Approximation Algorithms for Packing and Scheduling Problems

Klaus Jansen

Universität Kiel

Overview

- EPTAS for the Multiple Knapsack Problem (MKP)
- $3/2$ Approximation algorithm for Scheduling with Fixed Jobs

Multiple Knapsack Problem (MKP)

- Introduction
- Instances with similar capacities
- Instances with few bins
- General instances

Multiple knapsack problem (MKP)

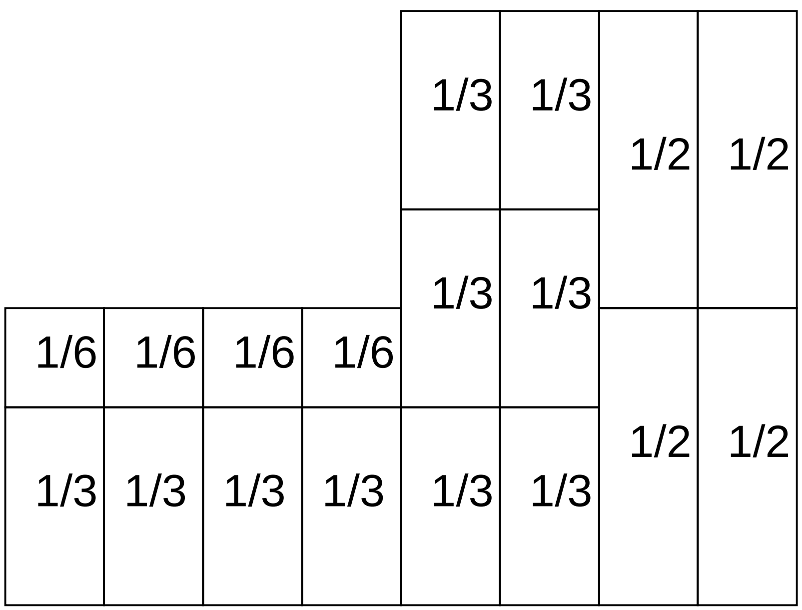
Given:

- a set A of n items with $size(a_j), profit(a_j) \in \mathbb{Z}^+$,
- a set B of m bins with capacities $c(b) \in \mathbb{Z}^+$.

Problem: find a subset $A' \subset A$ of maximum total profit $\sum_{a \in A'} profit(a)$ such that A' can be packed into B without exceeding the capacities.

Example

- 10 items of size $1/2$, $1/3$ and $1/6$ with profit $1/5$, $1/6$ and $1/15$.
- 4 bins of capacity 1 and 4 bins of capacity $1/2$.



profit(A’) = 4/5+10/6+4/15

Results

Known Results:

- MKP is strongly NP-hard (contains bin packing as special case)
- there is no FPTAS even for two bins (unless $P = NP$) **(Chekuri, Khanna), (Caprara, Kellerer, Pferschy)**
- there is a PTAS for MKP **(Chekuri, Khanna)** with running time

$$n^{O(1/\epsilon^8 \log(1/\epsilon))}.$$

Different Types of Approximation Schemes

- Polynomial Time Approximation Scheme (PTAS) with running time $|I|^{f(1/\epsilon)}$ for some function f.
- Efficient Polynomial Time Approximation Scheme (EPTAS) with running time $f(1/\epsilon)\, poly(|I|)$ for some function f.
- Fully Polynomial Time Approximation Scheme (FPTAS) with running time $poly(1/\epsilon, |I|)$.

Open Questions for Multiple Knapsack Problem

(1) Is there a PTAS for MKP with an improved running time $f(1/\epsilon)poly(n)$ **(Chekuri, Khanna 2000)**?

(2) Admits MKP an fixed parameter tractable (FPT) algorithm or is MKP W[1]-hard **(Fellows 2003)**?

Notice: If the standard parametrization of an optimization problem is W[1]-hard, then the optimization problem does not have an EPTAS (unless FPT=W[1]) **(Bazgan 1995, Cesati and Trevisan 1997)**.

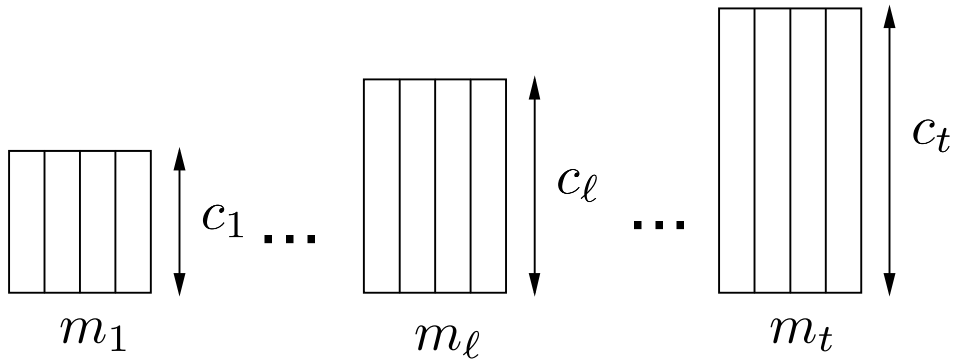
New Result

Theorem: (Jansen, SODA 2009)

There is an EPTAS for the multiple knapsack problem (MKP) with running time

$$2^{O(1/\epsilon^5 \log(1/\epsilon))} poly(n).$$

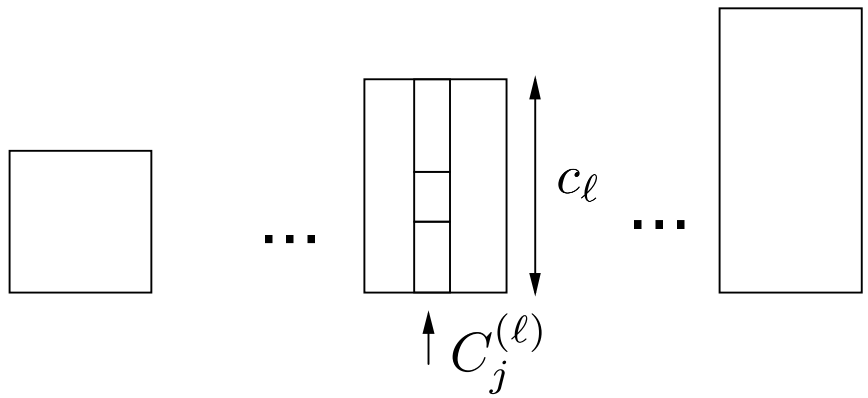
Instances with similar capacities



Let $c_1 < \ldots < c_t$ be the different capacities in the instances.

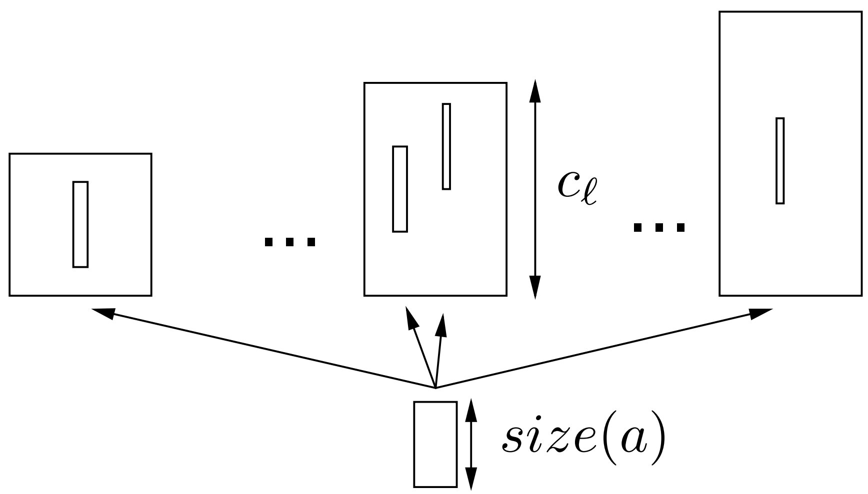
Suppose that there are $m_\ell \geq 1/\delta^3$ bins of capacity c_ℓ for each $\ell = 1, \ldots, t$.

LP-Relaxation



- a configuration $C_j^{(\ell)}$ is a subset $A' \subset A$ of items with $\sum_{a \in A'} size(a) \leq c_\ell$.
- use a fractional variable $y_j^{(\ell)}$ to denote the length of configuration $C_j^{(\ell)}$ in the solution.

LP-Relaxation



- use a variable $x_i \in [0, 1]$ to indicate a fractional piece of item a_i and allow this piece to be distributed among the t groups.

