Näherungsalgorithmen

(Approximationsalgorithmen)

WiSe 2012/13 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

3. Dezember 2012

Näherungsalgorithmen Gesamtübersicht

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

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

Einige Definitionen

Ein *Multigraph* ist ein Paar $M = (V, E)$, wobei V eine endliche Knotenmenge ist und F eine Multimenge von Kanten.
Eine Multimenge von Kanten (über V) ist eine Abbildung $F : \{\{u, v\} \mid u, v \in V\} \to \mathbb{N}_0$; sogenannte „Mehrfachkanten“ sind also zugelassen.
Wir werden im Folgenden auch Multigraphen mit *Kantengewichten* betrachten; dabei wird jeder Kante $e = (\{u, v\}, j)$, $j \leq F(\{u, v\})$ ein Gewicht $c(e)$ zugeordnet.

Ein Graph $G = (V, E)$ mit Kantengewichten erfüllt die *$\triangle$-Ungleichung*, wenn

$$\forall u, x, v \in V : c(\{u, v\}) \leq c(\{u, x\}) + c(\{x, v\})$$

für $\{u, v\}, \{u, x\}, \{x, v\} \in E$.

L. Euler

Eine Folge von Knoten $f \in V^*$ heißt *Eulersch*, wenn jedes $v \in V$ mindestens einmal in f vorkommt, der erste Knoten von f gleich dem letzten ist und für alle Zerlegungen $f = xuvy, u, v \in V$ gilt: $F(\{u, v\}) \geq 1$, sowie

$$\forall u, v \in V, u \neq v : F(\{u, v\}) = |\{x \in V^* \mid \exists y \in V^* : f = xuvy \vee f = xvuy\}| .$$

M.a.W.: jede Kante des Multigraphen wird genau einmal durchlaufen.
Ein Multigraph heißt Eulersch, wenn es eine Eulersche Knotenfolge gibt.

Mitteilung: Ein Multigraph ist genau dann Eulersch, falls alle seine Knoten (zusammenhängen und) geraden Grad besitzen.

Es gibt einen Polynomzeitalgorithmus, der entscheidet, ob ein gegebener Multigraph Eulersch ist und dabei auch eine zugehörige Knotenfolge konstruiert.

Ein wichtiges Lemma

— Ist $G = (V, E)$ ein vollständiger Graph mit Kantengewichten mit Gewichtsfunktion c, der die $\triangle$-Ungleichung erfüllt und
— ist $M = (V, F)$ irgendein Eulerscher Multigraph mit derselben Knotenmenge derart, dass alle Kanten zwischen zwei beliebigen Knoten $\{u, v\} \subseteq V$ das Gewicht $c(\{u, v\})$ haben,
$\rightsquigarrow$ dann findet man in Polynomzeit eine Tour I in G mit einem Wert, der höchstens gleich der Summe der Gewichte der Kanten in M ist.

Beweis: Es sei $V = \{v_1, \ldots, v_n\}$. Per Definition ist die Summe der Gewichte aller Kanten, die in einer Eulerschen Knotenfolge „vorkommen“, gleich der Summe der Gewichte der Kanten in M.
Wir bestimmen als eine Eulersche Knotenfolge $f = v_{i_1} \ldots v_{i_m}$ in M.
Es sei $j(r)$ der erste Index in $1, \ldots, m$ mit $v_{i_{j(r)}} = v_r$, $1 \leq r \leq n$.
Annahme: die $v_1, \ldots, v_n$ sind so sortiert, dass gilt:

$$j(1) \leq j(2) \leq \cdots \leq j(n).$$

$\triangle$-Ungleichung $\rightsquigarrow$

$$\forall 1 \leq r \leq n : c(\{v_r, v_{(r+1 \bmod m)+1}\}) = c\left(\left\{v_{i_{j(r)}}, v_{i_{(j(r+1) \bmod m)+1}}\right\}\right)$$

$$\leq c\left(\left\{v_{i_{j(r)}}, v_{i_{j(r) \bmod m+1}}\right\}\right) + c\left(\left\{v_{i_{j(r) \bmod m+1}}, v_{i_{j(r) \bmod m+2}}\right\}\right) + \ldots$$

$$\ldots + c\left(\left\{v_{i_{j(r+1) \bmod m}}, v_{i_{j(r+1) \bmod m+1}}\right\}\right)$$

Für die Tour $v_{i_{j(1)}}, \ldots, v_{i_{j(n)}}$ gilt daher die Behauptung. $\square$

Matchings

Ein *Matching* in einem Graphen ist eine Kantenmenge, in der keine zwei Kanten einen gemeinsamen Endpunkt haben.
Bekannt: In der Standard-Algorithmenvorlesung wird ein Polynomzeitalgorithmus zum Auffinden größtmöglicher (Maximum) Matchings vorgestellt.

