CMPUT 675: Approximation Algorithms **Fall 2014**

Lecture 8 (Sep 19): Bin Packing

Lecturer: Zachary Friggstad *Scribe: Yifeng Zhang*

8.1 Bin Packing

Definition 1 *In the* BIN PACKING *problem, we are given a set of items I (which we identify with $\{1, \ldots, n\}$) with sizes $0 \leq s_i \leq 1$, $i \in I$. The goal is to pack I into the fewest bins possible so that the items in any bin have total size ≤ 1. More explicitly, we should partition I into the fewest parts possible so that the sum of item sizes in each part is at most 1.*

In this lecture, we will see that there is no PTAS for this problem. However, we can get the next best thing: for any constant $\epsilon > 0$ there a polynomial time algorithm that uses at most $(1 + \epsilon) \cdot OPT + 1$ bins.

Definition 2 *For a subset $B \subseteq I$, we let* $\text{SIZE}(B) = \sum_{i \in B} s_i$.

8.1.1 A Simple ~ 2-Approximation

Algorithm 1 describes a greedy algorithm that tries to add the items one at a time. When adding an item, we only create a new bin if the item does not fit in any of the currently used bins.

Algorithm 1 Bin Packing Heuristic

Input: A set of all items I, $s_i \leq 1, \forall i \in I$.
Output: A partition $\{B_i\}$ of I where $\text{SIZE}(B_i) \leq 1$ for each i.

```
$b \leftarrow 0$
for each $i \in I$ do
  if $s_i + \text{SIZE}(B_j) \leq 1$ for any $1 \leq j \leq b$ then
    $B_j \leftarrow B_j \cup \{i\}$
  else
    $b \leftarrow b + 1$
    $B_b \leftarrow \{i\}$   {Create a new bin}
  end if
end for
return $B_1, \ldots, B_b$
```

Theorem 1 *Algorithm 1 uses at most $2 \cdot OPT + 1$ bins.*

Proof. First, observe that $OPT \geq \text{SIZE}(I)$ because each bin in the optimum solution can hold a total size of at most 1.

Let $B_1, \ldots, B_b$ be the bins returned by the algorithm. We claim that $\text{SIZE}(B_j) \geq 1/2$ for at least $b - 1$ of the bins $B_1, \ldots, B_b$. Otherwise, we would have $\text{SIZE}(B_j), \text{SIZE}(B_{j'}) < 1/2$ for some two indices $j < j'$. Consider the moment when bin $B_{j'}$ was created by the algorithm and say that item i was added to $B_{j'}$ at this time. Since $\text{SIZE}(B_{j'}) \leq 1/2$ at the end of the algorithm then $s_i \leq 1/2$. But then $s_i + \text{SIZE}(B_j) \leq 1$ so i could have been added to B_j, contradicting the fact that the algorithm created $B_{j'}$ to accommodate i.

Finally, since all but one of the bins are at least half full we have $(b-1)/2 \leq \text{SIZE}(I) \leq OPT$. Rearranging shows $b \leq 2 \cdot OPT + 1$. ■

The *First-Fit Decreasing* heuristic is essentially the same as Algorithm 1, except we consider the items in decreasing order of size and if the item fits in one of the current bins, we place it in the bin B_j with least index j. Johnson et al. show that this heuristic uses at most $11/9 \cdot OPT + 4$ bins [J+74].

8.1.2 A Constant Hardness Lower Bound

Now we show that there is no PTAS for Bin Packing problem. Bin Packing problem is related to Partition problem which is well-known to be NP-complete.

Definition 3 *In the* Partition *problem, we are given positive integers a_1,a_2,...,a_n. The goal is to determine if we can partition the set of indices $\{1, 2, ..., n\}$ into sets S and T such that $\sum_{i \in S} a_i = \sum_{i \in T} a_i$.*

Notice we can make this problem into Bin Packing problem by normalizing a_i values so their sum is 2. The items can be packed into two bins if and only if there is a solution to the original Partition instance. If there is no solution to the Partition problem, then at least three bins are required. The following theorem follows immediately from these facts.

Theorem 2 *There is no $c < \frac{3}{2}$-approximation unless* $\text{P} = \text{NP}$.

8.1.3 An Aysmptotic PTAS

Definition 4 *An aysmptotic polynomial-time approximation scheme is a family of algorithms parameterized by $\epsilon > 0$ where, for each constant $\epsilon > 0$, the algorithm finds a solution with cost at most $(1 + \epsilon) \cdot OPT + c$ in polynomial time where c is a constant independent of ϵ.*

In this section, we present an asymptotic PTAS for bin packing with constant $c = 1$.

