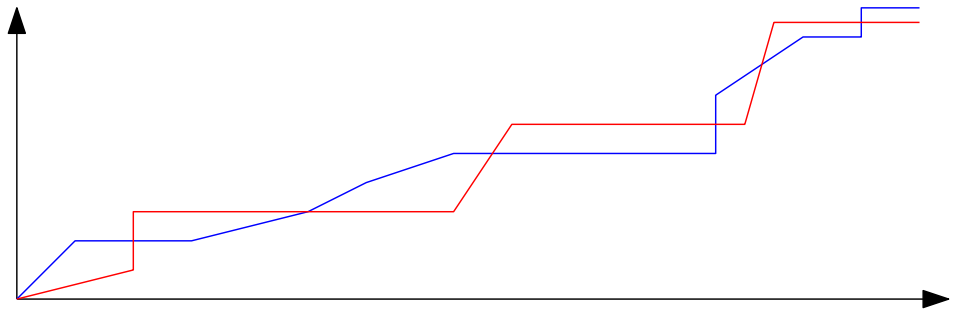
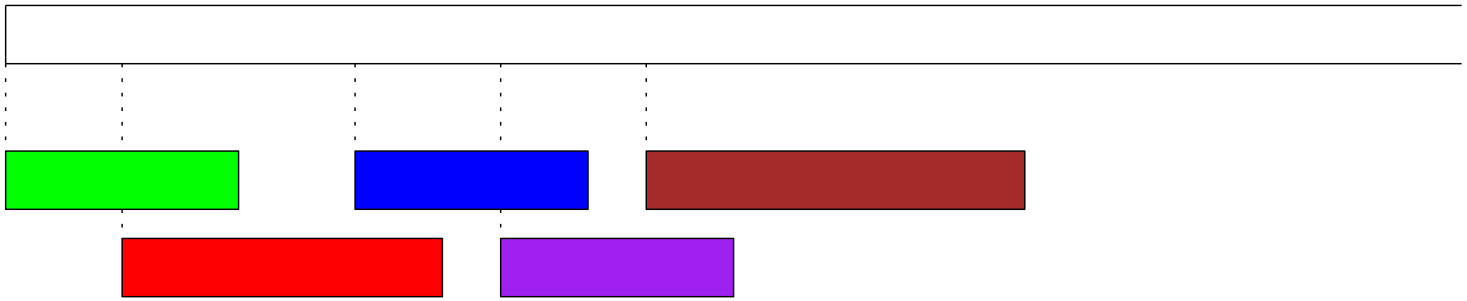
How Unsplittable-Flow-Covering helps Scheduling with Job-Dependent Cost Functions

Wiebke Höhn, Julián Mestre, and Andreas Wiese

TU Berlin, University of Sydney, MPI Saarbrücken



Scheduling with job-dependent cost functions



Problem setting

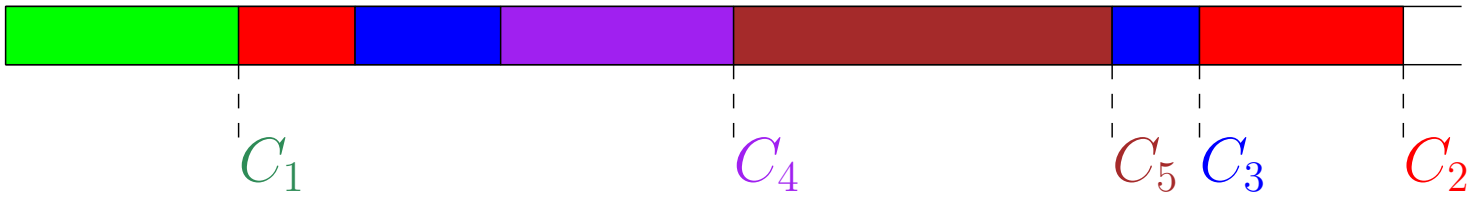
- One machine, jobs with release dates
- Preemption allowed

