $A=(a_{ij})$ und ein Vektor $b=(b_1,\dots,b_m)^T$ mit ebenfalls ganzzahligen Einträgen.
Frage: Existiert ein Vektor $x=(x_1,\dots,x_n)^T$ mit nichtnegativen ganzzahligen Komponenten, der die Gleichung Ax=b erfüllt ?

Es ist also eine Aufgabe Ax=b gesucht, die dann und nur dann eine Lösung besitzt, wenn G Hamiltonsch ist. [13]

Behauptung: Die durch die folgenden Bedingungen induzierte Aufgabe hat die geforderte Eigenschaft:

$$\sum_{k=1}^{N} x_{ik} = 1 \;;\quad i=1,\dots,N$$

$$\sum_{i=1}^{N} x_{ik} = 1 \;;\quad k=1,\dots,N$$

$$x_{ik} + x_{j,k+1} \le 1 + e_{ij} \;;\quad i,j=1,\dots,N;\ k=1,\dots,N-1$$

$$x_{iN} + x_{j1} \le 1 + e_{ij} \;;\quad i,j=1,\dots,N$$

Hierbei können die Ungleichungen durch Einführung von Schlupfvariablen s_{ijk} in Gleichungen der Form $x_{ik} + x_{j,k+1} + s_{ijk} = 1 + e_{ij}$ bzw. $x_{iN} + x_{j1} + s_{ij1} = 1 + e_{ij}$ überführt werden, wobei $x=\left((x_{11},\dots,x_{1N})^T,\dots,(x_{N1},\dots,x_{NN})^T\right)^T$ ein Vektor mit N^2 Einträgen ist. Man erhält dann ein Gleichungssystem $A\begin{pmatrix} x \\ s \end{pmatrix} = b$ mit einer $(N^3+2N)\times(N^3+N^2)$ – Matrix A.

Beweis: Setze $x_{ik} := \begin{cases} 1 \;;\; v_i \text{ ist k - te Ecke des Hamiltonschen Kreises} \\ 0 \;;\; \text{sonst} \end{cases}$ (i,k=1,...,N)

und $e_{ij} := \begin{cases} 1 \;;\; \{i,j\} \in E \\ 0 \;;\; \{i,j\} \notin E \end{cases}$ (i,j=1,...,N) . Sei nun x eine Lösung von Ax=b.

[12] ILP steht hier für Integer Linear Program.
[13] Ein Graph heißt Hamiltonsch, wenn er einen Hamiltonschen Kreis enthält.

Dann gilt insbesondere $\sum_{k=1}^{N} x_{ik} = 1 \ (i = 1,\ldots,N)$, d.h. jedes v_i ist in dem (potentiellen) Hamiltonschen Kreis enthalten. Wegen $\sum_{i=1}^{N} x_{ik} = 1 \ (k = 1,\ldots,N)$ wird jede Ecke des Hamiltonschen Kreises genau einmal durchlaufen. Damit entspricht also jedes v_i genau einer Ecke des Kreises. Die Ungleichungen schließlich bewirken, daß der Hamiltonsche Kreis tatsächlich Teil des gegebenen Graphen ist, d.h. im Hamiltonschen Kreis werden nur dort Ecken verbunden, wo der Graph Kanten besitzt. Sei nun umgekehrt durch x ein Hamiltonscher Kreis definiert. Dann folgen aus der Definition des Hamiltonschen Kreises sofort die zu erfüllenden Gleichungen, also $A\begin{pmatrix} x \\ s \end{pmatrix} = b$. ■

Die polynomiale Transformation von (HK) auf (ILP) besteht nun darin, zu einem vorgegebenen Graph ein Gleichungssystem Ax=b zu konstruieren und dieses auf Lösbarkeit zu überprüfen. Wie oben bereits erwähnt, ist die Anzahl der Gleichungen und der Variablen jeweils durch ein Polynom bestimmt, so daß die Konstruktion des Systems Ax=b in polynomialer Zeit erfolgt.

Bezeichnung: Ist ein Problem A auf ein Problem B polynomial transformierbar, so schreibt man $A \propto B$. Nach obigem Beispiel gilt somit $HP \propto ILP$.