LP-Relaxation $LP(A, B)$

$$
\begin{array}{lcll}
\max \sum_{i=1}^{n} profit(a_i) x_i & & & \\
\sum_{\ell=1}^{t} \sum_{j : a_i \in C_j^{(\ell)}} y_j^{(\ell)} & = & x_i & \text{for } i = 1, \ldots, n, \\
\sum_{j=1}^{H_\ell} y_j^{(\ell)} & \leq & m_\ell & \text{for } \ell = 1, \ldots, t, \\
y_j^{(\ell)} & \geq & 0 & \text{for } j = 1, \ldots, H_\ell \text{ and } \ell = 1, \ldots, t, \\
x_i & \in & [0, 1] & \text{for } i = 1, \ldots, n.
\end{array}
$$

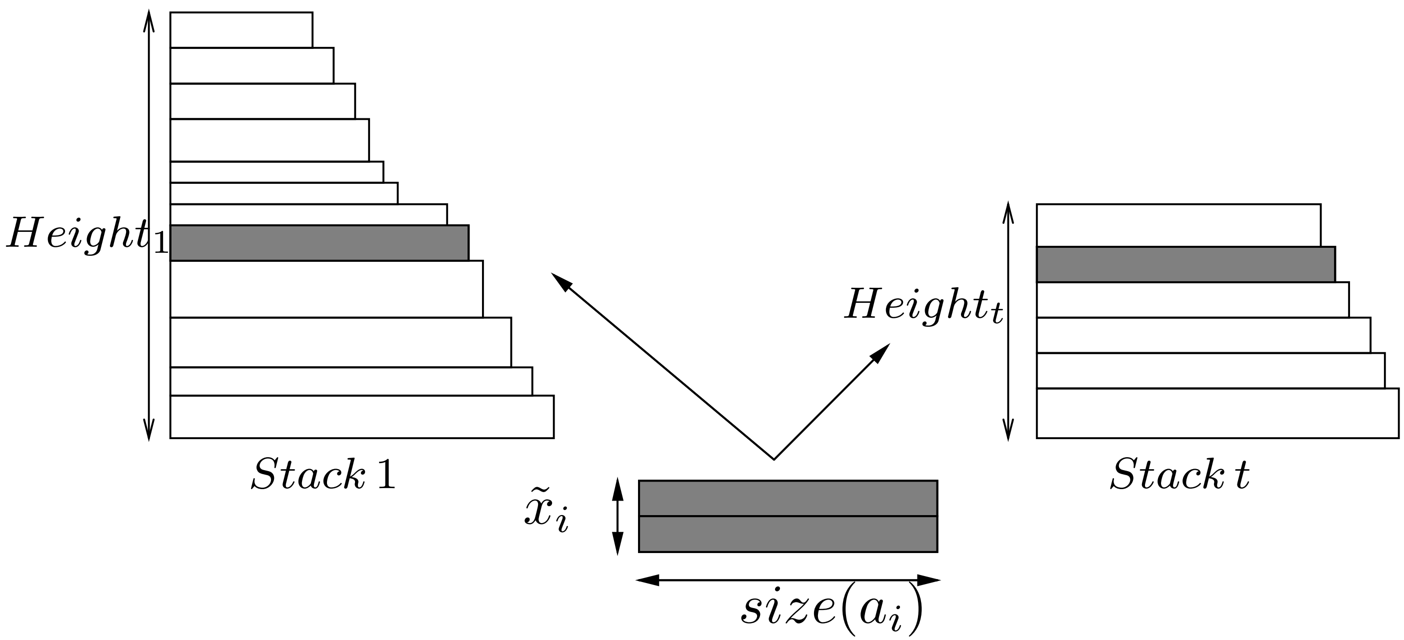
First Results

1) The linear program $LP(A,B)$ is a relaxation of the multiple knapsack problem: $OPT(LP(A,B)) \geq OPT_{MKP}(A,B)$.

2) We can compute an approximate solution $(\tilde{x},\tilde{y})$ of the LP in time polynomial in n and $1/\alpha$ where $\sum_j \tilde{y}_j^{(\ell)} \leq m_\ell(1+2\alpha)$ and objective value at least $(1-3\alpha)OPT(LP(A,B))$.

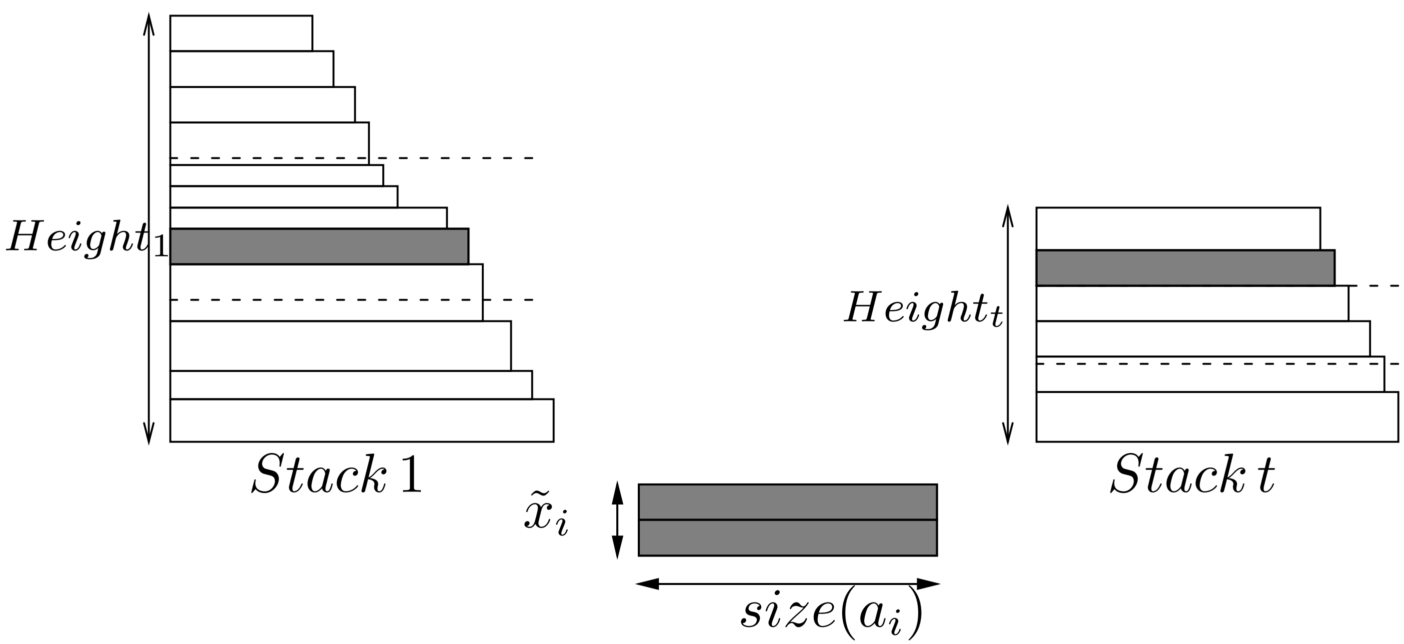
Rounding the LP-solution

Let $z_i^{(\ell)} = \sum_{j:a_i \in C_j^{(\ell)}} \tilde{y}_j^{(\ell)}$ be a piece of item a_i in group ℓ. Use rectangles $(size(a_i), z_i^{(\ell)})$ and notice that $\sum_{\ell=1}^{t} z_i^{(\ell)} = \tilde{x}_i$.



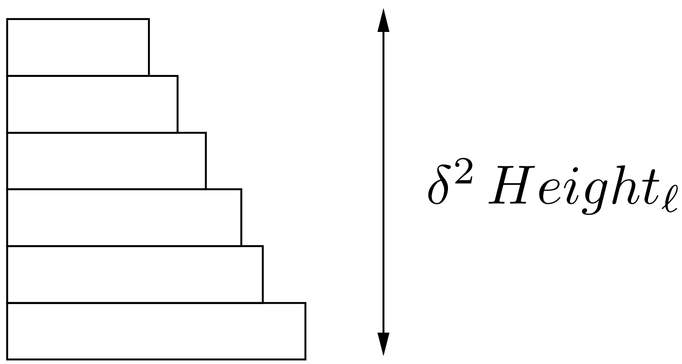
Rounding the LP-solution

We can round the solution such that there are at most $1/\delta^2$ items with values $\bar{z}_i^{(\ell)} \in (0, \tilde{x}_i)$ for $\ell = 1, \ldots, t$.



