Approximationsklassen für Optimierungsprobleme

Matthias Erbar

19. September 2007

Inhaltsverzeichnis

1 Einleitung **1**

2 Approximationsalgorithmen mit garantierter Güte **2**

2.1 Terminologie 2

2.2 Garantierte absolute Güte 2

2.3 Garantierte relative Güte 2

2.4 Approximationsschemata 4

2.5 Erweiterte Klassen 5

1 Einleitung

Viele Optimierungsprobleme aus der Klasse $\mathcal{NPO}$ sind effizient nicht exakt lösbar. Algorithmen, die in polynomieller Zeit eine exakte Lösung liefern, sind uns nicht bekannt und wir wissen, dass im Fall $\mathcal{P} \neq \mathcal{NP}$ für viele Probleme keine solchen Algorithmen existieren können. Das macht die Entwicklung von Approximationsalgorithmen notwendig. Dabei ist es wichtig zu wissen, wie gut die gelieferten approximativen Lösungen sind und es ist wünschenswert, eine Güte der Approximation im vorhinein vorzugeben.
Unterschiedliche Optimierungsprobleme zeigen diesbezüglich unter Umständen sehr unterschiedliche Approximierbarkeitseigenschaften. Manche lassen sich mit beliebiger Genauigkeit effizient lösen, andere weisen eine intrinsiche Grenze auf, ab der genauere Lsungen nicht mehr effizient gewonnen werden können.
Im folgenden wollen wir verschiedene Begriffe von Approximierbarkeit vorstellen und miteinander vergleichen. Anhand dieser Begriffe wollen wir eine Klassifikation der $\mathcal{NPO}$-Probleme vornehmen. Wir betrachten Beispiele, die unterschiedliches Approximationsverhalten illustrieren und erhalten, $\mathcal{P} \neq \mathcal{NP}$ voraussgesetzt, negative Resultate, was die Approximierbarkeit einiger Probleme betrifft.

2 Approximationsalgorithmen mit garantierter Güte

2.1 Terminologie

Wir klären zunächst kurz unsere Terminologie für Optimierungsprobleme.

Definition 2.1. *Ein Optimierungsproblem $P = (\mathcal{I}, \mathcal{S}, m)$ ist gegeben durch:*

- *$\mathcal{I}$: Menge der Instanzen*
- *$\mathcal{S}$: Funktion, die $x \in \mathcal{I}$ die Menge der Lösungen von x zuordnet*
- *m: Maßfunktion, gibt Wert einer Lösung an*

Der Optimalwert wird mit $m^*(x)$ bezeichnet.

In dieser Formulierung sieht zum Beispiel das Knapsack-Problem wiefolgt aus:

- $\mathcal{I} = \{$Menge von Gegenständen x_i mit Wert p_i u. Größe g_i, Kapazität B$\}$
- $\mathcal{S}(x) = \{$Teilmenge von x mit $\sum g_i < B\}$
- $m(x) = \sum p_i$

2.2 Garantierte absolute Güte

Wichtig bei Approximationsalgorithmen ist, im vorhinein zu wissen, wie gut der Wert der gelieferten Lösung mit dem Optimum übereinstimmt. Man kann fordern, dass der Algorithmus immer eine Lösung liefert, deren absulute Abweichung vom Optimum kleiner als eine gewisse Schranke ist. Genauer:

Definition 2.2. *Algorithmus A heißt **absoluter Approximationsalgorithmus** zu Problem P, falls er für jedes $x \in \mathcal{I}$ eine Lösung $A(x) \in \mathcal{S}(x)$ liefert und eine Konstante k existiert, mit:*

$$\forall x \in \mathcal{I} : \ |m^*(x) - m(x, A(x))| < k \ .$$

In der Praxis ist dies meist zu restriktiv. Beispielsweise lässt bekanntlich das Knapsack-Problem wegen seiner Skalierungseigenschaften keinen polynomiellen absoluten Approximationsalgorithmus zu, vorausgesetzt, dass $\mathcal{P} \neq \mathcal{NP}$. Wir wollen uns also im folgenden Auf relative Abweichungen konzentrieren.

2.3 Garantierte relative Güte

Als Maß für die relative Güte verwenden wir die so genannte *performance ratio*.

