Näherungsalgorithmen

(Approximationsalgorithmen)

WiSe 2010/11 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Näherungsalgorithmen Gesamtübersicht

- Organisatorisches
- Einführung / Motivation
- Grundtechniken für Näherungsalgorithmen
- Approximationsklassen (Approximationstheorie)

Näherungsalgorithmen Gesamtübersicht

- Organisatorisches
- Einführung / Motivation
- Grundtechniken für Näherungsalgorithmen
- Approximationsklassen (Approximationstheorie)

Grundtechniken

- Greedy-Verfahren
- Partitionsheuristiken
- Lokale Suche
- Lineares Programmieren
- Dynamisches Programmieren

Dynamisches Programmieren

- exaktes Verfahren
- Buchführung vermeidet Mehrfachberechnung (Tabelle)
- Tabelle wird “bottom-up” aufgebaut
- Beispiel: CYK-Algorithmus zum
  Parsen von kontextfreien Sprachen, bekannt aus
  Theoretischer Informatik / Compilerbau

Erinnerung: **Maximum Knapsack (Rucksackproblem)**

I:	endliche Menge X	bzw. „Liste“ $X = \{x_1, \ldots, x_n\}$
	Profitfunktion $p : X \to \mathbb{N}$	bzw. „Liste“ $\{p_1, \ldots, p_n\}$
	Größenfunktion $a : X \to \mathbb{N}$	bzw. „Liste“ $\{a_1, \ldots a_n\}$
	Fassungsvermögen $b \in \mathbb{N}$ des Rucksacks	
	[O.E.: $\forall 1 \leq i \leq n : a_i \leq b$ im Folgenden]	
S:	$\{Y \subseteq X \mid s(Y) = \sum_{x_i \in Y} a_i \leq b\}$	
m:	$p(Y) = \sum_{x_i \in Y} p_i$	
opt.:	max	

Mitteilung: Knapsack ist NP-vollständig.

Ein exakter Algorithmus für das Rucksackproblem

Für das dynamische Programmieren betrachten wir für jedes $1 \leq k \leq n$ und jedes $0 \leq p \leq \sum_{i=1}^{n} p_i$ das Problem, eine Teilmenge von $\{x_1, \ldots, x_k\}$ zu finden mit Gesamtprofit p und Gesamtgröße höchstens b.

Es bezeichnen $M^*(k,p) \subseteq \{x_1, \ldots, x_k\}$ eine optimale Lösung dieses Problems und $S^*(k,p)$ die entsprechende Gesamtgröße.

Sonderfall: $S^*(k,p) := 1 + \sum_{i=1}^{n} a_i$ ("unendlich"), falls $M^*(k,p)$ undefiniert.

Damit gilt

$$M^*(1,p) = \begin{cases} \emptyset, & p = 0 \\ \{x_1\}, & p = p_1 \\ \text{undefiniert}, & \text{sonst} \end{cases}$$

Für $k \geq 2$ gilt der folgende (rekursive) Zusammenhang:

$$M^*(k,p) = \begin{cases} M^*(k-1,p-p_k) \cup \{x_k\} & \text{falls gilt: (i) } p_k \leq p, \\ & \text{(ii) } M^*(k-1,p-p_k) \text{ ist definiert,} \\ & \text{(iii) } S^*(k-1,p) \\ & \qquad \geq S^*(k-1,p-p_k) + a_k \\ & \text{(iv) } S^*(k-1,p-p_k) + a_k \leq b \\ M^*(k-1,p) & \text{sonst} \end{cases}$$

KnapsackDynProg $(X = \{x_1, \ldots, x_n\}, \{p_1, \ldots, p_n\}, \{a_1, \ldots, a_n\}, b)$

1. Für $p := 0$ bis $\sum_{i=1}^{n} p_i$ tue

   1a. Setze $M^*(1, p) :=$ undefiniert

   1b. Setze $S^*(1, p) := 1 + \sum_{i=1}^{n} a_i$

2a. $M^*(1, 0) := \emptyset; S^*(1, 0) := 0;$

2b. $M^*(1, p_1) := \{x_1\}; S^*(1, p_1) := a_1;$

3. Hauptprogramm...

4. Sei p^* das maximale p, für das $M^*(n, p)$ definiert ist.

   (Beachte: $M^*(n, 0) = \emptyset$ ist stets definiert!)