Scheduling with job-dependent cost functions

Problem setting

- One machine, jobs with release dates
- Preemption allowed

Scheduling with job-dependent cost functions



Problem setting

- One machine, jobs with release dates
- Preemption allowed
- Cost function: $\sum_j f_j(C_j)$

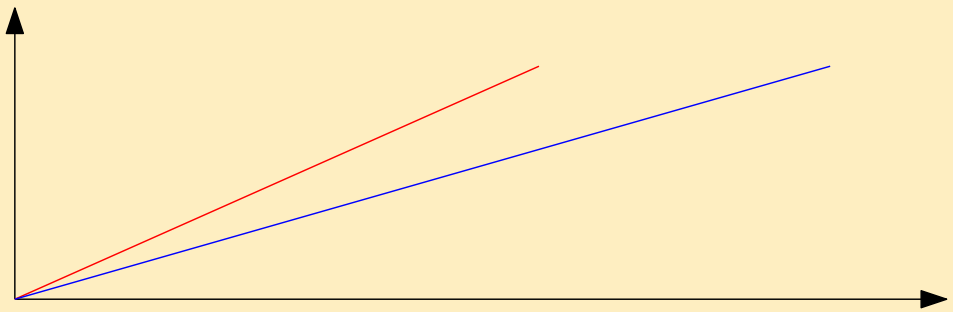
Examples

Problem setting

- One machine, jobs with release dates
- Preemption allowed
- Cost function: $\sum_j f_j(C_j)$

Weighted sum of completion time

- Cost function: $\sum_j w_j \cdot C_j$



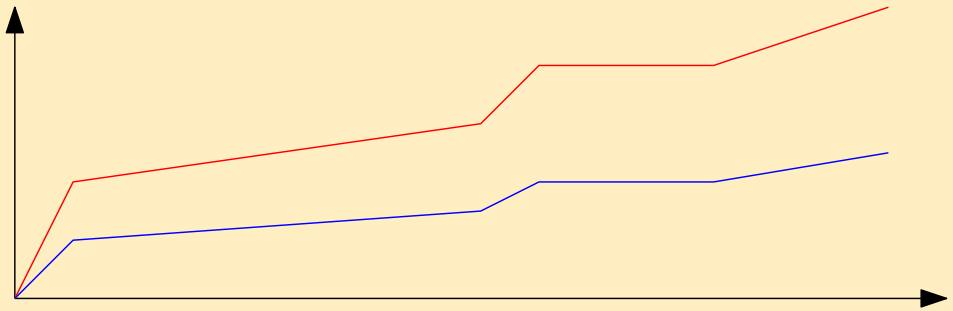
Examples

Problem setting

- One machine, jobs with release dates
- Preemption allowed
- Cost function: $\sum_j f_j(C_j)$

Weighted sum of completion time, speed varying machine

- Cost function: $\sum_j w_j \cdot g(C_j)$



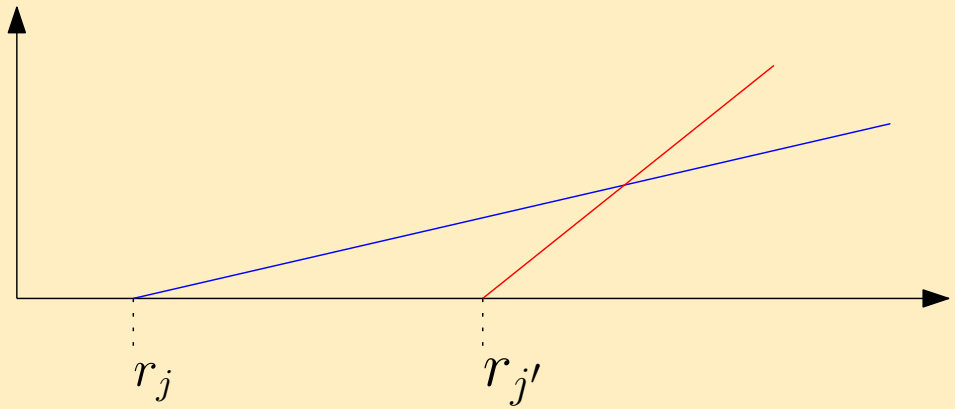
Examples

Problem setting

- One machine, jobs with release dates
- Preemption allowed
- Cost function: $\sum_j f_j(C_j)$

