Approximation schemes

Bin packing problem

Bin Packing problem

- *Given* n items with sizes $a_1,\ldots,a_n \in (0,1]$.
- *Find* a packing in unit-sized bins that minimizes the number of bins used.

First Fit

- Consider items in an arbitrary order. In this i-th step, it has a list of partially packed bins, say $B_1,\ldots, B_k$. It attempts to put the next item, a_i, in one of these bins, in this order. If a_i does not fit into any of these bins, it opens a new bin B_{k+1}, and puts a_i in it.

First Fit Algorithm

Input $(a_1,\ldots,a_n)$

1) $i \leftarrow 1,\ k \leftarrow 1,\ \mathrm{bin}(1) \leftarrow 1,\ \mathrm{bin}(2) \leftarrow 1.$

2) **While** $i \leq n$ **do:**

$r \leftarrow \min\{j \leq k+1 \mid a_i \leq \mathrm{bin}(j)\}$

if $(r \leq k)$ **then** $\mathrm{bin}(r) \leftarrow \mathrm{bin}(r) - a_i$

otherwise $k \leftarrow k + 1$

$\mathrm{bin}(k) \leftarrow 1 - a_i$

$\mathrm{bin}(k+1) \leftarrow 1$

Output (k)

Approximation ratio of First Fit Algorithm

Теорема 6.1

First Fit algorithm is a 2-approximation for the bin packing problem.

Proof: If the algorithm use k bins.

$$OPT \geq \sum_{i=1}^{n} a_i > \frac{k-1}{2} \Rightarrow 2OPT > k-1 \Rightarrow 2OPT \geq k$$

Inapproximability

Theorem 6.2

For any $\varepsilon > 0$, there is no approximation algorithm having a guarantee of $3/2 - \varepsilon$ for the bin packing problem, assuming $P \neq NP$.

.

Proof

- If there were such an algorithm, then we show how to solve the NP-hard problem of deciding if there is a way to partition n nonnegative numbers $a_1,\ldots, a_n$ into two sets, each adding up to $\frac{1}{2}\sum_i a_i$.
- Clearly, the answer to this question is 'yes' iff the n items can be packed in 2 bins of size $\frac{1}{2}\sum_i a_i$.
- If the answer is "yes' the $3/2 - \varepsilon$ factor algorithm will have to give an optimal packing, and thereby solve the partitioning problem.

An asymptotic PTAS

- **Theorem 6.3**

  For any ε, $0 < \varepsilon \leq 1/2$, there is an algorithm A_ε that runs in time polynomial in n and finds a packing using at most $(1+2\varepsilon)OPT + 1$ bins.

Packing of big items with fixed number of item sizes

- **Lemma 6.4**

  Let $\varepsilon > 0$ be fixed, and let $K > 0$ be a fixed nonnegative integer. Consider the restriction of the bin packing problem to instances in which each item is of size at least ε and the number of distinct item sizes is K. There is a polynomial time algorithm that optimally solves the restricted problem.

Proof

- The number of items in a bin is bounded by $\lfloor 1/\varepsilon \rfloor := M$.
- Therefore, the number of different bin types is bounded by $\binom{M+K-1}{M} := R$, which is a constant.

- The total number of bins used is at most n.
- The number of possible feasible packing is bounded by $\binom{n+R-1}{R} := P,$ which is polynomial in n.
- Enumerating them and picking the best packing gives the optimal answer.