Ein Matching heißt *perfekt*, wenn jeder Knoten des Graphen Endpunkt einer Kante aus dem Matching ist.
Es gibt einen Polynomzeitalgorithmus zum Auffinden minimaler perfekter Matchings, wobei „Minimalität“ im Sinne der jetzt angenommenen Kantengewichte des Graphen zu sehen ist.

Wir stellen jetzt den Algorithmus von Christofides vor.

Algorithmus von Christofides

MMTS-CHR$(G = (V, E), c)$ (G vollständiger Graph, c Kantengewichtsfkt.)

1. Finde einen (bzgl. c) minimalen Spannbaum $T = (V, E_T)$ von G.
2. Es sei $U \subseteq V$ die Menge der Knoten in T mit ungeradem Grad.
3. Finde ein (bzgl. c) minimales perfektes Matching H in $G[U]$.
4. Bilde den Multigraphen $M = (V, E_T + H)$.
   (In M gibt es nur Einfach- oder Doppelkanten.)
5. Finde eine Eulersche Knotenfolge f in M.
6. Bilde (wie in obigem Lemma) eine Tour I aus f.
7. Liefere I zurück.

Satz: Ist $(G = (V, E), c)$ eine Instanz von MMTS, so liefert MMTS-CHR in Polynomzeit eine Lösung mit Leistungsgüte $\leq \frac{3}{2}$.

Beweis: Mit obigen Vorbemerkungen und Grundkenntnissen aus einer Algorithmenvorlesung ist klar, dass MMTS-CHR in Polynomzeit arbeitet. Wir zeigen jetzt die Leistungsgüteschranke:
Dazu betrachten wir den Multigraph $M = (V, E_T + H)$ und eine Eulersche Knotenfolge f.
(Beachte: M ist Eulersch wegen obiger Mitteilung!)
Nach dem vorigen Lemma können wir eine Tour I finden, die

$$m(G, I) \leq c(E_T) + c(H)$$

genügt. Im Folgenden schätzen wir $c(E_T)$ und $c(H)$ nach oben mit Hilfe von $m^*(G)$ ab.

Behauptung 1:

$$c(H) \leq \frac{m^*(G)}{2}.$$

Beweis: Es sei $(v_{i_1}, \ldots, v_{i_{|U|}})$ die Folge der Knoten aus U in der Reihenfolge, in der sie in einer fixierten optimalen Tour I^* vorkommen.
Offensichtlich sind

$$H_1 = \left\{(v_{i_1}, v_{i_2}), \ldots, (v_{i_{|U|-1}}, v_{i_{|U|}})\right\} \text{ und}$$

$$H_1 = \left\{(v_{i_2}, v_{i_3}), \ldots, (v_{i_{|U|-2}}, v_{i_{|U|-1}}), (v_{i_{|U|}}, v_{i_1})\right\}$$

perfekte Matchings. Wegen der Gültigkeit der $\triangle$-Ungleichung gilt:

$$c(H_1) + c(H_2) \leq m^*(G) = c(I^*).$$

Da H ein perfektes Matching mit minimalem Gewicht ist, gilt $c(H) \leq c(H_1)$ und $c(H) \leq c(H_2)$, also gilt $2c(H) \leq m^*(G)$.

Behauptung 2:

$$c(E_T) \leq m^*(G).$$

Hierzu beobachte man: Eine (optimale Tour) ist ein Spannbaum (ein „Spannpfad“) plus einer zusätzlichen Kante. Da T minimaler Spannbaum ist, gilt die Behauptung.
Zusammengenommen haben wir gezeigt:

