Komplexitätstheorie: Ergänzende Anmerkungen

Hans Ulrich Simon

A Entscheidungs- versus Optimierungsprobleme

Es scheint so, daß die Theorie der NP-vollständigen Probleme die Komplexität von Sprachen erfaßt, aber nicht adäquat mit sogenannten Optimierungsproblemen umgeht, welche in der Praxis eine große Rolle spielen. Dies täuscht jedoch insofern, als das Lösen von Optimierungsproblemen auf das Erkennungsproblem von Sprachen reduziert werden kann. Das wollen wir am Beispiel des Problems des Handelsreisenden (Travelling Salesman Problem oder kurz TSP) illustrieren.

TSP: ermittle eine optimale Lösung
Gegeben eine Matrix $D \in \mathbb{N}_0^{n \times n}$, bei der Eintrag d_{ij} die Distanz zwischen Städten C_i, C_j angibt. Finde eine kürzeste Rundreise, die durch alle Städte $C_1, \ldots, C_n$ führt (in beliebiger Reihenfolge).

In mathematischer Sprechweise: Finde eine Permutation σ von $[1:n]$, die

$$\mathrm{COST}(\sigma) := \sum_{i=1}^{n-1} d_{\sigma(i),\sigma(i+1)} + d_{\sigma(n),\sigma(1)}$$

minimiert. Die optimale Permutation zu Eingabe D notieren wir als OPTIMAL-SOLUTION(D).

Verwandte Problemvarianten sind:

TSP: berechne Kosten einer optimalen Lösung
Finde die Länge der kürzesten Rundreise. Diese Länge notieren wir als OPTIMAL-VALUE(D).

TSP: berechne Zeugen zu einer gegebenen Kostenschranke
Gegeben zusätzlich eine Kostenschranke K. Falls möglich, finde eine Rundreise deren Länge K nicht überschreitet. Falls eine solche Rundreise existiert notieren wir sie als WITNESS(D, K). (Falls sie nicht existiert ist WITNESS(D, K) undefiniert.)

TSP: entscheide Einhaltbarkeit einer gegebenen Kostenschranke Entscheide, ob eine Rundreise existiert, deren Länge die gegebene Kostenschranke K nicht überschreitet. Das Ergebnis TRUE oder FALSE wird als RECOGNITION(D, K) notiert.

Die letzte Problemvariante entspricht dem Wortproblem für die formale Sprache:

$$L_{TSP} := \{D, K |\ \exists \sigma : \mathrm{COST}(\sigma) \leq K\}.$$

Bemerkung A.1 *(ohne Beweis)*

L_{TSP} ist NP-vollständig, sogar wenn wir verlangen, daß D symmetrisch ist und die Dreiecksungleichung erfüllt, d.h., sogar wenn

(i) $\forall i, j : d_{ij} = d_{ji}$,

(ii) $\forall i, j, k : d_{ik} \leq d_{ij} + d_{jk}$.

Wir wollen nun zeigen, daß alle vier Problemvarianten polynomiell verknüpft sind, d.h., ein deterministischer Polynomialzeitalgorithmus für eine Variante kann in einen nicht wesentlich aufwendigeren Algorithmus für alle anderen Varianten transformiert werden.

Die in Abbildung 1 dargestellt Hierarchie ist offensichtlich. Hierbei bedeutet $A \to B$, daß eine polynomiell zeitbeschränkte DTM M', die Problem (oder Problemvariante) B löst, in eine polynomiell zeitbeschränkte DTM M für Problem (oder Problemvariante) A transformiert werden kann.

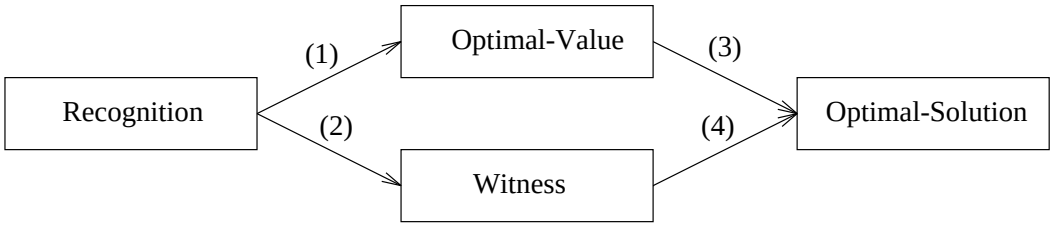

Abbildung 1: Reduktion des Entscheidungsproblems auf das Optimierungsproblem.