   Liefere $M^*(n, p^*)$.

Hauptprogramm

3. Für $n := 2$ bis n tue:
   Für $p := 0$ bis $\sum_{i=1}^{n} p_i$ tue:
   3a. Falls $p_k \leq p$ und $M^*(k-1, p-p_k)$ ist definiert
   und $S^*(k-1, p-p_k) + a_k \leq \min\{b, S^*(k-1, p)\}$
   dann
   3a1. Setze $M^*(k, p) := M^*(k-1, p-p_k) \cup \{x_k\}$
   3a2. Setze $S^*(k, p) := S^*(k-1, p-p_k) + a_k$
   3b. sonst
   3b1. Setze $M^*(k, p) := M^*(k-1, p)$;
   3b2. Setze $S^*(k, p) := S^*(k-1, p)$

Satz 1: KnapsackDynProg findet stets eine optimale Lösung, und zwar in Zeit

$$O(n \cdot \sum_{i=1}^{n} p_i).$$

Hinweis: Diese Laufzeit ist *nicht* polynomiell, da die p_i als binär codiert angenommen werden. Solche so genannten *pseudopolynomialen* Algorithmen untersuchen wir näher im Kapitel über Approximationsschemata.

Ein Näherungsverfahren für das Rucksackproblem

KnapsackFPTAS $(X = \{x_1, \ldots, x_n\}, \{p_1, \ldots, p_n\}, \{a_1, \ldots a_n\}, b, r)$
($r \in \mathbb{Q}, r > 1$ ist der angestrebte Approximations-Leistungs-Faktor)

1. $p_{max} := \max\{p_i \mid 1 \leq i \leq n\}$;
2. $t := \left\lfloor \log_2(\frac{r-1}{r} \cdot \frac{p_{max}}{n}) \right\rfloor$;
3. $x' :=$ Instanz mit Profiten $p'_i := \left\lfloor p_i / 2^t \right\rfloor$;
4. Liefere KnapsackDynProg(x').

Satz 2: Ggb: Instanz x von MaximumKnapsack (mit n Gegenständen) und $r > 1, r \in \mathbb{Q}$. KnapsackFPTAS(x) liefert eine Lösung in Zeit $O(r \cdot n^3/(r-1))$, deren Maß $m_{AS}(x,r)$ der Ungleichung $m^*(x)/m_{AS}(x,r) \leq r$ genügt.

Beweis: Es bezeichne $Y(x,r)$ die von KnapsackFPTAS gelieferte Lösung und $Y^*(x)$ eine optimale Lösung von Instanz x (mit Maß $m^*(x)$).

$$
\begin{array}{lll}
m_{AS}(x,r) = \sum_{x_j \in Y(x,r)} p_j & \geq & \sum_{x_j \in Y(x,r)} 2^t \cdot \lfloor p_j/2^t \rfloor \\
\text{(nach Def. der } \lfloor\cdot\rfloor\text{-Funktion)} & & \\
 & \geq & \sum_{x_j \in Y^*(x)} 2^t \cdot \lfloor p_j/2^t \rfloor \\
\text{(denn } \sum_{x_j \in Y(x,r)} \lfloor p_j/2^t \rfloor \text{ ist} & & \\
\text{der optimale Wert des} & & \\
\text{„skalierten Problems“)} & & \\
 & \geq & \sum_{x_j \in Y^*(x)} (p_j - 2^t) \\
\text{(nach Def. der } \lfloor\cdot\rfloor\text{-Funktion)} & & \\
 & \geq & \left(\sum_{x_j \in Y^*(x)} p_j\right) - 2^t \cdot |Y^*(x)| \\
 & = & m^*(x) - 2^t \cdot |Y^*(x)|
\end{array}
$$

$$\left.\begin{array}{l} \rightsquigarrow m^*(x) - m_{AS}(x,r) \leq 2^t \cdot |Y^*(x)| \leq 2^t \cdot n \\ \text{Offenbar gilt ferner: } n \cdot p_{max} \geq m^*(x) \geq p_{max} \end{array}\right\} \rightsquigarrow$$

(Hierbei Annahme, dass alle Gegenstände der Instanz in den leeren Rucksack passen.)

