Gliederung

1. Einführung
    1. Definitionen P , NP, coNP, EXP, NEXP
    2. Bekannte Zusammenhänge zwischen dem Klassen
    3. Hypothesen zu deren zusammenhängen und deren Konsequenzen
2. Polynominal Reduktion und NP-Vollständigkeit
    1. Definition von NP-Vollständigkeit und (PReduktion) Polynominal Reduktion
    2. NP Probleme
    3. Bsp. Für eine Reduktion
3. Geschichte
    1. Stephen A. Cook , Richard Karp
    2. SAT als „ERSTES" Problem
    3. Beweis von SAT- Grundidee und Schwierigkeit
4. Reale Probleme und Lösungsansetze
5. Zusammenfassung

P, NP

Komplexitätsklasse P(auch: **PTIME**) ist diejenige Komplexitätsklasse, welche die Entscheidungsprobleme enthält, die in Polynomialzeit für deterministische Turingmaschinen lösbar sind. Diese Problemklasse wird allgemein als die Klasse der "praktisch lösbaren" Probleme betrachtet.

(Komplexitätsklasse NP). *NP ist die Klasse aller Entscheidungsprobleme L, die in polynomieller Zeit verifizierbar sind. D.h. es gibt einen Algorithmus A polynomieller Laufzeit und es gibt ein k ∈ N, so dass*

L = {w| ∃ *Wort x mit* |x| ≤ |w|^k, *so dass A(x,w) = 1}*

x *heißt auch Zeuge für w.*

coNP

coNP ist eine Komplexitätsklasse, in der genau die Sprachen enthalten sind , deren Komplemente zu NP gehören.

Einschub (Subset-Sum, Untermengensumme)

Gegeben sei eine Menge von ganzen Zahlen $M = \{a_1, a_2, ..., a_n\}$. Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke b ist (oft ist auch gefragt, die Schranke b exakt zu erreichen).

Komplement Bilden

coSubset-Sum

Eingabe dieselbe

Frage gibt es eine Untermenge deren summe kleiner, gleich b ist ?

Mein coSubset-Sum ist Falsch

1. Lösung der Frage in linearer zeit möglich
2. Richtige Definition

   Co − NP = {L|LComplement ∈ NP}

Es gibt ein Algorithmus A mit polynomieller Laufzeit und ein k ∈ N, so dass L = {w|∀x mit |x| ≤ |w|^k gilt A(x,w) = 0}.

Frage, ob *alle* nichtleeren Teilmengen *nicht b* als Summe haben.

Bei NP Wahr es

L = {w| ∃ *Wort x mit* |x| ≤ |w|^k, *so dass A(x,w) = 1}*

∃- es gibt eine Teilmenge

∀-alle möglichen Teilmengen sind nicht maximal und kleiner als b

EXPT, NEXPT

Komplexitätsklasse **EXPTIME (EXP)** ist die Komplexitätsklasse der Entscheidungsprobleme, die von einer deterministischen TM in durch $O(2^{p(n)})$ beschränkter Zeit entschieden werden können .p(n) ist dabei ein beliebiges Polynom von der Eingabelänge *n*.

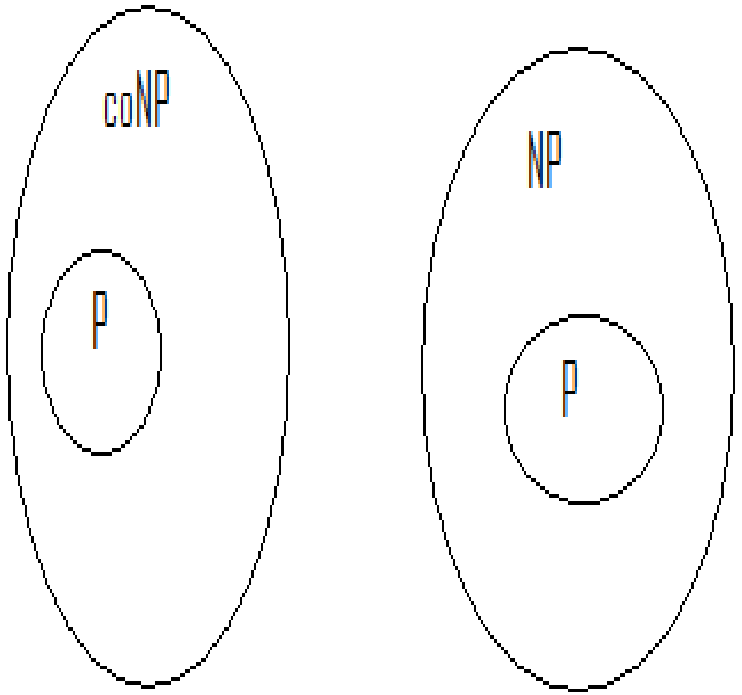
Komplexitätsklasse **NEXPTIME** (**NEXP**) ist die Komplexitätsklasse der Entscheidungsprobleme, die von einer nichtdeterministischen TM in durch $O(2^{p(n)})$ beschränkter Zeit akzeptiert werden können. p(n) ist dabei ein beliebiges Polynom von der Eingabelänge *n*.

