1 Approximation schemes (Continued)

2 Bin packing (Continued)

During the last lecture, the bin packing problem was tackled first by FirstFit, which we showed to be a 2-approximation algorithm. We also described $A_\epsilon$, an exact algorithm which solves bin packing under two assumptions:

1. All items have at least size $\epsilon$
2. There are only $k$ different possible sizes (for some constant $k$).

Towards the end, we worked towards removing these two assumptions.

**Definition 1** (Bin packing problem). Given a set $S$ with $n$ items where each item $i$ has size $i \in (0, 1]$, find the minimum number of unit-sized (size 1) bins that can hold all $n$ items.

For any problem instance $I$, let $OPT(I)$ be an optimum bin assignment and $|OPT(I)|$ be the corresponding minimum number of bins required. One can see that $\sum_{i=1}^{n} size(i) \leq |OPT(I)|$.

2.1 Special case where items have sizes larger than $\epsilon$, for some $\epsilon > 0$

**Algorithm 1** PTAS-BinPacking ($I = S, \epsilon$)

\[ k \leftarrow \lceil \frac{1}{\epsilon^2} \rceil \]
\[ Q \leftarrow \lfloor n\epsilon^2 \rfloor \]

Partition $n$ items into $k$ non-overlapping groups, each with $Q$ items \> See Figure 1

for $i \in \{1, \ldots, k\}$ do

\[ k_{\text{max}} \leftarrow \max_{j \text{ in group } i} \text{size}(j) \]

for item $j$ in group $i$ do

\[ \text{size}(j) \leftarrow k_{\text{max}} \]

end for

end for

Denote the modified instance as $J$

return $A_\epsilon(J)$

\[ J_1 = J'_1 \]
\[ J_2 = J'_2 \]
\[ J \text{ rounds up} \]
\[ J_k = J'_k \]

\[ \leq Q \text{ items} \]
\[ \leq Q \text{ items} \]
\[ J' \text{ rounds down} \]
\[ \leq Q \text{ items} \]

**Figure 1**: Partition items into $k$ groups, each with at most $Q$ items. Label groups in ascending size ordering. $J$ rounds up item sizes, $J'$ rounds down item sizes.

Algorithm 1 pre-processes the sizes of a given input instance, then calls the exact algorithm $A_\epsilon$ to solve the modified instance. Since $J$ only rounds up sizes, $A_\epsilon(J)$ will yield a satisfying bin assignment for instance $I$, with possibly “spare slack”. For analysis, let us define another modified instance $J'$ as rounding down item sizes. Since we rounded down item sizes in $J'$, $|OPT(J')| \leq |OPT(I)|$. 

Lemma 2. $|OPT(J)| \leq |OPT(J')| + Q$

Proof. Label the $k$ groups in $J$ by $J_1, \ldots, J_k$ where the items in $J_i$ have smaller sizes than the items in $J_{i+1}$. Label the $k$ groups in $J'$ similarly. See Figure 1. For $i = \{1, \ldots, k-1\}$, since the smallest item in $J'_{i+1}$ has size larger to the largest item in $J_i$, any valid packing for $J'_i$ serves as a valid packing for the $J_{i-1}$. For $J_k$ (the largest group of $Q$ items), let us use separate bins (hence the additive $Q$ term). \hfill \Box

Lemma 3. $|OPT(J)| \leq |OPT(I)| + Q$

Proof. By Lemma 2 and the fact that $|OPT(J')| \leq |OPT(I)|$. \hfill \Box

Theorem 4. Algorithm 1 is an $(1 + \epsilon)$-approximation algorithm.

Proof. Assumption (1) tells us that all items have at least size $\epsilon$, so $|OPT(I)| \geq n \epsilon$. Then, $Q = [n \epsilon^2] \leq n \epsilon^2 \leq \epsilon \cdot OPT(I)$. By Lemma 2, $|OPT(J)| \leq (1 + \epsilon) \cdot |OPT(I)|$. \hfill \Box

2.2 Full PTAS for bin packing without assumptions

Algorithm 2 Full-PTAS-BinPacking($I = S, \epsilon$)

\begin{algorithmic}
  \State $\epsilon' \leftarrow \min\{\frac{1}{2}, \frac{\epsilon}{2}\}$ \Comment{See analysis why we chose such an $\epsilon'$}
  \State $X \leftarrow$ Items with size $< \epsilon'$ \Comment{Ignore small items to fulfill assumption of sizes $\geq \epsilon'$}
  \State $P \leftarrow$ PTAS-BinPacking($S \setminus X, \epsilon'$) \Comment{By theorem 4, $|P| = (1 + \epsilon') \cdot |OPT(S \setminus X)|$}
  \State $P' \leftarrow$ Using FirstFit, add items in $X$ to $P$ \Comment{Handle small items}
  \State \Return Resultant packing $P'$
\end{algorithmic}