Weighted flow time

- Cost function: $\sum_j w_j \cdot (C_j - r_j)$



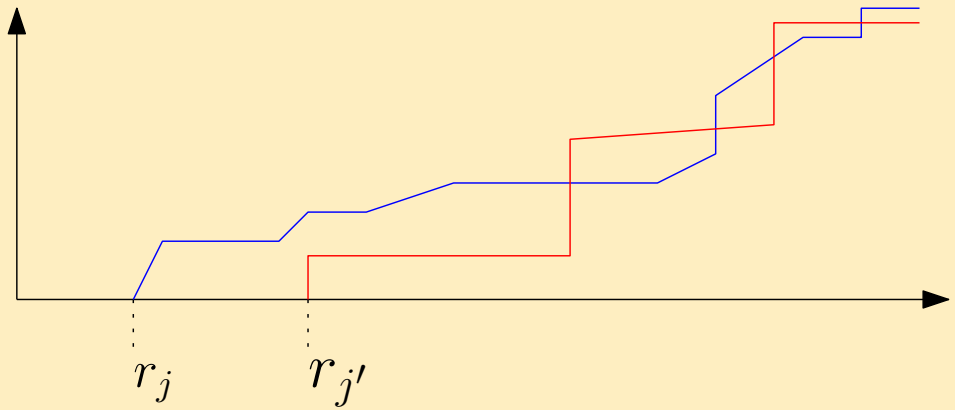
Examples

Problem setting

- One machine, jobs with release dates
- Preemption allowed
- Cost function: $\sum_j f_j(C_j)$

General case: general scheduling problem (GSP)

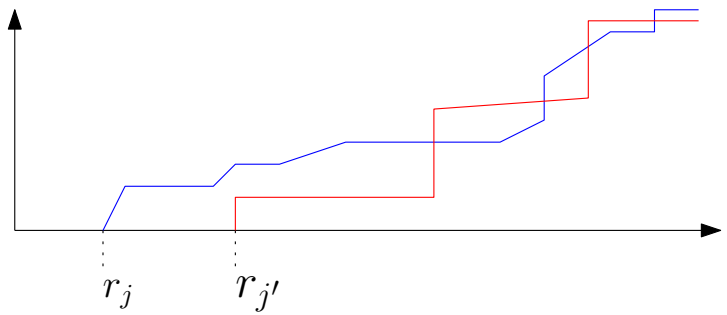
- Cost function: $\sum_j f_j(C_j)$



Previous work

With release dates

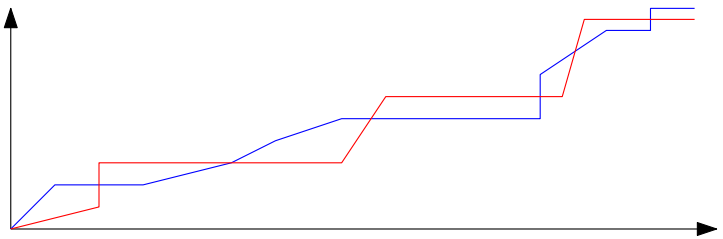
- Weighted sum of completion time
  - $(1+\varepsilon)$-approximation [Afrati et al., FOCS 1999]
- Weighted flow time
  - $O(\log^2 P)$ [Chekuri, Khanna, Zhu, STOC 2001]
  - $O(\log W)$, $O(\log nP)$ [Bansal and Dhamdhere, SODA 2003]
  - QPTAS [Chekuri and Khanna, STOC 2002]
- General case
  - $O(\log\log P)$-approximation [Bansal and Pruhs, FOCS 2010, ESA 2012]



