Komplexitätstheorie

Komplexität bezeichnet die „Kompliziertheit“ von Problemen, Algorithmen und Daten. Die *Komplexitätstheorie* befasst sich dabei mit dem Ressourcenverbrauch von Algorithmen. Die betrachteten Ressourcen sind fast immer die Anzahl der benötigten Rechenschritte (*Zeitkomplexität*) oder der Speicherbedarf (*Platzkomplexität*). Algorithmen und Probleme werden in der Komplexitätstheorie gemäß ihrer so bestimmten Komplexität in so genannte *Komplexitätsklassen* eingeteilt. Diese sind ein wichtiges Werkzeug, um bestimmen zu können, welche Probleme *„gleich schwierig“* sind.[1]

Komplexität
Komplexitätstheorie
Zeitkomplexität
Platzkomplexität
Komplexitätsklassen

Komplexitätsklassen

P enthält alle Probleme, für die eine *deterministische* Turingmaschine in *polynomieller* Zeit eine Lösung berechnen kann.[2]

deterministische
polynomieller

NP enthält alle Probleme, für die eine *nicht-deterministische* (= NP) Turingmaschine in *polynomieller* Zeit eine Lösung berechnen kann.[3]

nicht-deterministische
polynomieller

NP-schwer (engl. NP-hard) ist eine Eigenschaft eines algorithmischen Problems. Ein NP-schweres Problem ist dabei mindestens genauso „schwer“ wie alle Probleme in NP. Das bedeutet, dass ein Algorithmus, der ein NP-schweres Problem löst, mithilfe einer *Reduktion* benutzt werden kann, um alle Probleme in NP zu lösen.[4]

Reduktion

NP-vollständig sind die Probleme, die in NP liegen und NP-schwer sind.[5]

Millennium-Problem P=NP

Das `P=NP`-Problem gilt als eines der großen offenen Probleme in der Informatik. Es wird angenommen, dass es nicht gilt. Was würde es bedeuten, wenn wir `P=NP` zeigen könnten? Fast alle kryptographischen Sicherheitsalgorithmen wie Verschlüsselungen usw. wären knackbar. Es ergäben sich immense Optimierungsmöglichkeiten bei verschiedenen Problemen.

In dem Moment, wo für ein NP-vollständiges Problem ein deterministischer Algorithmus in Polynomialzeit gefunden wird, sind alle Probleme in `NP` auch in `P` und damit `P=NP`.[6]

---

[1] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 58.
[2] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 59.
[3] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 60.
[4] *Wikipedia-Artikel „NP-Schwere“*.
[5] *Wikipedia-Artikel „NP-Vollständigkeit“*.
[6] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 61.

Problemreduktion

Problem A heißt reduzierbar

B löst, auch verwendet werden kann, um A zu lösen

Probleminstanz von A umrechnet in eine Instanz von B

Um die Schwere von Problemen zu vergleichen, werden in der theoretischen Informatik Problemreduktionen benutzt: Ein *Problem A heißt reduzierbar* auf ein anderes Problem B, wenn jeder Algorithmus, der *B löst, auch verwendet werden kann, um A zu lösen*, indem man eine *Probleminstanz von A umrechnet in eine Instanz von B* und diese anschließend löst. Sei $L' \subseteq \Sigma$ eine formale Sprache. L' heißt dann NP-schwer, wenn gilt:

$$\forall L \in \mathrm{NP}: \; L \leq_p L'$$

(Alle L aus NP sind polynomiell reduzierbar auf L'.) Dies bedeutet, dass L' mindestens so schwer wie jedes beliebige Problem aus NP ist. Es gibt einen Algorithmus A, der L' in Polynomialzeit löst, für jedes Problem aus NP ebenfalls ein polynomialer Algorithmus konstruieren ließe:

- führe zuerst die Reduktion auf L' aus und

- anschließend Algorithmus A.

L' selbst kann jedoch auch schwerer sein. Insbesondere muss L' nicht notwendigerweise in NP liegen (liegt L' jedoch zusätzlich in NP, so heißt L' NP-vollständig).[7]

Polynomialzeitreduktion

Eine Polynomialzeitreduktion (auch polynomielle Reduktion) ist eine spezielle Form der Reduktion in der theoretischen Informatik. Zusätzlich zur Reduzierbarkeit wird hier gefordert, dass die Reduktion deterministisch in Polynomialzeit berechnet werden kann.[8]

L_1 heißt polynomiell reduzierbar auf L_2 ($L_1 \leq_p L_2$), wenn es eine total, berechenbare Funktion $f = \Sigma^* \rightarrow \Sigma^*$ gibt, so dass gilt: $w \in L_1 \Leftrightarrow f(w) \in L_2$ Es gibt eine deterministische Turingmaschine M, die f in Polynomialzeit berechnet. L_2 ist also *„mindestens genauso schwer“* wie L_1.[9]

Was muss gezeigt werden:

- Es gibt eine Funktion, die Instanzen von L_1 auf Instanzen von L_2 „umbaut“.