Theorem 3 *For any constant $0 < \epsilon \leq 1$ we can efficiently find a solution with at most $(1 + \epsilon)OPT + 1$ bins in time $n^{O(1/\epsilon^2)}$.*

The idea is much like the PTAS for Minimizing Makespan on Identical Parallel Machines: ignore the small items, approximate the instance with large items, and then greedily add small items to this solution. The key difference is that we cannot naively scale the items down to solve via DP. If we did this and packed the bins according to these scaled sizes, then returning the items to their original size might result in overpacked bins.

The approach taken here is a bit more subtle, instead we will scale item sizes *up* so that a packing under the new sizes yields a packing under the old sizes. Of course, the concern here is that if we scale item sizes up then the optimum solution value might go up. We will scale them carefully (i.e. using *linear scaling* described below) to ensure the optimum solution value does not go up by much.

To begin, let $I_{large} = \{i \in I : s_i \geq \epsilon/2\}$ and let $I_{small} = I - I_{large}$.

Claim 1 *Given a packing of I_{large} into b bins, we can efficiently find a packing of I using* $\max\{b, (1+\epsilon)OPT+1\}$ *bins.*

Proof. Extending packing of the large items by adding the small items one at a time. Create a new bin only if none of the current bins can accommodate the item. This can be viewed as running Algorithm 1 on items I_{small}, starting with the given packing of I_{large} (rather than an empty collection of bins).

Case 1: If no new bins were created, we use b bins.

Case 2: Otherwise, say b' is the total number of bins used. Since $s_i < \epsilon/2$ for $i \in I_{small}$, we only create bins if the other bins contains total size $\geq 1 - \epsilon/2$. Therefore,

$$(b'-1)(1-\epsilon/2) \leq \text{SIZE}(I) \leq OPT.$$

This is because all bins except, perhaps, the last bin we created will contain total size of at least $1 - \epsilon/2$. This shows

$$b' \leq \frac{OPT}{1-\frac{\epsilon}{2}} + 1 \leq (1+\epsilon)OPT + 1$$

where we have used $1/(1-\epsilon/2) \leq 1 + \epsilon$ for any $0 < \epsilon \leq 1$.

■

Linear Grouping: For a given value k (which we will specify soon), create a new instance I' as follows. Order I_{large} in decreasing order of size so $s_1 \geq s_2 \geq \ldots \geq s_{n_\ell}$ where $n_\ell = |I_{large}|$. Create groups $G_1 = \{s_1, \ldots, s_k\}, G_2 = \{s_{k+1}, \ldots, s_{2k}\}, G_3 = \{s_{2k+1}, \ldots, s_{3k}\}, \ldots$ where all but, perhaps, the last group G_h have size k and G_h has size at most k. The new instance I' consists of items $\{k+1, k+2, \ldots, n_\ell\} = \cup_{j=2}^{h} G_h$. For an item $i \in I'$ that is in group, say, G_a, set $s'_i = \max\{s_i : i \in G_a\}$. This process is called *linear grouping.*

The following is clear from this process.

Lemma 1 *For each $i \in I'$, $s_{i-k} \geq s'_i \geq s_i$.*

For clarity, we will let $OPT(I_{large})$ denote the optimum solution value for instance I_{large}, $OPT(I)$ for the original instance I and $OPT(I')$ denote the optimum solution value for instance I'. Clearly, $OPT(I_{large}) \leq OPT(I)$.

Lemma 2 *For the instance I' with sizes s'_i obtained from I_{large} using linear grouping, $OPT(I') \leq OPT(I_{large}) \leq OPT(I') + k$. Finally, given a packing of I' into b bins we can efficiently find a packing of I_{large} into at most $b + k$ bins.*

Proof. For the first inequality, consider an optimum solution for I_{large}. Pack each item $i \in I'$ in the bin where $i - k \in I_{large}$ is packed. Since $s_{i-k} \geq s'_i$, this produces a feasible packing of I' using at most $OPT(I_{large})$ bins.

For the second part, consider a packing of I' into b bins. We obtain a packing of I_{large} by packing each of items $1, \ldots, k$ in their own bin and then packing the remaining items $i \geq k+1$ in the same bin as item $i \in I'$ is packed. Since $s_i \leq s'_i$, this produces a feasible packing of I_{large} using at most $b + k$ bins. ■

To complete the description of the asymptotic PTAS for Bin Packing, we use the linear grouping scheme to reduce to a problem we already know how to solve in polynomial time. To start, use the linear grouping scheme with $k := \lfloor \epsilon \cdot \text{SIZE}(I_{large}) \rfloor$.

We will assume $n_l > 2/\epsilon^2$, as otherwise there are a constant number of items in I_{large} and such instances can easily be solved optimally in constant time by brute force.