Packing of big items

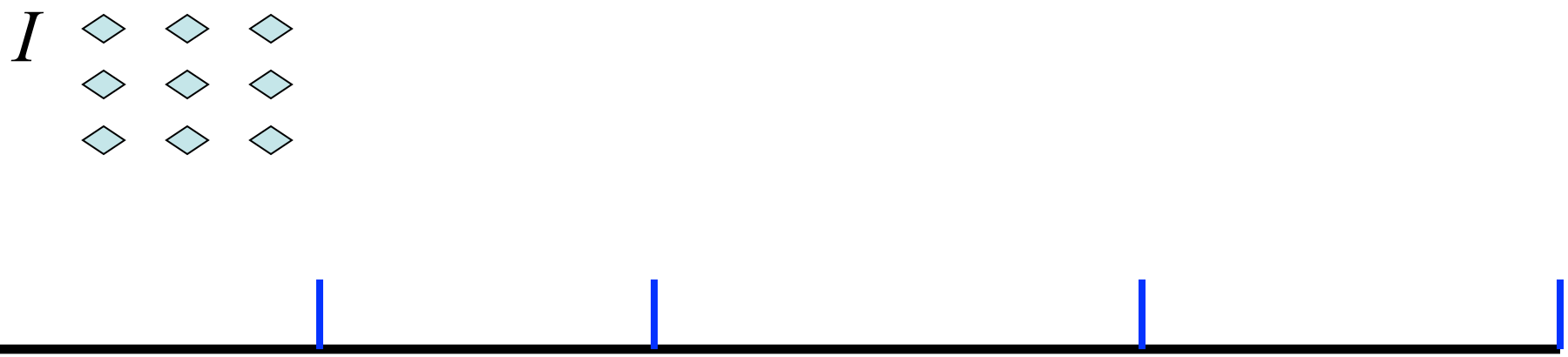
- **Lemma 6.5**

  Let $\varepsilon > 0$ be fixed. Consider the restriction of the bin packing problem to instances in which each item is of size at least ε. There is a polynomial time approximation algorithm that solves the restricted problem within a factor of $(1+\varepsilon)$.

Proof

Let I denote the given instance. Sort the n items by increasing size.

Partition them into $K=\lceil 1/\varepsilon^2 \rceil$ groups each having at most $Q=\lfloor n\varepsilon^2 \rfloor$ items.



Proof

Let I denote the given instance. Sort the n items by increasing size.

Partition them into $K=\lceil 1/\varepsilon^2 \rceil$ groups each having at most $Q=\lfloor n\varepsilon^2 \rfloor$ items.

Rounding up the size of each item to the size of the largest item in its group.

I

Proof

Let I denote the given instance. Sort the n items by increasing size.

Partition them into $K=\lceil 1/\varepsilon^2 \rceil$ groups each having at most $Q=\lfloor n\varepsilon^2 \rfloor$ items.

Rounding up the size of each item to the size of the largest item in its group.

A new instance J has at most K different item sizes.

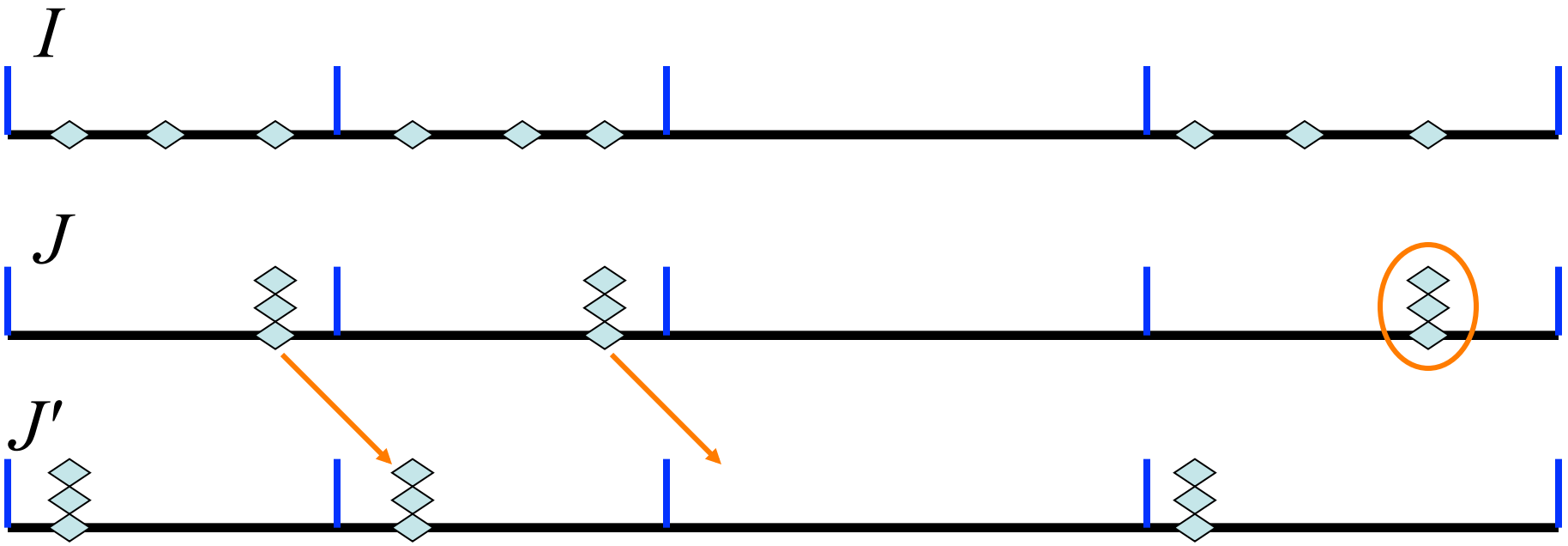
By Lemma 6.5 we can find an optimal packing for J.

