Näherungsalgorithmen

(Approximationsalgorithmen)

WiSe 2012/13 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

29. Januar 2013

Näherungsalgorithmen Gesamtübersicht

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

Approximationstheorie

- Absolute Approximation
- Relative Approximation: die Klasse APX
- Polynomzeit-Approximationsschemata PTAS
- Zwischen APX und NPO
- Zwischen PTAS und APX
- Approximationsklassen und Reduktionen

FPTAS

Störend an den bislang vorgestellten Approximationsschemata:
Die Laufzeit hängt exponentiell von der angestrebten Güte der Approximation ab!
Wir stellen jetzt einen „schöneren“ PTAS-Begriff vor.

Definition Es sei $\mathcal{P} \in \mathrm{NPO}$. Ein Algorithmus $\mathcal{A}$ heißt *volles Polynomzeit-Approximations-Schema* (engl. fully PTAS, kurz FPTAS), falls $\mathcal{A}$ für jedes Paar von Eingaben (x, r), x eine Instanz von $\mathcal{P}$ und $r > 1$, eine r-approximative Lösung zurück liefert in einer Zeit, die polynomiell in $|x|$ und in $\frac{1}{r-1}$ ist.
Die zugehörige Problemklasse nennen wir FPTAS.

Rucksack und FPTAS

Weiter oben haben wir bereits ein (einfaches) Beispiel für ein Problem aus FPTAS kennen gelernt, nämlich das Rucksackproblem.
Tatsächlich gibt es nur verhältnismäßig wenige (natürliche) Probleme in FPTAS; die meisten von ihnen sind Verwandte des Rucksackproblems.
Der Grund scheint darin zu liegen, dass man von einer ganzen Reihe von NPO-Problemen leicht nachweisen kann, dass sie **nicht** zu FPTAS gehören.

Polynomielle Beschränktheit und FPTAS

Ein Optimierungsproblem heißt *polynomiell beschränkt*, wenn es ein Polynom p gibt derart, dass (für jede Instanz $x \in I_{\mathcal{P}}$ und jede zulässige Lösung $y \in S_{\mathcal{P}}(x)$) $m(x, y) \leq p(|x|)$ gilt.

Satz: Gilt P $\neq$ NP, so gehört kein NP-hartes polynomiell beschränktes Optimierungsproblem zu FPTAS.

Beweis: Angenommen, es gibt ein FPTAS $\mathcal{A}$ für das Maximierungsproblem $\mathcal{P}$, dessen Laufzeit durch $q(|x|, \frac{1}{r-1})$ für ein Polynom q beschränkt ist (für jede Instanz x und jedes $r > 1$). Da $\mathcal{P}$ polynomiell beschränkt ist, gibt es ein Polynom p mit $m^*(x) \leq p(|x|)[*]$ für jede Instanz x. Wir zeigen: für $r = 1 + \frac{1}{p(|x|)}$ liefert $\mathcal{A}(x, r)$ eine optimale Lösung für x (in Polynomzeit $q(|x|, p(|x|))$:

Da $m(x, \mathcal{A}(x, r))$ r-Approximation ist, gilt:

$$m(x, \mathcal{A}(x, r)) \geq m^*(x)\frac{p(|x|)}{p(|x|)+1} = m^*(x) - \frac{m^*(x)}{p(|x|)+1} \overset{[*]}{>} m^*(x) - 1$$

Da $m(x, \mathcal{A}(x, r)), m^*(x) \in \mathbb{N}$ und $m^*(x)$ größtmöglich, folgt

$$m^*(x) = m(x, \mathcal{A}(x, r)).$$

Aus der NP-Härte von $\mathcal{P}$ folgt nun die Behauptung. □

NP-Härte und Pseudo-Polynomialität

Eine gewisse Sonderrolle spielen Optimierungsprobleme, deren Schwierigkeit von der gewählten Codierung der (ganzen) Zahlen (unär oder binär) abhängt. Ist x eine Instanz eines NPO-Problems, so bezeichne $\max(\mathrm{x})$ den Betrag der betragsmäßig größten vorkommenden Zahl.