Zu (1): RECOGNITION(D, K) kann genau dann auf TRUE gesetzt werden, wenn OPTIMAL-VALUE$(D) \leq K$.

Zu (2): RECOGNITION(D, K) kann genau dann auf TRUE gesetzt werden, wenn WITNESS(D, K) nicht undefiniert ist.

Zu (3): Die minimalen Kosten zur Eingabe D ergeben sich aufgrund der Gleichung OPTIMAL-VALUE(D) = COST(OPTIMAL-SOLUTION(D)).

Zu (4): WITNESS(D, K) kann gleich OPTIMAL-SOLUTION(D) gesetzt werden, falls die Kosten der optimalen Lösung Kostenschranke K nicht überschreiten. Andernfalls ist WITNESS(D, K) undefiniert.

Die Begründungen zu (1), ..., (4) gelten für alle Optimierungsprobleme, bei denen die Kosten- bzw. Profitfunktion in Polynomialzeit berechenbar ist.

Überraschender ist, daß die Problemreduktionen in der obigen Hierarchie auch in der umgekehrten Richtung möglich sind, wie es in Abbildung 2 zu sehen ist.

Zu (5): Wenn wir die Kostenschranke K auf OPTIMAL-VALUE(D) setzen, ergibt sich eine optimale Lösung aufgrund der Gleichung OPTIMAL-SOLUTION(D) = WITNESS(D, K).

Zu (6): Wir zeigen zunächst: OPTIMAL-VALUE $\to$ RECOGNITION.
Eine Rundreise mit Kosten

$$\text{MAX} := n \cdot \max_{1 \leq i,j \leq n} d_{ij}$$

ist sicher möglich. OPTIMAL-VALUE(D) kann dann durch Binärsuche im Kostenintervall [0 : MAX] ermittelt werden:

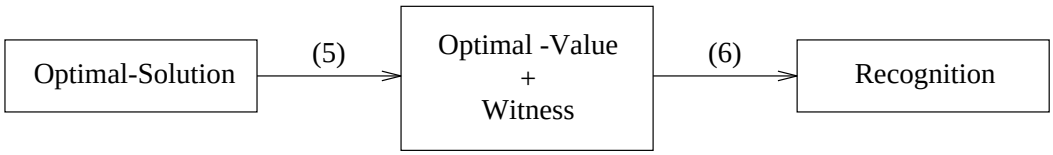

Abbildung 2: Reduktion des Optimierungproblems auf das Entscheidungsproblem.

$L \leftarrow 0$; (Initialisierung einer unteren
$U \leftarrow$ MAX; und oberen Kostenschranke)
while $L \neq U$ (Binäre Suche)
do $M \leftarrow \lfloor \frac{L+U}{2} \rfloor$;
  if RECOGNITION(D, M)
    then $U \leftarrow M$ **else** $L \leftarrow M + 1$
  fi
od
OPTIMAL-VALUE$(D) \leftarrow L$. (Ausgabe)

Bleibt zu zeigen: WITNESS $\rightarrow$ RECOGNITION

Idee Teste für jede Kante (i, j), ob sie für WITNESS gebraucht wird.

Methode Für jede Kante (i, j) wird probeweise $d_{ij} = K + 1$ gesetzt. Falls RECOGNITION immer noch die Existenz einer Rundreise σ mit Kosten $\leq K$ signalisiert, ist Kante (i, j) für σ unwichtig.

Eine formale Beschreibung der Methode liest sich wie folgt:

if not (RECOGNITION(D, K)) (Vorabtest: WITNESS definiert?)
  then output UNDEFINED and STOP
fi;
for all $1 \leq i, j \leq n$ with $i \neq j$
do puffer $\leftarrow d_{ij}$; (Retten der Distanz)
  $d_{ij} \leftarrow K + 1$;
  if not (RECOGNITION(D, K)) (Kante wichtig)
    then $d_{ij} \leftarrow$ puffer;
      include (i, j) into WITNESS(D, K)
  fi
od

Auf diese Weise werden genau die Kanten einer brauchbaren Rundreise in WITNESS(D, K) aufgenommen.
Begründungen (5), (6) sind auf alle Optimierungsprobleme anwendbar, die

(i) eine obere Schranke $2^{poly(n)}$ für die Kosten (oder den Profit) einer optimalen Lösung zu einer Eingabe der Kodierungslänge n haben,

(ii) selbstreduzierbar sind.