Definition 2.3. *Sei y Lösung zur Instanz x. Die **performance ratio (PR)** von x und y ist gegeben durch:*

$$r(x, y) = max(\frac{m^*(x)}{m(x, y)}, \frac{m(x, y)}{m^*(x)})$$

Diese Definition umfasst Maximierungs- und Minimierungsprobleme. Die PR ist immer ≥ 1 und $= 1$ nur für eine optimale Lösung. Analog zu eben können wir von einem Algorihmus fordern, immer Lösungen zu liefern, deren performance ratio kleiner als eine gewisse Schranke ist.

Definition 2.4. *Sei $r \geq 1$. Ein Algorithmus A heißt* ***r-Approximationsalgorithmus*** *zu einem Problem P, falls:*

$$\forall x \in \mathcal{I}: \quad r(x, A(x)) < r \ .$$

Wir sagen in diesem Fall, der Algorithmus A hat eine garantierte relative Güte von r.

Alle $\mathcal{NPO}$-Probleme die einen polynomiellen r-Approximationalgorithmus zulassen für ein $r \geq 1$ wollen wir in einer Klasse zusammenfassen, die wir mit $\mathcal{APX}$ bezeichnen. Nun stellt sich die Frage, ob nicht jedes $\mathcal{NPO}$-Problem sich in dieser Weise approximieren lässt. Wie der nächste Satz zeigt ist dies äquivalent zu der Frage, ob $\mathcal{P} \neq \mathcal{NP}$. Wir werden zeigen, dass ein polynomieller r-Approximationalgorithmus für das *Traveling-Salesperson-Problem* (TSP) genutzt werden kann um ein $\mathcal{NP}$-vollständiges Problem in polynomieller Zeit zu lösen.

Theorem 2.5. *Falls TSP zu $\mathcal{APX}$ gehört, gilt $\mathcal{P} = \mathcal{NP}$.*

Beweis. Sei $r \geq 1$ und A ein polynomieller r-Approximationsalgorithmus für TSP. Wir reduzieren das $\mathcal{NP}$-vollständige Entscheidungsproblem, ob ein gerichteter Graph einen Hamiltonkreis enthält auf TSP.
Sei der gerichtete Graph $G = (E, V)$ eine Instanz des Hamiltonproblems. Wir konstruieren eine Instanz von TSP als den vollständigen Graphen über V mit den Gewichten:

$$d(v, w) = \begin{cases} 1 & (v, w) \in E \\ 1 + nr & \text{sonst} \end{cases}$$

Falls ein Hamiltonkreis existiert, so hat eine optimale Lösung von TSP den Wert n. Die nächstbeste approximative Lösung hat mindestens den Wert $n(1+r)$. Falls kein Hamiltonkreis existiert, so ist der Wert einer Optimallösung mindestens $n(1 + r)$. Da der Approximationsalgorithmus eine Lösung mit $PR < r$ liefert, erhalten wir genau dann eine Lösung mit Wert n, falls ein Hamiltonkreis in (V, E) existiert. Somit wäre ein polynomieller Entscheidungsalgorithmus für ein $\mathcal{NP}$-vollständiges Problem gefunden. □

Bemerkung 2.6. Der Beweis verwendet die Technik des sogenannten **Gap Amplifying**. Die diskrete Struktur der Wertfunktion wird ausgenutzt, um den Algorithmus in einem skalierten Problem zu zwingen, nur eine Optimallösung als Lösung mit hinreichend kleiner PR zu akzeptieren. Diese Technik wird häufig benutzt und uns später nochmals begegnen.

Innerhalb der Klasse $\mathcal{APX}$ zeigt sich sehr unterschiedliches Approximationsverhalten. Von vielen wichtigen Problemen lässt sich zeigen, dass sie sich nur bis zu einer inherenten Grenze effizient approximieren lassen. Approximationsalgorithmen mit einer garantierten PR unterhalb dieser Grenze können nicht existieren, falls $\mathcal{P} \neq \mathcal{NP}$. Solche Resultate beweist man ähnlich dem obigem, wie wir bei *Minimum Bin Packing* sehen werden.
Manche Probleme erlauben jedoch einen Algorithmus der bei beliebig vorgegebner PR effizient eine Lösung liefert, ein sogenanntes Approximationsschema.

2.4 Approximationsschemata

Oft benötigt man Lösungen einer frei wählbaren Approximationsgüte. Für eine bessere PR wird dabei höherer Rechenaufwand in Kauf genommen. Dies leisten die Approximationsschemata, genauer:

Definition 2.7. *Ein Algorithmus A heißt **polynomielles Approximationschema** für ein Problem P, falls er für alle $x \in \mathcal{I}, r > 1$ eine Lösung $A(x,r)$ liefert mit $r(x, A(x,r)) < r$, wobei die Laufzeit polynomiell in $|x|$ ist.*

Wir untersuchen zunächst ein Beispiel eines solchen Schemas für das Problem *Partition*.