Definition: Ein NPO-Problem $\mathcal{P}$ heißt *pseudo-polynomiell*, wenn es eine Algorithmus $\mathcal{A}$ zu Lösung von $\mathcal{P}$ gibt, der, für jede Instanz x, mit einer Laufzeit arbeitet, die durch ein Polynom in $|\mathrm{x}|$ und $\max(\mathrm{x})$ beschränkt ist.

Beispiel: Das Rucksackproblem ist pseudo-polynomiell, wie oben vermerkt.
Für eine Instanz $x = (X = \{x_1, \ldots, x_n\}, \{p_1, \ldots, p_n\}, \{a_1, \ldots, a_n\}, b)$
würde $\max(x) = \max(p_1, \ldots, p_n, a_1, \ldots, a_n, b)$ sein.
Für die Laufzeit des früher betrachteten Algorithmus ist zu beachten:

$$O\left(n \sum_{i=1}^{n} p_i\right) \subseteq O\left(n^2 p_{\max}\right) \subseteq O\left(|x|^2 \max(x)\right).$$

Satz: Es sei $\mathcal{P} \in$ FPTAS für $\mathcal{P}$. Gibt es ein Polynom p, sodass für alle Instanzen $x \in I_{\mathcal{P}}$ gilt:

$$m^*(x) \leq p(|x|, \max(x)),$$

so ist $\mathcal{P}$ pseudo-polynomiell.

Beweis: Es sei $\mathcal{A}$ ein FPTAS für $\mathcal{P}$. Betrachte

$$\mathcal{A}'(x) = \mathcal{A}\left(x, 1 + \frac{1}{p(|x|, \max(x)) + 1}\right).$$

Aufgrund der Ganzzahligkeit der Lösungen liefert $\mathcal{A}'(x)$ ein Optimum. Ist die Laufzeit von $\mathcal{A}(x, r)$ durch $q\left(|x|, \frac{1}{r-1}\right)$ beschränkt, so ist die Laufzeit von $\mathcal{A}(x)$ durch $q(|x|, p(|x|, \max(x)) + 1)$ beschränkt, d.h. $\mathcal{P}$ ist pseudo-polynomiell. □

Definition: Es sei $\mathcal{P}$ ein NPO-Problem und p ein Polynom. $\mathcal{P}^{\max,p}$ bezeichne die Einschränkung von $\mathcal{P}$ auf Instanzen x mit $\max(x) \leq p(|x|)$. $\mathcal{P}$ heißt *stark NP-hart*, wenn es ein Polynom p gibt, sodass $\mathcal{P}^{\max,p}$ NP-hart ist.

Satz: Ist P $\neq$ NP, so ist kein stark NP-hartes Problem pseudo-polynomiell.

Beweis: Angenommen, $\mathcal{P}$ ist ein stark NP-hartes Problem, das auch pseudo-polynomiell ist. Dann gibt es einen Algorithmus $\mathcal{A}$, der $\mathcal{P}$ in der Zeit $q(|x|, \max(x))$ löst für ein geeignetes Polynom q und jede Instanz $x \in I_{\mathcal{P}}$. Für jedes Polynom p kann $\mathcal{P}^{\max,p}$ daher in Zeit $q(|x|, p(|x|))$ gelöst werden. Da $\mathcal{P}$ stark NP-hart ist, müsste für ein p jedoch $\mathcal{P}^{\max,p}$ NP-hart sein, woraus P $=$ NP folgt. $\square$.

