Online-Algorithmen

Proseminar von
Prof. Dr. Rolf Klein, Dr. Elmar Langetepe, Dipl. Inform. Thomas Kamphans
im Wintersemester 00/01

Vortrag Bin Packing
von Thilo Geertzen

25. Oktober 2000

Online Algorithmen - Binpacking

Inhalt:

1. Bin Packing - Problemvorstellung

Was heißt Bin Packing?
Bin Packing kommt aus dem Englischen und setzt sich aus zwei Wörtern zusammen: "bin" bedeutet Kiste, Eimer oder Behälter und "packing" steht für packen/verpacken. Wie wir hier schon erkennen können, geht es also um Kisten verpacken.
Das Bin Packing-Problem ist nicht nur ein theoretisches Problem sondern hat durchaus praktische Relevanz. Zum Beispiel
bei der Post: Wie belade ich meine LKWs ohne Platzverschwendung mit Paketen?
bei einer Zeitung: Wie kann ich eine Zeitungs-Seite optimal mit verschieden großen Artikeln und Werbung ausfüllen?

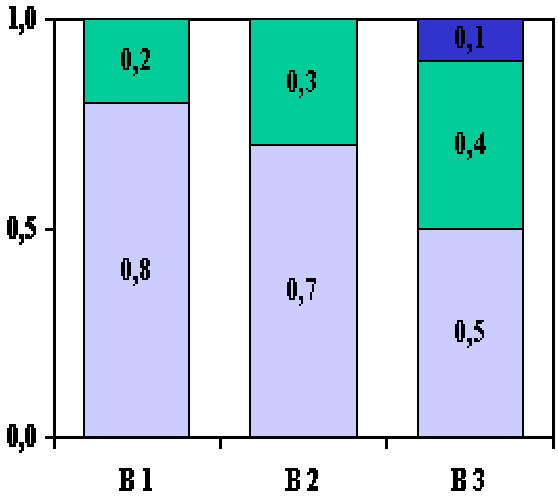
Eine Animation von Binpacking, die das ganze Problem veranschaulicht und insbesondere die Strategie First-Fit verdeutlicht, gibt es hier.

Bin Packing wird folgendermaßen formal beschrieben:

Gegeben: n Objekte mit Größen s1, ..., sn mit
$0 < si \leq 1$ für $1 \leq i \leq n$

Gesucht: Die kleinstmögliche Anzahl von Kisten (Bins) mit Größe 1, die alle Objekte aufnehmen kann.

Beispiel: 7 Objekte mit den Größen:
0.2, 0.5, 0.4, 0.7, 0.1, 0.3, 0.8



Optimale Packung (offline)

top

1.1. Grundlegende Begriffe

Das vorangehende Beispiel zeigte die optimale Offline-Packung der 7 Objekte. Uns interssieren jedoch überwiegend die Möglichkeiten, den Algorithmus online auszuführen. Dazu muß man einige Begriffe beherrschen, um die folgenden Kapitel auch verstehen zu können. Zuerst wird der Unterschied zwischen Offline- und Online-Algorithmus tabellarisch dargestellt, danach wenden wir uns der kompetitiven Analyse zu. Diese wird gebraucht, um die Güte bzw. Qualität der benutzten Online-Strategie festzustellen. Hierbei wird der Online Algorithmus mit dem optimalen Offline-Algorithmus verglichen.

Offline-Algorithmus:	**Online-Algorithmus:**
• alle benötigten Daten liegen zu Beginn der Berechnung vor	• relevante Daten treffen erst nach und nach im Laufe der Zeit ein • kennt nicht die zukünftigen Eingaben • weiß nicht, wieviele Daten eintreffen (Wann ist Schluß?) • die aktuelle Anfrage muß sofort bearbeitet werden (Kein Aufschieben möglich)

Kompetitiv:

OnlineAlgorithmus(A) ≤ Optimaler Offline-Algorithmus (OPT) + B

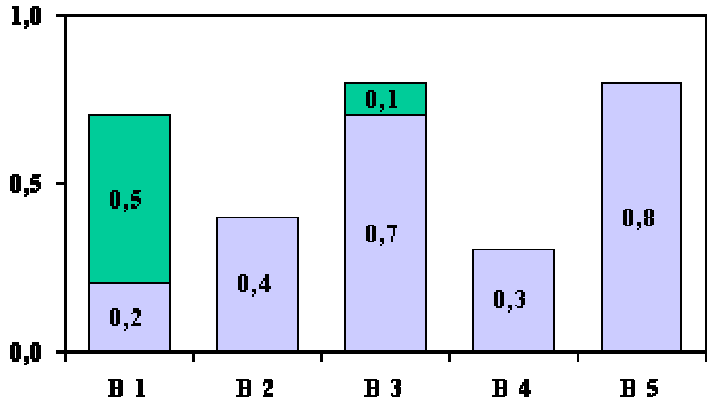
Ein Online-Algorithmus A heißt c-kompetitiv, wenn für alle möglichen Eingaben die von A berechnete Lösung nicht schlechter als das c-fache einer optimalen Offline-Lösung (OPT) ist.

top

2. Die Strategie Next Fit

Next Fit: Packe nächstes Objekt in dieselbe Kiste wie das Vorherige, wenn es dort noch hineinpasst, sonst schließe aktive Kiste und öffne eine neue.

