Advanced Algorithms 2 October 2018

Lecture 3: Approximation Algorithms III

Lecturer: Mohsen Ghaffari *Scribe: Davin Choo*

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 $\mathcal{A}_\epsilon$, an exact algorithm which solves bin packing under two assumptions:

1. All items have at least size ϵ
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 $\mathcal{S}$ 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 ϵ, for some $\epsilon > 0$

Algorithm 1 PTAS-BinPacking$(I = \mathcal{S}, \epsilon)$

$k \leftarrow \lceil \frac{1}{\epsilon^2} \rceil$
$Q \leftarrow \lfloor n\epsilon^2 \rfloor$
Partition n items into k non-overlapping groups, each with Q items ▷ See Figure 1
for $i \in \{1, \ldots, k\}$ **do**
  $k_{max} \leftarrow \max_{\text{item } j \text{ in group } i} size(j)$
  for item j in group i **do**
    $size(j) \leftarrow k_{max}$
  end for
end for
Denote the modified instance as J
return $\mathcal{A}_\epsilon(J)$

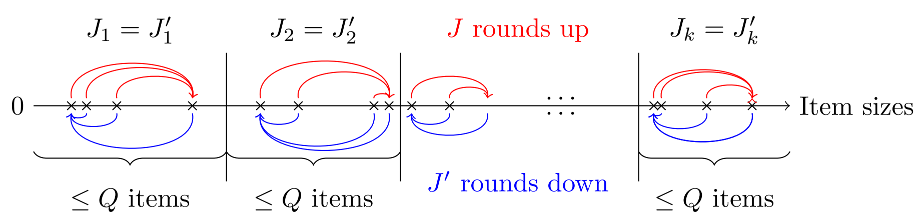


Figure 1: Partition items into k groups, each with at most Q items. Label groups in ascending size ordering. J rounds up item sizes, J' rounds down item sizes.

Algorithm 1 pre-processes the sizes of a given input instance, then calls the exact algorithm $\mathcal{A}_\epsilon$ to solve the modified instance. Since J only *rounds up* sizes, $\mathcal{A}_\epsilon(J)$ 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). □

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

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

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

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

2.2 Full PTAS for bin packing without assumptions

Algorithm 2 FULL-PTAS-BINPACKING($I = \mathcal{S}, \epsilon$)

$\epsilon' \leftarrow \min\{\frac{1}{2}, \frac{\epsilon}{2}\}$ ▷ See analysis why we chose such an ϵ'
$X \leftarrow$ Items with size $< \epsilon'$ ▷ Ignore small items to fulfill assumption of sizes $\geq \epsilon'$
$P \leftarrow$ PTAS-BINPACKING($\mathcal{S} \setminus X, \epsilon'$) ▷ By theorem 4, $|P| = (1+\epsilon') \cdot |OPT(\mathcal{S} \setminus X)|$
$P' \leftarrow$ Using FIRSTFIT, add items in X to P ▷ Handle small items
return Resultant packing P'

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.

$$\begin{array}{rcll} |OPT(I)| & \geq & \sum_{i=1}^{n} size(i) & \text{Lower bound on } |OPT(I)| \\ & > & (m-1)(1-\epsilon') & \text{From above observation} \end{array}$$

Hence,

$$\begin{array}{rcll} m & < & \frac{|OPT(I)|}{1-\epsilon'} + 1 & \text{Rearranging} \\ & < & |OPT(I)| \cdot (1+2\epsilon') + 1 & \text{Since } \frac{1}{1-\epsilon'} \leq 1 + 2\epsilon', \text{ for } \epsilon' \leq \frac{1}{2} \\ & \leq & (1+\epsilon) \cdot |OPT(I)| + 1 & \text{By choice of } \epsilon' = \min\{\frac{1}{2}, \frac{\epsilon}{2}\} \end{array}$$

□

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.*

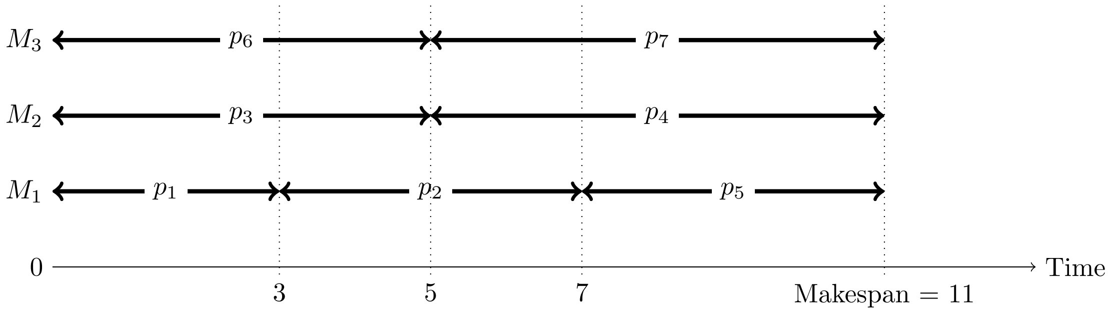
For any problem instance I, let $OPT(I)$ be an optimum job assignment and $|OPT(I)|$ be the corresponding makespan. One can see that:

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

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 $|OPT(I)|$.