Mit *Selbstreduzierbarkeit* ist die oben praktizierte Reduktion der Problemvariante WITNESS auf die Problemvariante RECOGNITION gemeint.

Definition A.2 *Falls $A \leftrightarrow B$ (gleichbedeutend mit $A \rightarrow B$ und $B \rightarrow A$), heißen A, B* polynomiell verknüpft.

Es hat sich somit gezeigt, daß unter ein paar Voraussetzungen (die für die meisten klassischen Optimierungsprobleme erfüllt sind) die Problemvarianten RECOGNITION, OPTIMAL-VALUE, WITNESS und OPTIMAL-SOLUTION polynomiell verknüpft sind. Dies liefert die Legitimation sich beim Studium der (P $\neq$ NP)-Frage auf formale Sprachen zurückzuziehen.

B Entwurf von Approximationsalgorithmen

Wir beschränken uns in diesem Abschnitt darauf, ein Beispiel für einen Approximationsalgorithmus zum Problem des Handelsreisenden (TSP) zu präsentieren.

Ein *Approximationsalgorithmus* zu einem Optimierungsproblem Π ist ein polynomiell zeitbeschränkter Algorithmus A, der zu einer Eingabe I eine "legale Lösung" σ ausgibt, die eine "mathematische Gütegarantie" besitzt. Die Begriffe "legale Lösung" und "mathematische Gütegarantie" wollen wir informell am Beispiel TSP erläutern. Die legalen Lösungen zur Eingabe D (die Distanzmatrix vom Format $n \times n$) sind Rundreisen durch n Städte (formal Permutationen von $1, \ldots, n$). Wir sagen A hat *Güte* k, wenn garantiert ist, daß die Länge der von A konstruierten Rundreise schlimmstenfalls k mal so groß ist wie die Länge der kürzesten Rundreise. Das Optimum wird also maximal um den Faktor k verfehlt.

Für TSP kann man sich klar machen, daß für kein konstantes k eine solche Gütegarantie existieren kann (außer wenn $P = NP$). Wir reduzieren zu diesem Zweck das Problem des Hamiltonschen Kreises (HC) in geeigneter Weise auf das Problem des Handelsreisenden (TSP). Sei $G = (V, E)$ der Eingabegraph zu HC. Knotenmenge V bestehe aus $n \geq 2$ Knoten, die wir mit den Nummern von 1 bis n identifizieren. Will man lediglich HC$\leq_{pol}$TSP nachweisen, genügt folgende Reduktion:
Setze in der $(n \times n)$-Distanzmatrix D den Eintrag $d_{i,j}$ auf 1 falls $\{i, j\} \in E$, und andernfalls auf 2. Es folgt, daß bez. D genau dann eine Rundreise der Länge n existiert, wenn G einen Hamiltonschen Kreis enthält. Die kürzeste Rundreise, die mindestens eine Kante außerhalb von E verwendet, hat nämlich die Länge $n + 1$.
Diese Reduktion zeigt zwar (zusammen mit den Resultaten in Abschnitt **??**, daß TSP NP-vollständig ist. Es ist aber noch nicht ausgeschlossen, daß vernünftige Approximationsalgorithmen existieren. Betrachten wir aber nun die folgende leichte Modifikation der soeben geschilderten Reduktion:
Setze in der $(n \times n)$-Distanzmatrix D den Eintrag $d_{i,j}$ auf 1 falls $\{i, j\} \in E$, und andernfalls auf kn. Es folgt, daß bez. D genau dann eine Rundreise der Länge n existiert, wenn G einen Hamiltonschen Kreis enthält. Falls jedoch in G kein Hamiltonscher Kreis existiert,

muß die Rundreise mindestens eine Kante außerhalb von E verwenden. In diesem Fall hat sie mindestens die Länge $n-1+kn$. Gäbe es einen Approximationsalgorithmus für TSP der Güte k, so würde im Falle der Existenz eines Hamiltonschen Kreises in G eine Rundreise der Länge höchstens kn produziert, andernfalls jedoch eine Rundreise der Länge mindestens $kn+n-1 > kn$. Mit anderen Worten: mit Hilfe der approximativen Lösung des TSP könnten wir HC exakt lösen. Falls $P \neq NP$, ist dies jedoch nicht möglich. Somit gibt es keine garantierte Güte k für Approximationsalgorithmen zu TSP (außer wenn $P = NP$).