J

Proof (2)

- Clearly, this will also be a valid packing for the original item sizes.
- We show below that OPT(J) ≤ (1+ε) OPT(I).
- Let us construct another instance, say J', by rounding down the size of each item to that of the smallest item in its group.
- The crucial observation is that a packing for instance J' yields a packing for all but the largest Q items of instance J.

Proof
OPT(J) ≤(1+ε) OPT(I)



$OPT(J') \le OPT(I)$

$OPT(J) \le OPT(J') + Q \le OPT(I) + Q$

$OPT(I) \ge n\varepsilon$ ⇒ $Q = n\varepsilon^2 \le \varepsilon OPT(I)$ ⇒ $OPT(J) \le (1+\varepsilon) OPT(I)$

Fernandes de la Vega and Lueker Algorithm

Input $(a_1,\ldots,a_n)$

1) Remove items of size less than ε.
2) Round to obtain constant number of item sizes.
3) Find optimal packing.
4) Use this packing for original item sizes.
5) Pack items of size < ε using First-Fit.

Output (k)

An asymptotic PTAS

- **Theorem 6.3**

  For any ε, $0 < \varepsilon \leq 1/2$, there is an algorithm A_ε that runs in time polynomial in n and finds a packing using at most $(1+2\varepsilon)OPT + 1$ bins.

Proof of Theorem 6.3

- Let k be the number of used bins.
- Let I denote the given instance, and I' denote the instance obtained by discarding items of size $< \varepsilon$ from I.
- By Lemma 6.5 we can find a packing for I' using at most $(1+\varepsilon)\mathrm{OPT}(I')$ bins.
- If no additional bins are needed, then we have a packing in $k = (1+\varepsilon)\mathrm{OPT}(I') \leq (1+\varepsilon)\mathrm{OPT}(I)$ bins.
- In the second case, all but the last bin must be full to the extent of at least $1-\varepsilon$.
- Therefore, the sum of items in I is at least $(k-1)(1-\varepsilon) \leq \mathrm{OPT}(I)$.
- We get $k \leq \mathrm{OPT}(I)/(1-\varepsilon) + 1 \leq (1+2\varepsilon)\mathrm{OPT}(I)+1$, $(0 < \varepsilon < 1/2)$.

$P||C_{max}$

- *Given* processing times for n jobs, $p_1,\ldots,p_n$, and an integer m.
- *Find* an assignment of the jobs to m identical machines so that the completion time, also called the makespan, is minimized

Greedy Algorithm (GL)

- Schedule the jobs one by one, in an arbitrary order.
- Each job being assigned to a machine with least amount of work so far.

The first approximation algorithm

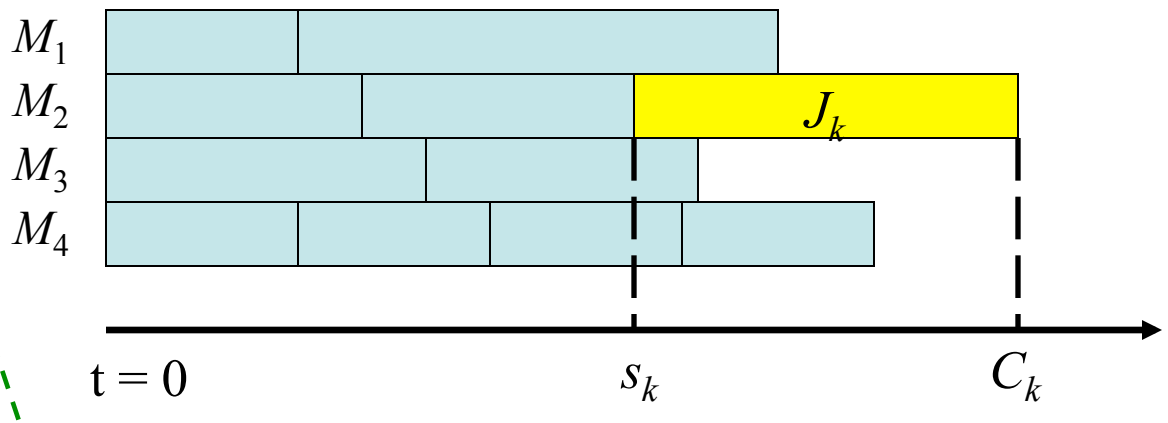
Theorem 6.6 (Graham [1966])

Алгоритм GL *is* $(2 - 1/m)$-*approximation algorithm for* $P||C_{max}$.