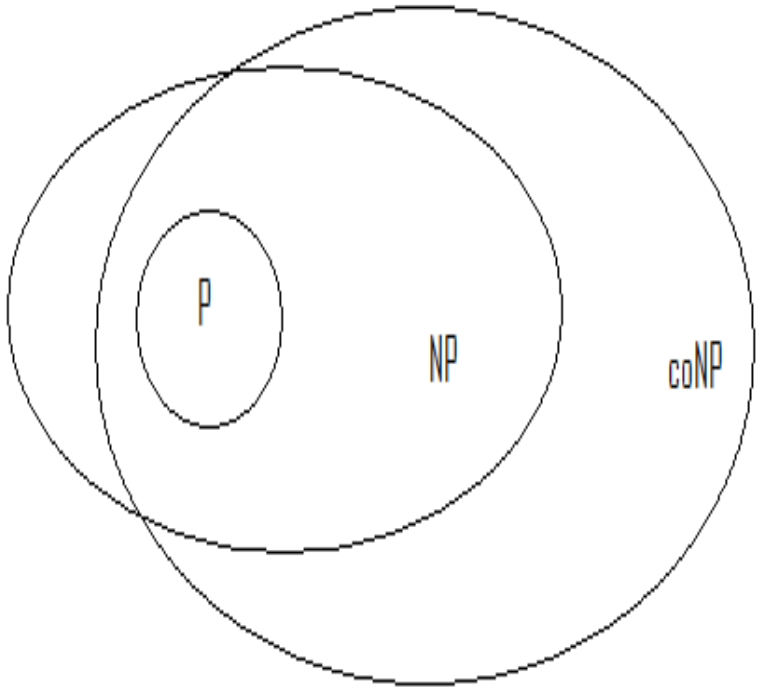
Was wissen wir

P Teilmenge von NP

P Teilmenge von coNP

Also NP und CoNP sind nicht disjunkt





Die Große 1 Mio Frage und andere

Fragen

1 MIO IST P=NP

Ist NP=coNP

Antwort wahrscheinlich nicht

Wieso?