Let m be the number of distinct sizes in I'. Note that $m \leq h - 1$ where h is the number of groups created in the linear grouping scheme. Therefore

$$\begin{array}{rcll} m & = & h-1 & \\ & \leq & n/k & \\ & = & n/\lfloor \epsilon \cdot \text{SIZE}(I_{large}) \rfloor & \\ & \leq & n/\lfloor (\epsilon^2/2) \cdot n \rfloor & \text{(each } i \in I_{large} \text{ has } s_i \geq \epsilon/2) \\ & \leq & 2n/((\epsilon^2/2) \cdot n) & \text{(see below)} \\ & = & 4/\epsilon^2. & \end{array}$$

The last inequality holds because $n \geq 2/\epsilon^2$ and $\lfloor \alpha \rfloor \geq \alpha/2$ for any $\alpha \geq 1$.

Say these distinct items items sizes are $a_1, a_2, \ldots, a_m$. Any bin with items from I' can be identified with a tuple of nonnegative integers $(b_1, b_2, \ldots, b_m)$ where $\sum_{j=1}^m b_j \cdot a_j \leq 1$. Furthermore, since $a_j \geq \epsilon/2$ for each j, then the number of items in this bin is at most $2/\epsilon$, i.e. $\sum_{j=1}^m b_j \leq 2/\epsilon$.

Let $\mathcal{C}$ be the set of all nonzero tuples $(b_1, \ldots, b_m)$ that represent a bin of size at most 1. Since $0 \leq b_j \leq 2/\epsilon$ the the number of such tuples is at most $(2/\epsilon + 1)^m \leq (3/\epsilon)^{4/\epsilon^2}$.

For any tuples $(n_1, \ldots, n_m)$ of nonnegative integers let $f(n_1, \ldots, n_m)$ be the minimum number of bins required to pack the set of items consisting of n_j items of size a_j. Since the number of configurations $\mathcal{C}$ is constant, then the dynamic programming algorithm from the last lecture can be used to compute $f(\overline{n}_1, \ldots, \overline{n}_m)$ in $n^{O(1/\epsilon^2)}$ time where $\overline{n}_j$ is the number of items in I' having size a_j (view bins $\equiv$ machines and item sizes $\equiv$ processing times). Thus, we can compute $OPT(I')$ in polynomial time. Of course, an optimum packing of I' can be constructed through sufficient bookkeeping in the dynamic programming algorithm.

We summarize these argument below to complete the proof of our main statement.

Proof of Theorem 3. Compute an optimum packing of I' in time $n^{O(1/\epsilon^2)}$ using the dynamic programming routine. By Lemma 2, we can transform this to a packing of I_{large} using at most $b := OPT(I_{large}) + k \leq OPT(I) + \epsilon \cdot OPT(I)$ bins. By Claim 1, greedily packing the items of I_{small} into these bins, creating a new bin only when an item does not fit in any of the current bins, finds a solution using at most $\max\{b, (1+\epsilon) \cdot OPT(I) + 1\} = (1+\epsilon) \cdot OPT(I) + 1$ bins. ∎

This algorithm is a variant of the asymptotic PTAS for Bin Packing proposed by Fernandez de la Vega and Lueker [FL81]. Their algorithm is more refined and has a much better running time bound of $O(f(\epsilon) + \log(1/\epsilon) \cdot n)$ where $f(\epsilon)$ is a constant that depends only on ϵ. In essence, for constant ϵ this is a linear-time algorithm.

The best approximation for Bin Packing is very recent. Rothvoss demonstrated an algorithm that uses at most $OPT + O(\log OPT \cdot \log\log OPT)$ bins [R13]. Currently, the best lower bounds do not rule out the possibility of using only $OPT + 1$ bins.

The dynamic programming subroutine we used shows that if each item has size at least some constant and if there are a constant number of different item sizes, then the problem can be solve exactly. Relaxing the second constraint does not help (i.e. if each item is at least some constant but there can be many different item sizes) as the problem remains NP-hard. However, another very recent algorithm of Goemans and Rothvoss shows how to solve Bin Packing exactly if we are only guaranteed a constant number of different item sizes [GR14].

References

FL81 W. Fernandez de la Vega and G. L. Lueker, Bin packing can be solved within $1 + \epsilon$ in linear time, *Combinatorica*, 1:349–355, 1981.

GR14 M. X. Goemans and T. Rothvoss, Polynomiality for bin packing with a constant number of item types, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 830–839, 2014.

J+74 D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal on Computing*, 1974.

R13 T. Rothvoss, Approximating bin packing within $O(\log OPT \cdot \log\log OPT)$ bins, *Proceedings of IEEE Foundations of Computer Science*, 20–29, 2013.