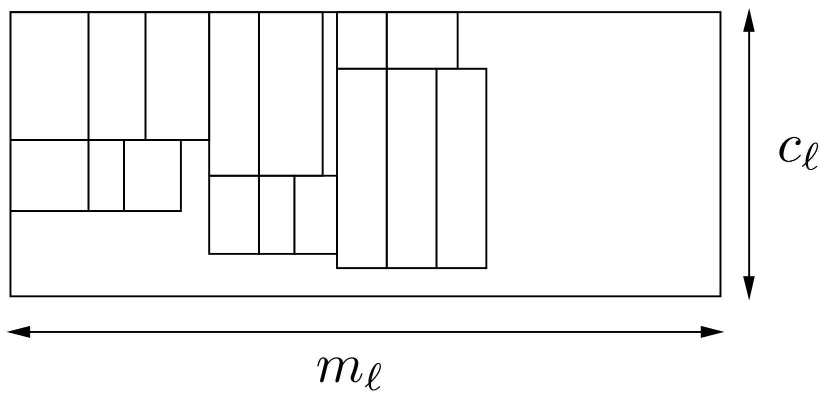
Selecting the items

Remove the items a_i with values $\bar{z}_i^{(\ell)} \in (0, \tilde{x}_i)$. Then, each remaining item with $\tilde{x}_i > 0$ is assigned to one group ℓ and one part j.



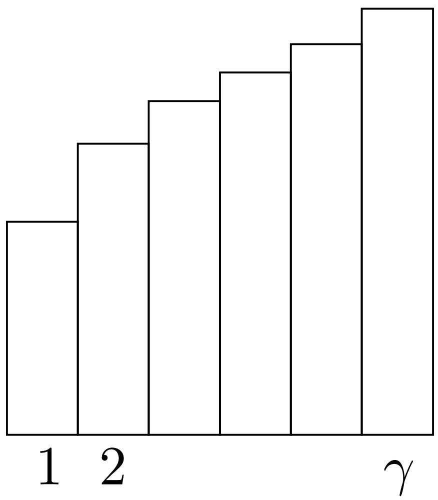
Use **fractional 1-dimensional knapsack** to select items.

Packing of items into $m_\ell \geq 1/\delta^3$ bins



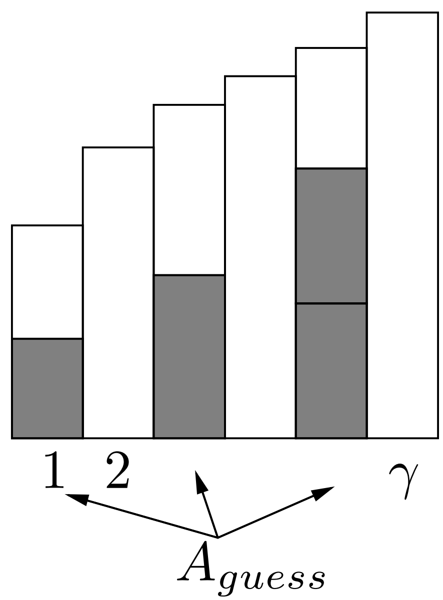
- Use AFPTAS by Kenyon and Rémila for strip packing and pack selected and removed items into $m_\ell(1 + 6\alpha) + 5/\delta^2$ bins.
- Apply shifting technique to select a subset of items with high profit.

Instances with few bins



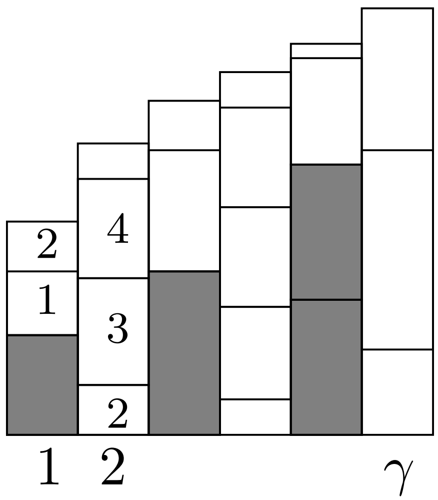
- Consider items with high profit $profit(a_i) \geq \rho/\gamma OPT(A, B)$.
- Round the profit of these items to values $k[\epsilon'\rho/\gamma]OPT(A, B)$ where $k \in \{1/\epsilon', \ldots, \gamma/\epsilon'\rho\}$.

Instances with few bins



- Reduce the number of high profit items to $O([\gamma/\epsilon^2]\log[\gamma/\epsilon^2])$.
- Choose subsets A_{guess} with at most γ/ρ items and test whether they fit into the bins.

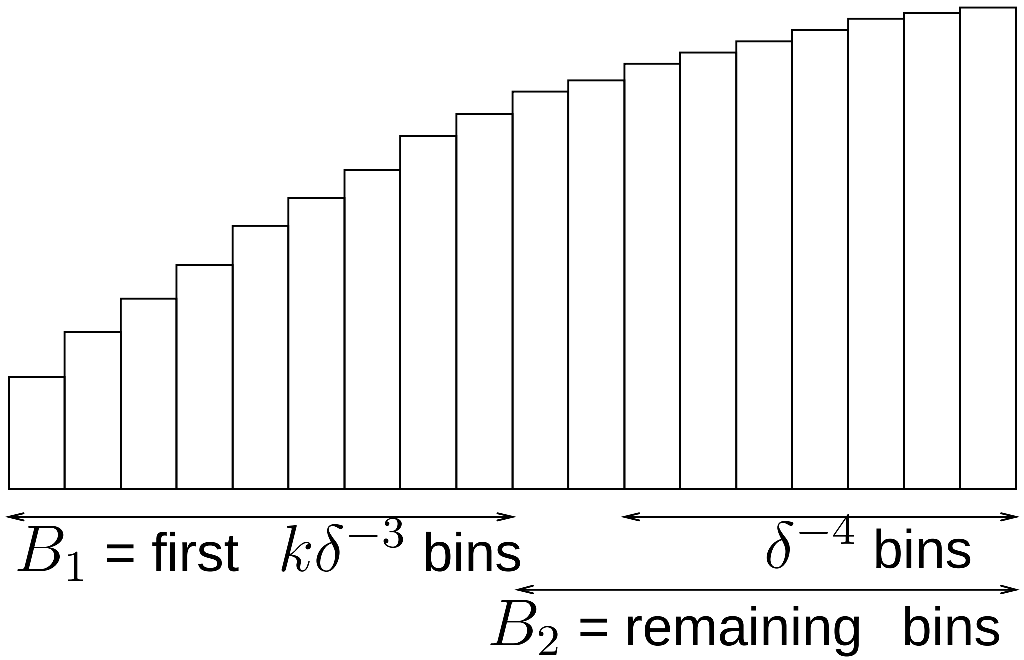
Instances with few bins



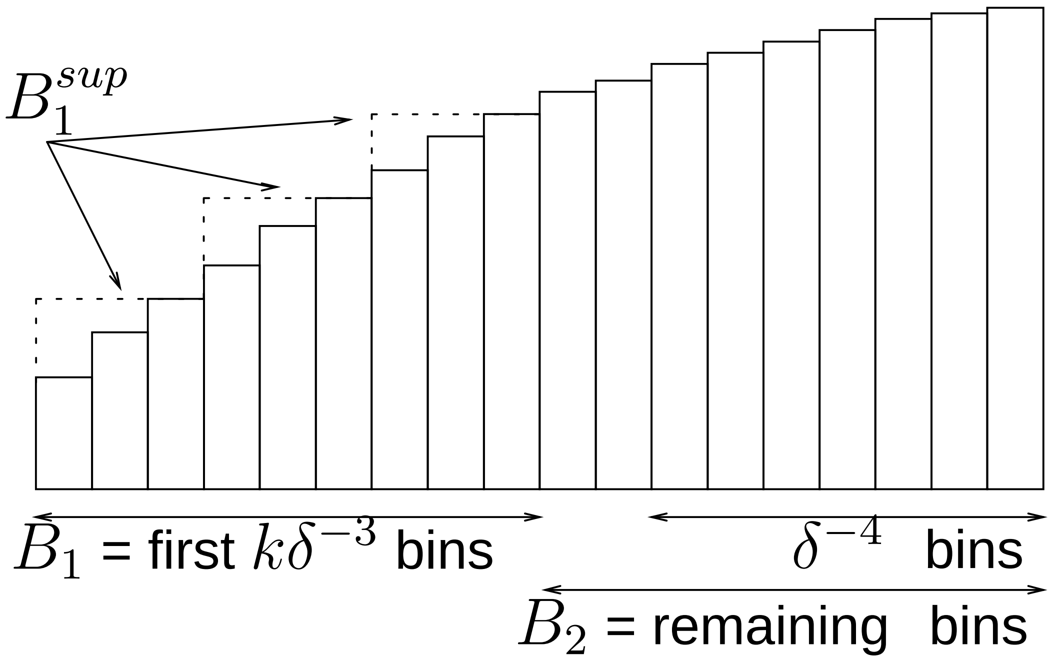
- Use fractional 1-dimensional knapsack to choose the remaining items with small profit and capacity $\sum_{i=1}^{\gamma} c(b_i) - size(A_{guess})$.