Folgerung: Ist $\mathcal{P}$ ein stark NP-hartes NPO-Problem und gibt es ein Polynom p, sodass $m^*(x) \leq p(|x|, \max(x))$ für jede Instanz $x \in I_{\mathcal{P}}$, so gilt $\mathcal{P} \notin$ FPTAS, wenn nicht P $=$ NP. $\square$

Die Sätze aus diesem Abschnitt gestatten es, für ein Problem zu schließen, dass es (unter gewissen komplexitätstheoretischen Annahmen) kein FPTAS besitzt. Dazu dient auch der folgende Zusammenhang, für dessen genaueres Verständnis wir auf die Vorlesung über parametrisierte Algorithmen verweisen.

Mitteilung: Besitzt das einem Optimierungsproblem $\mathcal{P}$ entsprechende parametrisierte Entscheidungsproblem (unter gewissen komplexitätstheoretischen Annahmen) *keinen* Festparameteralgorithmus, so liegt $\mathcal{P}$ nicht in FPTAS.

Zwischen APX und NPO

Wie oben gesehen, gibt es Probleme (wie das Handelsreisendenproblem), die sich (vorbehaltlich P $\neq$ NP) nicht bis auf einen konstanten Faktor mit einem Polynomzeitalgorithmus angenähert lösen lassen.
In solchen Fällen stellt sich die Frage, ob es denn wenigstens möglich ist, Leistungsgarantien für Approximationsalgorithmen zu finde, die von der Länge der Instanz abhängen.

Jetzt lernen wir Beispiele für solche Näherungsalgorithmen kennen.

Betrachten wir zunächst eine weitere Verallgemeinerung des Knotenüberdeckungsproblems:

Das Mengenüberdeckungsproblem (Set Cover, SC)

I : Eine Kollektion C von Teilmengen einer endlichen Grundmenge U.

S : Eine Mengenüberdeckung $C' \subseteq C$ für S, d.h. eine Teilkollektion, sodass jedes Element der Grundmenge U zu wenigstens einem Element der Teilkollektion gehört.

m : $|C'|$

opt : min

Mögliche Verallgemeinerung durch Einführung einer Kostenfunktion $k : U \to \mathbb{N}$

$\rightsquigarrow$ **Gewichtetes Mengenüberdeckungsproblem (Weighted Set Cover, WSC)**

Zusammenhang mit Knotenüberdeckung

Das Knotenüberdeckungsproblem ist insofern ein Mengenüberdeckungsproblem, als dass dort die Grundmengenelemente „Kanten“ heißen und ein Element der Kollektion einem „Knoten“ entspricht, genauer der Menge von Kanten, die mit nämlichen Knoten inzidieren.

$\rightsquigarrow$ Der folgende Greedy-Algorithmus für das Mengenüberdeckungsproblem eine unmittelbare Verallgemeinerung eines Greedy-Algorithmus des Knotenüberdeckungsproblems, denn die Auswahl eines Elementes der Kollektion mit größter Mächtigkeit entspricht gerade der Wahl eines Knotens von größtem Grad.

In ähnlicher Weise lässt sich auch das Hitting-Set-Problem als Mengenüberdeckungsproblem auffassen (und umgekehrt).

GreedyWSC (U, C, k)

1. Es sei U' das von $C' \subseteq C$ bereits abgedeckte Teiluniversum, anfangs also $C' \leftarrow \emptyset$, $U' \leftarrow \emptyset$.
2. Solange $U' \neq U$, tue:
   - 2a. Bestimme $c \in C \setminus C'$, sodass $\alpha(c) = \frac{k(c)}{|c \setminus U'|}$ kleinstmöglich ist.
   - 2b. Definiere $\pi(x) := \alpha(c)$ für $x \in c \setminus U'$.
   - 2c. Setze $C' \leftarrow C' \cup \{c\}$ und $U' \leftarrow U' \cup c$.
3. Liefere C' zurück.

$\pi(x)$ ist anschaulich gesprochen der Preis, der für das Überdecken von Element x gezahlt wurde. $\rightsquigarrow$ Kosten der Überdeckung C': $\sum_x \pi(x)$.