$$\rightsquigarrow \frac{m^*(x) - m_{AS}(x,r)}{m^*(x)} \leq \frac{n \cdot 2^t}{p_{max}}$$

$$\begin{array}{rcl} \Rightarrow m^*(x) & \leq & \frac{p_{max}}{p_{max} - n \cdot 2^t} m_{AS}(x,r) \\ & \leq & \frac{p_{max}}{p_{max} - n \cdot \frac{r-1}{r} \cdot \frac{p_{max}}{n}} m_{AS}(x,r) \\ & = & r \cdot m_{AS}(x,r) \end{array}$$

Laufzeitabschätzung:

$$\begin{array}{rcl} O\left(n \cdot \sum_{i=1}^{n} p_i'\right) & = & O\left(n \cdot \sum_{i=1}^{n} \frac{p_i}{2^t}\right) \\ & \leq & O\left(n^2 \frac{p_{max}}{2^t}\right) \\ & \leq & O\left(n^2 \frac{p_{max}}{\frac{r-1}{r} \frac{p_{max}}{n}}\right) \\ & = & O\left(n^3 \cdot \frac{r}{r-1}\right) \end{array} \qquad \square$$

Was tun bei konkreten Problemen?

- Probiere bisher kennengelernte Strategien.

- Gibt es bekannte Polynomzeitalgorithmen
  für “ähnliche” Probleme?

Billigste Erweiterung zu starkem Zusammenhang

I : $G = (V, A)$ gerichteter, vollständiger Graph
mit Kantenkostenfunktion $c : A \to \mathbb{N}$,
eine vorliegende Kantenmenge A_0

S : $\{A_1 \subseteq A \setminus A_0 \mid (V, A_0 \cup A_1)$ ist stark zusammenhängend$\}$
Ein gerichteter Graph heißt *stark zusammenhängend*, wenn es zwischen je zwei Knoten einen (gerichteten) Pfad gibt.

m : $c(A_1)$

opt : min.

Hinweis: Das Problem ist NP-vollständig.

Ein mögliche Idee: (siehe Frederickson / Jájá)

Sie haben den Algorithmus von Dijkstra zur Bestimmung eines billigsten Weges zwischen zwei Knoten in einem kantengewichteten gerichteten Graphen kennengelernt. Diesen Algorithmus können wir verwenden, um in einem vollständigen gerichteten Graphen mit Kantengewichtsfunktion einen von der Wurzel zu den Blättern gerichteten minimalen Spannbaum in Polynomzeit zu berechnen. Ist die Wurzel zunächst unbestimmt, so kann man durch eine äußere Schleife über alle Knoten den absolut billigsten Spannbaum finden.

Kann man nun erzwingen, dass (durch geeignete Modifikation der Kantengewichte) zwei gerichtete Bäume T_1 und T_2 (aufgefasst als Kantenmenge) mit gleicher Wurzel r, aber unterschiedlicher Pfeilrichtung gefunden werden, so gibt es in dem Graphen, der nur aus den so gewichteten Baumkanten besteht, zwischen je zwei Knoten x und y jedenfalls einen Pfad von x nach y, der über r führt.

So ließe sich eine 2-Approximation gewinnen, denn eine optimale Erweiterung A^* müsste (in gewissem Sinne) auch die Bäume T_1 und T_2 enthalten.

STC$(G = (V, A), c, A_0)$

1. Definiere neue Kantengewichtsfunktion c_1 auf gerichtetem vollständigen Graph G_1 mit Knotenmenge V wie folgt:

$$c_1((u,v)) = \begin{cases} 0, & \text{falls } (v,u) \in A_0 \\ c((v,u)), & \text{falls } (v,u) \notin A_0 \end{cases}$$

2. Finde absolut minimalen Spannbaum T_1 in G_1 (aufgefasst als Kantenmenge).
   Sei r die Wurzel von T_1. Setze $A_1 = \{(u,v) \mid (v,u) \in T_1\}$. (Kantenumdrehen)
3. Definiere Kantengewichtsfunktion c_2 auf gerichtetem vollständigen Graph G_2 mit Knotenmenge V wie folgt:

$$c_2((u,v)) = \begin{cases} \infty, & v = r \\ 0, & (u,v) \in (A_0 \cup A_1) \wedge v \neq r \\ c((u,v)), & \text{sonst} \end{cases}$$

4. Finde minimalen Spannbaum T_2 in G_2 (aufgefasst als Kantenmenge).
5. Liefere $(A_1 \cup T_2) \setminus A_0$.

Satz: STC ist eine 2-Approximation für das Problem, eine billigste Erweiterung zu starkem Zusammenhang zu finden.