Das oben konstruierte Problem Ax=b gehört zu einer speziellen Klasse der Probleme der Gestalt (ILP). Für sämtliche Komponenten von x galt die Einschränkung $x_{ij} \in \{0,1\}$ bzw. $x_{ij} \leq 1$. Bezeichnet man die Klasse linearer Programme dieser Gestalt mit (ZOLP) [14] , so ist klar, daß man ein (ZOLP)-Programm durch Einführung von zusätzlichen Variablen $\tilde{x}_{ij}$ mit $x_{ij} + \tilde{x}_{ij} = 1$ in ein (ILP)-Programm überführen kann. Insbesondere ist also das durch Aufgabe der Bedingung $x_{ij} \in \{0,1\}$ erhaltene Problem mindestens so schwer wie das ursprüngliche Problem. Erstaunlicher ist der hier nicht bewiesene

Satz: Es existiert dann und nur dann ein polynomialer Algorithmus für ILP, falls es einen polynomialen Algorithmus für ZOLP gibt. Außerdem gilt $ILP \propto ZOLP$. [15]

Im folgenden soll nun die Komplexitätsklasse NP eingeführt werden. Wie schon erwähnt, ist diese Klasse nur für Entscheidungsprobleme definiert. Zunächst werden drei Problemklassen beschrieben, die danach als identisch entlarvt und mit NP bezeichnet werden.

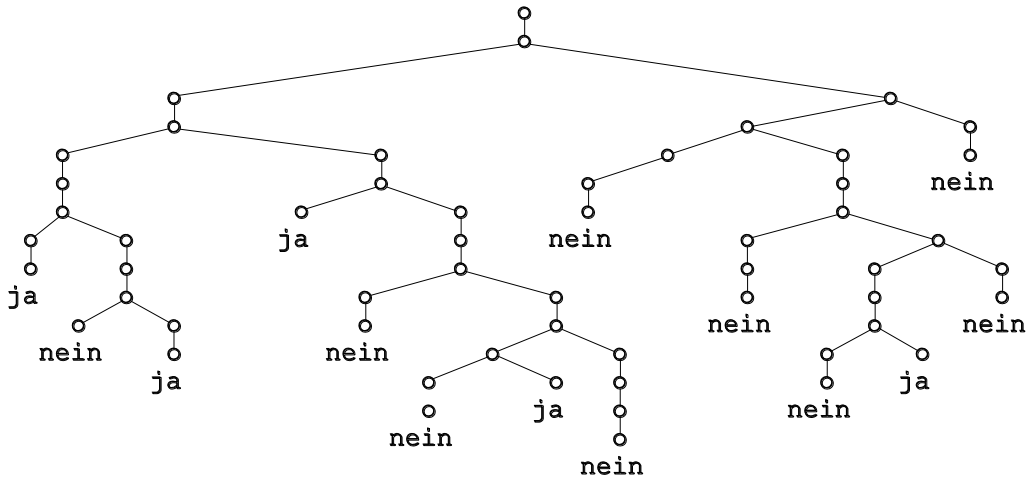
---

[14] ZOLP steht abkürzend für Zero-One Linear Program.

[15] Einen Beweis findet man z.B. bei C. H. Papadimitriou, K. Steiglitz, a. a. O., S. 323.

♒ Ein Entscheidungsproblem habe die Verifikations-Eigenschaft, wenn eine positive Lösung in polynomialer Zeit verifiziert werden kann. Am Beispiel des Traveling Salesman Problems soll dies näher erläutert werden. Angenommen, jemand behauptet, zu einer vorgegebenen Entfernungsmatrix C eine Tour T der Länge $\leq S$ gefunden zu haben. Die positive Lösung zur Vorgabe (C,S) stellt in diesem Fall die Tour T dar. Nun muß es in polynomialer Zeit möglich sein, zu überprüfen, ob T wirklich ein Hamiltonscher Kreis ist und ob die Schranke S tatsächlich nicht überschritten wird. Ist es nun für alle Tupel (C,S) und für alle Touren T der Länge $\leq S$ möglich, dies in polynomialer Zeit nachzuweisen, so hat das Traveling Salesman Problem die Verifikations-Eigenschaft. Dies ist in der Tat der Fall, denn die Länge jeder Tour T kann in polynomialer Zeit berechnet und mit der Schranke S verglichen werden; ebenso ist es in polynomialer Zeit möglich, die Tour auf die Hamiltonsche Eigenschaft zu überprüfen. Ferner kann man zeigen, daß alle aus kombinatorischen Optimierungsproblemen abgeleiteten Entscheidungsprobleme die Verifikations-Eigenschaft besitzen, zumindest sofern sich der Zielfunktionswert effizient berechnen läßt. [16] [17]