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 achieveable 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 $\frac{4}{3}$-approximation algorithm (Algorithm 4). Finally, we end off the section with a PTAS for minimum makespan scheduling.

3.1 Greedy approximation algorithms

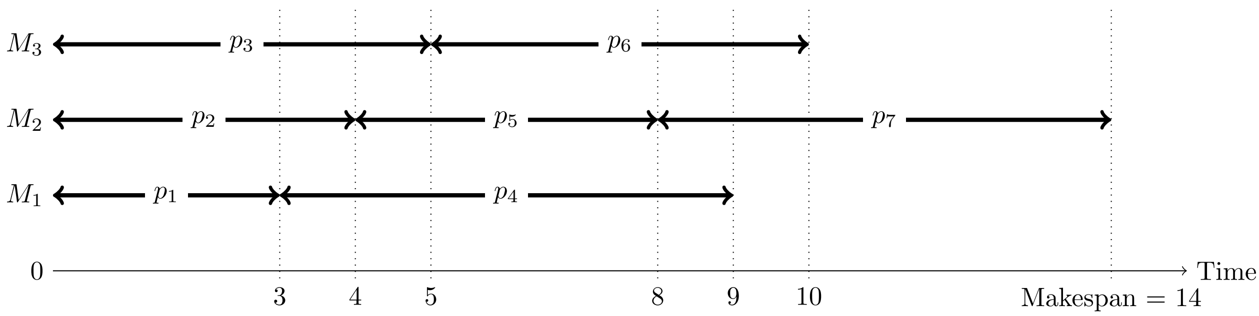
Algorithm 3 $\textsc{Graham}(I = \{p_1, \ldots, p_n\}, m)$

$M_1, \ldots, M_m \leftarrow \emptyset$ ▷ All machines are initially free
for $i \in \{1, \ldots, n\}$ **do**
  $j \leftarrow \text{argmin}_{j \in \{1,\ldots,m\}} \sum_{p \in M_j} p$ ▷ Pick the least loaded machine
  $M_j \leftarrow M_j \cup \{p_i\}$ ▷ Add job i to this machine
end for
return $M_1, \ldots, M_m$

Theorem 7. *$\textsc{Graham}$ (Algorithm 3) is a 2-approximation to minimum makespan scheduling.*

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

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 $m = 3$. $\textsc{Graham}$ will schedule $M_1 = \{p_1, p_4\}, M_2 = \{p_2, p_5, p_7\}, M_3 = \{p_3, p_6\}$, yielding a makespan of 14. As expected, $14 = |\textsc{Graham}(I)| \leq 2 \cdot |OPT(I)| = 22$.



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

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

Algorithm 4 ModifiedGraham($I = \{p_1, \ldots, p_n\}, m$)

$I' \leftarrow I$ in descending order
return Graham(I', m)

Proof. For $m \geq n$, $|\text{ModifiedGraham}(I)| = |OPT(I)|$ by trivially putting one job on each machine. For $m < n$, we may assume that every machine has a job[1].

Suppose, for a contradiction, that $|\text{ModifiedGraham}(I)| > |OPT(I)|$. Then, there exists[2] a sequence of jobs with descending sizes $I = \{p_1, \ldots, p_n\}$ such that the last smallest job p_n causes ModifiedGraham(I) to have a makespan larger than $OPT(I)$. That is, $p_{last} = p_n$ and $|\text{ModifiedGraham}(I \setminus \{p_n\})| \leq |OPT(I)|$. Let $\mathcal{C}$ be the configuration of machines after 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 ≥ 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 $OPT(I)$.