es ist eine von dem Möglichkeiten und keiner

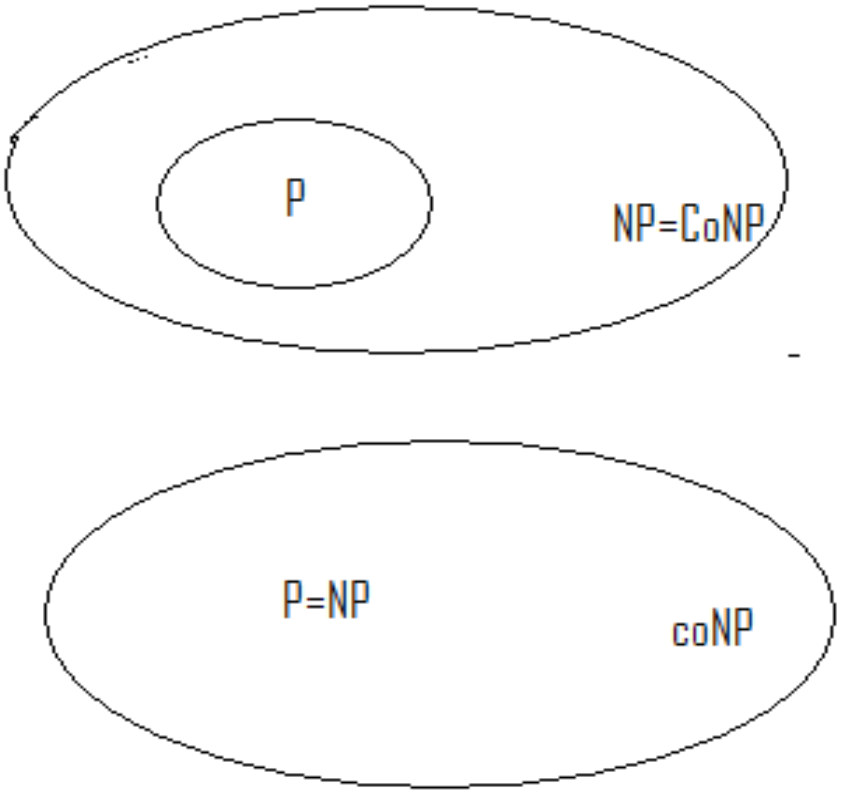
hat dem Gegenteil bewiesen,

Die Intuition sagt so

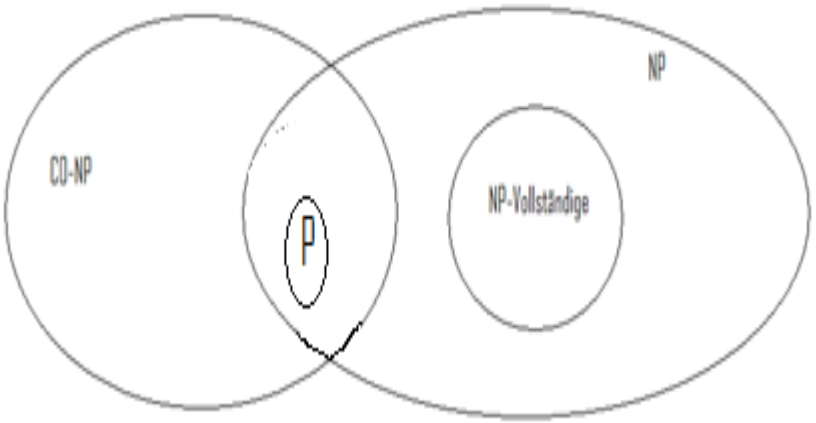
Diese Begründung ist mathematisch nicht haltbar

Aber P=NP => NP = coNP

DA P unter Komplement abgeschlossen ist



Annahme P!= NP



Wenn *A* NP-vollständig ist, dann ist A in co-NP genau dann, wenn

co-Np=NP

Bis 2002 wurde vermutet das Primalität sich zwischen NP und co-NP Befindet, bis es gezeigt würde dass, ob eine Zahl prim ist oder nicht man in O((logn)^12) herausfinden kann

Der Beweis wurde von Manindra Agrawal , Neeraj Kayalund Nitin Saxena Erbracht

Hypothesen

P=NP ?

Konsequenzen :

Mathematiker sind arbeitslos und Informatiker haben nicht so viel zu tun

Approximationsalgorithmen sind Geschichte

Perfektes AI, Mächtige Computerprograme

Dein Geld auf deinem Konto ist nicht sicher, alles was „digital“ ist lässt sich „Knacken“

Kryptographen hoffen dass P!=NP

EXP!= NEXT => P!=NP

Andere weg um die 1 Mio $ zu bekommen.

2. Polynomzeit Reduktion

1. L1, L2 ∈ E ^* *seien Sprachen.*
   L1 heißt in polynomieller Zeit auf L2 reduzierbar (L1 ≤P L2)
   genau dann wenn eine in polynomieller Zeit berechenbare Funktion

F : E^* → E^* existiert, mit w ∈ L1 genau dann wenn f(w) ∈ L2.

2. Polynomzeit Reduktion ist Transitiv

Sein g zwei Funktionen die im n^d und n^c wachsen dann wächst g(f(w)) in n^(c*d)

L2 ∈ NP und L1 ≤P L2, dann ist auch L1 ∈ NP.