Die Theorie der NP-Vollständigkeit hat uns signalisiert, daß es vermutlich Zeitverschwendung ist nach einem Approximationsalgorithmus für TSP zu suchen. Betrachten wir nun aber die Einschränkung von TSP auf Distanzmatrizen D, die symmetrisch sind und die Dreiecksungleichung erfüllen (vgl. Abschnitt B). Diese Einschränkung ist für praktische Anwendungen durchaus vernünftig. Denn diese Einschränkungen besagen in salopper Formulierung:

Symmetrie Von A nach B ist es soweit wie von B nach A.

Dreiecksungleichung Von A nach C kann es nicht weiter sein als von A über B nach C.

Dieses eingeschränkte Problem wird als *metrisches TSP* bezeichnet.

Um einen Approximationsalgorithmus für das metrische TSP anzugeben, benötigen wir zunächst ein paar graphentheoretische Vorausetzungen. Ein *Graph* $G = (V, E)$ ist gegeben durch eine *Knotenmenge* V und eine *Kantenmenge* E. Bei *gerichteten Graphen* gilt $E \subseteq V \times V$, d.h., eine Kante $e \in E$ hat die Form (v, w) mit $v, w \in V$. Wir betrachten e als von v nach w gerichtet. Bei *ungerichteten Graphen* ist eine Kante $e = \{v, w\}$ eine zweielementige Teilmenge von V. Falls V_2 die Menge der zweielementigen Teilmengen von V bezeichnet, gilt also dann $E \subseteq V_2$. Falls $E = V \times V$ (gerichteter Fall) bzw. $E = V_2$ (ungerichteter Fall), sprechen wir von einem *vollständigen Graphen* oder auch einer *Clique*. Ein Graph mit *Kantengewichten* ist versehen mit einer Funktion w die jeder Kante $e \in E$ ein Gewicht $w(e)$ zuordnet. Für unsere Zwecke können wir im folgenden $w(e) \in \mathbb{N}_0$ voraussetzen. Eine Eingabe des metrischen TSP läßt sich auf die offensichtliche Weise als vollständiger, ungerichteter Graph mit Kantengewichten auffassen. Die Knoten repräsentieren die Städte und ein Kantengewicht die Distanz zwischen zwei Städten. Wegen obiger Symmetriebedingung können wir den Graphen als ungerichtet auffassen.

Im folgenden betrachten wir einen ungerichteten Graphen G. Ein *Weg* in G ist eine Folge $v_1, \ldots, v_r$ von $r \geq 1$ Knoten, wobei für alle $i = 1, \ldots, r-1$ Knoten v_i und v_{i+1} durch eine Kante verbunden sein müssen. (Ein Grenzfall ist der aus nur einem Knoten bestehende "Punktweg".) Falls $r \geq 2$ und $v_1 = v_r$, dann heißt der Weg auch *geschlossener Weg* oder *Kreis*. Ein *Hamiltonscher Kreis* in G ist ein Kreis in G, der jeden Knoten genau einmal durchläuft. Ein *Euklidischer Kreis* in G ist ein Kreis, der jede Kante in G genau einmal durchläuft. G heißt *zusammenhängend*, wenn zwei Knoten sich stets durch einen Pfad miteinander verbinden lassen. Ein *ungerichteter Baum* ist ein zusammenhängender, kreisloser ungerichteter Graph. Wenn man aus einem Baum eine Kante entfernt (ohne die Randknoten der Kante dabei mitzuentfernen), zerfällt er in zwei Teile. Ein Baum ist gewissermaßen die ökonomischste Art, alle Knoten durch Pfade miteinander zu verbinden. Ein *Untergraph* von $G = (V, E)$ ist gegeben durch eine Knotenmenge $V' \subseteq V$ und alle Kanten aus E, die

Knoten aus V' miteinander verbinden. Man spricht auch von dem *durch V' in G induzierten Untergraphen*. Ein *Teilgraph* von $G = (V, E)$ ist ein Graph $G' = (V', E')$ mit $V' \subseteq V$ und $E' \subseteq E$. Ein *Spannbaum (spanning tree)* von G ist ein Teilgraph der Form $T = (V, E')$, der ein Baum ist. Er enthält also alle Knoten von G und verbindet diese baumartig. Ein solcher Spannbaum kann natürlich nur dann existieren, wenn G zusammenhängend ist. Im Falle von Kantengewichten können wir $T = (V, E')$ die Kosten

$$c(T) = \sum_{e \in E'} w(e)$$

zuordnen. Ein *minimaler Spannbaum (minimum spanning tree)* ist ein Spannbaum minimaler Kosten. Es ist bekannt, daß minimale Spannbäume effizient berechnet werden können (zum Beispiel durch den Algorithmus von Kruskal).