♓ Von S. A. Cook (1971) und R. M. Karp (1972) wurden diejenigen Entscheidungsprobleme zu einer Klasse zusammengefaßt, die sich mit sog. nichtdeterministischen Algorithmen in polynomialer Zeit lösen lassen. Unter einem nichtdeterministischen Algorithmus wird dabei ein um eine Eigenschaft erweiterter gewöhnlicher (deterministischer) Algorithmus verstanden. Diese zusätzliche Eigenschaft besteht darin, in jedem Schritt in zwei Teilalgorithmen verzweigen zu können, die dann parallel weiterarbeiten. Die Anzahl parallel laufender Algorithmen kann dabei exponentiell zunehmen, was verdeutlicht, daß es sich bei einem nichtdetermini-



[16] Der Begriff der Verifikations-Eigenschaft ("succinct certificate property") wurde 1965 von J. Edmonds eingeführt. Probleme mit dieser Eigenschaft nannte er "problems with good characterizations".

[17] Eine ganz andere Frage ist es, ob auch jede negative Lösung in polynomialer Zeit verifiziert werden kann. Eine negative Lösung des Traveling Salesman Problems ist die Behauptung, es gebe keine Tour T zur Vorgabe (C,S). Die Klasse der Probleme, für die jede negative Lösung in polynomialer Zeit verifiziert werden kann, wird mit co-NP bezeichnet. Es wird vermutet, daß NP $\neq$ co-NP gilt; ein Beweis ist jedoch nicht bekannt.

stischen Algorithmus lediglich um ein theoretisches Modell handelt. Ferner liefert jeder dieser Algorithmen (Zweige) ein Ergebnis, also "ja" oder "nein" (bzw. 0 - 1). Die Antwort des nichtdeterministischen Algorithmus sei "ja", falls mindestens ein Zweig die Antwort "ja" liefert. Nur falls alle Zweige die Lösung "nein" haben, soll auch die Lösung des nichtdeterministischen Algorithmus "nein" sein. Insbesondere kann die Berechnung gestoppt werden, sobald das erstemal die Lösung "ja" gemeldet wird. Nun sagt man, ein nichtdeterministischer Algorithmus löse ein Problem in polynomialer Zeit, falls er zu jeder Eingabe das korrekte Ergebnis liefert und die Anzahl der benötigten Operationen durch ein Polynom in der Inputlänge beschränkt ist, sofern man parallel durchgeführte Rechenschritte jeweils als eine einzige Operation auffaßt.

Dieses Konzept soll nun an einem Beispiel erläutert werden. Man betrachte dazu den folgenden nichtdeterministischen Algorithmus, der unter der Restriktion $x_j \in \{0,1\}\ (1 \le j \le n)$ ein Gleichungssystem Ax=b auf Lösbarkeit überprüft.

Input: Eine $m \times n$ – Matrix A und ein Vektor b mit jeweils ganzzahligen Einträgen.
Für j=1,...,n :
    Führe parallel aus :
        $x_j := 0$ $x_j := 1$
Falls Ax=b, dann :
    Output: "ja"
Andernfalls :
    Output: "nein"

Hier wird zunächst bei x_1 in die Fälle x_1=0 und x_1=1 verzweigt, dann in beiden Fällen analog für x_2 verfahren usw. In der letzten Stufe werden alle 2^n möglichen Fälle parallel betrachtet. Wird mindestens einmal als Output "ja" geliefert, so ist das System lösbar. Werden nun parallele Rechenschritte als ein einziger Schritt aufgefaßt, so liefert der Algorithmus offensichtlich in n Schritten die richtige Lösung, d.h. das Problem wird von diesem nichtdeterministischen Algorithmus in polynomialer Zeit gelöst.

♑ Im Jahr 1960 klassifizierte G. B. Dantzig diejenigen Probleme, die in polynomialer Zeit in ganzzahlige Optimierungsaufgaben transformiert werden können. Die Entdeckung des Simplex-Algorithmus und neuartige Schnittebenenverfahren für ganzzahlige Optimierungsprobleme schürten die Hoffnung, derartige Probleme bald effizient lösen zu können. Aus diesem Grund schien es sinnvoll, verschiedene Probleme auf ganzzahlige Optimierungsaufgaben zurückzuführen, und diese zu einer Klasse zusammenzufassen.

Wie schon erwähnt, sind die in ♒ bis ♑ dargestellten Problemklassen identisch. Dies konnte 1971 von S. A. Cook gezeigt werden; Karp führte daraufhin die Bezeichnung NP ein. [18] Es gilt also der folgende

[18] Das Akronym NP steht für nondeterministic polynomial.

Satz. Für ein Entscheidungsproblem A sind folgende Aussagen äquivalent:

(a) A hat die Verifikations-Eigenschaft.