Beispiel: 7 Objekte mit den Größen:
0.2, 0.5, 0.4, 0.7, 0.1, 0.3, 0.8

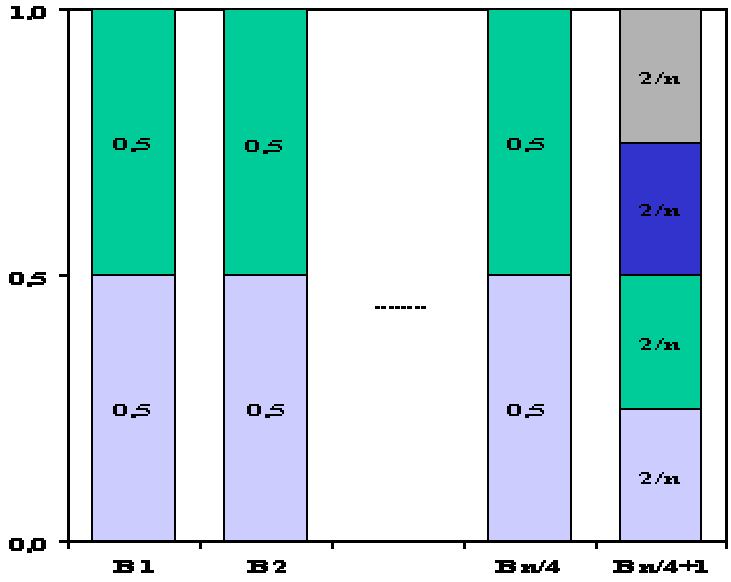


ist 2-kompetitiv:
Denn wenn in der Kiste i noch p Platz ist, ist in Kiste i+1 mindestens p Platz belegt.
Höchstens die Hälfte des Platzes bleibt unbenutzt.

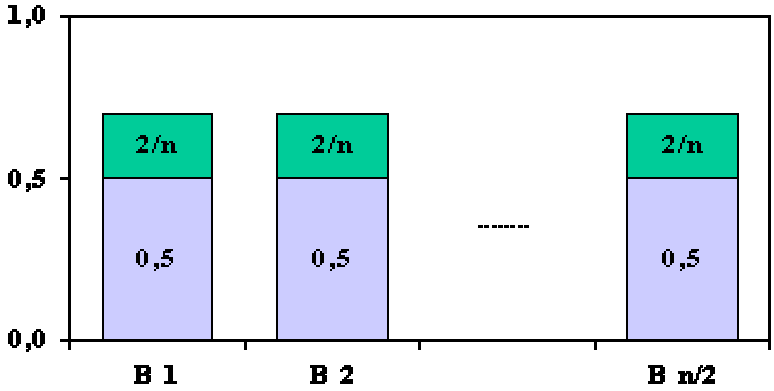
Im Beispiel werden die gleichen Werte wie im Beispiel der Offline-Version in Kap. 1 benutzt. Hier muß allerdings beachtet werden, dass die Folge nicht schon vorher vorliegt, sondern die Werte im Laufe der Zeit, d.h. ein Objekt nach dem anderen eintrifft. Nach Ausführung der Strategie kann man leicht sehen, dass Next Fit 5 Kisten benutzen mußte, um alle Objekte unterzubringen, während der optimale Offline-Algorithmus nur Kosten von 3 hat.

2.1. Next Fit kann diese Schranke wirklich erreichen

Gegeben: Inputfolge I mit Länge n; n = 0 mod 4
Inputfolge: 0.5, 2/n, 0.5, 2/n, ..., 0.5, 2/n



Next Fit liefert:



OPT(I) = n/4 + 1 NF (I) = n/2

Einsetzen:
OPT(I) = (NF/2) + 1
OPT(I) - 1 = NF/2
2 * OPT(I) - 2 = NF

Also:

NF(I) = 2 OPT(I) - 2

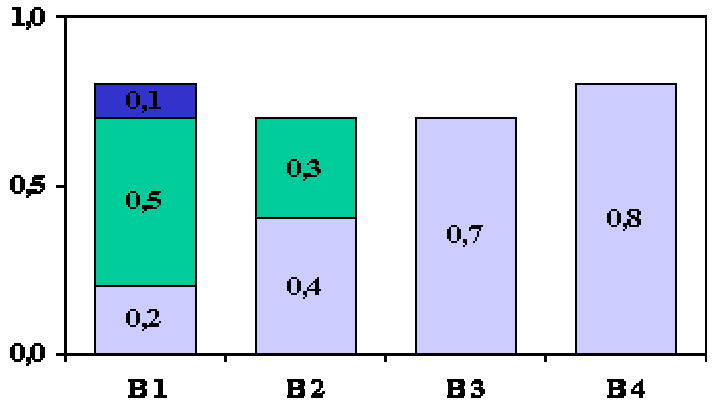
Um zu zeigen, dass NF diese Schranke auch wirklich erreichen kann, geben wir dem Algorithmus eine alternierende Folge von zwei Elementen 0.5 und 2/n. Dabei muß die Länge der Inputfolge durch 4 teilbar sein. Das hat den Vorteil, dass man bei OPT sehr schnell sehen kann, wie viele Kisten mit dem Wert 0.5 doppelt belegt werden, nämlich genau n/4. Die nachfolgende Kiste n/4+1 ist komplett mit den 2/n Objekten gefüllt.
Nehmen wir an, wir haben eine Input-Länge von 16, so liegen acht 0.5 und acht 2/n-Objekte vor, die abwechselnd eintreffen. Somit wären 16/4 = 4 Kisten mit jeweils zwei 0.5-Objekten belegt und die 16/4+1 = 5. Kiste wäre mit acht 2/16 = 1/8 Objekten gefüllt. So wird jede Kiste bis 1 beladen, d.h. der Platz wird optimal genutzt.
Next Fit hat das Problem, dass es maximal jeweils zwei Objekte in eine Kiste packen kann, nämlich immer die Größen 0.5 und 2/n, wie man leicht der obigen Grafik entnehmen kann. Folglich benötigt NF für die gegebene Inputfolge n/2 Kisten. OPT hat also die Kosten n/4+1 und NF n/2. Nun setzen wir NF in OPT ein und erhalten durch Umformung den Ausdruck NF(I) = 2 OPT(I) - 2 mit der relevanten Multiplikation 2 OPT. Somit ist bewiesen, dass NF die Schranke erreicht.

top

3. Die Strategie First Fit

First Fit: Lege nächstes Objekt in die erste Kiste von links, in die es hineinpasst.

Beispiel: 7 Objekte mit den Größen:
0.2, 0.5, 0.4, 0.7, 0.1, 0.3, 0.8



ist 2-kompetitiv:
Denn bis auf höchstens eine sind alle benutzten Kisten mindestens halbvoll.
Jedes Objekt ≤ 0.5 kann in eine halbleere Kiste gepackt werden.

ist sogar 1.7-kompetitiv (ohne Beweis)

Zum Vergleich mit NF und OPT wurde wieder die gleiche Inputfolge gewählt. FF benutzt nun 4 Kisten, was um eine Kiste schlechter als OPT ist, jedoch auch um eine Kiste günstiger als NF ist.

top

4. Vergleich der Laufzeit NextFit - FirstFit

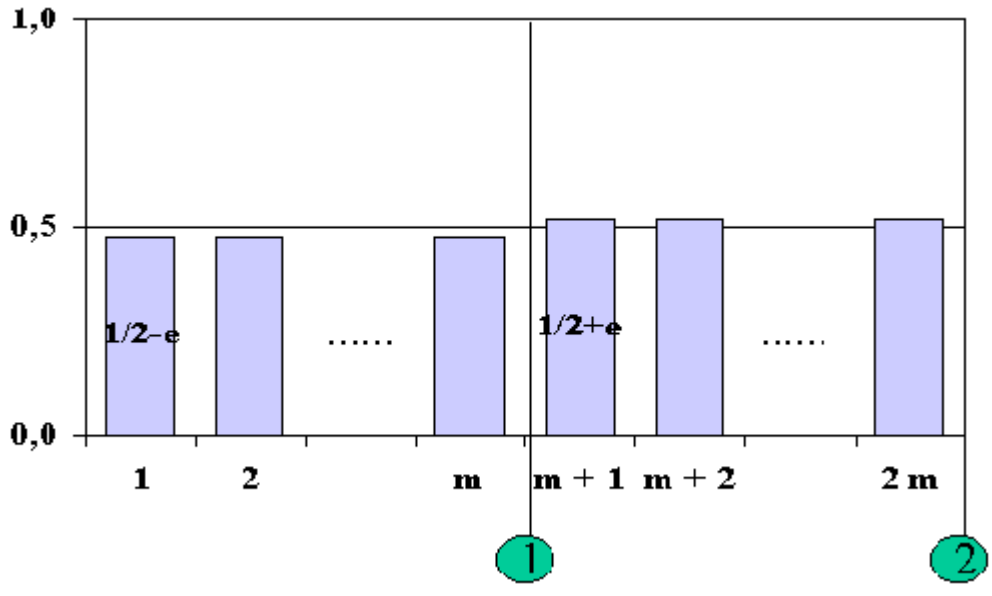
Next Fit:
Entweder passt ein Objekt in die aktuelle Kiste oder nicht. Falls nicht, wird eine neue Kiste geöffnet. Die Kosten steigen linear mit der Anzahl der Objekte. **O(n)**
First Fit:
Methode 1: Wenn jede Kiste, beginnend mit der ersten von links, abgefragt werden muß, ob ein Objekt hineinpasst, wird im worst case jede Kiste abgefragt. **O(n²)**
Methode 2: Balancierter Baum. Je nach Gewicht werden die Kisten in einem Baum angeordnet. **O(n log n)**