Theorem 5. Algorithm 2 uses at most $(1 + \epsilon)|OPT(I)| + 1$ bins

Proof. If FirstFit does not open a new bin, the theorem trivially holds. Suppose FirstFit opens a new bin (using $m$ bins in total), then we know that at least $(m - 1)$ bins are strictly more than $(1 - \epsilon')$-full.

\[
|OPT(I)| \geq \sum_{i=1}^{n} \text{size}(i) \quad \text{Lower bound on } |OPT(I)|
\]

\[
> \frac{n}{m - 1} (1 - \epsilon') \quad \text{From above observation}
\]

Hence,

\[
m < \frac{|OPT(I)| + 1}{1 - \epsilon'} \quad \text{Rearranging}
\]

\[
< |OPT(I)| \cdot (1 + 2\epsilon') + 1 \quad \text{Since } \frac{1}{1 - \epsilon'} \leq 1 + 2\epsilon', \text{ for } \epsilon' \leq \frac{1}{2}
\]

\[
\leq (1 + \epsilon) \cdot |OPT(I)| + 1 \quad \text{By choice of } \epsilon' = \min\{\frac{1}{2}, \frac{\epsilon}{2}\}
\]

\hfill \Box

3 Minimum makespan scheduling

Definition 6 (Minimum makespan scheduling problem). Given $n$ jobs $I = \{p_1, \ldots, p_n\}$, find an assignment of jobs to $m$ identical machines such that the completion time (called the makespan) is minimized.

For any problem instance $I$, let $OPT(I)$ be an optimum job assignment and $|OPT(I)|$ be the corresponding makespan. One can see that:

- $p_{max} = \max_{i \in \{1, \ldots, n\}} p_i \leq |OPT(I)|$
- $\frac{1}{m} \sum_{i=1}^{n} p_i \leq |OPT(I)|$

Denote $L(I) = \max\{p_{max}, \frac{1}{m} \sum_{i=1}^{n} p_i\}$. We see that $L(I) \leq |OPT(I)| \leq p_{max} + \frac{1}{m} \sum_{i=1}^{n} p_i \leq 2L(I)$.

Remark To prove approximation factors, it is often useful to relate to lower bounds of $|OPT(I)|$. 

\hfill 2
Example Suppose we have 7 jobs \( I = \{p_1 = 3, p_2 = 4, p_3 = 5, p_4 = 6, p_5 = 4, p_6 = 5, p_7 = 6\} \) and \( m = 3 \) machines. Then, the lower bound on the makespan is \( L(I) = \max\{6, 11\} = 11 \). This is achievable via \( M_1 = \{p_1, p_2, p_5\}, M_2 = \{p_3, p_4\}, M_3 = \{p_6, p_7\} \).

We now describe a simple greedy algorithm (Algorithm 3) due to Graham [Gra66] and show that it is a 2-approximation algorithm. With slight modifications, we improve it to a \( \frac{3}{2} \)-approximation algorithm (Algorithm 4). Finally, we end off the section with a PTAS for minimum makespan scheduling.

### 3.1 Greedy approximation algorithms

**Algorithm 3** Graham \((I = \{p_1, \ldots, p_n\}, m)\)

1. \( M_1, \ldots, M_m \leftarrow \emptyset \) \(^{\triangleright} \) All machines are initially free
2. For \( i \in \{1, \ldots, n\} \) do
   1. \( j \leftarrow \text{argmin}_{j \in \{1, \ldots, m\}} \sum_{p \in M_j} p \) \(^{\triangleright} \) Pick the least loaded machine
   2. \( M_j \leftarrow M_j \cup \{p_i\} \) \(^{\triangleright} \) Add job \( i \) to this machine
3. Return \( M_1, \ldots, M_m \)

**Theorem 7.** Graham (Algorithm 3) is a 2-approximation to minimum makespan scheduling.

**Proof.** Consider the last job that finishes running. Suppose it takes time \( p_{\text{last}} \) and it was assigned to machine \( j \) whereby \( \sum_{p \in M_j} p = t \). Then, \( |\text{Graham}(I)| = t + p_{\text{last}} \). As Graham assigns greedily to the least loaded machine, all machines take at least \( t \) time, so \( t \cdot m \leq \sum_{i=1}^n p_i \leq m \cdot |\text{OPT}(I)| \). Since \( p_{\text{last}} \leq p_{\text{max}} \leq |\text{OPT}(I)| \), \( |\text{Graham}(I)| = t + p_{\text{last}} \leq 2 \cdot |\text{OPT}(I)| \). \( \square \)

Recall the example where \( I = \{p_1 = 3, p_2 = 4, p_3 = 5, p_4 = 6, p_5 = 4, p_6 = 5, p_7 = 6\} \) and \( m = 3 \). Graham will schedule \( M_1 = \{p_1, p_4\}, M_2 = \{p_2, p_5, p_7\}, M_3 = \{p_6, p_7\} \), yielding a makespan of 14. As expected, \( 14 = |\text{Graham}(I)| \leq 2 \cdot |\text{OPT}(I)| = 22 \).