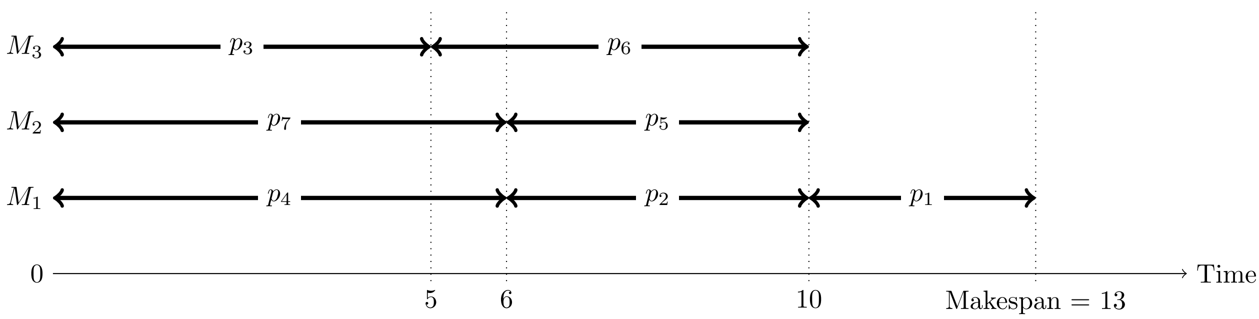
Suppose there are k heavy jobs occupying a machine each in $OPT(I)$. Then, there are $2(m-k)+1$ jobs (two non-heavy jobs per machine in $\mathcal{C}$, and p_n) to be distributed across $m-k$ machines. By pigeonhole principle, at least one machine M^* will get ≥ 3 jobs in $OPT(I)$. However, since the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

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

Proof. Case 1: $p_{last} \leq \frac{1}{3} \cdot |OPT(I)|$
By similar arguments as per Theorem 7, $|\text{ModifiedGraham}(I)| = t + p_{last} \leq \frac{4}{3} \cdot |OPT(I)|$

Case 2: $p_{last} > \frac{1}{3} \cdot |OPT(I)|$
By Lemma 8, $|\text{ModifiedGraham}(I)| = |OPT(I)|$. □

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 $m = 3$. $I' = \{p_4 = 6, p_7 = 6, p_3 = 5, p_6 = 5, p_2 = 4, p_5 = 4, p_1 = 3\}$ and ModifiedGraham will schedule $M_1 = \{p_4, p_2, p_1\}, M_2 = \{p_7, p_5\}, M_3 = \{p_3, p_6\}$, yielding a makespan of 13. As expected, $13 = |\text{ModifiedGraham}(I)| \leq \frac{4}{3} \cdot |OPT(I)| = 14.666\ldots$



[1]Suppose there is a machine M_i without a job, then there must be another machine M_j with more than 1 job (by pigeonhole principle). Shifting one of the jobs from M_j to M_i will not increase the makespan.
[2]If adding p_j for some $j < n$ already causes $|\text{ModifiedGraham}(\{p_1, \ldots, p_j\})| > |OPT(I)|$, we can truncate I to $\{p_1, \ldots, p_j\}$ so that $p_{last} = p_j$. Since $p_j \geq p_n > \frac{1}{3} \cdot |OPT(I)|$, the antecedent still holds.

3.2 PTAS for minimum makespan scheduling

Recall that any makespan scheduling instance (I, m) has a lower bound $L(I) = \max\{p_{max}, \frac{1}{m}\sum_{i=1}^{n} p_i\}$. 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, we can relate $Bin(I, t)$ to the bin packing problem. One can see that $|OPT(I)| = \min_t\{Bin(I, t) = m\}$, 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($I = \{p_1, \ldots, p_n\}, m$)

```
L = max{p_max, (1/m) Σ_{i=1}^n p_i}
for t ∈ {L, L + ε, L + 2ε, ..., 2L} do        ▷ Binary search on t also works, but it is still poly-time
    X ← Jobs with sizes ≤ εt                  ▷ Remaining jobs have sizes ∈ (εt, t]
    I' ← I \ X                                 ▷ Ignore small jobs
    h ← ⌈log_{1+ε}(1/ε)⌉      ▷ Partition (εt, t] into powers of (1 + ε): tε · (1, (1 + ε), ..., (1 + ε)^h = ε^{-1}]
    for p_i ∈ I' do
        k ← min_{j∈{0,...,h}} {p_i ≥ tε(1 + ε)^j}      ▷ Find lowest power of (1 + ε) for rounding down
        p_i ← tε(1 + ε)^j                              ▷ Round down job sizes
    end for
    P ← A_ε(I')                ▷ See Section 3.2.1 in Lecture 2 notes for A_ε, but with size t bins
    α(I, t, ε) ← Use bins of size t(1 + ε) to emulate P          ▷ Use extra (1 + ε) buffer
    α(I, t, ε) ← Using FirstFit, add items in X to α(I, t, ε)    ▷ Handle small items
    if α(I, t, ε) uses ≤ m bins then          ▷ Since |OPT(I)| ∈ [L, 2L], this will occur at some point
        return Assign jobs to machines according to bin assignment α(I, t, ε)
    end if
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 $\mathcal{A}_\epsilon$ (exact solving on rounded down non-small items) that are more than $(t(1+\epsilon) - \epsilon t) = t$-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 ϵ, 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 poly(I, m).*

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

References

[Gra66] Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.