Beweis: Durch die Definition von c_2 wird erzwungen, dass r (ebenfalls) die Wurzel von T_2 ist, sodass tatsächlich eine Erweiterung geliefert wird, die starken Zusammenhang garantiert. In Schritt 2 wird eine Kantenmenge T_1 gefunden, sodass jeder Knoten mit der Wurzel r durch einen Pfad verbunden ist. In einer optimalen Erweiterung A^* gibt es natürlich auch einen solchen Pfad. Wegen der Minimalität von T_1 ist dieser Pfad in A^* sicher nicht billiger.
Da T_1 minimal ist, gilt $c(A_1 \setminus A_0) = c_1(T_1) \leq c(A^*)$.
Entsprechend sieht man: $c(T_2 \setminus (A_0 \cup A_1)) = c_2(T_2) \leq c(A^*)$.
$\leadsto c((A_1 \cup T_2)) \setminus A_0) = c(A_1 \setminus A_0) + c(T_2 \setminus (A_0 \cup A_1)) \leq 2c(A^*)$. □

Literatur:

- K. P. Eswaran and R. E. Tarjan: Augmentation problems. SIAM J. Computing 5(1976), 653–665.

- G. N. Frederickson and J. Jájá: Approximation algorithms for several graph augmentation problems. SIAM J. Computing 10(1981), 270–283.

- H. Nagamochi and T. Ibaraki: An approximation for finding the smallest 2-edge connected subgraph containing a specified spanning tree. In: T. Asano u.a. (Herausgeber), Proc. COCOON'99, LNCS Band 1627, pp. 221–230. Springer, 1999.

Approximationsklassen

- $\leadsto$ (strukturelle) Komplexitätstheorie
- Phänomenologie der Komplexitätsklassen:
  Welche Effekte können bei Näherungsalgorithmen auftreten?
- Insbesondere:
  Feinstruktur von NPO

Absolute Approximation

- Ziel: Annäherung bis auf konstanten Summanden
- Natürlich “schöner” als relative Approximation
- Aber: fast “nie” möglich!
  Warum?
  $\leadsto$ Dieses Kapitel!

Absoluter Fehler

Gegeben sei ein Optimierungsproblem $\mathcal{P}$. Dann ist für jede Instanz x von $\mathcal{P}$ und jede zulässige Lösung y von x der *absolute Fehler* von y bezüglich x definiert als

$$D(x, y) := |m^*(x) - m(x, y)|$$

Dabei ist wiederum $m^*(x)$ der Wert einer optimalen Lösung der Instanz x und $m(x, y)$ sei der Wert der Lösung y.
Ist $\mathcal{A}$ ein Approximationsalgorithmus (Näherungsalgorithmus) für $\mathcal{P}$, d.h. gilt

$$\forall x \in I_{\mathcal{P}} : \mathcal{A}(x) \in S_{\mathcal{P}}(x),$$

so heißt $\mathcal{A}$ *absoluter Approximationsalgorithmus*, wenn es eine Konstante K gibt, so dass für alle $x \in I_{\mathcal{P}} : D(x, \mathcal{A}(x)) \leq K$ gilt.

Eulerscher Polyedersatz: Für planare Graphen gilt:

$$f - m + n = 2.$$

$f =$ # Flächen des (als eingebettet gedachten) Graphen.
$m =$ # Kanten
$n =$ # Knoten
Daraus folgt: $\boxed{m \leq 3n - 6}$.
Bezeichnet nämlich ℓ_i die Anzahl der an Fläche F_i angrenzenden Kanten, so gilt, da $\forall i(\ell_i \geq 3)$:

$$2m = \sum_{i=1}^{f} \ell_i \geq 3f = 3m - 3n + 6.$$

Daher gilt:
$(*)$ Jeder planare Graph enthält wenigstens einen Knoten von Grad 5.
Sonst wäre $2m = \sum_{i=1}^{n} d(v_i) \geq 6n$, also $m \geq 3n$, im Widerspruch zum soeben Hergeleiteten.