$\Longrightarrow$ at most γ fractional items

General Instances

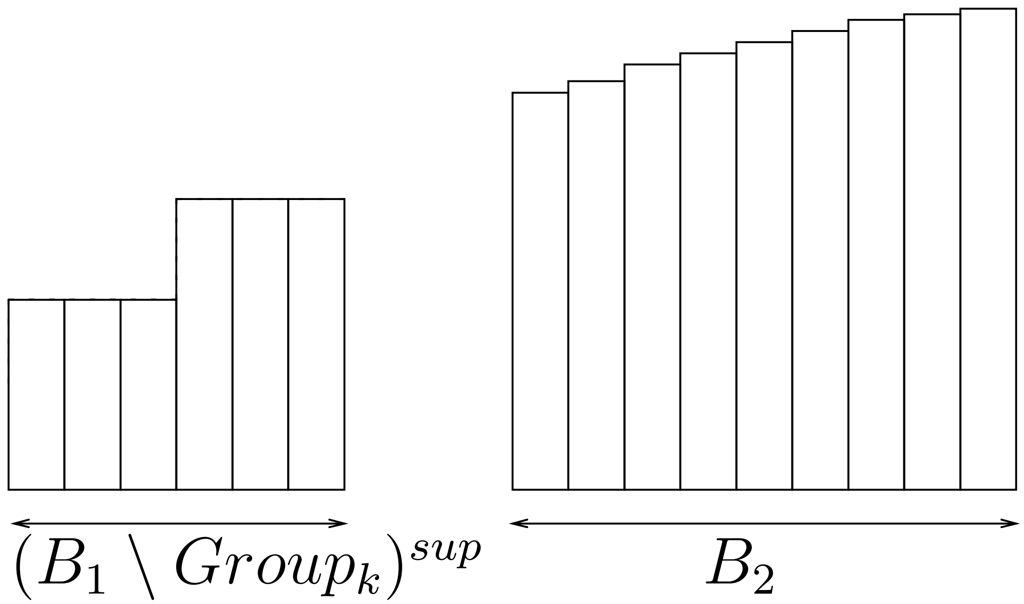


Rounding the bin capacities



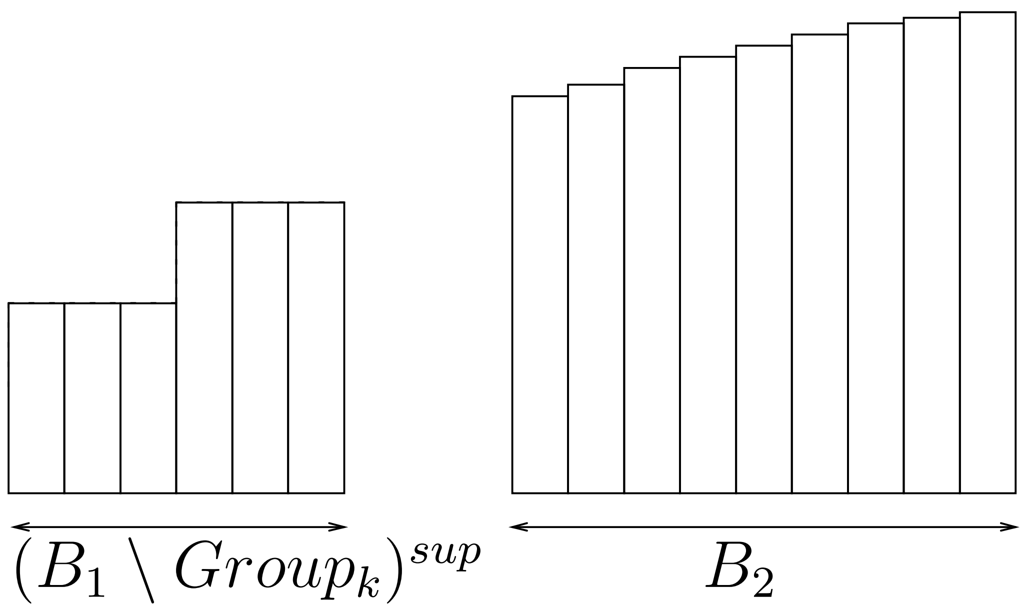
- Round up the capacity of each bin in the first k groups to the largest capacity in the group.

Rounding the bin capacities



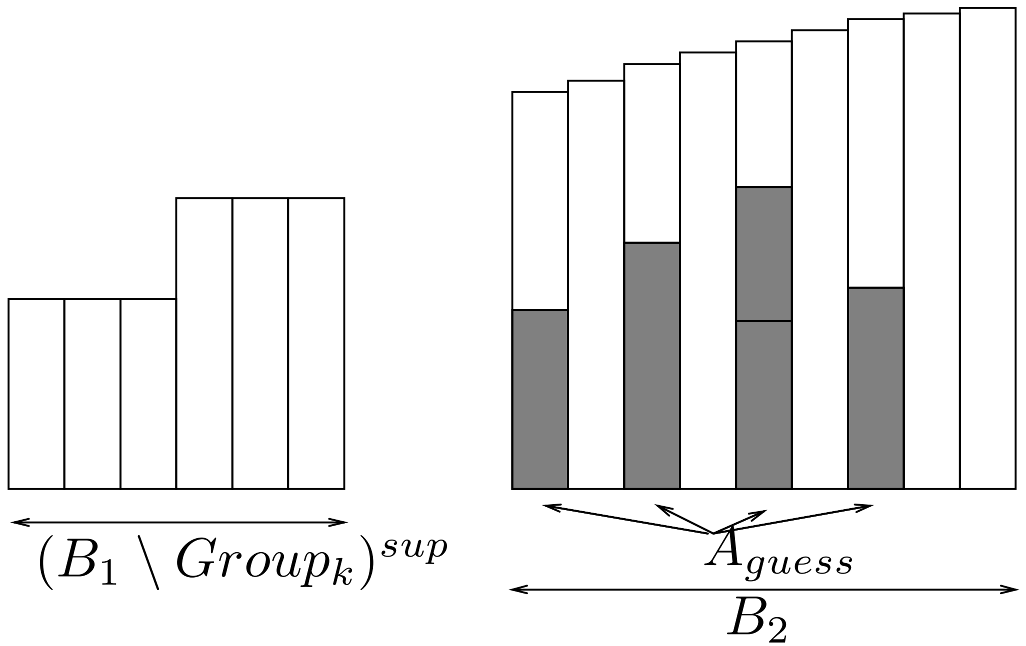
- Each optimum solution for instance (A, B) can be transformed into a solution for the rounded instance where the profit loss is $\leq \delta OPT(A, B)$.

Modified instance



- We obtain a modified instance $(A, B_1' \cup B_2)$ where B_1' consists of $(k-1)$ groups of $1/\delta^3$ bins with the same capacity and B_2 consists of $\leq 2\delta^{-4}$ bins.

Modified instance



1) Guess the high profit items to be placed into B_2.

2) Solve a modified linear program relaxation to select the other items.

Summary

there is an EPTAS for MKP **(Jansen, SODA 2009)** with running time

$$2^{O(1/\epsilon^5 \log(1/\epsilon))} poly(n).$$

New upcoming results I

1) there is an EPTAS for MKP **(Jansen, 2009)** with running time

$$2^{O(1/\epsilon^2 \log(1/\epsilon)^4)} poly(n)$$

(improving the SODA result).

New upcoming results II

2) there is an EPTAS for scheduling on uniform machines **(Jansen, ICALP 2009)** with running time

$$2^{O(1/\epsilon^2 \log(1/\epsilon)^3)} poly(n)$$

(improving the classical PTAS by Hochbaum and Shmoys 1988).

Open question

Is there a lower bound on the running time?

Use the exponential time hypothesis (ETH):

$$FPT \neq M[1]$$

or equivalently:

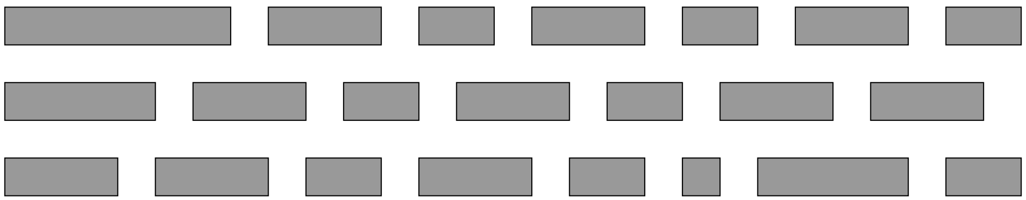
there is no algorithm for 3-SAT (with n variables)
with running time $2^{o(n)}$

Scheduling with Fixed Jobs

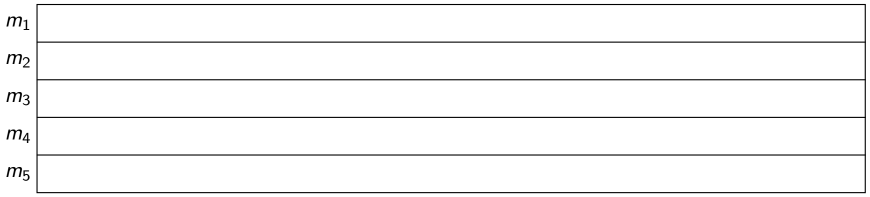
study classical non-preemptive scheduling problems