Beispiel 2.8 (Optimierungsproblem Partition)**.** Menge X von Elementen x_i mit Gewichten w_i, soll in zwei möglichst gleich schwere Mengen Y, Z aufgeteilt werden. Das Gewicht der schwereren der beiden Mengen $max(w(Y), w(Z))$ soll minimiert werden.

Code 2.9 (Partition PTAS)**.**

```
input: Menge X von Elementen mit ganzen Gewichten w_i, r > 1;
output: Partition von X in Y u. Z;
begin
if r > 2 then return X, leer
else
Sortiere Elemente in nicht aufsteigender Reihenfolge bezgl. Gewicht;
k(r) := ⌈(2-r)/(r-1)⌉;
Finde optimale Lsung Y, Z fr x_1, ..., x_k(r);
for j:=k(r)+1 to n do
if Σ_Y w_i < Σ_Z w_i then
Y := Y ∪ {x_j}
else
Z := Z ∪ {x_j};
return Y, Z;
end;
end.
```

Analyse 2.10. Der Algorithmus löst eine Teilinstanz des Problems, die $k(r)$ schwersten Elemente, optimal und verwendet eine Greedy-Algorithmus für den Rest. Da k nicht voin $|x|$ abhängt, ist die Laufzeit polynomiell. Bleibt zu zeigen, dass der Algorithmus immer Lösungen mit $PR < r$ liefert.
OBdA sei $r < 2$, denn jede Lösung hat $PR < 2$. Sei weiter OE $w(Y) \geq w(Z)$ und x_l das letzte zu Y hinzugefügte Element. Setze $L = \frac{w(X)}{2}$.
Falls x_l in erster Phase hinzugefügt wurde, so ist die Lösung optimal. Falls nicht, so gilt: $w(Y) \leq L + \frac{w_l}{2}$. Weiter gilt: $w(X) \geq a_l(k(r)+1)$, denn: $w_l \leq w_j,\ 1 \leq j \leq k(r)$. Also erhalten wir zusammen:

$$PR = \frac{w(Y)}{m^*(X)} \leq \frac{w(Y)}{L} \leq 1 + \frac{w_l}{2L} \leq 1 + \frac{1}{\frac{2-r}{r-1} + 1} = r\ .$$

Bemerkung 2.11. Das hier verwendete Prinzip wird häufig zur Kontruktion von Approximationsschemata benutzt. Zunächst wir eine ausreichend große Teilinstanz des Problems optimal gelöst, und dann ein bekannter r-Approximationsalgorithmus auf den Rest angewendet.

Alle $\mathcal{NPO}$-Probleme die einen polynomielles Approximationschema zulassen wollen wir in einer Klasse zusammenfassen, die wir mit $\mathcal{PTAS}$ (für **P**olynomial **T**ime **A**pproxsimation **S**cheme) bezeichnen. Offenbar ist $\mathcal{PTAS}$ in $\mathcal{APX}$ enthalten. Wieder stellt sich die Frage, ob nicht schon jedes Problem in $\mathcal{APX}$ ein polynomielles Approximationsschema zulässt. Der folgende Satz zeigt, dass dies äquivalent ist zu der Frage, ob $\mathcal{P} \neq \mathcal{NP}$.

Theorem 2.12. *Falls Minimum Bin Packing zu $\mathcal{PTAS}$ gehört, so $\mathcal{P} = \mathcal{NP}$.*

2.13 (Optimierungsproblem Minimum Bin Packing)**.** Eine endliche Menge von Elementen u mit Größe $g(u)$ soll passend in möglichst wenige Dosen mit Kapazität B gepackt werden.