Das Färben von planaren Graphen

Aus früheren Überlegungen folgern wir:

Satz 1: Betrachte die Knotenmenge des Eingabegraphen $G = (V, E)$ als geordnet: $(v_1, \ldots, v_n)$. (Reihenfolge, in der die Partitionsheuristik arbeitet.)
Es sei $G_i = G(\{v_1, \ldots, v_i\})$; $d_i(v)$ bezeichne den Grad von v in G_i.
Die Partitionsheuristik benötigt dann höchstens

$$1 + \max_{1 \leq i \leq n} d_i(v_i)$$

viele Farben zum Färben von G_i. □

Wir wollen jetzt die Ordnung der Knoten genauer festlegen:
v_n sein ein Knoten minimalen Grades in $G = (V, E)$.
Induktiv gehen wir davon aus, die Reihenfolge $(v_{i+1}, \ldots, v_n)$ sei bereits festgelegt. Dann sei v_i ein Knoten minimalen Grades in $G(V \setminus \{v_{i+1}, \ldots, v_n\})$.

Satz 2: Ist G ein planarer Graph, so benötigt die Partitionsheuristik mit der soeben definierten Knotenordnung höchstens 6 Farben zum Färben von G.

Beweis: Wegen Satz 1 reicht es zu zeigen, dass

$$\max\{d_i(v_i) \mid 1 \leq i \leq n\} \leq 5.$$

Aus $(*)$ ergibt sich $\forall i : d_i(v_i) \leq 5$, da stets ein kleinstgradiger Knoten v_i in $G_i = G(\{v_1, \ldots, v_i\})$ ausgewählt wird und alle G_i planar sind, weil $G = G_n$ planar ist und die planaren Graphen unter der Operation „Fortlassen von Knoten und inzidenten Kanten“ abgeschlossen sind. □

Folgerung: Das Satz 2 zu Grunde liegende Verfahren ist ein absoluter Approximationsalgorithmus zum Färben planarer Graphen. Die absolute Fehlerschranke von 5 lässt sich noch auf 3 reduzieren, da 2-färbbare Graphen mit einer Greedy-Strategie optimal gefärbt werden können.

Typischer ist aber die folgende Situation:

Satz 3: Unter der Annahme P $\neq$ NP gilt:
Es gibt keinen polynomiellen absoluten Approximationsalgorithmus für MaximumKnapsack.

Beweis: Es sei X eine Menge Gegenstände sowie zugehörigen Profiten p_i und Größen a_i und b die Rucksackkapazität. Gäbe es einen polynomiellen absoluten Approximationsalgorithmus für MaximumKnapsack mit Fehlerschranke K, so könnten wir das Rucksackproblem in Polynomialzeit wie folgt lösen:

Wir generieren eine neue Instanz, indem wir alle Profite mit $(K + 1)$ multiplizieren. Die Menge der zulässigen Lösungen wird dadurch nicht verändert. Der Wert jeder Lösung ist nun aber ein Vielfaches von $K+1$, was zur Folge hat, dass jede Lösung mit absolutem Fehler K bereits optimal ist. $\square$

Relative Approximation; die Klasse APX

- unser bisheriger “Standard”
- im Folgenden: Grundbegriffe und Beispiele
- Gap-Technik

Ist $\mathcal{P}$ ein Optimierungsproblem, so ist —für jede Instanz x von $\mathcal{P}$ und für jede zulässige Lösung y von x— der *relative Fehler* von y bezüglich x definiert als

$$E(x,y) := \frac{|m^*(x) - m(x,y)|}{\max\{m^*(x), m(x,y)\}} = \frac{D(x,y)}{\max\{m^*(x), m(x,y)\}}.$$

Ist $\mathcal{A}$ ein Approximationsalgorithmus für $\mathcal{P}$, so heißt $\mathcal{A}$ *ϵ-Approximation*, wenn

$$\forall x \in I_{\mathcal{P}} : E(x, \mathcal{A}(x)) \leq \epsilon.$$

Bem.: Der relative Fehler ist gleich Null, wenn die Lösung optimal ist, und er liegt nahe bei Eins, wenn die Lösung schlecht ist.