- sequential jobs
- identical parallel machines
- objective to minimize makespan

Classical scheduling



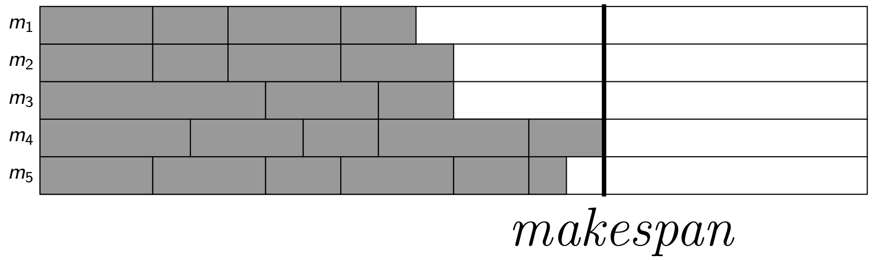
jobs to schedule



parallel machines

Classical scheduling

jobs to schedule



solution

resulting problems can be modeled as

- $\mathsf{P}m||C_{\mathsf{max}}$ (m constant)

  complexity NP-hard

  algorithm FPTAS **(Sahni 1976)**

- $\mathsf{P}||C_{\mathsf{max}}$ (m part of input)

  complexity strongly NP-hard

  algorithm PTAS **(Hochbaum & Shmoys 1988)**

  algorithm EPTAS **(Hochbaum & Shmoys 1988, Alon et al. 1997)** – running time doubly exponential in $1/\epsilon$

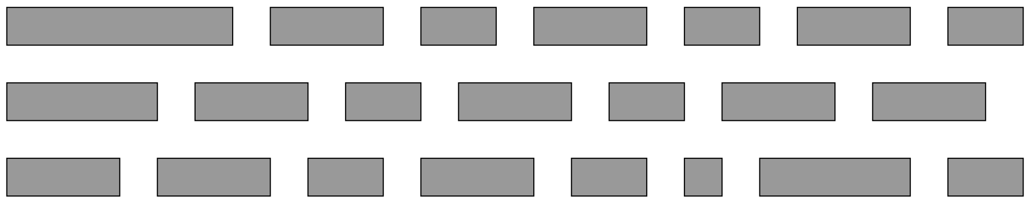
  algorithm EPTAS **(Jansen 2009)** – running time singly exponential in $1/\epsilon$

Problems

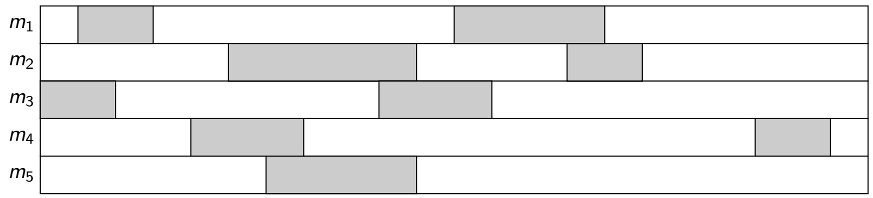
reality more difficult

- fixed jobs
  - high-priority system jobs
  - jobs of other users

Scheduling with fixed jobs



jobs to schedule



parallel machines with fixed jobs

Scheduling with fixed jobs

jobs to schedule

solution

Problem formally

given

- jobs $p_1, \ldots, p_n$
- number m of machines
- list $(m_1, s_1), \ldots, (m_k, s_k)$ fixing first k jobs

objective

- no preemption of jobs
- no overlap of jobs
- makespan C_{max} minimized

Related work

Scharbrodt, Steger & Weisser **(SODA 99, J'Sched 99)**

for m constant

complexity strongly NP-hard

algorithm PTAS

for m part of input

complexity no ratio better than $3/2$ unless $\mathsf{P} = \mathsf{NP}$

algorithm ratios of 3 and $2 + \epsilon$ (via a PTAS for $\mathsf{P}||C_{\mathsf{max}}$)

New results

for m part of input

algorithm ratio $3/2 + \epsilon$ **(Diedrich & Jansen, SODA 2009)**

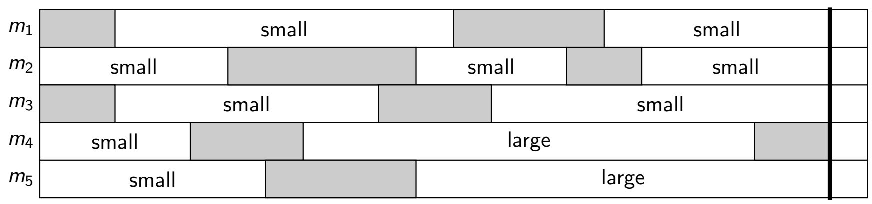
algorithm ratio $3/2$ **(Diedrich & Jansen, 2009)**

matches lower bound from SODA 1999

Used techniques

- classification of jobs and gaps
- cyclic rounding
- network flow
- (E)PTAS for MSSP (“multiple subset sum problem”)
- list scheduling