Mit diesem Wissen ausgestattet ist es nun leicht einen Approximationsalgorithmus der Güte 2 für das metrische TSP zu skizzieren. Es sei $G = (V, E)$ der vollständige ungerichtete Graph mit n Knoten $1, \ldots, n$. Knoten i repräsentiert dabei die i-te Stadt. Kante $\{i, j\}$ erhält als Gewicht die Distanz $d_{i,j}$ zwischen den Städten i und j. Wir berechnen einen minimalen Spannbaum $T = (V, E')$ von G (zum Beispiel mit dem Algorithmus von Kruskal). Seien c die Gesamtkosten von T. Wir erhalten eine Rundreise

$$R(T) = 3, 5, 8, 5, 1, 5, 7, 10, 7, 5, 3, 2, 9, 4, 9, 6, 9, 2, 3,$$

wenn wir (wie in Abbildung 3 angedeutet) einmal um T herumlaufen und dabei die angetroffenen Knoten (=Städte) der Reihe nach auflisten. Da Rundreise $R(T)$ jede Kante von T zweimal durchläuft, hat sie Länge $2c$. $R(T)$ hat noch einen Schönheitsfehler: die Städte werden i.A. mehrfach besucht. Wir erhalten eine legale Lösung von TSP — also eine Permutation $P(T)$ von $1, \ldots, n$ —, wenn wir in $R(T)$ alle Vorkommen von Knoten außer dem ersten (also alle Duplikate) streichen. In unserem Beispiel führt dies zu

$$P(T) = 3, 5, 8, 1, 7, 10, 2, 9, 4, 6.$$

Wegen der Dreiecksungleichung kann $P(T)$ nicht länger als $R(T)$ sein. Die Kosten von $P(T)$ sind also maximal $2c$.

Wie verhält es sich nun mit einer optimalen Rundreise? Im Graphen G formt diese einen Hamiltonschen Kreis C. Entfernen wir aus C (irgend-) eine Kante, entsteht ein (sehr spezieller) Spannbaum $T(C)$ (s. Abbildung 4). Die Länge von Rundreise C ist nicht kleiner als die Gesamtkosten von $T(C)$. Diese wiederum betragen mindestens c. Also hat C mindestens die Länge c und der skizzierte Approximationsalgorithmus die Güte 2.

Eine leichte (aber trickreiche) Variation dieses Approximationsalgorithmus hat sogar die Güte 1.5. (Dies ist bislang der Weltrekord für das eingeschränkte TSP.)

C Praktischer Umgang mit harten Problemen

In der Programmierpraxis bei großen Softwareprojekten entstehen nach einer arbeitsreichen Strukturierungs- und Modellierungsphase klar umrissene Teilprobleme. Ein Lernziel beim Studium der Komplexitätstheorie besteht darin

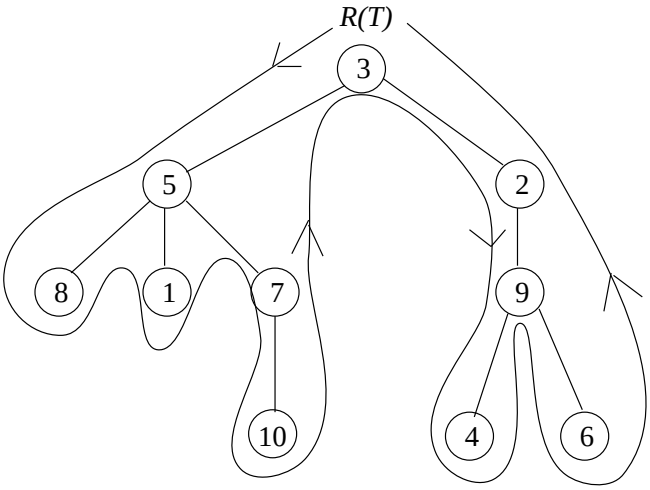

Abbildung 3: Die aus einem Minimum Spanning Tree abgeleitete Rundreise.