(b) $A \in \mathsf{NP}$.

(c) A ist in polynomialer Zeit in eine ganzzahlige Optimierungsaufgabe transformierbar.

Beweis: Hier soll lediglich der Beweis einer der Implikationen skizziert werden. Der übrige Beweis setzt weiterführende Resultate voraus. Insbesondere der Teil $(c) \Rightarrow (a)$ erfordert ein präziseres Modell von nichtdeterministischen Algorithmen. [19]

$(a) \Rightarrow (b)$: Habe nun das Problem A die Verifikations-Eigenschaft. Wie sieht nun ein nichtdeterministischer Algorithmus aus, der A in polynomialer Zeit löst ? Zuerst muß ermöglicht werden, alle denkbaren positiven Lösungen parallel zu überprüfen (vgl. auch obiges Beispiel). Auf Grund der Verifikations-Eigenschaft von A kann jede potentielle positive Lösung in polynomialer Zeit verifiziert werden, d.h. der konstruierte nichtdeterministische Algorithmus hat polynomiale Laufzeit. Falls A nun tatsächlich eine positive Lösung besitzt, so liefert mindestens ein Zweig in polynomialer Zeit die Antwort "ja", womit das Ergebnis des nichtdeterministischen Algorithmus ebenfalls "ja" ist. Hat A nun keine Lösung, so liefern alle Zweige die Antwort "nein", was bedeutet, daß auch der nichtdeterministische Algorithmus die Lösung "nein" liefert. Damit gilt aber schon $A \in \mathsf{NP}$. (■)

Nun stellt sich die Frage nach Beziehungen zwischen den Klassen P und NP. Zunächst ist leicht einzusehen, daß ein deterministischer - d. h. gewöhnlicher - Algorithmus lediglich ein Spezialfall eines nichtdeterministischen Algorithmus ist. Ein gewöhnlicher Algorithmus ist in der Weise nichtdeterministisch, als daß er von der Möglichkeit, in zwei parallele Algorithmen zu verzweigen, keinen Gebrauch macht. Jedes Problem, das von einem gewöhnlichen Algorithmus in polynomialer Zeit gelöst werden kann, wird also auch von einem nichtdeterministischen Algorithmus in polynomialer Zeit gelöst. Damit ist die Aussage $\mathsf{P} \subseteq \mathsf{NP}$ bewiesen. [20] Ein offenes Problem hingegen ist die Frage, ob P eine echte Teilmenge von NP ist. Es ist sogar unbekannt, ob dieses Entscheidungs(meta)problem entscheidbar oder unentscheidbar im zu Beginn angegebenen Sinne ist. [21] Die sehr große Anzahl von Problemen aus NP, für die trotz großer Anstrengungen kein polynomialer Algorithmus gefunden werden konnte, läßt allerdings $\mathsf{P} \subset \mathsf{NP}$ vermuten. Diese empirische Beobachtung kann allerdings letzte Zweifel daran nicht ausräumen.

---

[19] Vgl. z. B. M. R. Garey, D. S. Johnson, a. a. O.

[20] Ferner zeigt dies auch, daß mit der Bezeichnung "Klasse" (für P und NP) nicht etwa eine Äquivalenzklasse im mathematischen Sinne zu verstehen ist.

[21] Vgl. S. Zelewski, a. a. O., S. 11 ff.

Zum Schluß sei noch darauf hingewiesen, daß es entscheidbare Probleme gibt, die nachweislich außerhalb von NP liegen, die sich also selbst mit nichtdeterministischen Algorithmen nicht in polynomialer Zeit lösen lassen. Berücksichtigt man, daß die Anzahl der Verzweigungen in einem nichtdeterministischen Algorithmus exponentiell beschränkt ist, so läßt sich folgendes Resultat festhalten.

Satz: Ist ein Problem mit einem nichtdeterministischen Algorithmus in polynomialer Zeit lösbar, so ist es mit einem deterministischen Algorithmus (zumindest) in exponentieller Zeit lösbar.

Der Beweis ergibt sich sofort aus der Tatsache, daß das Produkt einer polynomial beschränkten Funktion mit einer exponentiell beschränkten Funktion wieder exponentiell beschränkt ist. Für entscheidbare Probleme außerhalb von NP folgt daraus unmittelbar, daß es keine deterministischen Algorithmen geben kann, die diese Probleme in polynomialer Zeit lösen. Exponentielle Lösbarkeit solcher Probleme liegt ferner genau dann vor, wenn P =NP gilt.

----- ENDE -----