Ist $\mathcal{P}$ ein Optimierungsproblem, so ist —für jede Instanz x von $\mathcal{P}$ und für jede zulässige Lösung y von x— die *Leistungsgüte* (engl: performance ratio) von y bezüglich x definiert als

$$R(x,y) := \max\left\{\frac{m^*(x)}{m(x,y)}, \frac{m(x,y)}{m^*(x)}\right\}.$$

Ist $\mathcal{A}$ ein Approximationsalgorithmus für $\mathcal{P}$, so heißt $\mathcal{A}$ *r-Approximation*, wenn

$$\forall x \in I_{\mathcal{P}} : R(x, \mathcal{A}(x)) \leq r.$$

Bem.: Die Leistungsgüte ist gleich Eins, wenn die Lösung optimal ist, und sie ist sehr groß, wenn die Lösung schlecht ist. Offenbar gilt:

$$E(x,y) = 1 - \frac{1}{R(x,y)}.$$

Relativer Fehler versus Leistungsgüte

Offenbar gilt: $$E(x,y) = 1 - \frac{1}{R(x,y)}.$$

Ist nämlich $\mathcal{P}$ ein Maximierungsproblem, so ist:

$$1 - \frac{1}{R(x,y)} = 1 - \frac{1}{\frac{m^*(x)}{m(x,y)}} = 1 - \frac{m(x,y)}{m^*(x)} = \frac{m^*(x) - m(x,y)}{m^*(x)}.$$

Ist $\mathcal{P}$ ein Minimierungsproblem, so gilt:

$$1 - \frac{1}{R(x,y)} = 1 - \frac{m^*(x)}{m(x,y)} = \frac{m(x,y) - m^*(x)}{m(x,y)}.$$

Die (diversen) Näherungsalgorithmen für das Knotenüberdeckungsproblem sind sowohl $\frac{1}{2}$-Approximation (im Sinne des relativen Fehlers) als auch 2-Approximation (im Sinne der Leistungsgüte).

Die Klasse APX

$\mathcal{P}$ heißt ϵ*-approximierbar* (bzw. r*-approximierbar*), wenn es eine Polynomzeit-ϵ-Approximation (bzw. r-Approximation) für $\mathcal{P}$ gibt.

APX ist die Klasse aller NPO-Probleme, die r-approximierbar für ein $\mathrm{r} \geq 1$ sind.

Asymptotische Approximation

Algorithmen $\mathcal{A}$, die (im Minimierungsfall)

$$m_{\mathcal{A}}(x, y) \leq r \cdot m^*(x) + k$$

erfüllen, sind „nur“ $(r + k)$-Approximationen (im Sinne von unserer Definition), werden aber oft als r-Approximationen in der Literatur angesprochen.
Wir werden solche Algorithmen *asymptotische r-Approximationen* nennen.
Im Zusammenhang mit Approximationsschemata ist dieser Unterschied bedeutender, siehe unten !

Ein (weiteres) Beispiel aus APX: MAXSAT

I : Menge C von Klauseln über einer Menge V von Variablen
(Eine Klausel ist eine Disjunktion von Literalen.
Ein Literal ist eine Variable oder ihre Negation.
Formal ist C als Menge von Mengen von Literalen aufzufassen.)

S : Menge der Wertzuweisungen $f : V \to \{\text{true}, \text{false}\}$

m : $|\{c \in C \mid f(c) = \text{true}\}|$

opt : max

Beispiel: Die KNF-Formel $(x \vee \bar{y}) \wedge (\bar{x} \vee y)$ würden wir als $C = \{\{x, \bar{y}\}, \{\bar{x}, y\}\}$ schreiben.

GreedySAT(C, V)

1. Für alle $v \in V$ setze $f(v) := \text{true}$; (Anfangsbelegung)
2. Wiederhole, solange $C \neq \emptyset$:
   - 2a. Es sei ℓ ein Literal, das in einer maximalen Zahl von Klauseln vorkommt; es sei x die zugehörige Variable.
   - 2b. Bestimme die Menge C_x von Klauseln, in denen x vorkommt.
   - 2c. Bestimme die Menge $C_{\overline{x}}$ von Klauseln, in denen $\overline{x}$ vorkommt.
   - 2d. Ist $\ell = x$, so
     - 2d(i) $C := C \setminus C_x$;
     - 2d(ii) Lösche $\overline{x}$ aus allen Klauseln in $C_{\overline{x}} \setminus C_x$
     - 2d(iii) Lösche alle leeren Klauseln aus C.
   - 2e. sonst:
     - 2e(0) $f(x) := \text{false}$;
     - 2e(i) $C := C \setminus C_{\overline{x}}$;
     - 2e(ii) Lösche x aus allen Klauseln $C_x \setminus C_{\overline{x}}$;
     - 2e(iii) Lösche alle leeren Klauseln aus C.
3. Liefere f zurück.