top

5. Jede Strategie ist mindestens 4/3-kompetitiv

Annahme:A benötigt stets 4/3 OPT Kisten

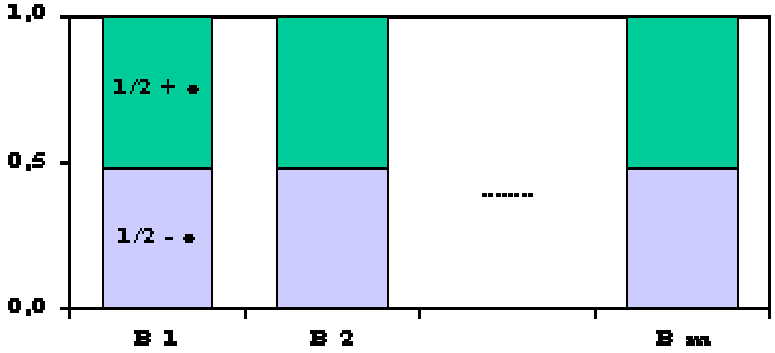
Gegeben: Inputfolge I mit Länge 2m:
- eine Folge aus m Objekten mit der Größe ½ - e
- eine Folge aus m Objekten mit der Größe ½ + e



Abschnitt 1:

OPT = m/2
Die Anzahl der von A benötigten Kisten sei b
b < c * m/2
b / m/2 < c

Abschnitt 2:



OPT = m
A ≥ 2m - b

Es gilt:

2m - b ≤ c * m
(2m - b) / m ≤ c

Insgesamt folgt:

c ≥ 4/3

Um zu zeigen, dass jede Strategie mindestens 4/3-kompetitv ist, betrachten wir eine Eingabe mit einer Anzahl von 2m Objekten, wobei die erste Hälfte aus Objekten der Größe ½-e (e := mathem. Epsilon) und die zweite Hälfte von Objekten den Wert ½+e annehmen.
Im ersten Bild aus Kap. 5 wird diese Folge grafisch dargestellt, wobei mit "1" und "2" Zeitabschnitte gesetzt werden. Wir betrachten zuerst den Zeitabschnitt 1, der nach der Eingabe der m ½-e Werte endet. Es ist offensichtlich, dass OPT hier alle Kisten doppelt belegt, also ist OPT = m/2. Die Anzahl von der verwendeten Online-Strategie sei mit b bezeichnet, also folgt hier b/(m/2) < c.
Kommen wir nun zum zweiten Abschnitt, der alle 2m Eingaben umfasst, also in der Grafik von Objekt 1 bis 2m geht. Was würde OPT tun? ½+e und ½-e ergibt 1, somit ist jede Kiste mit zwei Objekten belegt, also stehen die Kosten bei m. Da wir wissen, dass die Online-Strategie A im Abschnitt 1 b Kisten verbraucht hat (also maximal b Kisten doppelt belegt sein können), ziehen wir diesen Teil von 2m ab und erhalten 2m-b. Wendet man nun die kompetitive Analyse an, ergibt sich (2m-b)/m ≤ c. Insgesamt folgt dann c ≥ 4/3.

top

6. Die Offline-Version ist NP-hart

Bin Packing gehört zur Klasse der sogenannten NP-harten Probleme, d.h. ein Algorithmus, der in polynomieller Zeit eine optimale Lösung berechnet, ist nicht bekannt.

top

Links für Binpacking-Animationen:
JCAT - Binpacking und andere Animationen
Binpacking (First Fit)

Quellen:
Garey, Johnson: Computers and Intractability, S. 124-126
Fiat, Woeginger (eds.): Online Algorithms, S. 150
Ottmann, Th.: Online-Algorithmen, *Script*
Speckenmeyer, E.,: Online-Algorithmen, *Script*
Albers, Susanne: Online-Algortihmen, *Script*

Best view with:
Internet Explorer 4.0 or higher
Resolution: 1024x768
Last Modification: 28.10.2000

top