Wenn man weist wie es geht, dann ist es einfach

PARTITION<pSUBSET-SUM

Subset -Sum wurde schon vorgestellt jetzt kommt Partition

Partition

Geg: Natürliche Zahlen a1,……,ak k gehört zu N

Frage :kann ich die Zahlen in zwei mengen Teilen so dass, deren summe Gleich ist

Existiert ein Algorithmus in Polynomieller Laufzeit für SUBSETSUM, so existiert auch einer für PARTITION.

Richtige Richtung beachten

Denn: Sei $a_1, ..., a_n$ eine Eingabe für PARTITION. $a_1, ..., a_n$ hat eine positive Antwort $\Leftrightarrow$ $a_1, ..., a_n$,b hat eine positive Antwort bei SUBSET-SUM, wobei $b = \frac{1}{2}\sum_{i=1}^{n} a_i$.

Jede Eingabe w für PARTITION ist leicht zu übersetzen in eine Eingabe w' für SUBSET-SUM, so dass w eine positive Antwort liefert gdw. w' eine positive Antwort liefert.

Schreibweise: PARTITION $\leq_p$ SUBSET-SUM.

Partition <= Bin Packing

Bin Packing

Geg.: k Behälter der Größe b, n Objekte mit der Größe/Gewicht <= b

Frage kann man diese Objekte in die k Behälter einsetzten so dass sie nicht überlaufen?

Wenn man weis ist es einfach

Vorschläge ????

Partition <p Bin Packing

Super Einfache Lösung:

Nehme 2 Behälter mit der Größe ein halb von der Summe der Größe aller Objekte

Eingabe Partition

(a1,……..,an)

