UNIVERSITÄT KONSTANZ
LEHRSTUHL FÜR PRAKTISCHE INFORMATIK
Prof. Dr. D. Wagner / Annegret Liebers

September 1998

Grundlagen: Die Komplexitätsklassen $\mathcal{P}$, $\mathcal{NP}$ und $\mathcal{NPC}$

Zu einer gegebenen Problemstellung interessieren wir uns dafür, ob es einen Algorithmus mit polynomialer Laufzeit gibt, der das Problem löst[1], oder ob dies „wahrscheinlich" nicht der Fall ist. Wir interessieren uns also für den Schwierigkeitsgrad oder die *Komplexität* eines Problems. Die folgenden Ausführungen können nur versuchen, die Bedeutung der in diesem Zusammenhang wichtigen Problemklassen $\mathcal{P}$, $\mathcal{NP}$ und $\mathcal{NPC}$ zu erläutern, eine präzise Definition auf der Grundlage von Turing-Maschinen, wie sie im Rahmen der Vorlesung *Informatik III: Theoretische Informatik* erfolgt, kann zum Beispiel in [GJ79, GJ91][2] nachgelesen werden.

Wir illustrieren die Frage nach der Komplexität eines Problems anhand der folgenden beiden Graphenprobleme[3]:

Input: Ein ungerichteter Graph $G = (V, E)$.

Output: Ein **Kantenschnitt minimaler Kardinalität**, das heißt eine möglichst kleine Kantenmenge $E' \subseteq E$ so daß der durch die Wegnahme der in E' enthaltenen Kanten entstehende Teilgraph G' von G aus mindestens zwei Zusammenhangskomponenten besteht. (Falls G unzusammenhängend ist, ist die leere Menge ein minimaler Kantenschnitt für G.)

Es gibt polynomiale Algorithmen, die dieses Problem für jeden Eingabegraphen lösen, d.h. deren Laufzeit durch ein Polynom in der Größe des Eingabegraphen beschränkt ist[4]. Dagegen ist für das folgende Problem kein solcher Algorithmus bekannt:

Input: Ein ungerichteter Graph $G = (V, E)$ und eine Menge von Farben F.

Output: Eine **Knotenfärbung** von G, das heißt eine Abbildung $f : V \to F$ mit

$$u, v \in V \wedge \{u, v\} \in E \implies f(u) \neq f(v)$$

oder die Aussage, daß es eine solche Knotenfärbung nicht geben kann.

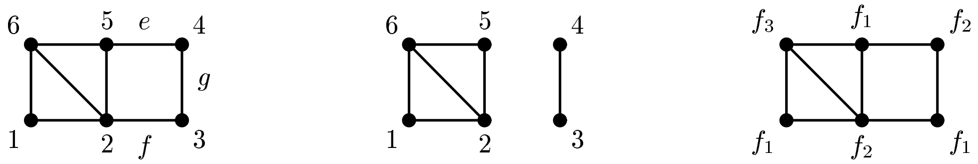

Abbildung 1: **Links:** Ein Graph G. **Mitte:** Ein Kantenschnitt minimaler Kardinalität für G besteht aus zwei Kanten. G besitzt mehrere solcher Schnitte, z.B. $\{e, f\}$ oder $\{e, g\}$. Hier ist der durch Wegnahme der Kanten e und f entstandene Teilgraph von G dargestellt. **Rechts:** Eine Knotenfärbung von G unter Verwendung der drei „Farben" f_1, f_2 und f_3.

1 Entscheidungs- versus Optimierungsprobleme

Um Problemstellungen in einheitlicher Weise formulieren und um ihren „Schwierigkeitsgrad" miteinander vergleichen zu können, legen wir folgende Definitionen fest:

Problem. Ein *Problem* Π besteht aus einer allgemeinen Beschreibung aller seiner Parameter sowie aus einer Beschreibung aller Eigenschaften, die eine Lösung haben soll.

Instanz. Eine *Instanz* eines Problems Π erhalten wir durch Festlegung aller seiner Parameter auf konkrete Werte.