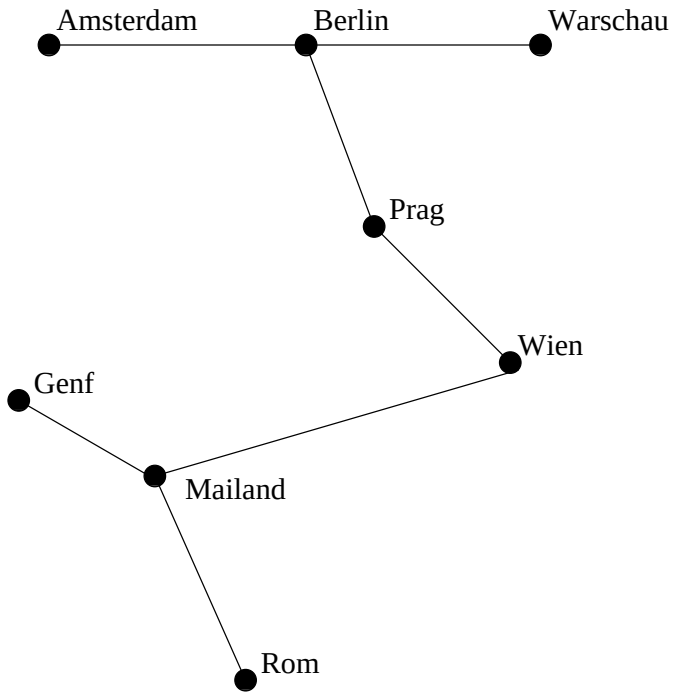
$$m(G, I) \leq m^*(G) + \frac{m^*(G)}{2} = \frac{3}{2} m^*(G).$$

□

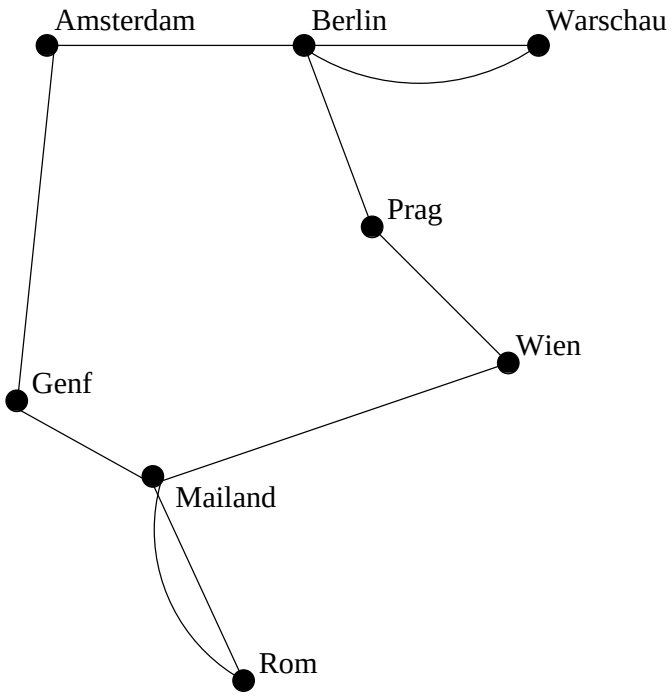
Praktischen Beispiel: Straßen-km-Entfernungstabelle zwischen den Städten Amsterdam (AMS), Berlin (BER), Genf (GEN), Mailand (MAI), Prag (PRA), Rom, Warschau (WAR) und Wien (WIE):

AMS	685	925	1180	960	1755	1235	1180
	BER	1160	1105	340	1530	585	630
		GEN	325	950	880	1575	1025
			MAI	870	575	1495	830
				PRA	1290	625	290
					ROM	1915	1130
						WAR	765
							WIE