Proof

$$C^*_{\max} \geq (1/m)\sum_{j=1}^{n} p_j,$$

$$C^*_{\max} \geq p_k,$$



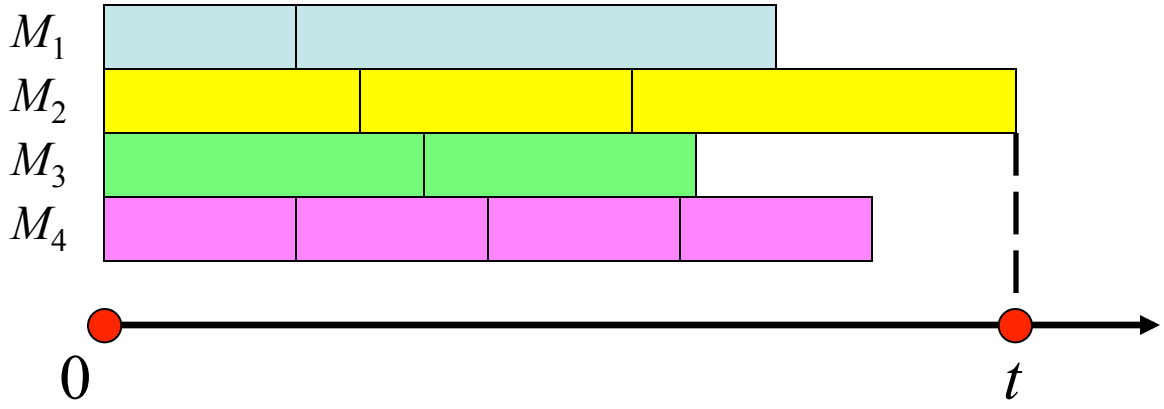
$$C^{LS}_{\max} = C_k = s_k + p_k \leq (1/m)\sum_{j\neq k} p_j + p_k =$$

$$(1/m)\sum_{j=1}^{n} p_j + (1-1/m)p_k \leq C^*_{\max} + (1-1/m)C^*_{\max} = (2-1/m)C^*_{\max}.$$

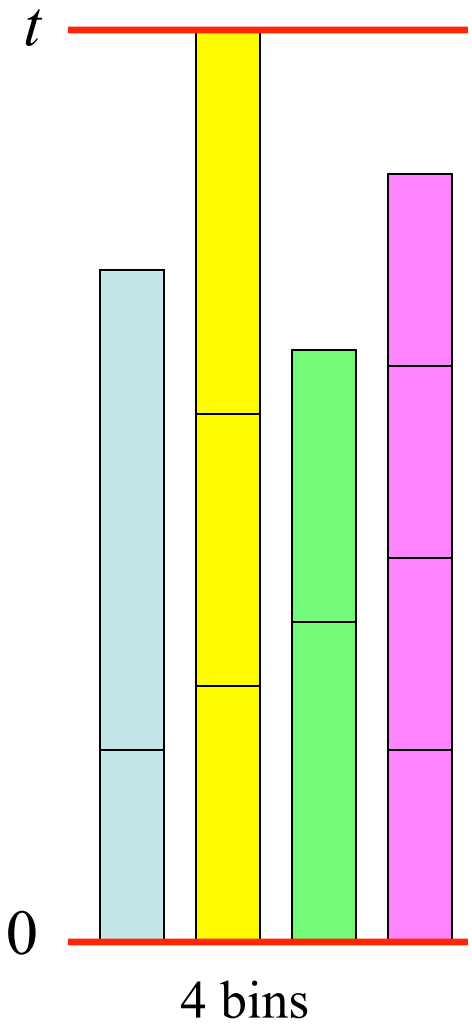
$P||C_{max}$

- The minimum makespan problem is strongly NP-hard; thus it does not admit an FPTAS, assuming P ≠ NP. We will obtain a PTAS for it. The minimum makespan problem is closely related to the bin packing problem by the following observation.