- Die Funktion ist total.

- Die Funktion arbeitet in Polynomialzeit.

- Korrektheit der Reduktion: $w \in L_1 \Leftrightarrow f(w) \in L_2$[10]

Satz von Cook

Teilmenge der Klasse NP existiert

Stephen A. Cook zeigte, dass eine *Teilmenge der Klasse NP existiert*, auf die sich aus NP reduzieren

alle *Probleme aus NP reduzieren* lassen. Diese Teilmenge ist damit repräsentativ für die Schwierigkeit von NP und wird als NP-vollständig (englisch: NP complete) bezeichnet. Der nach ihm benannte Satz von Cook sagt aus, dass das *Erfüllbarkeitsproblem der Aussagenlogik* (SAT, v. engl. satisfiability) NP-vollständig ist.

Erfüllbarkeitsproblem der Aussagenlogik

Es ist allerdings nicht das einzige schwerste Problem, denn Richard M. Karp zeigte, dass es in NP Probleme gibt, auf die SAT reduziert werden können, die also *genauso schwer sind wie SAT*.[11]

genauso schwer sind wie SAT

SAT und k-SAT

Das **Erfüllbarkeitsproblem der Aussagenlogik** SAT und k-SAT mit $k \geq 3$, $k \in \mathbb{N}$ (Satz von Cook) fragt, ob eine aussagenlogische Formel erfüllbar ist.[12] Das Erfüllbarkeitsproblem der *Aussagenlogik* ist in exponentieller Zeit in Abhängigkeit der Anzahl der Variablen mit Hilfe einer Wahrheitstabelle entscheidbar. Diese *Wahrheitstabelle* kann nicht in polynomieller Zeit aufgestellt werden.[13]

Karps 21 NP-vollständige Probleme

Nachdem Stephen Cook 1971 den Nachweis erbrachte, dass SAT NP-vollständig ist, griff Richard Karp 1972 diese Idee auf und zeigte die NP-Vollständigkeit für 21 weitere Probleme. Diese werden als klassische NP-vollständige Probleme bezeichnet.[14]

Seit 1972 wurden mehr als 10000 Probleme als NP-vollständig charakterisiert.[15]

- **SATISFIABILITY:** das Erfüllbarkeitsproblem der Aussagenlogik für Formeln in konjunktiver Normalform
  - **CLIQUE:** Cliquenproblem
    - **SET PACKING:** Mengenpackungsproblem
    - **VERTEX COVER:** Knotenüberdeckungsproblem
      - **SET COVERING:** Mengenüberdeckungsproblem
      - **FEEDBACK ARC SET:** Feedback Arc Set
      - **FEEDBACK NODE SET:** Feedback Vertex Set
      - **DIRECTED HAMILTONIAN CIRCUIT:** siehe Hamiltonkreisproblem
        - **UNDIRECTED HAMILTONIAN CIRCUIT:** siehe Hamiltonkreisproblem
  - **0-1 INTEGER PROGRAMMING:** siehe Integer Linear Programming

[7] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 63.
[8] *Wikipedia-Artikel „Polynomialzeitreduktion“.*
[9] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 64.
[10] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 65.
[11] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 70.
[12] *Wikipedia-Artikel „Erfüllbarkeitsproblem der Aussagenlogik“.*
[13] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 71.
[14] *Wikipedia-Artikel „Karps 21 NP-vollständige Probleme“.*
[15] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 80.

3-SAT: siehe 3-SAT

CHROMATIC NUMBER: graph coloring problem

CLIQUE COVER

EXACT COVER: Problem der exakten Überdeckung

3-dimensional MATCHING: 3-dimensional matching (Stable Marriage mit drei Geschlechtern)

STEINER TREE: Steinerbaumproblem

HITTING SET: Hitting-Set-Problem

KNAPSACK: Rucksackproblem

JOB SEQUENCING: Job sequencing

PARTITION: Partitionsproblem

- MAX-CUT: Maximaler Schnitt

Aussagenlogische Formeln

Negation: $\neg A$

Disjunktion: Verknüpfung mit Oder $\vee$ $A \vee B$

Konjunktion: Verknüpfung mit Und $\wedge$ $A \wedge B$

Konjunktive Normalform: Konjunktion von Disjunktionstermen $(A \vee B) \wedge D \wedge (A \vee B \vee C)$

3-SAT => höchstens drei Literale (also Variablen) pro Klausel Aber beliebig viele verschiedene Variablen: $x_1, x_2, x_3, x_4, \ldots$ Klausel ist ein Disjunktionsterm (=Oder-Term)

Beispiel:

$F = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x$[16]

Problem des Handlungsreisenden

(TSP: engl. traveling salesman problem, Optimierungsproblem der Kombinatorik): Die Aufgabe besteht darin, eine Reihenfolge für den Besuch mehrerer Orte so zu wählen, dass keine Station außer der ersten mehr als einmal besucht wird, die gesamte Reisestrecke des Handlungsreisenden möglichst kurz und die

erste Station gleich der letzten Station ist. Eingabe: Ungerichteter Graph mit Längen als Beschriftungen für Kanten

Ausgabe: Kürzeste Tour, auf der alle Knoten

genau einmal vorkommen[17]

[16] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 72.
[17] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 73.