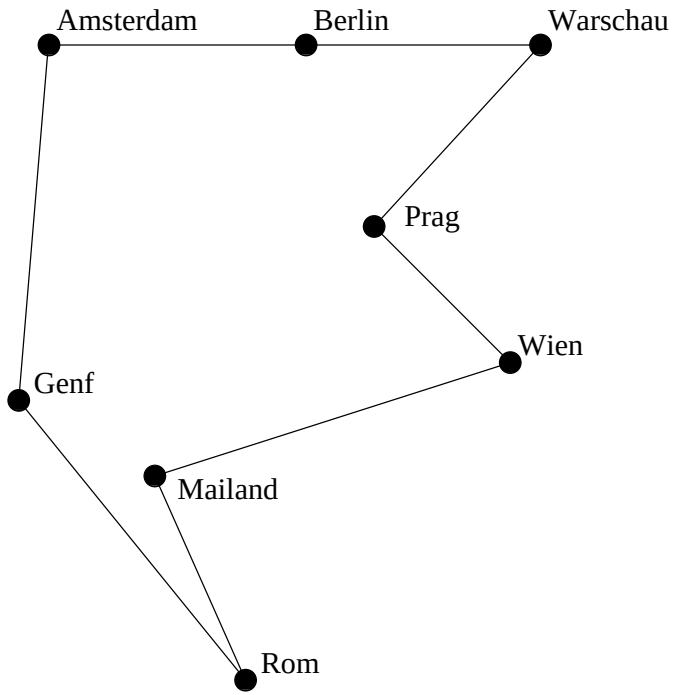
Ein minimaler Spannbaum



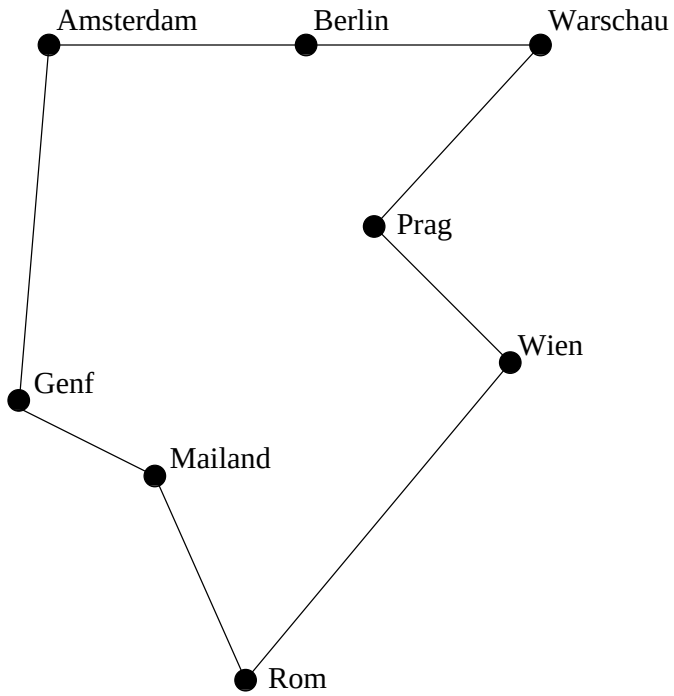
Der so erzeugte Multigraph



Die von Christofides erhaltene Tour I (vom Wert 5395)



Ein optimale Tour I^* (vom Wert 5140)



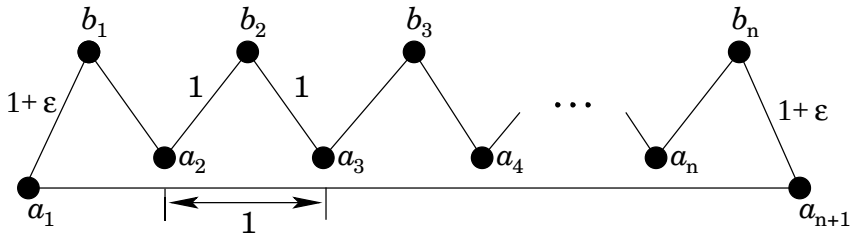
Die ersten drei Bilder zeigen **die drei Hauptphasen** des Algorithmus:
Schritt 1, 4 und 6.
Als Eulersche Knotenfolge nehmen wir dabei

AMS-BER-WAR-BER-PRA-WIE-MAI-ROM-MAI-GEN-AMS

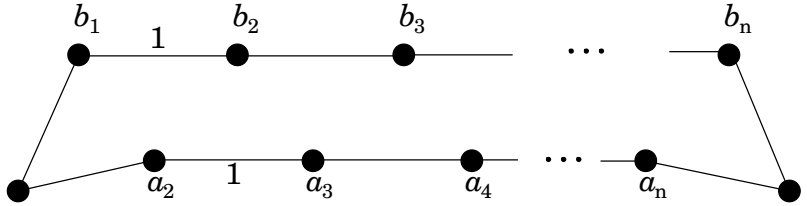
Eine optimale Tour zeigt das letzte Bild.
Diese ist nicht durch bloßes zyklisches Verschieben der Eulerschen Knotenfolge zu erzielen, da die Kante Wien-Rom bereits im Spannbaum fehlt.

Die **asymptotische Schärfe**

der für den Algorithmus von Chriftofides gefundene Schranke:



(a)The approximate tour



(b)The optimal tour

Erläuterungen:

Ein möglicher minimaler Spannbaum besteht aus der in der ersten Abbildung angegebenen Tour (ohne die Kante $\{a_1, a_{n+1}\}$). Ausgehend von diesem Spannbaum bestimmt MMTS-CHR die angegebene Tour vom Wert $3n + 2\epsilon$.
Der Wert der im zweiten Bild angegebenen (optimalen) Tour ist nur $2n + 1 + 4\epsilon$.

Ist Christophides optimal?