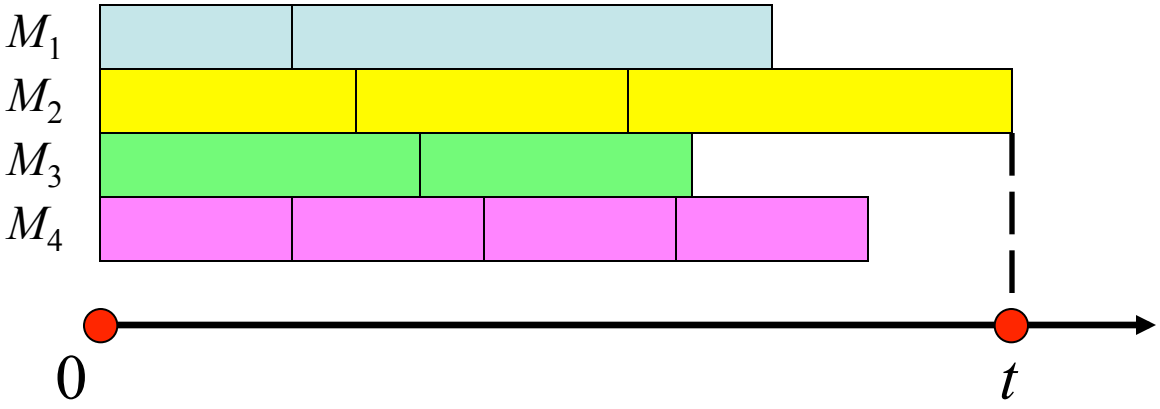
P||$C_{\max}$ and the bin packing problem



There exists a schedule with makespan t iff n objects of sizes $p_1, p_2, \ldots, p_n$ can be packed into m bins of capacity t each.

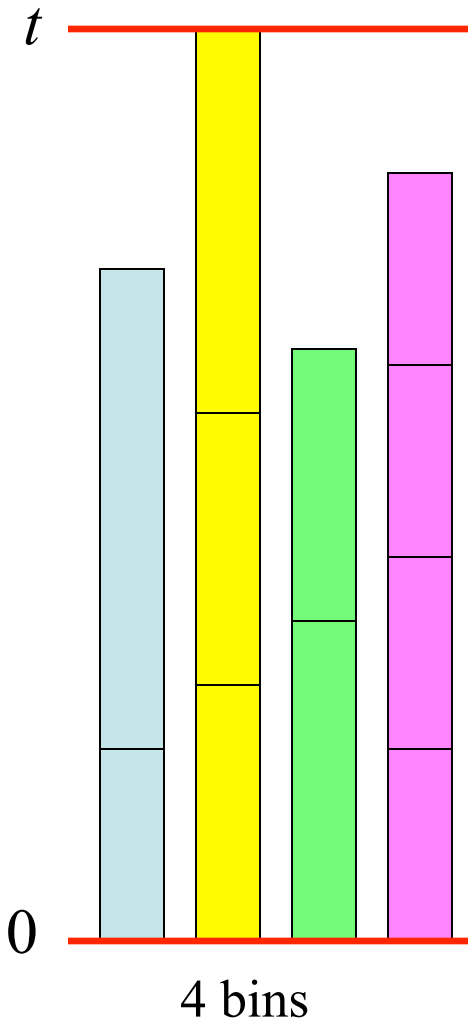


$P||C_{max}$ and the bin packing problem



bins(i,t) is the minimum number of bins of size t, required to pack these n objects.

$$C^*_{max} = \min\{t : \text{bins}(I,t) \le m\}$$



Binary search

- Thus if we know or forecast $C_{\max}$ in an optimal solution we can reduce $P||C_{\max}$ to the bin packing problem.

$$LB = \max\left\{(1/m)\sum_{j=1}^{n} p_j, \max_j\{p_j\}\right\} \text{ is a lower bound.}$$

$$LB \le C^*_{\max} \le 2LB$$

- We can determine the minimum makespan by a binary search in the interval $[LB, 2LB]$.

