CS 663: Pattern Matching Algorithms

Scribe: Chen Jiang 11/29/2010

PTAS for Bin-Packing

1. Introduction

The Bin-Packing problem is NP-hard. If we use approximation algorithms, the Bin-Packing problem could be solved in polynomial time. For example, the simplest approximation algorithm is the First-fit algorithm, which solves the Bin-Packing problem in time $O(n\log n)$.

We use the **approximation factor** to determine how good our approximation algorithm is. Let $A(I)$ be the number of bins required by the approximation algorithm, and let $OPT(I)$ be the optimal number of required bins for input I. We say that algorithm A has approximation factor C if for every input I:

$$A(I) \le C \cdot OPT(I) \qquad (C \ge 1)$$

This inequality means that the approximation algorithm would not use more than C times the optimal number of bins, which is also the upper bound of the approximation algorithm. Obviously, the closer C is to 1, the better the approximation.

Claim: The Bin-Packing problem has a **PTAS (Polynomial Time Approximation Scheme)**. I.e., Given $\forall \varepsilon > 0$, one can always produce approximation algorithm whose (1) *time* is polynomial in n and ε; and (2) whose *approximation factor* is,

$$A(I) \le (1+\varepsilon)OPT(I)$$

2. Special Cases for Bin-Packing

Before proving that the Bin-Packing problem has a PTAS, first consider two Special Cases for Bin-Packing to get some interesting conclusions.

2.1 Case 1: All item sizes less than δ

Claim 1:

If item size $u_i < \delta$ for i=1 to n, then

$$FF(I) \le (1+2\delta)OPT(I)+1$$

$FF(I)$ = *the number of bins of First-fit algorithm.*

Proof

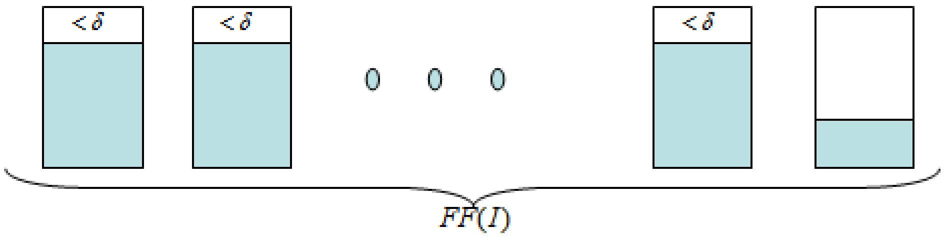
① if $\delta \geq 1/2$

$\because FF(I) \leq 2OPT(I)+1$ (Proved by First-fit)

and $\delta \geq 1/2 \Rightarrow 2 \leq 1+2\delta$

$\therefore FF(I) \leq 2OPT(I)+1 \leq (1+2\delta)OPT(I)+1$

② if $\delta < 1/2$



The above figure shows the situation at the end of running the First-fit. Since all item sizes are less than δ, there would be less then δ empty space in each bin (except, possibly, for the last bin).

So, the filled part (blue) = $\sum_{i=1}^{n} u_i \geq (FF(I)-1)(1-\delta)$, $FF(I)-1$ is the number of bins, $1-\delta$ is the smallest empty space within any bin.

Another way of thinking $\sum_{i=1}^{n} u_i$ is that suppose we could smash all the items & blend them, and then put them into bins without leaving any spare space. $\left\lceil \sum_{i=1}^{n} u_i \right\rceil$ can not be more than $OPT(I)$.

Therefore, $(FF(I)-1)(1-\delta) \leq \sum_{i=1}^{n} u_i \leq OPT(I)$

$\Rightarrow FF(I)-1 \leq OPT(I)/(1-\delta)$

But $1/(1-\delta) \leq 1+2\delta$,

$\because (1+2\delta)(1-\delta) = 1+\delta(1-2\delta)$

and $\delta < 1/2 \;\Rightarrow\; 0 < 1-2\delta \;\Rightarrow\; (1+2\delta)(1-\delta) \geq 1$

$\therefore FF(I)-1 \leq OPT(I)/(1-\delta) \leq (1+2\delta)OPT(I) \Rightarrow FF(I) \leq (1+2\delta)OPT(I)+1$

Claim 1 proved.