Bis heute wurde kein besserer Approximationsalgorithmus („besser“ nur im Sinne der Approximationsgüte) gefunden als MMTS-CHR **für den metrischen Fall**.

Beschränkt man sich jedoch weiter auf den Euklidischen Fall, so gibt es sogar ein PTAS ($\rightarrow$ nächstes Kapitel).

Verallgemeinerungen: Die β-Dreiecksungleichung:

$$\forall u, v, x : d(u, v) \leq \beta(d(u, x) + d(x, v))$$

$(\beta^2 + \beta)$-Approximation: Andrae: On the traveling salesman problem. . . in: Networks 38 (2001), 59–67.

4β-Approximation: Bender / Chekuri: Performance guarantees for the TSP with a parameterized triangle inequality. In: Proc. WADS 1999.

$3/2 \cdot \beta^2$-Approximation: Böckenhauer u.a.: Towards the notion of stability of approximation . . . In: TCS 285 (2002), 3–24.

Grenzen der Approximierbarkeit — Die Gap-Technik

Allgemeine Darstellung einer bereits kennen gelernten Beweistechnik:

Satz: Es seien $\mathcal{P}'$ ein NP-vollständiges Entscheidungsproblem und $\mathcal{P}$ ein Minimierungsproblem aus NPO. Angenommen, es gibt zwei in Polynomzeit berechenbare Funktionen

$$f : I_{\mathcal{P}'} \to I_{\mathcal{P}} \text{ und}$$

$$c : I_{\mathcal{P}'} \to \mathbb{N} \text{ sowie}$$

eine Konstante $gap > 0$, sodass für jede Instanz x von $\mathcal{P}'$ gilt:

$$m^*(f(x)) = \begin{cases} c(x), & x \text{ ist positive Instanz von } \mathcal{P}' \\ c(x)(1+gap), & \text{sonst.} \end{cases}$$

Dann gibt es keine Polynomzeit-r-Approximation für $\mathcal{P}$ mit $r < 1 + gap$, falls P $\neq$ NP gilt.

Beweis: Nehmen wir an, es gäbe solch eine r-Approximation $\mathcal{A}, r < 1 + gap$, für $\mathcal{P}$.
Benutze den Polynomzeitalgorithmus $\mathcal{A}$ dazu, um $\mathcal{P}'$ in Polynomzeit zu lösen:

$\mathcal{A}'(x)$:*

1. Berechne $f(x)$.
2. Berechne $y := \mathcal{A}(f(x))$.
3. Berechne $c := c(x)$.
4. Gilt $m(f(x), y) < c \cdot (1 + gap)$, so antworte JA!
5. Gilt $m(f(x), y) \geq c \cdot (1 + gap)$, so antworte NEIN!

*x ist eine Instanz von $\mathcal{P}'$.

Warum ist $\mathcal{A}'$ korrekt?

Ist x eine positive Instanz von $\mathcal{P}'$, so gilt nach Voraussetzung

$$\frac{m(f(x), y)}{c} = \frac{m(f(x), y)}{m^*(f(x))} \leq r < 1 + gap$$

$\mathcal{A}'$ antwortet in diesem Fall also richtig (Schritt 4.).

Ist x eine negative Instanz von $\mathcal{P}'$, so gilt nach Voraussetzung

$$m(f(x), y) \geq m^*(f(x)) = c \cdot (1 + gap),$$

sodass $\mathcal{A}'$ auch in diesem Fall korrekt antwortet (Schritt 5.). □

Anwendung auf das Graphenfärbeproblem

Mitteilung: Für **planare** Graphen (die nach dem Vierfarben-Satz ja 4-färbbar sind) ist die Frage nach der 3-Färbbarkeit NP-vollständig.

Daher gilt: Das (allgemeine sowie das auf planare Graphen eingeschränkte) Graphenfärbeproblem lässt sich nicht besser als mit Faktor $r < 4/3$ approximieren. $(gap = 1/3)$

Man kann hier sogar noch mehr zeigen:
Mitteilung: Gilt P $\neq$ NP, so liegt MinimumGraphColoring in NPO $\setminus$ APX.

Polynomzeit-Approximationsschemata

Soeben: Beispiele

- für Probleme, die sich *nicht mit konstantem Faktor* approximieren lassen (sofern nicht P $=$ NP)

- und für Probleme, die sich *nicht mit beliebigem Faktor* approximieren lassen (Graphenfärben auf planaren Graphen).

Jetzt: Probleme, die *beliebig genaue Approximation*sverfahren gestatten!