Classification of gaps

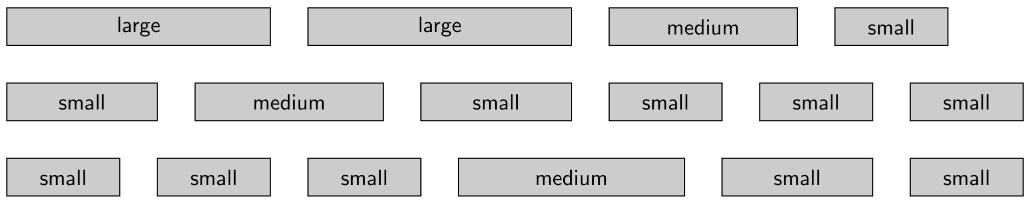


parallel machines with fixed jobs

large gap: at least $T/2$

small gap: smaller than $T/2$

Job classification



jobs to schedule

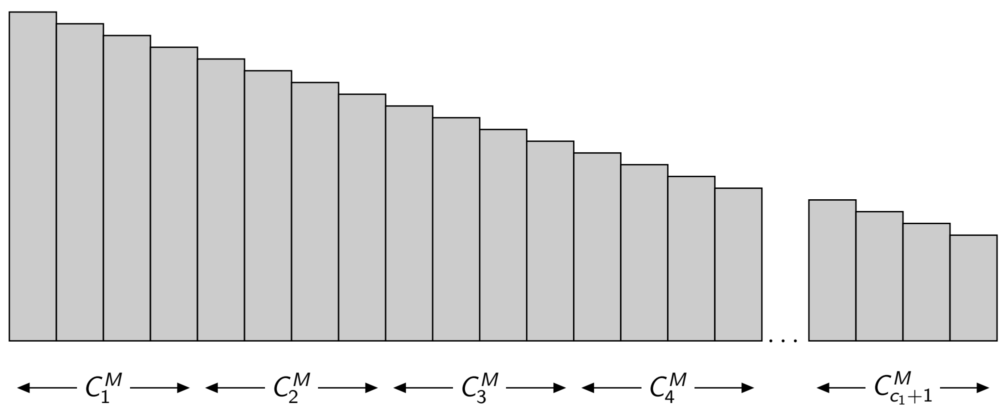
large job: more than $T/2$

medium job: between ϵT and $T/2$

small job: smaller than ϵT

Rounding medium jobs

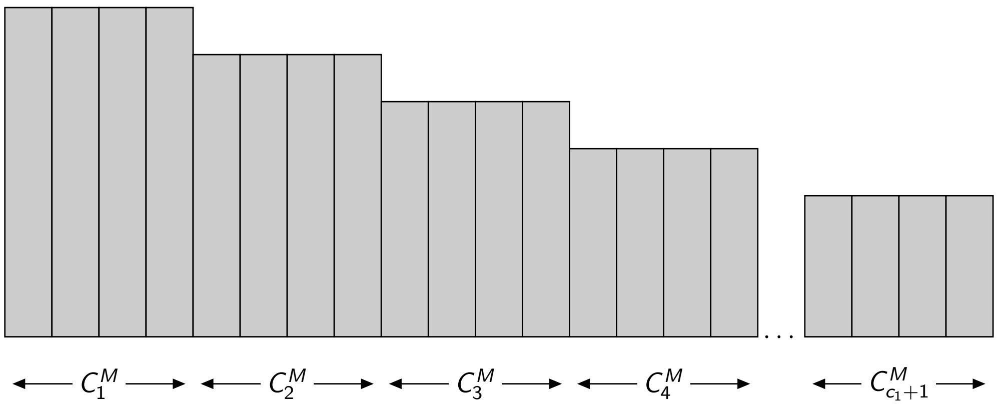
$$c_1 := \lceil 1/\epsilon^2 \rceil$$



arrange medium jobs in non-increasing order of processing time

Rounding medium jobs

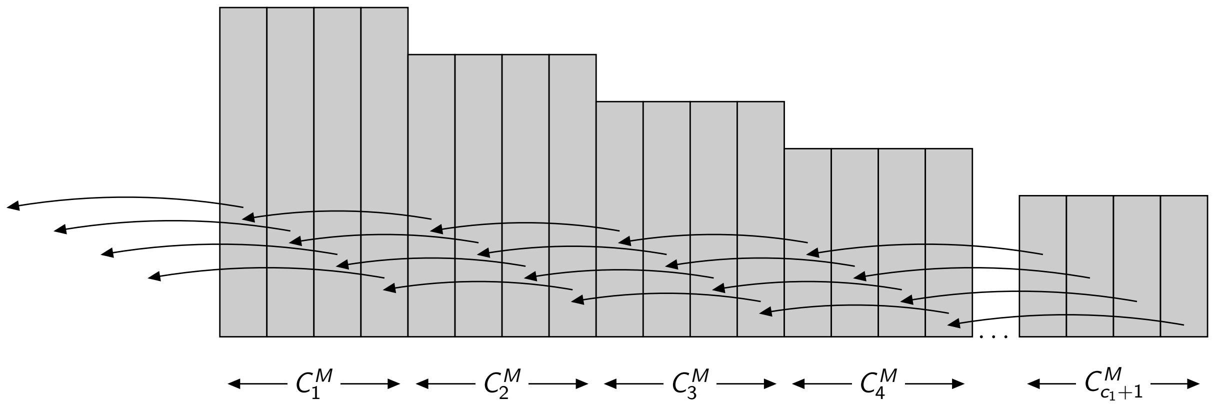
$$c_1 := \lceil 1/\epsilon^2 \rceil$$



round up each group to largest size

Rounding medium jobs

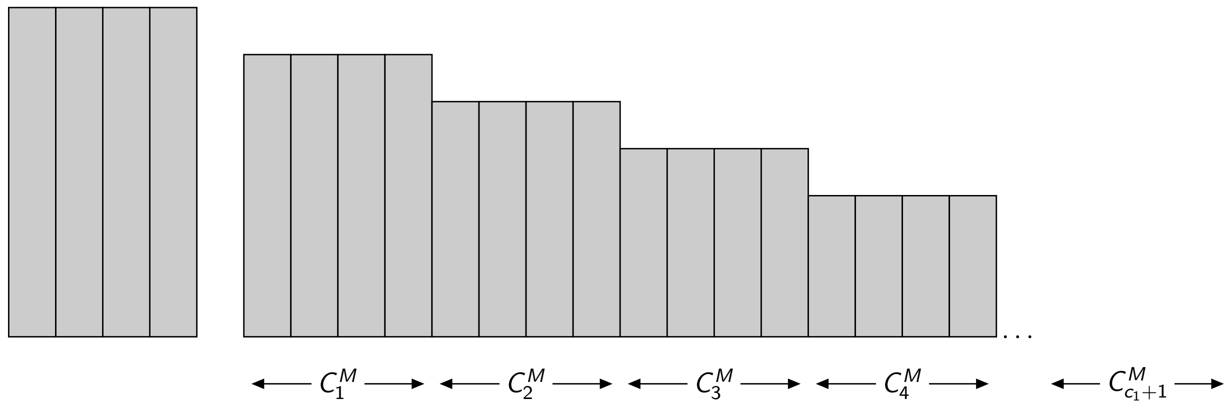
$c_1 := \lceil 1/\epsilon^2 \rceil$



embed each rounded group in previous group, except first group

Rounding medium jobs

$c_1 := \lceil 1/\epsilon^2 \rceil$



after rounding only first group is lost

Consequence

- only $c_1 = \lceil 1/\epsilon^2 \rceil$ sizes for medium jobs
- lose medium jobs of total processing time

$$P(C_1^M) \leq \frac{\epsilon T m}{2}$$

Configurations for gaps

- vector $(a_1, \ldots, a_{c_1})$ with $a_i \in \{0, \ldots, \lfloor 1/\epsilon \rfloor\}$
- models set of medium jobs which can occur together in a gap
- there are

$$c_2 \leq (\lfloor 1/\epsilon \rfloor + 1)^{c_1}$$

different configurations $\kappa^{(1)}, \ldots, \kappa^{(c_2)}$

Discretization of large jobs

Basic Idea: rounding up large jobs

packed together with a non-empty configuration

large job with configuration

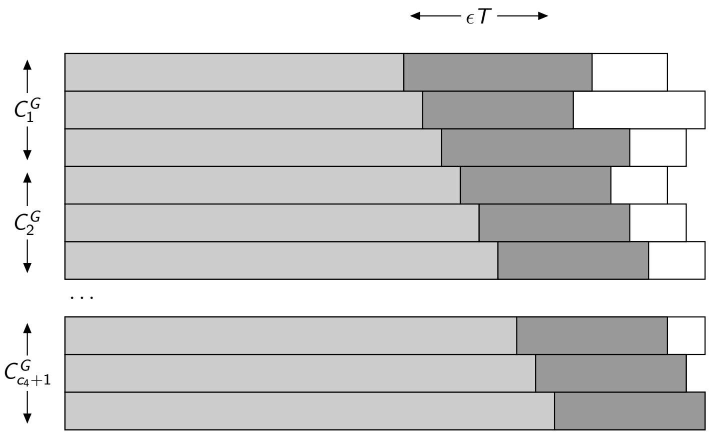
Cyclic shifting

consider large jobs with

- roughly equal size $\in (k\epsilon T, (k+1)\epsilon T]$
- packed together with a non-empty configuration