Sei $x_1, \ldots, x_n$ die Folge der Elemente von U in der Reihenfolge, in der sie von GreedyWSC überdeckt werden.
OPT: Kosten einer optimalen Mengenüberdeckung C^*.

Hilfssatz: Für alle $k = 1, \ldots, n$ gilt: $\pi(x_k) \leq \frac{OPT}{n-k+1}$.

Bei jedem Schleifendurchlauf könnten wir das noch abzudeckende Universum $U'' = U \setminus U'$ ganz einfach durch eine optimale Überdeckung der ursprünglichen Instanz abdecken, also mit Kosten OPT. Durch diese Strategie würden wir eine Durchschnittspreis von $\frac{OPT}{|U''|}$ für jedes $x \in U''$ bezahlen.

Wir wählen die Überdeckungsmenge jedoch so, dass der Preis pro Element kleinstmöglich ist. Also gilt $\pi(x_k) \leq \frac{OPT}{|U''|}$.

Kurz bevor x_k überdeckt wurde, enthielt U'' wenigstens $n-k+1$ viele Elemente. Somit gilt: $\pi(x_k) \leq \frac{OPT}{|U''|} \leq \frac{OPT}{n-k+1}$.

Satz: Ist $n = |S|$, so ist GreedyWSC ein $(\lfloor \ln n \rfloor + 1)$-approximativer Polynomzeitalgorithmus für das Mengenüberdeckungsproblem.

Starten wir zunächst mit einem kleinen mathematischen *Exkurs*:
$H(r) := \sum_{i=1}^{r} \frac{1}{i}$ bezeichnet die *r-te harmonische Zahl*.
Bekannt: $(H(r)) = \Theta(\log(r))$. Noch genauer gilt:

$$\lfloor \ln r \rfloor \leq H(r) \leq \lfloor \ln r \rfloor + 1$$

Ist C' die von GreedyWSC gefundene Lösung, so gilt für die betreffende Kostenfunktion π:

$$k(C') = \sum_{x} \pi(x) \leq \sum_{k} \frac{OPT}{n - k + 1} = H(n)OPT \leq (\lfloor \ln n \rfloor + 1)\, OPT$$

mit dem obigen Hilfssatz.

Zwei Bemerkungen

— Die in obigem Satz angegebene Abschätzung der Leistungsgüte von Greedy-WSC ist nicht verbesserbar. Es gibt eine Folge von Instanzen, für die die Schranke auch erreicht wird.

Betrachte z.B. eine Instanz aus n Elementen, die durch eine Menge mit Kosten $(1+\epsilon)$ abgedeckt werden kann und die ferner n einelementige Mengen mit Kosten $1/i$ für das $i$te Element enthält. Der Greedy-Algorithmus würde die einelementigen Mengen zur Überdeckung auswählen.

— GreedyWSC ist optimal in dem Sinne, dass es (unter einigen „wahrscheinlich“ komplexitätstheoretischen Annahmen) für kein $\epsilon > 0$ einen „$(\ln - \epsilon)$-Approximationsalgorithmus“ für SC gibt.

Graphenfärben

GreedyGC (G=(V,E))

1. $i := 0; U := V$
2. Solange $U \neq \emptyset$ tue:

   2a. $i := i + 1$;

   2b. $I :=$ GreedyIndependentSet$(G(U))$;

   2c. Färbe Knoten aus I mit „Farbe“ i;

   2d. $U := U \setminus I$
3. Liefere so erhaltene Färbung $f : V \to \mathbb{N}$ zurück.

Lemma: Ist $G = (V, E)$ k-färbbar, so benötigt GreedyGC höchstens $\frac{3|V|}{\log_k(|V|)}$ viele Farben zum Färben von G.