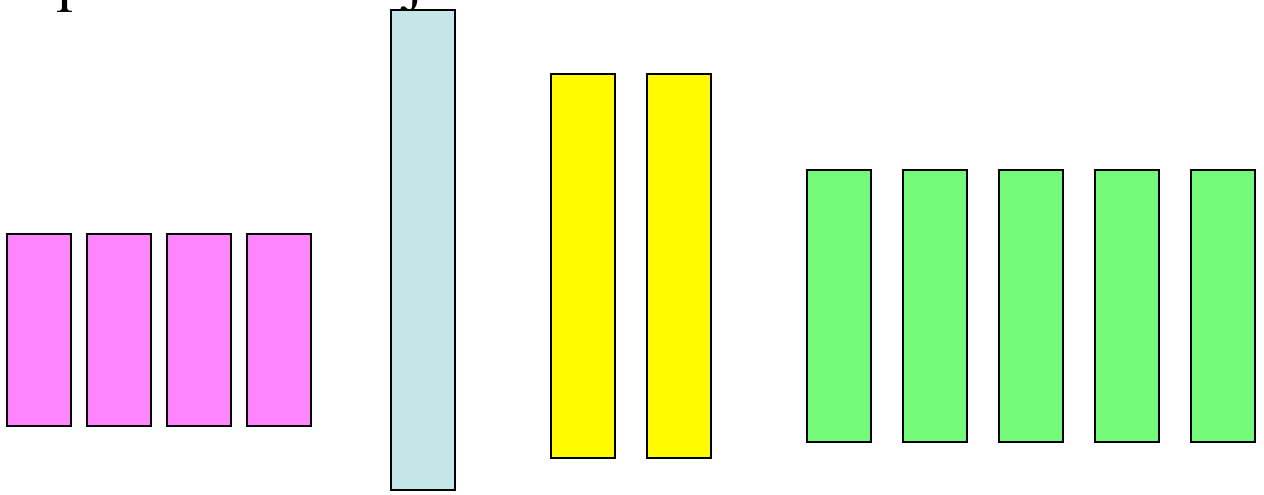
Observation

- At first sight, this reduction may not seem very useful since the bin packing problem is also NP-hard. However, it turns out that this problem is polynomial time solvable if the object sizes are drawn from a set of fixed cardinality. We will use this fact critically for solving the minimum makespan problem.

Bin packing with fixed number of object sizes

- Let k be the fixed number of object sizes.
- Fix an ordering on the object sizes.
- Now, an instance of the bin packing problem can be described by a ***k*-tuple**, $(i_1, i_2, \ldots, i_k)$ specifying the number of objects of each size.
- Let BINS$(i_1, i_2, \ldots, i_k)$ denote the minimum number of bins needed to pack these objects.

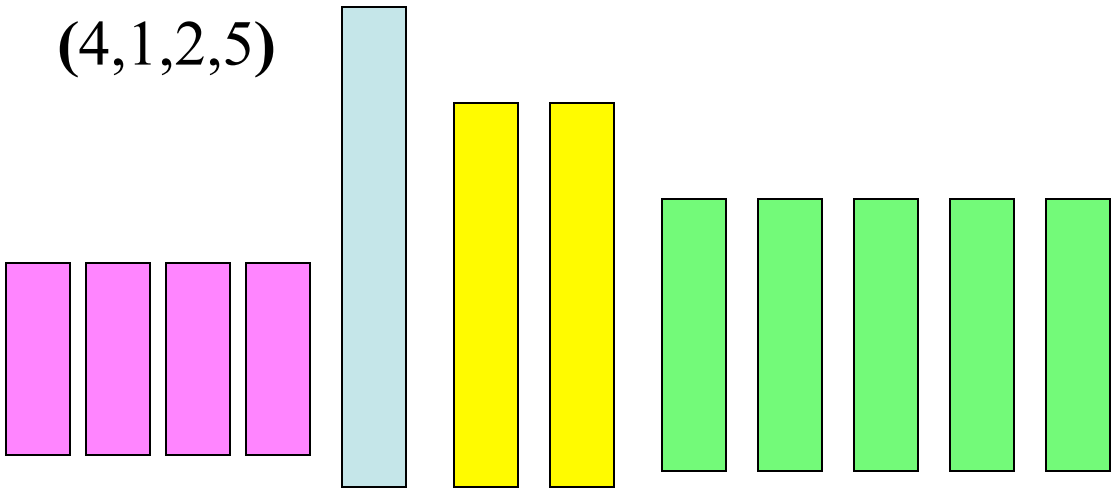
(4,1,2,5)



One bin

For a given instance $(n_1, n_2,\ldots, n_k)$, $\Sigma n_i = n$, we first compute Q, the set of all k-tuples $(q_1, q_2,\ldots, q_k)$ such that $\text{BINS}(q_1, q_2,\ldots, q_k) = 1$ and $0 \le q_i \le n_i$, $1 \le i \le k$.