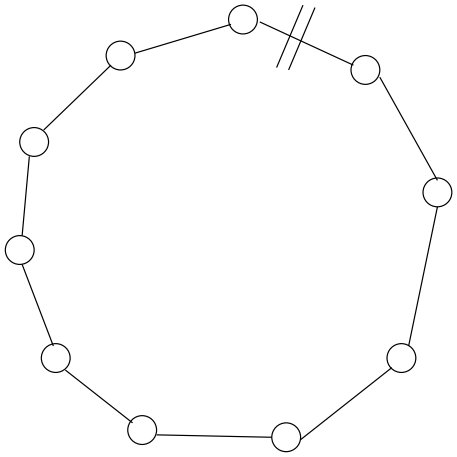

Abbildung 4: Der aus einer Rundreise abgeleitete Spanning Tree.

(a) zu erkennen, ob ein Teilproblem mit einem effizienten Algorithmus lösbar ist, oder ob es NP-hart ist,

(b) im Falle der NP-Härte effiziente Verfahren zu entwerfen, die das Problem zumindest teilweise lösen.

Zum Erreichen dieses Lernziels müssen wir auf Veranstaltungen des Hauptstudiums verweisen. Was diese kurze Einführung betrifft, beschränken wir uns auf die folgenden praktischen Hinweise:

1. Um zu erkennen, ob ein Problem mit effizienten Algorithmen perfekt lösbar ist, ist es wichtig, einen Vorrat an effizient lösbaren Grundlagenproblemen zu kennen. Diese Fähigkeit wird im Studium in den Vorlesungen Datenstrukturen und Effiziente Algorithmen vermittelt.

2. Um die NP-Härte eines Problems nachzuweisen, benötigt man eine gewisse Versiertheit im Umgang mit Polynomialzeitreduktionen, sowie einen Vorrat an bereits bekannten NP-harten Problemen. Einen leichten Vorgeschmack dafür haben wir in Kapitel **??** gegeben.

3. Es folgt eine Liste von populären Methoden, mit NP-harten Problemen umzugehen:

   - **Einschränkung auf Spezialfälle:** Mitunter ist man in einer praktischen Anwendung des NP-harten Problems mit einem Spezialfall zufrieden. Die Aufgabe besteht dann darin, das vorliegende Problem "vernünftig" einzuschränken und die Einschränkung exakt oder approximativ zu lösen. Im Abschnitt B haben wir das exemplarisch für TSP vorgeführt.
   - **Approximationsalgorithmen:** Bei manchen Optimierungsproblemen ist es möglich, fast-optimale Lösungen in Polynomialzeit zu konstruieren. Für TSP mit symmetrischen Distanzmatrizen und Dreiecksungleichung haben wir dies im Abschnitt B skizziert.
   - **Parametrisierte Komplexität:** Bei manchen NP-harten Problemen kann ein Parameter, sagen wir k, für die NP-Härte verantwortlich gemacht werden, so dass das Problem mit einer Laufzeit der Form $f(k)\text{poly}(n)$ gelöst werden kann. Hierbei bezeichnet n die Kodierungslänge der von k verschiedenen Parameter und f eine von k, aber nicht von n, abhängige Funktion.
   - **Probabilistische Verfahren:** Für manche Probleme kennt man keine deterministischen polynomiell zeitbeschränkten Verfahren, aber man kennt sogenannte *Monte-Carlo-* oder *Las-Vegas-Algorithmen*, die unter dem Einsatz von zufälligen Münzwürfen das zugrundeliegende Problem mit extrem kleiner Irrtumswahrscheinlichkeit effizient lösen. Allerdings ist diese Methode voraussichtlich nicht für NP-harte Probleme geeignet, da man vermutet, daß RP $\neq$ NP. RP bedeutet Random P. Diese Klasse liegt zwischen P und NP.
   - **(Meta-)Heuristiken:** Standardbeispiele dafür sind

- Branch und Bound
- Lokale Optimierung
- Simulated Annealing
- Genetische Algorithmen

Daneben gibt es ad-hoc-Heuristiken, die für ein konkretes Problem zusammengestrickt wurden. Meistens handelt es sich um Verfahren, die in der Praxis ganz gut funktionieren, die aber nicht ausreichend verstanden sind, um sie formal zu analysieren.