Cyclic shifting

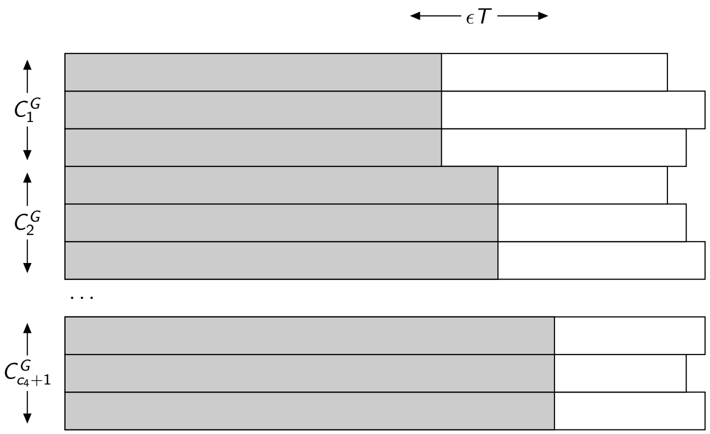
$$c_4 := \lceil 1/\epsilon \rceil$$



arrange gaps on stack, sorted by sizes of large jobs, create groups

Cyclic shifting

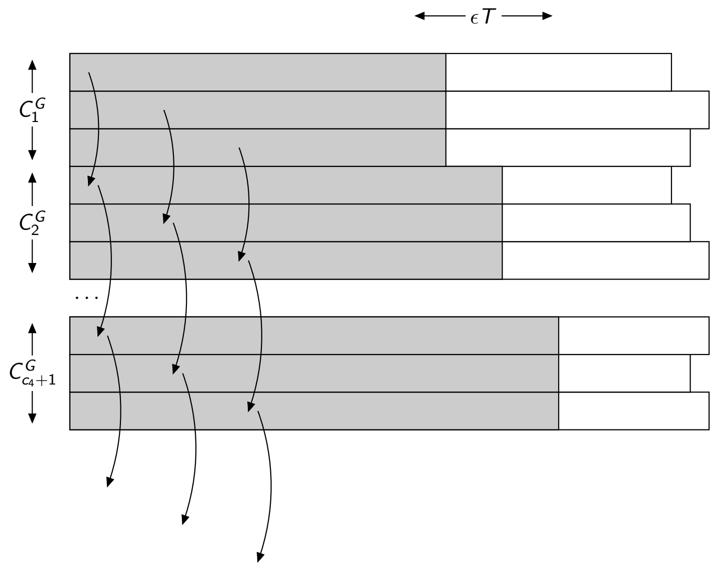
$c_4 := \lceil 1/\epsilon \rceil$



round up jobs in each group, configurations not shown

Cyclic shifting

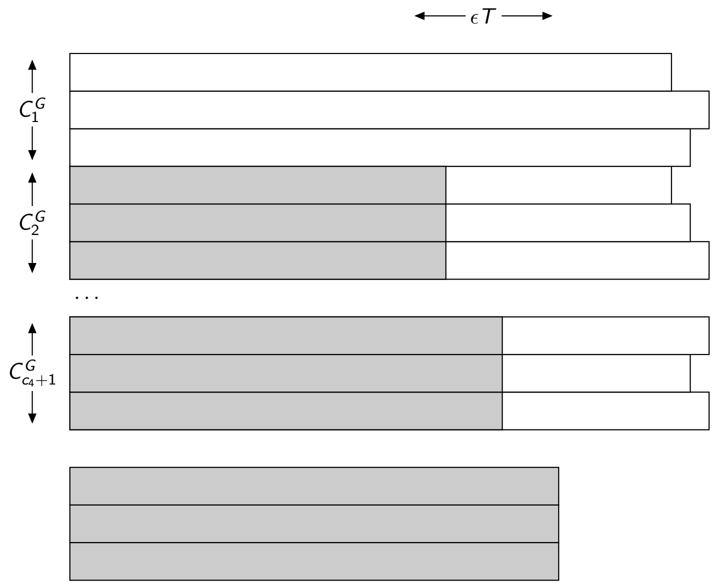
$c_4 := \lceil 1/\epsilon \rceil$



shift down rounded jobs, drop last jobs out of stack

Cyclic shifting

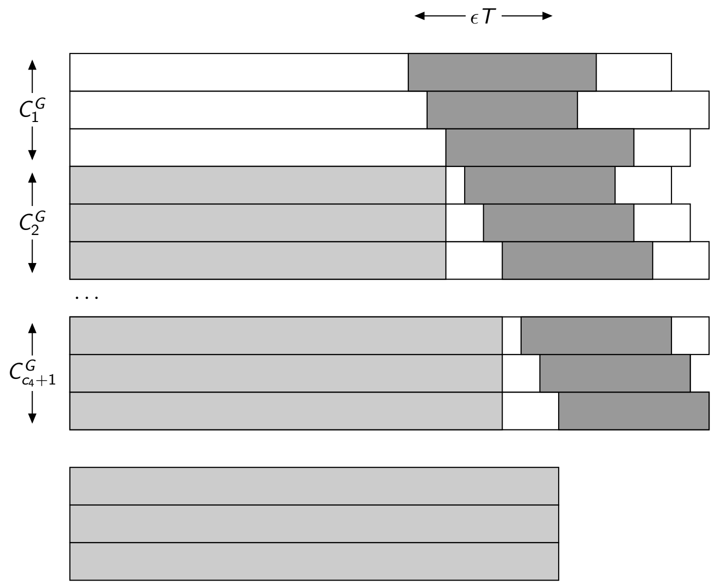
$$c_4 := \lceil 1/\epsilon \rceil$$



resulting arrangement

Cyclic shifting

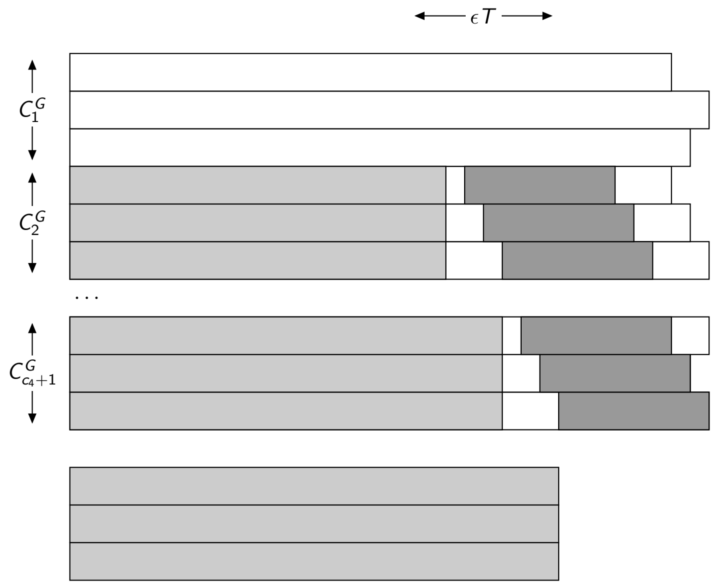
$$c_4 := \lceil 1/\epsilon \rceil$$



configurations shown again

Cyclic shifting

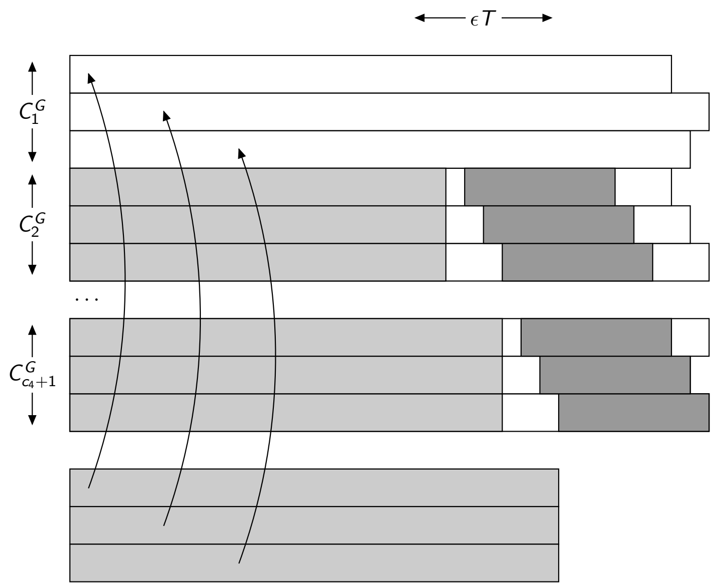
$$c_4 := \lceil 1/\epsilon \rceil$$



remove configurations in topmost gaps

Cyclic shifting

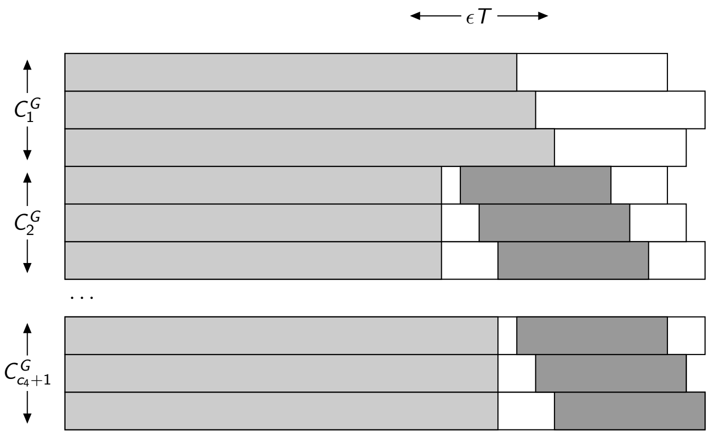
$c_4 := \lceil 1/\epsilon \rceil$



move large jobs from below stack in gaps

Cyclic shifting

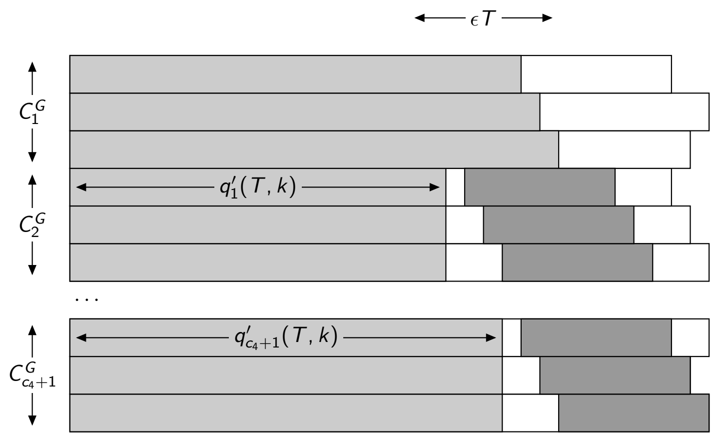
$c_4 := \lceil 1/\epsilon \rceil$



possible rounded large jobs packed, few configurations lost

Cyclic shifting

$c_4 := \lceil 1/\epsilon \rceil$



use $q_i'(T, k)$ to denote the rounded sizes of gaps let

- $G_L(T,k)$ – set of large gaps involved
- $I(T,k)$ – set of jobs lost by cyclic shifting

lose medium jobs of total processing time

$$P(I(T,k)) \leq \frac{T|G_L(T,k)|}{2c_4}$$

accumulating loss

over all k yields

$$P(\bigcup_{k=1}^{c_3} I(T,k)) = \sum_{k=1}^{c_3} P(I(T,k))$$

$$\leq T/(2c_4) \sum_{k=1}^{c_3} |G_L(T,k)| \leq \frac{T}{2c_4} |G_L(T)|$$

$$\leq \frac{Tm}{2c_4} \leq \frac{\epsilon Tm}{2}$$

Note

bad rounded sizes not known a priori

good for each k only $c_4 + 1$ values

$$q_i'(T, k) \text{ for } i \in \{1, \ldots, c_4 + 1\}$$

can hence be found

by enumeration

Schedule structure so far

- all small and large jobs scheduled
- in each large gap at most 3 objects
  - possibly rounded large job
  - configuration
  - set of small jobs
- $\forall\, k$, i, $\ell\, \exists$ integer $c(k, i, \ell) \leq m$

  indicating how often large job from interval k of rounded size $q_i'(T, k)$ packed together with configuration $\kappa^{(\ell)}$
- lost only medium jobs of processing time at most $\epsilon T m$