2.2 Case 2: There are only k different sizes of items

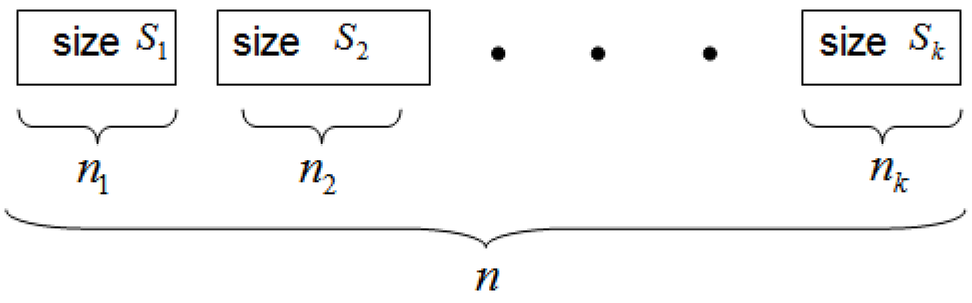
Claim 2:
If there are only k different sizes of items,

$OPT(I)$ *could be found deterministically in time* $O(n^{2k+1})$ *without using approximation algorithm.*

Proof

① Observation: The number of subsets of *n* elements is exponential (2^n) in the general case, while the number of subset is polynomial $O(n^k)$ in the case where *k* is fixed.

Why?



Sort the items by size, then divide the *n* items into *k* groups.
Denote a subset by a *k*-tuple of the number of elements of each size in the subset.

$$\langle i_1, i_2 \ldots i_k \rangle$$

Example
Suppose n items {1/2, 1/4, 1/3, 1/7, 1/9, 1/2, 1/3, 1/4, 1/9}
k(=5) groups {1/2, 1/4, 1/3, 1/7, 1/9}
We could denote subset {1/2, 1/4, 1/3, 1/9, 1/9} by <1, 1, 1, 0, 2>, where 2 represents the number of times 1/9 appeared. Obviously, all of these numbers cannot greater than *n*. So we have $O(n^k)$ subsets.

② Compute $OPT(I)$

Using dynamic programming.

Find the subsets that can be packed in 1 bin
Find the subsets that can be packed in <u>2 bins</u> $\Rightarrow$ Observation: each of these subsets must be composed of two subsets, each of which can be packed in 1 bin
…
…
Find the subsets that can be packed in *OPT(I)* bins

Since $OPT(I) \le n$. There are *n* iterations at most, and each iterations has $O(n^k)$ subsets.

Implementation

Step 1. Sort the subsets by the sum of their sizes. Those with sum not exceeding 1 can be packed in a single bin. Call them 1-bin subsets.

Step 2. For all pairs of 1-bin subsets, consider their union. Every union that is not a 1-bin subset is a 2-bin subset.

…

Step i+1. For all pairs consisting of a 1-bin subset and an *i*-bin subset, consider their union. If it is not an *i*-bin subset or less, then it is an *(i+1)*-bin subset.

If the entire set is packed – done, else continue to next step.

Time

We consider all pairs in each iteration, thus time is $O((n^k)^2) = O(n^{2k})$ per iteration. Since there are no more than *n* iterations, total time is $O(n^{2k+1})$.

Claim 2 proved.

3. Prove PTAS for Bin-Packing

3.1 Special cases

To prove PTAS for Bin-Packing problem, first consider two special cases.

Given $I = \{s_1, s_2 ... s_n\}$ and ε

3.1.1 Case 1: all item sizes are no more then $\varepsilon / 2$

Claim 3:

If item size $u_i \le \varepsilon / 2$ *for i=1 to n,*

$$A(I) \le (1+\varepsilon)OPT(I) + 1$$

Proof

Using Claim 1 directly.

3.1.2 Case 2: all item sizes are more then $\varepsilon / 2$

Claim 4:

If item size $u_i > \varepsilon / 2$ *for i=1 to n,*

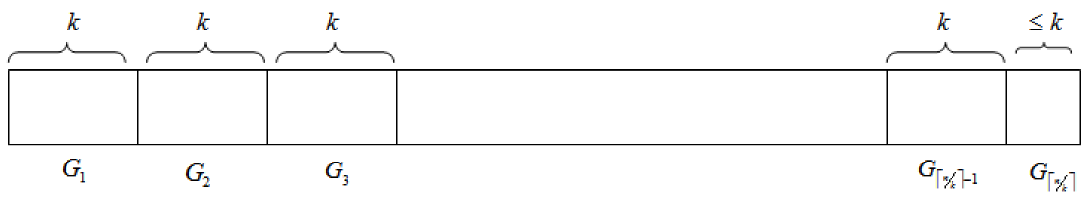
There exists a packing algorithm PA with approximation factor

$$PA(I) \le (1+\varepsilon)OPT(I)+1$$

Proof

① Fix k (we will later see how to choose it), then consider the following approximation algorithm.

Step 1. Sort I in non-decreasing order. Split into $\left\lceil \frac{n}{k} \right\rceil$ groups of k elements each.



Step 2. Pack the first group G_1 in at most k bins, because we have k elements in G_1

Step 3. Construct set I\`. Discard G_1 first, and then change all numbers in each group to largest number in group.

Step 4. Find $OPT(I`)$ by using the method in 2.2.