[1] Zum Begriff des Algorithmus und zur Laufzeit eines Algorithmus siehe auch das Handout *Grundlagen: Algorithmen und ihre Laufzeit*

[2] [GJ79] hat als erstes Buch die Komplexitätsklassen $\mathcal{P}$, $\mathcal{NP}$ und $\mathcal{NPC}$ behandelt. [GJ91] ist eine Neuauflage mit Korrekturen und Ergänzungen im Anhang. Inzwischen gibt es Lehrbücher zu diesem Thema wie z.B. [Weg93] und [Sch97], und viele allgemeinere Lehrbücher wie [CLR90, CLR94], [Jun94] oder das (sehr lesbare) Vorlesungsskript [EWHK$^+$96] enthalten Kapitel zu Komplexitätsklassen. [GJ79, GJ91] kann aber als das Standardwerk angesehen werden.

[3] Definitionen zu Graphen können im Handout *Grundlagen: Begriffe zu Graphen* nachgelesen werden.

[4] Siehe z.B. die Behandlung von Schnitten und Flüssen in [Jun94] oder den (ohne Flüsse auskommenden) Algorithmus in [SW94].

Entscheidungsproblem. Ein *Entscheidungsproblem* ist ein Problem, bei dem die gesuchte Lösung nur aus der Antwort „ja“ oder „nein“ besteht.

Zu den beiden Beispielproblemen lassen sich auf natürliche Art und Weise Entscheidungsprobleme formulieren. Zunächst zum erstgenannten Problem:

> **Gegeben:** Ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl K.
>
> **Frage:** Gibt es in G einen **Kantenschnitt** E' mit $|E'| \leq K$?

Der Graph G aus Abbildung 1 zusammen mit einer natürlichen Zahl K bildet eine Instanz dieses Entscheidungsproblems. Für $K \geq 2$ lautet die Antwort auf die im Problem gestellte Frage „ja“. Für $K = 1$ lautet die Antwort „nein“, denn es gibt keine Kante in G, durch deren Wegnahme der Graph in zwei Zusammenhangskomponenten zerfällt. Nun zum zweiten Problem:

> **Gegeben:** Ein ungerichteter Graph $G = (V, E)$ und eine Menge von Farben F.
>
> **Frage:** Gibt es eine **Knotenfärbung** $f : V \to F$?

Eine äquivalente Formulierung des so formulierten Knotenfärbungsproblems ist

> **Gegeben:** Ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl K.
>
> **Frage:** Gibt es eine Knotenfärbung für G, die K oder weniger Farben benutzt ?

Der Graph G aus Abbildung 1 zusammen mit einer natürlichen Zahl K stellt wieder eine Instanz dieses Problems dar. Für $K \geq 3$ lautet die im Problem gestellte Frage für diese Instanz „ja“, für $K \leq 2$ lautet die Antwort „nein“, denn es ist nicht möglich, die Knoten von G mit zwei oder weniger Farben zu färben.

Optimierungsproblem. Bei einem *Optimierungsproblem* wird durch eine zu erfüllende Bedingung eine Menge von *zulässigen Lösungen* festgelegt („potentielle Lösungen“). Eine Lösung des Optimierungsproblems ist eine zulässige Lösung, die ein gestelltes *Optimierungskriterium* (z.B. Minimierung oder Maximierung einer Funktion) bestmöglich erfüllt. Eine solche Lösung heißt auch *optimale Lösung*.

Das Kantenschnittproblem war bereits als Optimierungsproblemen formuliert:

> **Gegeben:** Ein ungerichteter Graph $G = (V, E)$.
>
> **Gesucht:** Ein **Kantenschnitt minimaler Kardinalität**.

Zu einer Instanz $G = (V, E)$ für dieses Problem ist jede Kantenmenge $E' \subseteq E$, durch deren Wegnahme aus G der Graph unzusammenhängend wird, eine zulässige Lösung. Die hier zu minimierende Funktion ist die Kardinalität dieser Menge, und jeder Kantenschnitt minimaler Kardinalität ist eine optimale Lösung. Abbildung 1 stellt einen Graph G und einen Kantenschnitt minimaler Kardinalität dar. Eine naheliegende Optimierungsversion des Knotenfärbungsproblems ist die folgende:

> **Gegeben:** Ein ungerichteter Graph $G = (V, E)$.
>
> **Gesucht:** Eine **Knotenfärbung** von G, die möglichst wenige Farben benutzt.