Previous work

Without release dates

- Weighted sum of completion time = Weighted flow time
  - Polytime solvable, order jobs by Smith's rule w_j/p_j
- Speed varying machine $\sum_j w_j \cdot g(C_j)$
  - NP-hard [Wang, Sun, Chu, Ann. Oper. Res 2005]
  - PTAS [Megow and Verschae, ICALP 2013]
- General case
  - 16-approximation [Bansal and Pruhs, FOCS 2010]
  - $(4+\varepsilon)$-approximation [Cheung and Shmoys, APPROX 2011][Mestre and Verschae 2013]



Our results

State of the art

- Speed varying machine $\sum_j w_j \cdot g(C_j)$
  - PTAS [Megow and Verschae, ICALP 2013]
- General case
  - $(4+\varepsilon)$-approximation [Cheung and Shmoys, APPROX 2011][Mestre and Verschae 2013]

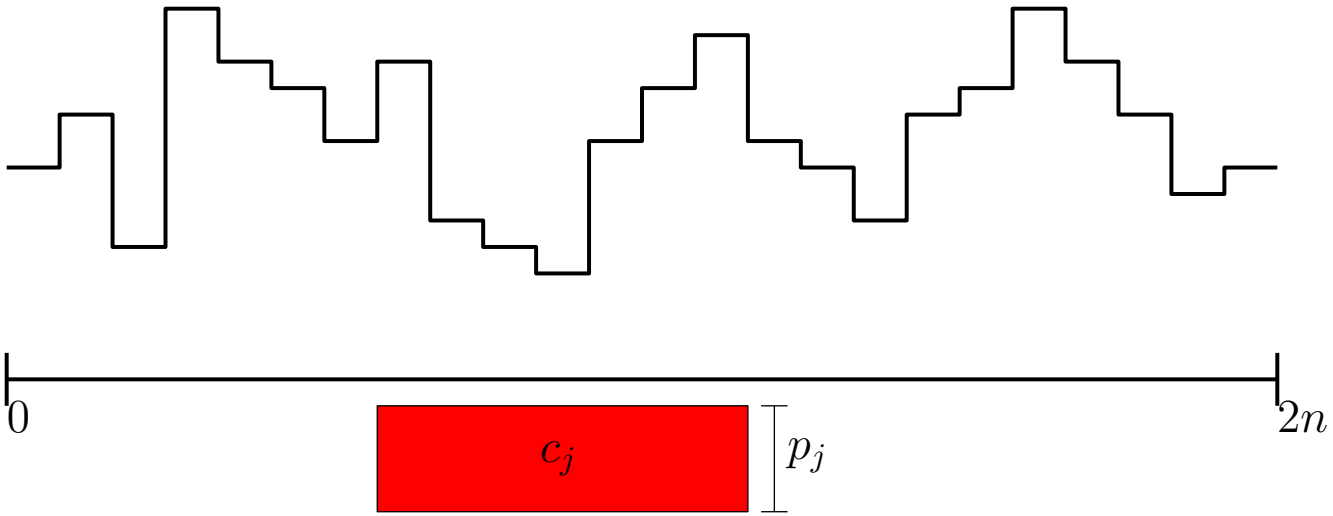
Our contributions

- $(e+\varepsilon)$-approximation in quasi-polynomial time: $n^{(\log n)^{O(1)}}$
- QPTAS if $f_j(C_j) \in \{0, c_j, \infty\}$
- QPTAS for few classes of cost functions $\sum_j w_j \cdot g_{\ell(j)}(C_j)$, $\ell(j) \in [\log n]$
- $1+\varepsilon$ speedup: schedule with *optimal* cost in polynomial time

Unsplittable Flow Covering

Problem definition

- time axis,
- intervals $[t, t+1)$ have demand d_t
- tasks j with
  - start- and end times
  - size p_j
  - cost c_j