**Remark** The approximation for Graham is loose because we have no guarantees on \( p_{\text{last}} \) beyond \( p_{\text{last}} \leq p_{\text{max}} \). This motivates us to order the job timings in descending order (see Algorithm 4).

**Lemma 8.** Let \( p_{\text{last}} \) be the last job that finishes running. If \( p_{\text{last}} > \frac{1}{3} |\text{OPT}(I)| \), then \( |\text{ModifiedGraham}(I)| = |\text{OPT}(I)| \).

3
Suppose, for a contradiction, that \( |\text{ModifiedGraham}(I)| > |OPT(I)| \). Then, there exists a sequence of jobs with descending sizes \( I = \{p_1, \ldots, p_n\} \) such that the last smallest job \( p_n \) causes \( \text{ModifiedGraham}(I) \) to have a makespan larger than \( OPT(I) \). That is, \( p_{\text{last}} = p_n \) and \( |\text{ModifiedGraham}(I \setminus \{p_n\})| \leq |OPT(I)| \). Let \( \mathcal{C} \) be the configuration of machines after \( \text{ModifiedGraham} \) assigned \( \{p_1, \ldots, p_{n-1}\} \).

**Observation 1** In \( \mathcal{C} \), each machine has either 1 or 2 jobs.

If there exists machine \( M_i \) with \( \geq 3 \) jobs, \( M_i \) will take \( > |OPT(I)| \) time because all jobs take \( > \frac{1}{3} \cdot |OPT(I)| \) time. This contradicts the assumption \( |\text{ModifiedGraham}(I \setminus \{p_n\})| \leq |OPT(I)| \).

Let us denote the jobs that are alone in \( \mathcal{C} \) as heavy jobs, and the machines they are on as heavy machines.

**Observation 2** In \( OPT(I) \), all heavy jobs are alone.

Assigning \( p_n \) to any machine (in particular, the heavy machines) in \( \mathcal{C} \) causes the makespan to exceed \( |OPT(I)| \). Since \( p_n \) is the smallest job, no other job can be assigned to the heavy machines otherwise \( |OPT(I)| \) cannot attained by \( OPT(I) \).

Suppose there are \( k \) heavy jobs occupying a machine each in \( OPT(I) \). Then, there are \( 2(m-k)+1 \) jobs (two non-heavy jobs per machine in \( \mathcal{C} \), and \( p_n \) ) to be distributed across \( m-k \) machines. By pigeonhole principle, at least one machine \( M^* \) will get \( \geq 3 \) jobs in \( OPT(I) \). However, since the smallest job \( p_n \) takes \( > \frac{1}{3} \cdot |OPT(I)| \) time, \( M^* \) will spend \( > |OPT(I)| \) time. Contradiction.

**Theorem 9.** \( \text{ModifiedGraham} \) (Algorithm 4) is a \( \frac{4}{3} \)-approximation to minimum makespan scheduling.

Proof. Case 1: \( p_{\text{last}} \leq \frac{1}{3} \cdot |OPT(I)| \)

By similar arguments as per Theorem 7, \( |\text{ModifiedGraham}(I)| = t + p_{\text{last}} \leq \frac{4}{3} \cdot |OPT(I)| \)

Case 2: \( p_{\text{last}} > \frac{1}{3} \cdot |OPT(I)| \)

By Lemma 8, \( |\text{ModifiedGraham}(I)| = |OPT(I)| \).
3.2 PTAS for minimum makespan scheduling

Recall that any makespan scheduling instance $(I, m)$ has a lower bound $L(I) = \max\{p_{\text{max}}, \frac{1}{m} \sum_{i=1}^{n} p_i\}$. We know that $|OPT(I)| \in [L(I), 2L(I)]$. Let $Bin(I, t)$ be the minimum number of bins of size $t$ that can hold all jobs. By associating job times with sizes, and scaling bin sizes up by a factor of $t$, in $O(\log n)$ rounding remaining jobs take $Bin(I, t)$ that $|\cdot|$.