Zu einem Graph G ist jede Knotenfärbung von G (auch die triviale Lösung, jedem Knoten eine eigene Farbe zuzuweisen, so daß die Knotenfärbung genau $|V|$ Farben benutzt) eine zulässige Lösung. In Abbildung 1 ist ein Graph G mit einer Färbung unter Benutztung von drei Farben dargestellt. Für diese Instanz ist die dargestellte Lösung optimal.

Wir teilen nun die Probleme nach ihrem „Schwierigkeitsgrad“ in Klassen ein, und beschränken uns dabei auf die Betrachtung von Entscheidungsproblemen. (Wie bereits an den beiden Beispielproblemen illustriert, steckt in jedem Optimierungsproblem immer auch ein Entscheidungsproblem.)

2 Die Klassen $\mathcal{P}$ und $\mathcal{NP}$

Definition: Die Problemklasse $\mathcal{P}$ ist die Menge derjenigen Entscheidungsprobleme, für die es einen Algorithmus gibt, der zu jeder möglichen Eingabe (d.h. zu jeder Instanz) in polynomialer Zeit die (korrekte) Antwort liefert. Der Name $\mathcal{P}$ steht dabei für *polynomial*.

Das Entscheidungsproblem Kantenschnitt ist also ein Element von $\mathcal{P}$. Für das Entscheidungsproblem Knotenfärbung ist dagegen kein polynomialer Lösungsalgorithmus bekannt, d.h. es ist offen, ob Knotenfärbung in $\mathcal{P}$ ist oder nicht.

Das Problem Knotenfärbung hat allerdings folgende schwächere Eigenschaft: Wenn zu einer Instanz G, K von Knotenfärbung eine Menge F und eine Abbildung $f \rightarrow F$ angegeben wird, dann läßt sich leicht (und zwar insbesondere in polynomialer Zeit) überprüfen, ob f eine Knotenfärbung für G mit K oder weniger Farben darstellt. Wenn also zu einer Instanz von Knotenfärbung bewiesen werden soll, daß die Lösung „ja" lautet, dann genügt es, eine Knotenfärbung f, die K oder weniger Farben benutzt, als *Zertifikat* anzugeben, und es kann (in polynomialer Zeit) nachgeprüft werden, ob f tatsächlich eine Knotenfärbung ist. Anders ausgedrückt kann ein Zertifikat dafür, daß die Antwort „ja" lautet, in polynomialer Zeit *verifiziert* werden. Alle Probleme, die diese schwächere Eigenschaft der polynomialen Verifizierbarkeit aufweisen, fassen wir in der Menge $\mathcal{NP}$ zusammen:

Definition: Die Problemklasse $\mathcal{NP}$ ist die Menge derjenigen Entscheidungsprobleme, für die es einen Algorithmus gibt, so daß für jede mögliche Eingabe gilt: Die Antwort lautet „ja" genau dann, wenn ein Zertifikat existiert, anhand dessen der Algorithmus in polynomialer Zeit verifizieren kann, daß die Antwort „ja" lautet. Der Name $\mathcal{NP}$ steht dabei für *nichtdeterministisch polynomial*.

Das Entscheidungsproblem Knotenfärbung gehört also zur Klasse $\mathcal{NP}$. Es gilt $\mathcal{P} \subseteq \mathcal{NP}$, und die Frage, ob $\mathcal{P} = \mathcal{NP}$ oder ob $\mathcal{P} \neq \mathcal{NP}$ gilt, ist ein wichtiges offenes Problem. Es wird vermutet, daß $\mathcal{P} \neq \mathcal{NP}$, aber ein Beweis ist nicht in Sicht. Falls tatsächlich $\mathcal{P} \neq \mathcal{NP}$ gilt, dann gibt es Probleme in $\mathcal{NP} \setminus \mathcal{P}$, und diese Probleme sind nicht in polynomialer Zeit lösbar.

Für viele andere Probleme wiederum ist es nicht offensichtlich, ob sie zu $\mathcal{NP}$ gehören oder nicht. Lautet ein Entscheidungsproblem zum Beispiel „Gegeben ein Graph G und eine natürliche Zahl K, gibt es zu G *keine* Knotenfärbung mit K oder weniger Farben ?", dann ist nicht klar, wie ein in polynomialer Zeit verifizierbares Zertifikat für dieses Problem aussehen sollte.