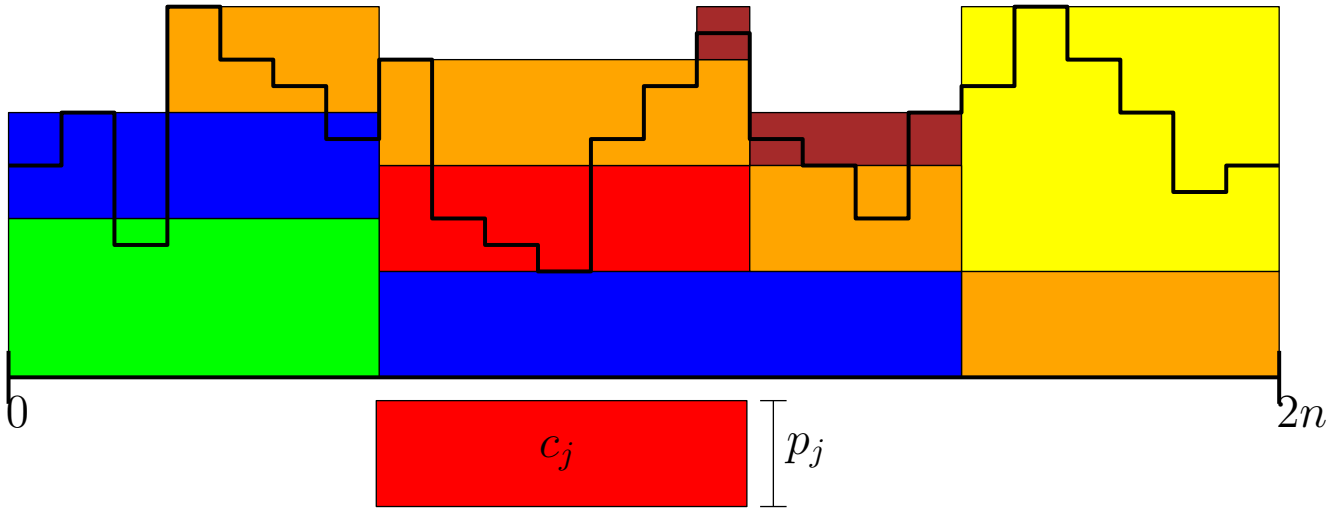
Unsplittable Flow Covering

Problem definition

- time axis,
- intervals $[t, t+1)$ have demand d_t
- tasks j with
  - start- and end times
  - size p_j
  - cost c_j

Goal

- select subset of tasks T'
- cover demands:
  $\sum_{j \in T' \text{ covers } t} p_j \geq d_t \quad \forall t$
- minimize total cost



Unsplittable Flow Covering

Problem definition

- time axis,
- intervals $[t, t+1)$ have demand d_t
- tasks j with
  - start- and end times
  - size p_j
  - cost c_j

Goal

- select subset of tasks T'
- cover demands:
  $\sum_{j \in T' \text{ covers } t} p_j \geq d_t \quad \forall t$
- minimize total cost

Lemma (Bansal and Verschae)

Special case where $f_j(C_j) \in \{0, c_j, \infty\}$ is equivalent to UFP-covering.

Unsplittable Flow Covering

Problem definition

- time axis,
- intervals $[t, t+1)$ have demand d_t
- tasks j with
  - start- and end times
  - size p_j
  - cost c_j

Goal

- select subset of tasks T'
- cover demands:
  $\sum_{j \in T' \text{ covers } t} p_j \geq d_t \quad \forall t$
- minimize total cost

Lemma (Bansal and Verschae)

Special case where $f_j(C_j) \in \{0, c_j, \infty\}$ is equivalent to UFP-covering.