Time

1. Sorting: $O(n \log n)$

2. Packing first group: $O(n)$

3. Constructing I\`: $O(n)$

4. Finding *OPT(I\`)*: $O(n^{2\lceil n/k \rceil + 1})$. Since we divide I into $\left\lceil \frac{n}{k} \right\rceil$ groups, there are no more than $\left\lceil \frac{n}{k} \right\rceil$-1 different sizes of items in I\`, so that we could use Claim 2.

Conclude: Time $O(n^{2\lceil n/k \rceil + 1})$

Approximation Factor

Since we used k bins to pack the first group and *OPT(I\`)* bins to pack the rest of groups, we have approximation factor:

$$PA(I) = OPT(I`) + k$$

② Lemma: $OPT(I`) \leq OPT(I)$

Proof

Construct set I\`\`. Discard the last group $G_{\lceil n/k \rceil}$ first, and then change all numbers in each group to the smallest number in group.

Example

	G1	G2	G3	G4
I	10 9 8	7 7 6	6 6 5	3 2
I``	8 8 8	6 6 6	5 5 5	discarded
I`	discarded	7 7 7	6 6 6	3 3

Clearly $OPT(I``) \le OPT(I)$, because there are less elements in I`` and the size of elements are smaller. Also $OPT(I`) \le OPT(I``)$, because if compare G$_i$ in set I`` with G$_{i+1}$ in set I` (showed in red arrow), we see that (1) all the size of elements in *I*`s group are *no more than* those in *I*``s group; and (2) the number of elements in *I*`s group is *no more than* those in *I*``s group.

Conclude, $OPT(I`) \le OPT(I``) \le OPT(I)$

③ Choose *k*

Approximation Factor

$\because PA(I) = OPT(I`) + k$

And $OPT(I`) \le OPT(I)$

$\therefore PA(I) \le OPT(I) + k$

Now choose $k = \left\lceil \varepsilon \sum_{i=1}^{n} S_i \right\rceil$

Since $\sum_{i=1}^{n} u_i \le OPT(I)$ is proved in 2.1,

$\sum_{i=1}^{n} u_i \le OPT(I) \Rightarrow k \le \varepsilon \cdot OPT(I) + 1$

$\Rightarrow A(I) \le OPT(I) + k \le (\varepsilon + 1)OPT(I) + 1$

Therefore,

$$PA(I) \le (\varepsilon + 1)OPT(I) + 1$$

Time

$$\frac{n}{k} = \frac{n}{\left\lceil \varepsilon \sum_{i=1}^{n} S_i \right\rceil} \leq \frac{n}{\varepsilon \sum_{i=1}^{n} S_i}$$

But $s_i > \varepsilon / 2$ for *i=1 to n* , so:

$$\frac{n}{k} \leq \frac{n}{\varepsilon \sum_{i=1}^{n} S_i} \leq \frac{n}{\varepsilon \sum_{i=1}^{n} \frac{\varepsilon}{2}} = \frac{2}{\varepsilon^2}$$

Therefore, Running Time $O(n^{2\frac{2}{\varepsilon^2}+1}) = O(n^{\frac{4}{\varepsilon^2}+1})$

Since ε is given (fixed), this packing algorithm is polynomial.

Claim 4 proved.

3.2 General case

Implementation

Step 1. Split items into two sets: small ($s_i \leq \varepsilon / 2$) and large ($s_i > \varepsilon / 2$).

Step 2. Handle large items as in Claim 4.

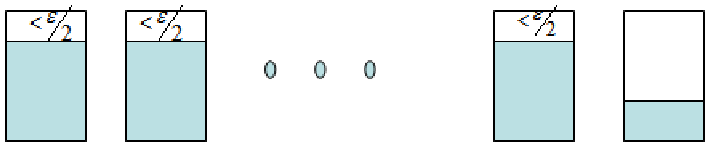
Step 3. Use FF to pack small items into remaining spaces of large item bins. When all such space is used, open new bins using FF.

Time

Obviously Polynomial $O(n^{\frac{4}{\varepsilon^2}+1})$

Approximation Factor

Case 1: All large item bins are filled, and new ones opened.



This is the same case as Claim 1, because we are using FF to pack the small elements of the set ($s_i \leq \varepsilon / 2$). So the Approximation Factor is $A(I) \leq (1+\varepsilon) OPT(I) + 1$

Case 2: Not all large item bins are filled.

Use conclusion of Claim 4, $A(I) = PA(I) \leq (1+\varepsilon) OPT(I') + 1 \leq (1+\varepsilon) OPT(I) + 1$

3.3 Conclusion

For Bin-Packing problem, Given $\forall \varepsilon$, we could always propose a Polynomial Time

Approximation Scheme whose approximation factor is $(1+\varepsilon)OPT(I)$, time is $O(n^{\frac{4}{\varepsilon^2}+1})$.

This conclusion also shows that if we want to get more close to optimal number of bins, we need to pay more time. For example, if we choose ε=1/100, time $O(n^{40001})$ would be awful although it is polynomial.