In verkürzter Sprechweise können wir die Definitionen von $\mathcal{P}$ und $\mathcal{NP}$ folgendermaßen prägnant formulieren: Die Klasse $\mathcal{P}$ besteht aus den in polynomialer Zeit *lösbaren* Entscheidungsproblemen, während die Klasse $\mathcal{NP}$ aus den in polynomialer Zeit *verifizierbaren* Entscheidungsproblemen besteht.

3 Die Klasse $\mathcal{NPC}$

Solange die Frage, ob $\mathcal{P} \neq \mathcal{NP}$ gilt, unbeantwortet ist, sollen wenigstens die „schwierigsten" Probleme aus $\mathcal{NP}$ identifiziert werden, wobei wir „schwierigst" folgendermaßen definieren: Wenn es für eines der schwierigsten Probleme aus $\mathcal{NP}$ einen polynomialen Lösungsalgorithmus gibt, dann gibt es für *alle* Probleme aus $\mathcal{NP}$ einen polynomialen Lösungsalgorithmus. Falls andererseits $\mathcal{P} \neq \mathcal{NP}$ gezeigt wird, dann ist für die schwierigsten Probleme bereits bewiesen, daß sie nicht in $\mathcal{P}$ sind (d.h. daß es keinen polynomialen Lösungsalgorithmus für sie gibt). Diese schwierigsten Probleme nennen wir **$\mathcal{NP}$-vollständig**. Die Klasse der $\mathcal{NP}$-vollständigen Probleme heißt $\mathcal{NPC}$ (für **$\mathcal{NP}$-complete**), und das Knotenfärbungsproblem gehört zu dieser Klasse.

Wenn wir für ein Problem $\Pi \in \mathcal{NP}$ trotz intensiver Suche keinen polynomialen Lösungsalgorithmus finden, dann wollen wir versuchen nachzuweisen, daß Π zu diesen „schwierigsten" Problemen gehört, d.h. daß $\Pi \in \mathcal{NPC}$ gilt. Wir müssen also zeigen, daß alle Probleme aus $\mathcal{NP}$ in polynomialer Zeit lösbar wären, wenn Π in polynomialer Zeit lösbar wäre. Wir definieren dazu die Transformation eines Problems in ein anderes:

Polynomiale Transformation. Seien Π_1 und Π_2 zwei Entscheidungsprobleme. Π_1 ist polynomial transformierbar in Π_2 (Schreibweise $\Pi_1 \propto \Pi_2$) genau dann, wenn es eine Funktion f gibt, die zu jeder Instanz I_1 von Π_1 eine Instanz I_2 von Π_2 erzeugt, so daß gilt: Die Berechnung von f benötigt eine durch ein Polynom in der Größe von I_1 beschränkte Laufzeit, und die Antwort für I_1 lautet „ja" genau dann, wenn die Antwort für $f(I_1) = I_2$ auch „ja" lautet.

Damit lautet die Definition $\mathcal{NP}$-vollständiger Probleme nun wie folgt:

$$\Pi \in \mathcal{NPC}, \text{ wenn } \quad \Pi \in \mathcal{NP} \text{ und wenn } \quad \forall\, \Pi' \in \mathcal{NP} : \Pi' \propto \Pi$$

Ein Problem Π mit $\Pi \notin \mathcal{NP}$ (oder für das nicht bekannt ist, ob $\Pi \in \mathcal{NP}$), aber für das die zweite Bedingung gilt, heißt **$\mathcal{NP}$-schwer** (engl. **$\mathcal{NP}$-hard**).

Polynomiale Transformierbarkeit ist transitiv: $\Pi_1 \propto \Pi_2 \wedge \Pi_2 \propto \Pi_3 \implies \Pi_1 \propto \Pi_3$. Daraus ergibt sich folgende Technik für den Beweis der $\mathcal{NP}$-Vollständigkeit eines Problems $\Pi \in \mathcal{NP}$: Ausgehend von einem Problem Π_1, für das $\mathcal{NP}$-Vollständigkeit bereits gezeigt wurde, zeige, daß wenn man das Problem Π in polynomialer Zeit lösen könnte, man dann auch das Problem Π_1 in polynomialer Zeit lösen könnte, d.h. zeige $\Pi_1 \propto \Pi$.

Um diese Technik anwenden zu können, benötigen wir ein „initiales“ $\mathcal{NP}$-vollständiges Problem, dessen $\mathcal{NP}$-Vollständigkeit auf andere Weise gezeigt wurde. Ein solches Problem liefert der

Satz von Cook (1971): Erfüllbarkeit ist $\mathcal{NP}$-vollständig.