Satz: GreedySAT ist eine Polynomzeit-2-Approximation für MAXSAT.

Beweis: Per Induktion über die Anzahl der Variablen zeigen wir:
Behauptung: Ist eine Instanz mit c Klauseln gegeben, so liefert GreedySAT stets eine Lösung, die mindestens $c/2$ der Klauseln erfüllt.
Da trivialer Weise c eine obere Schranke für den Wert einer optimalen Lösung ist, folgt so die Behauptung.

IA: Im Fall, dass es nur eine Variable gibt, folgt die Behauptung trivial.
IV: Es werde nun angenommen, die Behauptung sei für $n - 1$ Variablen $(n > 1)$ bewiesen.
IB: Dann gilt die Behauptung auch für n Variablen.

Beweis von IB: Von GreedySAT werde als erstes die Variable x betrachtet.

Es seien c_x bzw. $c_{\overline{x}}$ die Anzahl der Klauseln, in denen x bzw. $\overline{x}$ vorkommen.

O.E. betrachten wir $c_x \geq c_{\overline{x}}$, d.h., GreedySAT wird x auf true setzen.

Nach dieser Zuweisung müssen (nach Schritt 2d(i) bzw. 2d(iii)) noch $\tilde{c} \geq c - c_x - c_{\overline{x}}$ viele Klauseln (mit $n-1$ Variablen) betrachtet wurden.

Nach IV werden daher von GreedySAT wenigstens $\tilde{c}/2 \geq (c - c_x - c_{\overline{x}})/2$ der $\tilde{c}$ vielen Klauseln erfüllt.

Insgesamt werden daher wenigstens $c_x + (c - c_x - c_{\overline{x}})/2 = \frac{c+c_x-c_{\overline{x}}}{2} \geq \frac{c}{2}$ Klauseln der ursprünglichen Klauselmenge erfüllt. $\square$

Nicht alles in APX?!

Das Handelsreisendenproblem MTS

Satz: Wenn das allgemeine Handelsreisendenproblem MTS zu APX gehörte, dann wäre P $=$ NP.

Folgerung: Ist P $\neq$ NP, so ist APX $\neq$ NPO. □

Beweis: Wir beweisen das Resultat duch Reduktion vom Hamiltonschen Kreisproblem.
Ein *Hamiltonscher Kreis* ist dabei eine Permutation $(v_1, \ldots v_n)$ der Knoten eines ungerichteten Graphen $G = (V, E), |V| = n$, sodass $\forall 1 \leq i \leq n \{v_i, v_{i+1}\} \in E$
(der Einfachheit halber sei $v_{n+1} = v_1$).
Für jedes $r \geq 1$ konstruieren wir eine Instanz von MTS derart, dass aus der Existenz einer r-Approximation für MTS folgen würde, dass es einen Polynomzeitalgorithmus zum Hamiltonschen Kreisproblem gäbe. Da das Hamiltonsche Kreisproblem NP-vollständig ist, folgt die Behauptung.

Geben wir uns also ein $r \geq 1$ vor. Ist $G = (V, E)$ eine Hamiltonsche Kreisprobleminstanz, so definieren wir (auf derselben Knotenmenge V) eine Abstandsfunktion d, die zusammen mit V das fragliche MTS-Problem angibt. Setze dazu:

$$d(u, v) := \begin{cases} 1, & \{u, v\} \in E \\ 1 + |V| \cdot r, & \{u, v\} \notin E \end{cases}$$

MTS hat eine (minimale) Lösung vom Wert $|V|$ gdw. G einen Hamiltonschen Kreis enthält.

Eine r-Approximation $\mathcal{A}$ für MTS könnte wie folgt zum Lösen des Hamiltonschen Kreisproblems verwendet werden:

- Liefert $\mathcal{A}$ einen Wert $\leq r \cdot |V|$ zurück, so hatte das ursprüngliche Kreisproblem eine Lösung.
- Liefert $\mathcal{A}$ einen Wert $> r \cdot |V|$ zurück, so war das ursprüngliche Kreisproblem unlösbar.

□