(4,1,2,5)

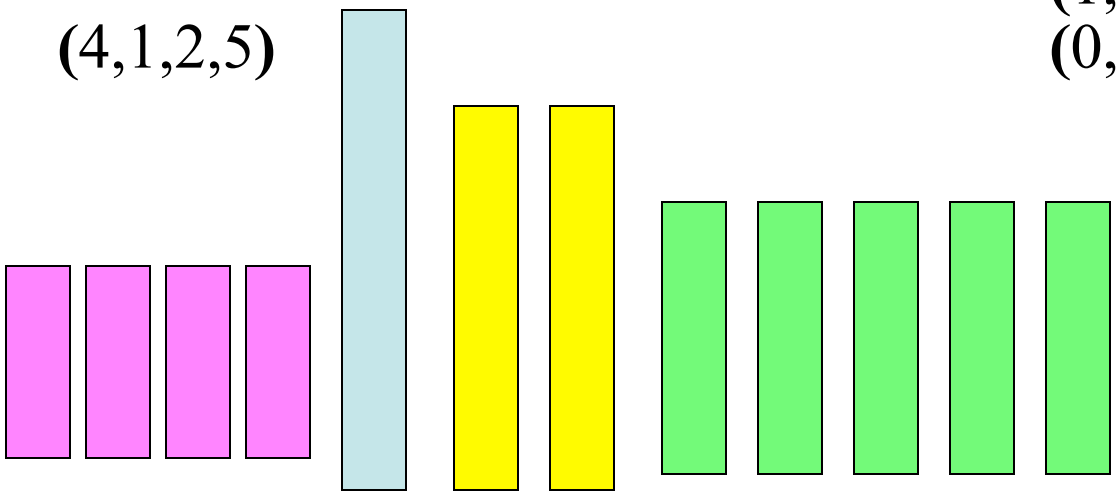


One bin

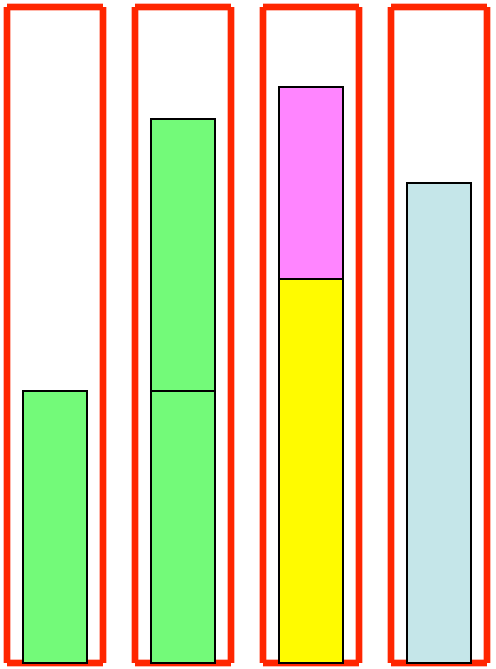
For a given instance $(n_1, n_2, \ldots, n_k)$, $\Sigma n_i = n$, we first compute Q, the set of all k-tuples $(q_1, q_2, \ldots, q_k)$ such that $\mathrm{BINS}(q_1, q_2, \ldots, q_k) = 1$ and $0 \le q_i \le n_i$, $1 \le i \le k$.

$$|Q| \le \prod_{i=1}^{k} n_i \le n^k$$

(4,1,2,5)



(0,0,0,1)
(0,0,0,2)
(1,0,1,0)
(0,1,0,0)



Dynamic Programming

- We compute all entries of the k-dimensional table BINS(i_1, i_2,…, i_k) for every $(i_1, i_2,\ldots, i_k)\in\{0,\ldots,n_1\}\times\{0,\ldots,n_2\}\times\ldots\times\{0,\ldots,n_k\}$.
- The table is initialized by setting BINS(q) = 1 for every $q\in Q$.
- Then, we use the following recurrence to compute the remaining entries

$$\mathrm{BINS}(i_1, i_2, \ldots, i_k) = 1 + \min_{q\in Q}\{\mathrm{BINS}(i_1 - q_1, i_2 - q_2, \ldots, i_k - q_k)\}.$$

- Computing each entry takes $O(n^k)$ time. Thus, the entire table can be computed in $O(n^{2k})$ time.