We know that $\sum_{i=1}^{n} p_i = m$, and that $Bin(I, t)$ is monotonically decreasing. To get a $(1 + \epsilon)$-approximate schedule, it suffices to find a $t \leq (1 + \epsilon) \cdot |OPT(I)|$ such that $Bin(I, t) \leq m$.

**Algorithm 5** PTAS-Makespan$(I = \{p_1, \ldots, p_n\}, m)$

1. $L = \max\{p_{\text{max}}, \frac{1}{m} \sum_{i=1}^{n} p_i\}$
2. for $t \in \{L, L + \epsilon, L + 2\epsilon, \ldots, 2L\}$ do
   1. $X \leftarrow$ Jobs with sizes $\leq \epsilon t$ (Remaining jobs have sizes $\in (\epsilon t, t]$)
   2. $I' \leftarrow I \setminus X$ (Ignore small jobs)
   3. $h \leftarrow \lceil \log_{1+\epsilon}(\frac{1}{\epsilon t}) \rceil$ (Partition $(\epsilon t, t]$ into powers of $(1 + \epsilon)$: $t \cdot (1, 1 + \epsilon), \ldots, (1 + \epsilon)^h = \epsilon^{-1}$)
   4. for $p_i \in I'$ do
      1. $k \leftarrow \min_{j \in \{0, \ldots, h\}} \{p_i \geq t(1 + \epsilon)^j\}$ (Find lowest power of $(1 + \epsilon)$ for rounding down)
      2. $p_i \leftarrow t(1 + \epsilon)^j$ (Round down job sizes)
   3. end for
3. $P \leftarrow \mathcal{A}_e(I')$ (See Section 3.2.1 in Lecture 2 notes for $\mathcal{A}_e$, but with size $t$ bins)
4. $\alpha(I, t, \epsilon) \leftarrow$ Use bins of size $t(1 + \epsilon)$ to emulate $P$ (Use extra $(1 + \epsilon)$ buffer)
5. $\alpha(I, t, \epsilon) \leftarrow$ Using FirstFit, add items in $X$ to $\alpha(I, t, \epsilon)$ (Handle small items)
6. if $\alpha(I, t, \epsilon)$ uses $\leq m$ bins then
   1. return Assign jobs to machines according to bin assignment $\alpha(I, t, \epsilon)$ (Since $|OPT(I)| \in [L, 2L]$, this will occur at some point)
   2. end if
7. end for

Given $t$, Algorithm 5 transforms a makespan scheduling instance into a bin packing instance, then solves for an approximate bin packing to yield an approximate scheduling. Let $\alpha(I, t, \epsilon)$ as the number of bins used by Algorithm 5.

**Lemma 10.** For any $t > 0$, $\alpha(I, t, \epsilon) \leq Bin(I, t)$.

**Proof.** If FirstFit does not open a new bin, then $\alpha(I, t, \epsilon) \leq Bin(I, t)$ since $\alpha(I, t, \epsilon)$ uses additional $(1 + \epsilon)$ buffer. If FirstFit opens a new bin (say, totaling $b$ bins), then there are at least $(b-1)$ produced bins from $\mathcal{A}_e$ (exact solving on rounded down non-small items) that are more than $(t(1 + \epsilon) - t) = t$-full. Hence, any bin packing algorithm must use strictly more than $\frac{(b-1)t}{t} = b - 1$ bins.

**Theorem 11.** PTAS-Makespan is a $(1 + \epsilon)$-approximation for minimum makespan scheduling.

**Proof.** Let $\min_t \{Bin(I, t) = m\} = t^*$. By Lemma 10, $\min_t \{\alpha(I, t, \epsilon) = m\} \leq \min_t \{Bin(I, t) = m\} = |OPT(I)|$. But since PTAS-Makespan checks for values of $t$ that differ by $\epsilon$, it may terminate with $t^* + \epsilon L$ instead. Since $L \leq |OPT(I)|$, $|\text{PTAS-Makespan}(I)| \leq t^* + \epsilon L \leq (1 + \epsilon) \cdot |OPT(I)|$.

**Theorem 12.** PTAS-Makespan is runs in $\text{poly}(I, m)$.

**Proof.** There are at most $\frac{L}{\epsilon} = \max\{p_{\text{max}}, \frac{1}{m} \sum_{i=1}^{n} p_i\}$ values of $t$ to try. Filtering small jobs and rounding remaining jobs take $O(n)$. From previous lecture, $\mathcal{A}_e$ runs in $O(\frac{L}{\epsilon} \cdot n \cdot \frac{1}{\epsilon})$ and FirstFit runs in $O(nm)$.

**References**