Hamiltonkreisproblem

Existiert ein geschlossener Pfad in einem Graphen, der jeden Knoten nur genau einmal enthält?

Wichtiges Problem der Graphentheorie Ist eine Verallgemeinerung des TSP, bei welchem nur nach dem kürzesten Hamiltonkreis in einem Graphen gefragt wird.

Teilsummenproblem

Das **Teilsummenproblem** (`Subset Sum` oder `SSP`) ist ein spezielles Rucksackproblem.[18] Gegeben sei eine Menge von ganzen Zahlen $I = \{w_1, w_2, \ldots, w_n\}$. Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke c ist.[19]

Rucksackproblem

(auch englisch knapsack problem) ist ein Optimierungsproblem der Kombinatorik.

Aus einer Menge von Objekten, die jeweils ein Gewicht und einen Nutzwert haben, soll eine *Teilmenge ausgewählt* werden, deren Gesamtgewicht eine vorgegebene *Gewichtsschranke nicht überschreitet*. Unter dieser Bedingung soll der *Nutzwert* der ausgewählten Objekte *maximiert* werden.[20]

Teilmenge ausgewählt
Gewichtsschranke nicht überschreitet
Nutzwert
maximiert

Cliquenproblem

Das **Cliquenproblem** fragt nach der Existenz einer Clique der Mindestgröße n in einem gegebenen Graphen.[21] Eine Clique ist eine Teilmenge von Knoten in einem ungerichteten Graphen, bei der *jedes Knotenpaar durch eine Kante* verbunden ist.[22]

Stabilitätsproblem

(Independent Set oder Co-Clique) fragt nach der Existenz einer Teilmenge von Knoten in einem gegebenen Graphen der Mindestgröße n, bei denen die Knoten zueinander *nicht adjazent* (= keine direkte Verbindung) sind.

nicht adjazent

Das Stabilitätsproblem ist das „Gegenproblem“ zum Cliquenproblem

Beispiel: Die neun blauen Knoten bilden im nebenstehenden Graphen eine (maximale) stabile Menge.[23]

[18] *Wikipedia-Artikel „Teilsummenproblem“*.
[19] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 74.
[20] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 75.
[21] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 76.
[22] *Wikipedia-Artikel „Cliquenproblem“*.
[23] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 77.

Knotenüberdeckungsproblem

Das **Knotenüberdeckungsproblem** (VERTEX COVER) fragt, ob zu einem gegebenen einfachen Graphen und einer natürlichen Zahl k eine Knotenüberdeckung der Größe von höchstens k existiert.[24]

Das heißt, ob es eine aus maximal k Knoten bestehende Teilmenge U der Knotenmenge gibt, so dass jede Kante des Graphen mit mindestens einem Knoten aus U verbunden ist.[25]

COL und k-COL

COL und k-COL mit $k \geq 3$, $k \in \mathbb{N}$ (graph coloring – Färbung eines Graphen) fragt, ob ein gegebener Graph mit k Farben so färbbar ist, dass zwei benachbarte Knoten niemals die gleiche Farben haben.

Literatur

[1] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*.

[2] *Wikipedia-Artikel „Cliquenproblem"*. `https://de.wikipedia.org/wiki/Cliquenproblem`.

[3] *Wikipedia-Artikel „Erfüllbarkeitsproblem der Aussagenlogik"*. `https://de.wikipedia.org/wiki/Erfüllbarkeitsproblem_der_Aussagenlogik`.

[4] *Wikipedia-Artikel „Karps 21 NP-vollständige Probleme"*. `https://de.wikipedia.org/wiki/Karps_21_NP-vollständige_Probleme`.

[5] *Wikipedia-Artikel „Knotenüberdeckung"*. `https://de.wikipedia.org/wiki/Knotenüberdeckung`.

[6] *Wikipedia-Artikel „NP-Schwere"*. `https://de.wikipedia.org/wiki/NP-Schwere`.

[7] *Wikipedia-Artikel „NP-Vollständigkeit"*. `https://de.wikipedia.org/wiki/NP-Vollständigkeit`.

[8] *Wikipedia-Artikel „Polynomialzeitreduktion"*. `https://de.wikipedia.org/wiki/Polynomialzeitreduktion`.

[9] *Wikipedia-Artikel „Teilsummenproblem"*. `https://de.wikipedia.org/wiki/Teilsummenproblem`.

[24] *Wikipedia-Artikel „Knotenüberdeckung"*.

[25] *Theoretische Informatik – Berechenbarkeit und Entscheidbarkeit*, Seite 78.