Basic idea

- The basic idea is that if we can tolerate some error in computing the minimum makespan, then we can reduce this problem to the restricted version of bin packing in polynomial time.
- There will be two sources of error:
  - rounding object sizes so that there are a bounded number of different sizes.
  - terminating the binary search to ensure polynomial running time.
- Each error can be made as small as needed, at the expense of running time. Moreover, for any fixed error bound, the running time is polynomial in *n,* and thus we obtain a polynomial approximation scheme.

Core Algorithm for fixed t ($LB \le t \le 2LB$)

Input $(p_1,\ldots,p_n,\varepsilon,t)$

1) Divide set of jobs (objects) into two sets **Big** $= \{ j \mid p_j \ge t\varepsilon\}$ and **Small** $= \{ j \mid p_j < t\varepsilon\}$.
2) Round a size of each big objects:
   - **if** $p_j \in [t\varepsilon(1+\varepsilon)^i, t\varepsilon(1+\varepsilon)^{i+1})$ **then** $p_j' \leftarrow t\varepsilon(1+\varepsilon)^i$**.**
3) Find an optimal packing $\boldsymbol{U}$ of the rounded big objects (p_j') in bins of size t using the dynamic programming algorithm.
4) Consider the original sizes of objects. Then $\boldsymbol{U}$ is valid for a bin sizes $t(1+\varepsilon)$.
5) Pack the small objects greedily in leftover spaces in the bins using the first fit rule. Denote with $\alpha(I,\varepsilon,t)$ the number of bins used by this algorithm.

Output ($\alpha(I,\varepsilon,t)$)

Running time of the Core Algorithm

- The number of distinct values of p_j' is $k = \lceil \log_{1+\varepsilon} 1/\varepsilon \rceil$.
- The running time of the Core Algorithm is the same as the running time of the dynamic programming algorithm and is equal to $O(n^{2k})$.
- For any fixed ε the running time of the Core Algorithm is polynomial in n.

Lower bound

- **Лемма 6.7**

$$\alpha(I,\varepsilon,t) \leq \text{bins}(I,t).$$

Proof

- If the algorithm does not open any new bins for the small objects, then the assertion clearly holds since the rounded down pieces have been packed optimally in bins of size t. It follows that at least $\alpha(I,\varepsilon,t)$ bins is required.
- In the other case, all but the last bin are packed at least to the extent of t. It follows that the total size of objects greater than t $(\alpha(I,\varepsilon,t) - 1)$. Hence, the optimal packing of I in bins of size t must also use at least $\alpha(I,\varepsilon,t)$ bins.

Lower bound on makespan

- **Corollary 6.8**

  $\min\{t: \alpha(I,\varepsilon,t) \leq m\} \leq OPT.$

Lower bound on makespan

- **Corollary 6.8**

$$\min\{t: \alpha(I,\varepsilon,t) \leq m\} \leq OPT.$$

Proof.

$OPT = \min\{t: \text{bins}(I,t) \leq m\}$.

Lemma 6.7 implies that for every t: $\alpha(I,\varepsilon,t) \leq \text{bins}(I,t)$.

Hence, $\min\{t: \alpha(I,\varepsilon,t) \leq m\} \leq OPT$.

How to determine t ?

- We perform the binary search on the interval [LB,$2LB$].
- The length of the interval is LB at the start of the search, and it reduces by a factor 2 in each iteration.
- We continue the search until the available interval drops to a length of εLB. Let T be the right endpoint of the interval we terminate with.
- This will require $\lceil \log_2 1/\varepsilon \rceil$ iterations.

Lower bound

- **Lemma 6.9**

$$T \leq (1+\varepsilon)OPT.$$

Proof.

$$\min\{t: \alpha(I,\varepsilon,t) \leq m\} \in [T - \varepsilon LB, T]$$

$$T \leq \min\{t: \alpha(I,\varepsilon,t) \leq m\} + \varepsilon LB \leq (1+\varepsilon)OPT.$$

PTAS

- **Theorem 6.10**

  For every $\varepsilon > 0$, there exists algorithm A_ε, that produces a valid schedule having makespan at most $(1+\varepsilon)^2 OPT \leq (1+3\varepsilon)OPT$ in $O(n^{2k} \lceil \log_2 1/\varepsilon \rceil)$ time, where $k = \lceil \log_{1+\varepsilon} 1/\varepsilon \rceil$.

Exercise

- Consider a more restricted algorithm than First-Fit, called Next-Fit, which tries to pack the next item only in the most recently started bin. If it does not fit, it is packed in a new bin. Show that this algorithm also achieves factor 2. Give a factor 2 tight example.