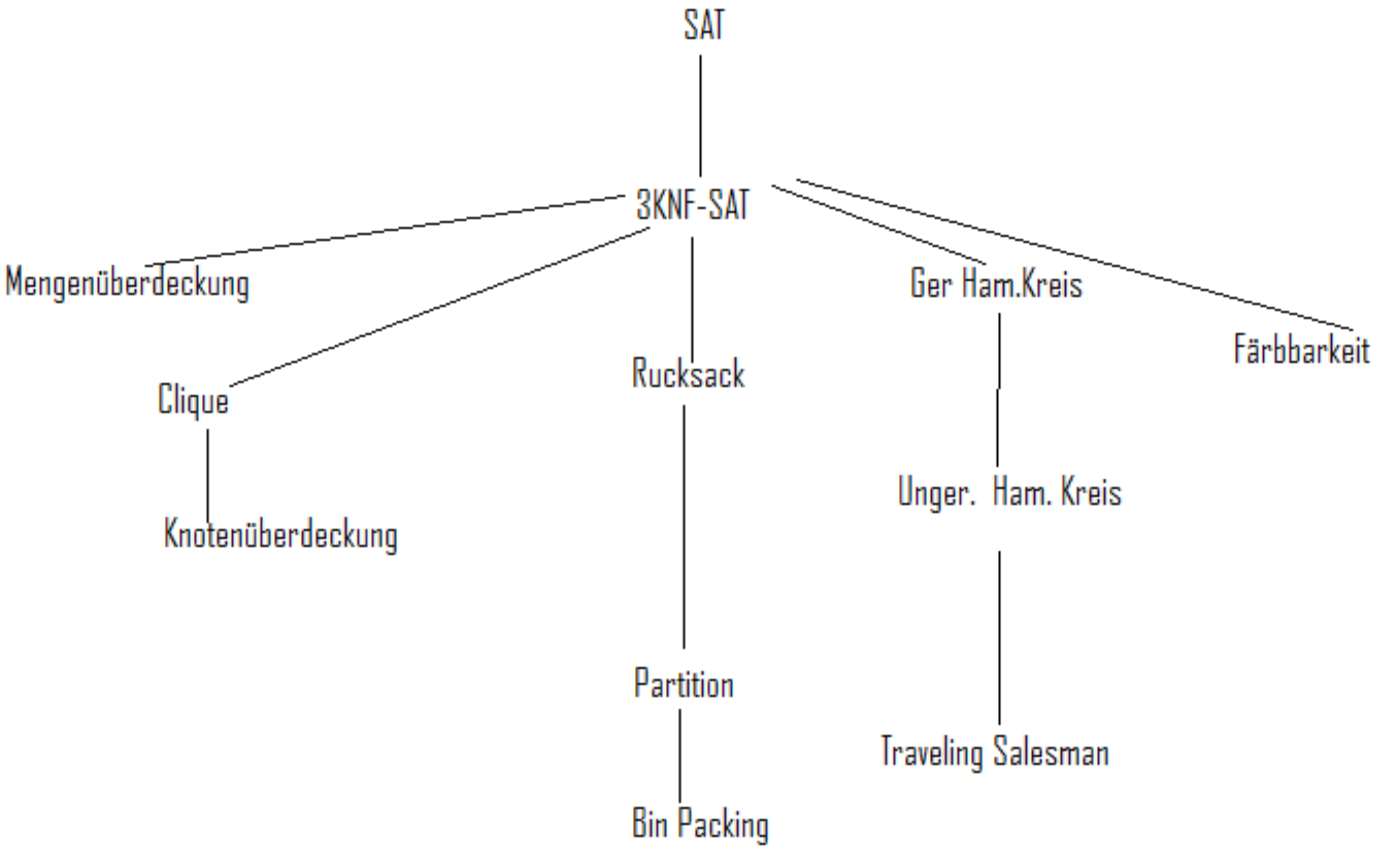
Eingabe Bin Packing

Objekte (a1,……,an)

K=2

B=1/2 Summ(a1,…….,an)

Reduktionsnetz



3. Geschichte und NP-Vollständigkeit

1. 1971 Satz von Cook und Lewin
   1. Der Große Durchbruch
      Es existiert eine Teilmenge der Klasse NP , auf die sich alle Probleme aus NP polynomiell reduzieren lassen.

   2. Satz SAT ist NP-vollständig(Lewin hat das auch unabhängig gezeigt) .

2.1 Richard M. Karp erweiterte 1972 auf diese Weise NPC um 21 weitere Probleme, bis heute hat man mehrere hundert reduziert.

Beim meisten von diesen Problemen ist die Verknüpfung zu einer Turing maschine nicht mehr zu erkennen.

NP-schwer, *NP-vollständig*

Falls L1 NP-schwer ist und L1< p L2, dann ist auch L2
NP-schwer.

Ein Problem L heißt NP-vollständig, genau dann wenn L
NP-schwer ist und L ∈ NP.

Um zu beweisen, dass ein Problem NP-vollständig ist, „genügt" es zu zeigen:

1. ER liegt in NP, d.h. es gibt einen Zeugen und dieser kann in polynomieller Zeit verifiziert werden.

2. Er ist NP-schwer, d.h. wir können ein bereits bekanntes Problem aus NP auf Ihn reduzieren.

Partition ist NP-vollständig

1.Nach dem Reduktionsnetz müssen wir wissen dass, Subset-Sum NP Vollständig ist, sonst hätte der beweis keine Aussagenkraft

2.Annahme wir wissen Subset-Sum ist NP-vollständig

3.Beweis von NP-Vollständigkeit

1. Partition gehört zu NP(einfache Teil)

Eingabe für Partition (a1,a2,a3,a4,……..,an)

Teile in 2 Mengen

(a1,a2,…….,a(n/2)) und (a(n/2)+1,…,an)

Man kann in Polynomiallzeit (sogar in linearer) nachprüfen ob der Zeuge dass erfühlt.

Schwierige Teil

2. Wir suchen jetzt die Überführungsfunktion

Wir müssen in Polynomieller Zeit die Eingabe für SubsetSum in Eingabe für Partition Umwandeln

Und müssen zeigen

Eingabe für SubsetSum sagt 1 oder ja

$\Leftrightarrow$

Eingabe für Partition sagt 1oder ja

Umwandlung

Eingabe SubsetSum => Eingabe Partition

S=(a1,……..,an,b)=>(a1,……..,an,x,y)
Wobei
S‘=Summ(Teilmenge von S)=b
A=Summ von allen ai
X=2*A-b und y=A+b

Geht in Polynomieller Zeit ???? Ja

X und Y können nicht in derselben Partition Liegen denn
X+Y=2*A-b+A+b=3A und die Summe für Ganze Partition ist 4A
Also hat jede Partition den Wert 2A => R1 = Vereinigung X mit S‘
R2= Vereinigung Y mit(S/S‘)

