Komplexitätstheorie

Andreas M. Hohenauer

Algorithmen und Datenstrukturen

14. September 2025

Grundlagen

Die Komplexitätstheorie ermöglicht es, gegebene Probleme in Bezug auf ihre Ressourcenanforderungen zu klassifizieren und zu vergleichen. *Lohnt es sich, weiter nach einem viel schnelleren Algorithmus zu suchen? Wie viel Ressourcen (Zeit / Speicher) braucht ein Problem mindestens?* Komplexitätstheorie liefert eine Landkarte der Schwierigkeit von Problemen.

Ressourcen und Kostenmodelle
Typischerweise sind die **Laufzeit** (Anzahl elementarer Schritte) und der **Speicherplatz** in Abhängigkeit der Eingabelänge n relevant. Wir betrachten asymptotisches Verhalten für großes n.

Grundlagen

- Zur theoretischen Analyse reduziert man Optimierungs-/Suchprobleme oft auf **Ja/Nein**-Fragen (Entscheidungsprobleme). Beispiel: SAT: "Hat die Formel eine erfüllende Belegung?". Es besteht ein enger Zusammenhang zwischen Entscheidungsproblemen und den zugehörigen Optimierungsproblemen.
- Weiters besteht ein enger Zusammenhang zum Gebiet der formalen Sprachen. Zu diesen existieren (formale) Automaten, die deartige Sprachen verarbeiten können. Hierzu seien genannt: Endlicher Automat, Keller-Automat, Turing-Maschine. Diesen Zusammenhang genauer zu betrachten, sprengt jedoch den Rahmen dieses Skriptums bei weitem.

Komplexitätsklassen

- **Die Klasse** P**:** Menge aller Entscheidungsprobleme, die ein *deterministischer* Algorithmus in polynomieller Zeit löst (z.B. Sortieren, Kürzeste Wege mit nicht-negativen Gewichten, bipartite Matching). Praktisch: gilt als effizient lösbar.
- **Die Klasse** NP**:** "Nondeterministic-Polynomial": Probleme, bei denen man eine **gegebene Lösung** (Zertifikat) in polynomieller Zeit prüfen kann. Wichtig: "NP" heißt *nicht* "nicht polynomial" – viele NP-Probleme könnten theoretisch in P liegen. Unbekannt: $P = NP$?

Komplexitätsklassen

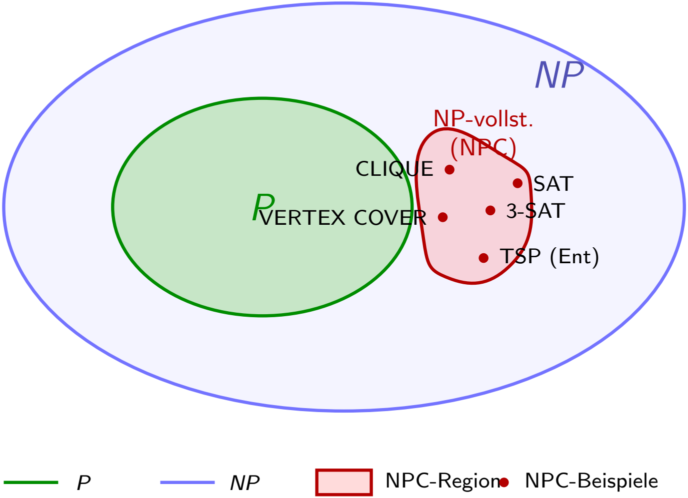

Abbildung: Beziehung der Klassen: $P \subseteq NP$; NP-vollständige Probleme (NPC)

Komplexitätsklassen

Eine nach wie vor offene Frage im Bereich der theoretischen Informatik ist, ob $P = NP$. Allgemein wird angenommen, dass $P \neq NP$ gilt. Anderenfalls wären viele der heute schwierigsten Probleme (NP-vollständige Probleme wie SAT, Travelling Salesman, Graph-Färbung usw.) plötzlich effizient lösbar. Praktisch hieße das: Alle Optimierungsprobleme, die bisher nur mit enormem Rechenaufwand oder Näherungsverfahren lösbar sind, könnten in polynomialer Zeit berechnet werden. Aufgrund zu vieler fehlgeschlagener Versuche, $P = NP$ zu beweisen, wird dies von vielen Experten als unwahrscheinlich angesehen.

NP-schwere und NP-Vollständigkeit

- **NP-schwer**: Mindestens so schwer wie jedes Problem in NP (alle NP-Probleme lassen sich darauf reduzieren).
- **NP-vollständig** (NPC): In NP *und* NP-schwer.

Intuition: NP-vollständige Probleme sind "universelle harte Knoten" der Klasse NP.

Reduktionen (Kernidee):
Um zu zeigen, dass neues Problem X schwer ist: Wähle bekanntes NP-vollständiges Problem Y, konstruiere *polynomielle* Abbildung $Y \to X$. Wenn X effizient wäre, wäre auch Y effizient. Damit: X ist mindestens so schwer wie Y.

Klassische NP-vollständige Probleme

Beispiele (alle Entscheidungsvarianten):

- SAT (Erfüllbarkeit boolescher Formel)
- 3-SAT (Klauseln mit je 3 Literalen)
- CLIQUE (enthält der Graph eine Clique Größe k?)
- VERTEX COVER (gibt es eine Überdeckungsmenge Größe k?)
- HAMILTONIAN CYCLE / PATH
- PARTITION / SUBSET-SUM
- TSP (Travelling Salesman Problem)

Was bedeutet NP-vollständig in der Praxis?
Für das Problem gibt es sehr wahrscheinlich keinen polynomiellen Algorithmus. Für kleine Instanzen (z.B. $n < 100$) sind exakte Verfahren (Backtracking, Branch-and-Bound, SAT-Solver) oft praktikabel. Für größere Instanzen helfen oft Approximationen, Parameterisierung oder Heuristiken.

Weitere Klassen:
PSPACE (polynomieller Speicher), EXPTIME (exponentielle Zeit).

Überblick P vs. NP

Klasse	Idee
P	effizient berechenbar
NP	Lösung schnell prüfbar
NPC	härteste Probleme in NP
NP-schwer	mindestens so schwer wie NP, evtl. nicht in NP
Co-NP	Komplemente zu NP-Sprachen
PSPACE	beschränkt durch polynomiellen Speicher

Typische Beweisstrategie für NP-Vollständigkeit

1. Zeige: Problem X liegt in NP (Zertifikat + Verifizierer in Polyzeit).
2. Wähle bekanntes NP-vollständiges Problem Y.
3. Konstruiere polynomielle Reduktion $Y \to X$.
4. Schluss: X ist NP-vollständig.

Zusammenfassung

Komplexitätstheorie erklärt Grenzen des Machbaren. Sie hilft zu entscheiden, ob wir (a) nach besserem exakten Algorithmus suchen, (b) approximieren, (c) heuristisch vorgehen oder (d) Problem einschränken. Kernmerksatz: *NP-vollständig* heißt sehr wahrscheinlich: "Keine allgemeine schnelle exakte Lösung – such Alternativen."