Main idea & network flow

for a fixed choice of values $c(k, i, \ell)$

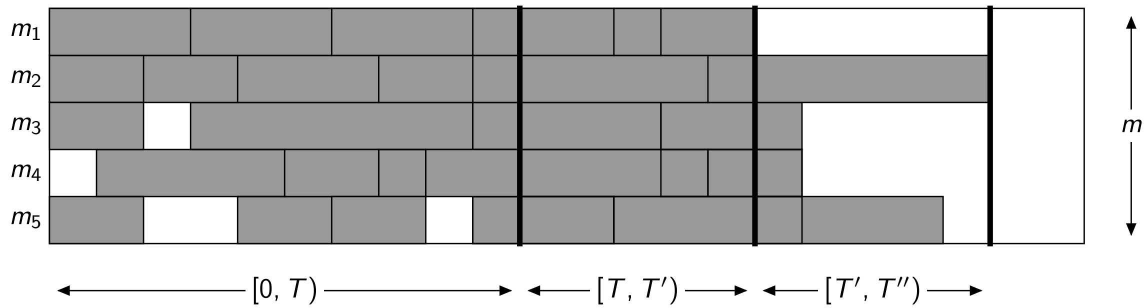
and rounded sizes $q'_i(T, k)$

- assignment of large jobs and configurations to gaps can be done via a network flow model
- assignment loses at most one small job per large gap
- total processing time of lost jobs at most $2\epsilon Tm$

final packing done in two steps

- use (E)PTAS for MSSP to fill almost all remaining jobs into $[0, T)$
  lose again at most $\epsilon T m$ processing time, hence
  total processing time lost at most $3\epsilon T m$
- remaining jobs are executed after T via list scheduling

List scheduling



structure of schedule

Graham-style analysis yields

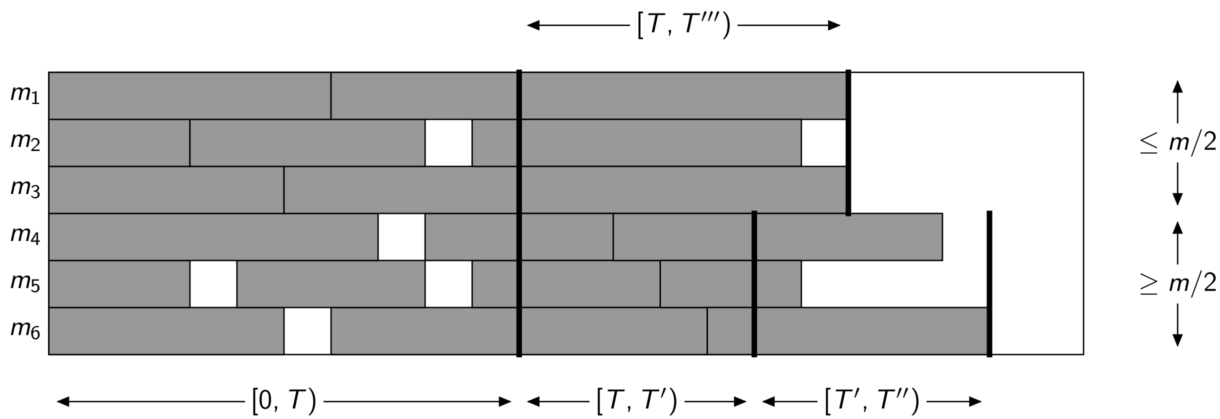
$$|[0,T)| + |[T,T')| + |[T',T'')|$$
$$\leq C^*_{\mathsf{max}} + 3\epsilon C^*_{\mathsf{max}} + \frac{1}{2} C^*_{\mathsf{max}} = (3/2 + 3\epsilon) C^*_{\mathsf{max}}$$

Final remark

for $\epsilon := 1/24$,

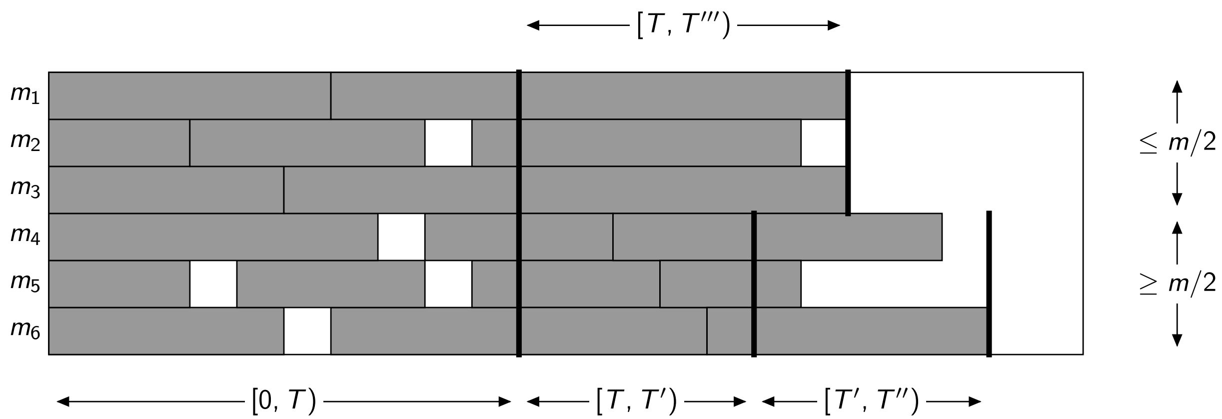
possible to modify list scheduling

to yield ratio of $3/2$



structure of schedule

- let n'' denote $\#$ non-scheduled jobs larger than $T/4$
- by using $\epsilon = 1/24$, this yields
  $n''T/4 \leq 3\epsilon Tm = Tm/8 \Rightarrow n'' \leq m/2$
- $|[T, T''')| \leq T/2$, since these jobs have medium sizes

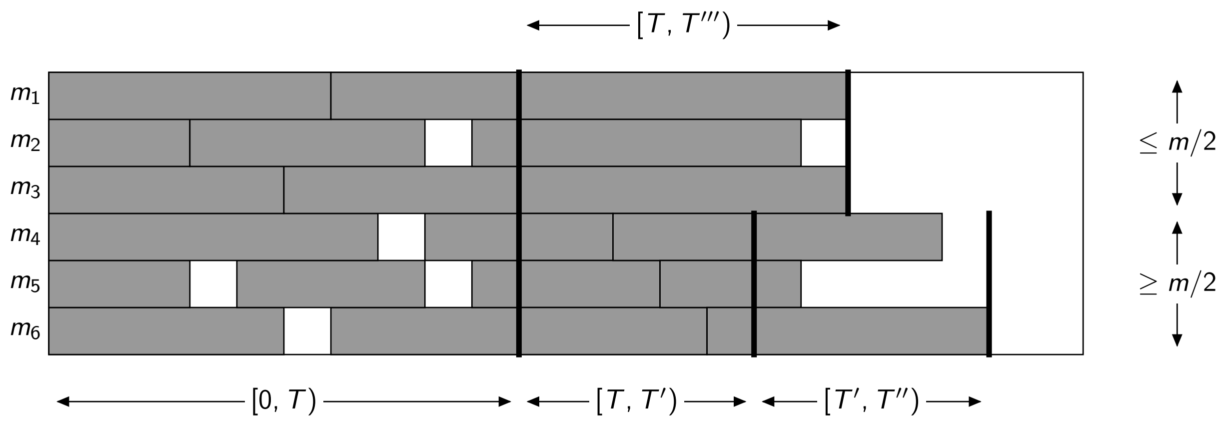


structure of schedule

- area argument:

  $$|[T, T')|m/2 \leq 3\epsilon Tm = Tm/8 \Rightarrow |[T, T')| \leq T/4$$

- there are only jobs of size $\leq T/4$, hence $|[T', T'')| \leq T/4$



structure of schedule

- in total $|[T, T')| \leq T/4$, $|[T', T'')| \leq T/4$ (Graham-style)

Summary

scheduling with fixed jobs

m part of input

complexity no ratio better than $3/2$ unless $\mathsf{P} = \mathsf{NP}$

algorithm approximation ratio of $3/2$

Open questions

(1) improvement of the running time of our algorithm via linear programming,

(2) lower bound on the running time of approximation algorithms with ratio $3/2$,

(3) more efficient approximation algorithms with ratio between $3/2$ and $2 + \epsilon$.