⇔

=>Eingabe für SubsetSum Liefert Lösung => Eingabe für Partition Liefert Lösung

S‘ ist Die Lösung für SubsetSum = b

=>

X+S‘=2A-b+b=2A

Y+Summ(S/S‘)=A+b+(A-b)=2A

<=

Eingabe für Partition Liefert Lösung => Eingabe für SubsetSum Liefert Lösung

R1 = Vereinigung X mit S‘, X = 2A-b

R2= Vereinigung Y mit(S/S‘), Y=A+b

=> S‘ ist die Lösung von Subset Sum

Cook-Lewin Theorem

1. SAT ist NP-Vollständig
    1. Beim dem Beweis sieht man noch die Verbindung zum TM

2. 3SAT ist NP COMPLETE
    1. Polynomzeit Reduktion auf SAT

    2. Nachdem man dass gezeigt hat , wurde es „einfacher“ zu bestimmen ob die Probleme in die Klasse NP Gehören

Extrem wichtig weil

Woher wissen wir das SAT ist NP-Vollständig ?

1.Problem (Sprache) L heißt NP-schwer (NP-hard) genau dann, wenn L‘<p L für alle L‘ die zu NP gehören.

2.Problem (Sprache) L heißt NP-vollständig genau dann, wenn es in NP liegt und NP-schwer ist.

Wir haben bisher benutzt das

„Falls L1 NP-schwer ist und L1< p L2, dann ist auch L2 NP-schwer.“

Woher kommt der ERSTE NP-Schwere Problem ?

Grundidee

1. Es muss gezeigt werden wie man aus einer beliebigen Sprache L in NP in Polynomialzeit eine Reduktion auf SAT machen kann.

2. Es wird mit Hilfe von einer nichtdeterministische Turingmaschine gemacht

3.Grundidee der Reduktion ist nun, die Aussage „Die Berechnung der Maschine M bei Eingabe *x* ergibt, dass *x* zur Sprache *L* gehört" in einer aussagenlogischen Formel auszudrücken

4. In dieser Formel müssen sich also eine Beschreibung der Maschine M, eine Beschreibung der Eingabe *x* sowie die „Rechenregeln", nach denen eine nichtdeterministische Turingmaschine arbeitet, sein.

5. Das alles muss in Polynomialzeit staatfinden.

Cook-Lewin Theorem

Sie haben gezeigt mit Hilfe von

„Problem (Sprache) L heißt NP-schwer (NP-hard) genau dann, wenn L‘<p L für alle L‘ die zu NP gehören.“

Und Turingmaschinen

dass:

SAT ist NP -Hard

4. Lösungsansetze für NP-Vollständige Probleme

Reales Problem Maschinenbelegung, TSP

1.Man schreibt tatsächlich einen optimalen Algorithmus , aber man weiß dass der Computer ist stark genug , um die Datenmenge in einer Zeitspanne die für uns ok ist zu schaffen

2. Benutze einen heuristischen Algorithmus und sucht nicht nach einer optimalen Lösung sondern nur nach einer Lösung die für die konkrete Aufgabe *gut genug* ist.

3.Man wählt einen approximativen Algorithmus der nicht die perfekte Lösung aber eine nahezu perfekte Lösung bietet.

3.1 Hilft nicht immer

Für kein a > 1 existiert ein a-approximativer Algorithmus polynomieller Laufzeit

für TSP, es sei denn P = NP.

5.Zusammenfassung

Was Möchte ich was, Hoffe ich, habt Ihr aus diesen Referat Mitgenommen

1. Die Klasse der NP-Vollständigen Problemen ist Jung und voller Geheimnisse
2. Sollte jemand es schaffen die 1 Mio $ Aufgabe zu lösen würde sich unsere Welt verändern weil , es keine Abstrakten Probleme sind
3. Das wichtigste Werkzeug ist hier die PReduktion, deswegen ist der Satz von Cook und Lewin so wichtig dass SAT NP-vollständig ist (Oder hat jemand Lust zu zeigen dass jede Sprache die zu NP gehört sich mit Hilfe von TM auf dieses Problem reduzieren lässt)
4. Die TM wahr sehr oft in Hintergrund, das ist auch so mit

Np-Vollständigen Problemen, manche sehen ob sie nichts mit TM zu tun hätten, aber sie ist immer in Hintergrund.

Vielen Dank, ich hoffe ihr habt etwas neues und interessantes erfahren.