Wermutstropfen: Eine erhöhte Genauigkeit der Näherung wird durch eine erhöhte Laufzeit eingehandelt.

Definition

Es sei $\mathcal{P} \in$ NPO.
Ein Algorithmus $\mathcal{A}$ heißt *Polynomzeit-Approximationsschema* (engl: polynomial-time approximation scheme, kurz PTAS), falls
$\mathcal{A}$ für jedes Paar von Eingaben (x, r), x Instanz von $\mathcal{P}$ und $r > 1$,
eine r-approximative Lösung zurück liefert in einer Zeit,
die polynomiell in $|x|$ ist
(aber möglicherweise nicht-polynomiell in $\frac{1}{r-1}$).
Die zugehörige Problemklasse nennen wir *PTAS*.

Kleinstmögliche Zweiteilung (Minimum Partition, MP)

I : Endliche Menge X von Gegenständen,
zu jedem $x_i \in X$ ein Gewicht $a_i \in \mathbb{N}$.

S : $\{Y_1, Y_2 \subseteq X \mid Y_1 \cap Y_2 = \emptyset, Y_1 \cup Y_2 = X\}$

m : $\max\left\{\sum_{x_i \in Y_1} a_i, \sum_{x_i \in Y_2} a_i\right\}$

opt : min.

Klar: MP liegt in APX; die Zweiteilung $(X, \emptyset)$ ist trivialer Weise eine 2-Approximation.

Satz: MP $\in$ PTAS.

Beweis: Wir zeigen, dass die Leistungsgüte des folgenden Algorithmus,

gegeben $x = (X, a)$ und $r > 1$, durch r beschränkt ist.

Klar ist dies, falls $r \geq 2$.

Sei nun also $r \in (1, 2)$.

Setze $a(Y) := \sum_{x_j \in Y} a_j$, $Y \subseteq X$ und

$L := \frac{a(X)}{2}$.

O. E. sei $a(Y_1) \geq a(Y_2)$, und

x_h das letzte Element, das zu Y_1 hinzugefügt wurde durch MP-PTAS.

MP-PTAS(X, a, r)

1. Ist $r \geq 2$, so liefere $(X, \emptyset)$.
2. Sortiere X absteigend bzgl. ihrer Gewichte.
   Es sei $(x_1, \ldots, x_n)$ die so erhaltene Folge.
3. $k(r) := \left\lceil \frac{2-r}{r-1} \right\rceil$;
4. Finde eine optimale Zweiteilung (Y_1, Y_2) für $(x_1, \ldots, x_{k(r)})$.
5. Für $j := k(r) + 1$ bis n tue

   5a. Falls

   $$\sum_{x_i \in Y_1} a_i \leq \sum_{x_i \in Y_2} a_i,$$

   5b. so $Y_1 := Y_1 \cup \{x_j\}$,

   5c. sonst $Y_2 := Y_2 \cup \{x_j\}$.
6. Liefere (Y_1, Y_2).

Also ist $a(Y_1) - a_h \leq a(Y_2)$

$$\rightsquigarrow 2 \cdot a(Y_1) - a_h \leq a(Y_1) + a(Y_2) = 2L$$

$$\rightsquigarrow \quad a(Y_1) - L \leq \frac{a_h}{2} \qquad (*)$$

Wurde x_h bereits in Schritt 4 eingefügt, so liefert MP-PTAS sogar eine optimale Lösung.
Wurde x_h in Schritt 5 eingefügt, so haben wir (natürlich) $a_h \leq a_j$ für $1 \leq j \leq k(r)$ (absteigende Sortierung!) und

$$2L \geq \sum_{1 \leq j \leq h} a_j \geq h \cdot a_h \geq (k(r)+1) \cdot a_h. \qquad [+]$$

Wegen $a(Y_1) \geq L \geq a(Y_2)$ und $m^*(x) \geq L$ gilt ferner:

$$\frac{a(Y_1)}{m^*(x)} \leq \frac{a(Y_1)}{L} \overset{(*)}{\leq} 1 + \frac{a_h}{2L} \overset{[+]}{\leq} 1 + \frac{1}{k(r)+1} \leq 1 + \frac{1}{\frac{2-r}{r-1}+1} = r.$$

Klar ist: Die Laufzeit von MP-PTAS ist für festes r polynomiell in n; allerdings führt Schritt 4 zu exponentieller Laufzeit in $k(r)$ mit $k(r) \in O\left(\frac{1}{r-1}\right)$. □