Dabei ist Erfüllbarkeit das folgende Entscheidungsproblem:

SAT. Eine Instanz des Problems **Erfüllbarkeit** (engl. *Satisfiability* oder kurz *SAT*) ist gegeben durch eine Boolesche Funktion $f(x_1, x_2, \ldots, x_n)$, die dargestellt ist als Konjunktion von Disjunktionen. Die Disjunktionen heißen auch „Klauseln“. Es ist nun zu entscheiden, ob es eine Belegung der Variablen $x_1, x_2, \ldots, x_n$ mit Wahrheitswerten „wahr“ und „falsch“ gibt, so daß die Funktion den Wert „wahr“ annimmt. Für die SAT-Instanz $f(a,b) = (a \vee \overline{b}) \wedge (\overline{a} \vee b)$ ist die Antwort auf die Frage nach der Erfüllbarkeit zum Beispiel „ja“, denn mit $a =$ wahr und $b =$ wahr wird $f(a,b)$ wahr. (Eine weitere erfüllende Belegung ist $a =$ falsch und $b =$ falsch.) Dagegen gibt es für $f(a,b) = (a \vee b) \wedge (a \vee \overline{b}) \wedge (\overline{a})$ keine erfüllende Belegung.

Um die $\mathcal{NP}$-Vollständigkeit des Problems Knotenfärbung zu zeigen, wurde die $\mathcal{NP}$-Vollständigkeit eines Spezialfalls von SAT benutzt: Das Problem 3SAT ist folgendermaßen formuliert: Gegeben ist eine Boolesche Funktion $f(x_1, x_2, \ldots, x_n)$, die dargestellt ist als Konjunktion von Disjunktionen, und bei der jede Disjunktion aus genau drei Literalen besteht. Gefragt ist wieder die Erfüllbarkeit von f. 3SAT ist $\mathcal{NP}$-vollständig (siehe z.B. [GJ79, Kapitel 3.1.1]), und es gilt 3SAT $\propto$ Knotenfärbung (siehe z.B. [Man89, Kapitel 11.4.5]).

Literatur

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw–Hill, 1990. ISBN 0-262-03141-8, 0-07-013143-0. Unibib KN: kid 112/c67, lbs 830/c67.

[CLR94] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw–Hill, 1994. ISBN 0-262-03141-8, 0-07-013143-0, 0-262-53091-0. Unibib KN: kid 112/c67c, lbs 830/c67.

[EWHK+96] Thomas Emden-Weinert, Stefan Hougardy, Bernd Kreuter, Hans-Jürgen Prömel, and Angelika Steger. Einführung in Graphen und Algorithmen. Humboldt-Universität zu Berlin, Institut für Informatik. Electronically available under http://www.informatik.hu-berlin.de/Institut/struktur/algorithmen/ga/, 1996.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$–Completeness*. W H Freeman & Co Ltd, 1979. Unibib KN: kid 112/g17, lbs 840/g17, y 16053.

[GJ91] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of $\mathcal{NP}$–completeness*. W H Freeman & Co Ltd, 1991. Unibib KN: kid 112/g17a.

[Jun94] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI-Wissenschaftsverlag, 1994. Unibib KN: kid 114/j96a(3), kid 604/j96, lbs 840/j96.

[Man89] Udi Manber. *Introduction to algorithms: a creative approach*. Addison–Wesley Publishing Company, 1989. Unibib KN: kid 112/m16, lbs 840/m16.

[Sch97] Uwe Schöning. *Theoretische Informatik — kurzgefasst*. Spektrum Akademischer Verlag, third edition, 1997. ISBN 3-8274-0250-6. Unibib KN: lbs 840/s23.

[SW94] Mechtild Stoer and Frank Wagner. A Simple Min Cut Algorithm. In Jan v. Leeuwen, editor, *Second European Symposium on Algorithms, ESA'94*, pages 141–147. Springer-Verlag, Lecture Notes in Computer Science, vol. 855, 1994. Unibib KN: kid 100:i/l22-855.

[Weg93] Ingo Wegener. *Theoretische Informatik. Eine algorithmenorientierte Einfürung*. Teubner, 1993. Unibib KN: kid 100/w23.