Beweis: Es sei $H = G(U)$ irgendein im Schritt 2b. verarbeiteter Teilgraph von G. Da G k-färbbar ist, ist auch H k-färbbar. (wird fortgesetzt)

Hilfssatz: Wird ein Graph H GreedyIndependentSet als Eingabe gegeben, so wird eine unabhängige Menge $I \subseteq U$ geliefert, die wenigstens $\lceil \log_k(|U|) \rceil$ viele Knoten enthält.

Beweis: Da H k-färbbar, enthält H eine unabhängige Menge $\bar{I}$ mit wenigstens $|U|/k$ Knoten. Jeder Knoten dieser Menge hat einen Maximalgrad $(|U| - |U|/k)$ (denn sonst gäbe es Verbindungen zwischen Knoten $\bar{I}$). Damit ist trivialerweise der Minimalgrad in H maximal $(|U| - |U|/k)$. Da als erster Knoten v_1 (für I) von GreedyIndependentSet einer minimalen Grades gewählt wird und er samt seinen höchstens $(|U| - |U|/k)$ vielen Nachbarn entfernt wird, wird der nächste Knoten v_2 in $H(U')$ gesucht mit $U' = U \setminus N[v_1]$ und

$$|U'| = |U \setminus N[v_1]| \geq |U| - (|U| - |U|/k) = |U|/k. \qquad (*)$$

Da $H(U \setminus N[v_1])$ k-färbbar, lässt sich das Argument wiederholen.
Abgebrochen wird diese Suche, wenn U erschöpft ist.
Dazu sind wegen $(*)$ mindestens $\lceil \log_k(|U|) \rceil$ viele Schritte nötig, und soviele Knoten werden auch wenigstens zu I hinzugenommen. □

Fortsetzung des Beweises des Lemmas:

Hilfssatz $\rightsquigarrow$ wenigstens $\lceil \log_k(|U|) \rceil$ viele Knoten mit Farbe i gefärbt.
Wie groß ist U vor dem Durchlaufen der Schritte 2a.-2d.?

Fall 1: Solange $|U| \geq |V|/\log_k(|V|)$, gilt wegen $|V|/\log_k(|V|) > \sqrt{|V|}$:

$$\lceil \log_k |U| \rceil \geq \log_k |U| \geq \log_k (|V| / \log_k (|V|)) > \log_k \sqrt{|V|} = \frac{1}{2} \log_k (|V|)$$

Da U bei jedem Durchlauf der Schleife „2“ um wenigstens $\frac{1}{2} \log_k (|V|)$ abnimmt, kommt der Fall 1 höchstens $2|V| / \log_k (|V|)$ oft zur Anwendung, und dabei werden höchstens $2|V| / \log_k (|V|)$ viele Farben benutzt.

Fall 2: Sobald $|U| < |V|/\log_k(|V|)$, reichen trivialer Weise $|V|/\log_k(|V|)$ viele Farben aus.

Fall 1 und Fall 2 zusammen liefern die Behauptung. □

Satz: Ist $n = |V|$, so ist GreedyGC ein $O(n/\log(n))$-approximativer Algorithmus fürs Graphenfärben.

Beweis: Nach dem Lemma benötiget GreedyGC höchstens $3n/\log_k(n)$ viele Farben mit $k = m^*(G)$. Also ist

$$\frac{m(G, \text{GreedyGC}(G))}{m^*(G)} \leq \frac{3n\log(m^*(G))/\log(n)}{m^*(G)} \leq \frac{3n}{\log(n)}. \qquad \square$$

Bem.: Der soeben dargestellte $O(n/\log n)$-approximative Algorithmus ist nicht bestmöglich: Es gibt einen $O\left(n\frac{(\log\log n)^2}{(\log n)^3}\right)$-approximativen Algorithmus.
Auf der anderen Seite ist bekannt, dass es —sofern nicht P $=$ NP— keinen $n^{1/7-\epsilon}$-approximativen Algorithmus geben kann für jedes $\epsilon > 0$.
Insbesondere dürfte das Graphenfärbeproblem nicht in APX liegen.