Beweis. Sei A ein polynomielles Approximationsschema für Minimum Bin Packing. Wir reduzieren das $\mathcal{NP}$-vollständige Entscheidungsproblem Partition auf Min Bin Packing mit Hilfe von A. Partition muss entscheiden, ob eine Menge mit gewichteten Elementen in zwei Mengen gleichen Gewichts geteilt werden kann.
Sei also X eine Instanz von Partition, B die Summe der Gewichte. Wir erhalten eine Instanz X' von Min Bin Packing wie folgt: für jedes Element von X mit Gewicht w hat X' ein Element der Größe $\frac{2w}{B}$.
Falls X partitionierbar ist, so ist $m^*(X') = 2$, sonst $m^*(X') = 3$. Das Approximationsschema liefert also für $r < \frac{3}{2}$ wegen des *gap amplifying* genau eine Lösung mit 2 Bins, falls X partitionierbar ist. Somit wäre ein polynomieller Entscheidungsalgorithmus für ein $\mathcal{NP}$-vollständiges Problem gefunden. □

Bei der Definition eines PTAS wurde nichts über die Abhängigkeit der Laufzeit von der vorgegebenen Approximationsgüte vorausgesetzt. Wächst die Laufzeit sehr schnell mit der geforderten Güte, so ist das Schema wenig nützlich. Wünschenswert sind strenge Schemata, deren Laufzeit polynomiell von $|x|$ **und** $\frac{1}{r-1}$ abhängt.

Definition 2.14. *Ein Algorithmus A heit **vollständig polynomielles Approximationsschema** für ein Problem P, falls er für alle $x \in \mathcal{I}, r > 1$ eine Lösung $A(x,r)$ liefert mit $r(x, A(x,r)) < r$, wobei die Laufzeit polynomiell in $|x|$ **und** $\frac{1}{r-1}$ ist.*

Die $\mathcal{NPO}$-Probleme, die ein vollständig polynomielles Approximationsschema zulassen, fassen wir wiederum in einer Klasse zusammen, die wir mit $\mathcal{FPTAS}$ (für **F**ully **P**olynomial **T**ime **A**pproxsimation **S**cheme) bezeichnen. Falls $\mathcal{P} \neq \mathcal{NP}$, gilt wieder $\mathcal{FPTAS} \subsetneq \mathcal{PTAS}$. Ein Beispiel für ein Problem, das ein FPTAS zulässt, ist *Maximum Knapsack*.

2.5 Erweiterte Klassen

Die bisher betrachteten Klassen von Optimierungsproblemen lassen sich natürlich noch erweitern. Wir haben bis jetzt recht strenge Forderungen an die Approximierbarkeit gestellt. In der Klasse $\mathcal{APX}$ haben wir zum Beispiel nur Probleme betrachtet, die einen Algorithmus mit einer konstanten Schranke an die PR zulassen. Von Interesse sind aber auch Approximationsalgorithmen, deren garantierte PR eine langsam wachsende Funktion der Inputgröße ist.
Beispielweise gibt es zum Problem der minimalen Knotenfärbung eines Graphen einen Algorithmus, der zu jeder Instanz G eine Lösung $A(G)$ liefert mit

$$r(x, A(G)) < \mathcal{O}(\frac{n}{log(n)}),$$

wobie n die Anzahl der Knoten ist.
Eine andere Möglichkeit, die Forderungen an die Güte abzuschwächen, ist, asymptotische Schranken zu betrachten. Oft sind für einen Approximationsalgorithmus die Werte der Lösungen nur affin linear durch die Optimallösung beschränkt, d.h.:

$$m(A(x)) < rm^*(x) + k$$

für Konstanten k und $r > 1$. Der Algorithmus liefert also Lösungen, deren PR $\frac{m(A(x))}{m^*(x)}$ für große Werte der optimalen Lösung asymptotisch durch r beschränkt ist. Das ist insbesondere bei Approximationsschemata interessant und wir definieren deshalb:

Definition 2.15. *Ein Algorithmus A heißt* ***asymptotisches polynomielles Approximationsschema*** *für ein Problem P, falls eine Konstante k existiert, so dass*

$$\forall x \in \mathcal{I},\ r > 1 :\ r(x, A(x, r)) \leq r + \frac{k}{m^*(x)}\ ,$$

wobei die Laufzeit polynomiell in $|x|$ ist.

Die Klasse der $\mathcal{NPO}$-Probleme, die ein asymptotisches polynomielles Approximationschema zulassen, bezeichnen wir mit $\mathcal{PTAS}^\infty$. Offenbar gilt:

$$\mathcal{PTAS} \subset \mathcal{PTAS}^\infty \subset \mathcal{APX}.$$

Ein Beispiel für ein Problem mit einem asymptotischen polynomiellen Approximationsschema ist *Minimum Bin Packing*.

Literatur

[Aus99] G. Ausiello et al. *Complexity and Approximation.* Springer, 1999.