Lemma

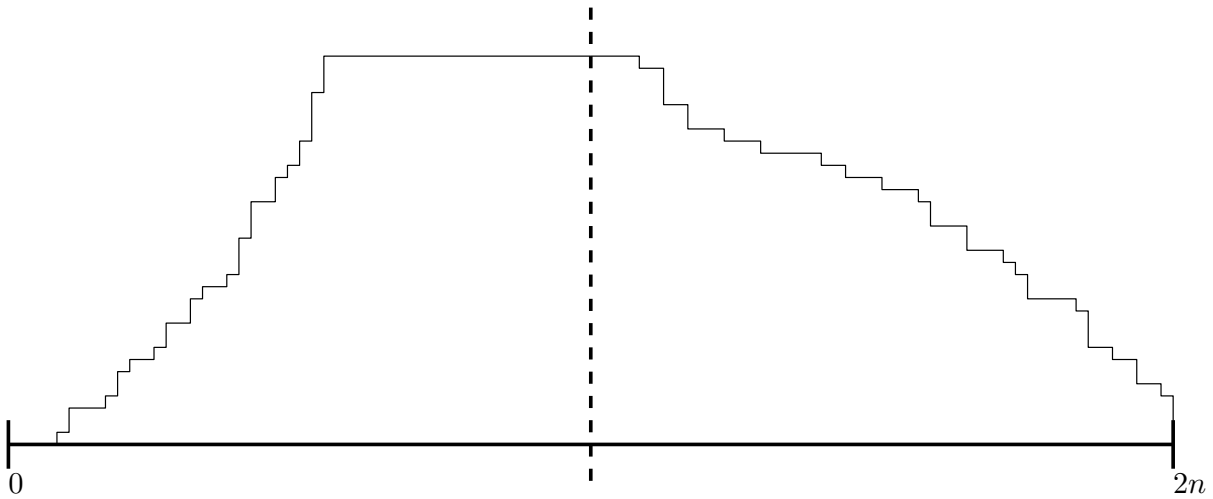
Lose factor $e \approx 2.718 \rightsquigarrow$ reduce to special case $f_j(C_j) \in \{0, c_j, \infty\}$.

QPTAS for UFP-covering

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

group tasks with almost same size and cost $\rightsquigarrow$ $O(\log^2 n)$ groups

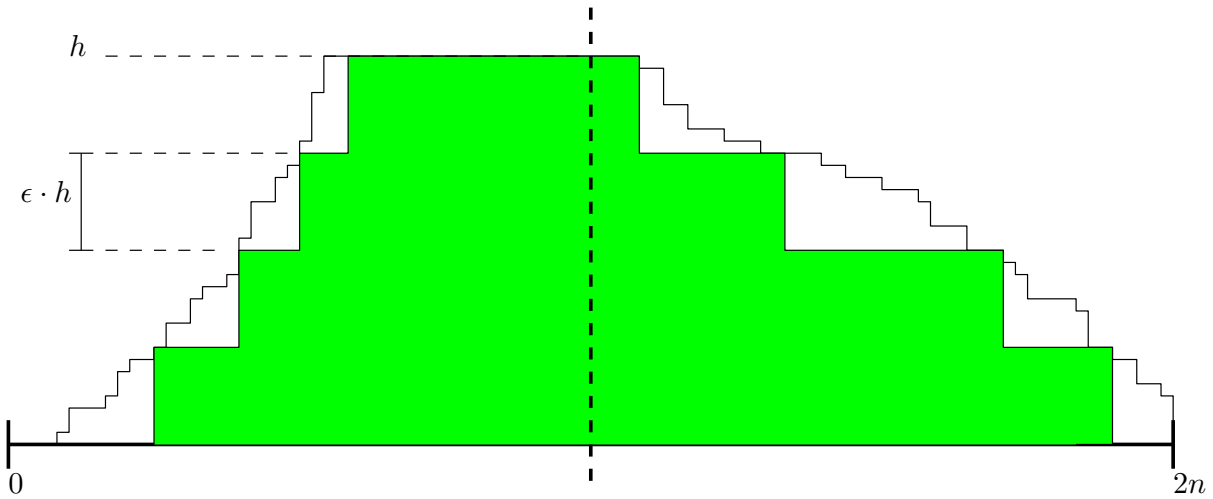


QPTAS for UFP-covering

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

group tasks with almost same size and cost $\rightsquigarrow$ $O(\log^2 n)$ groups

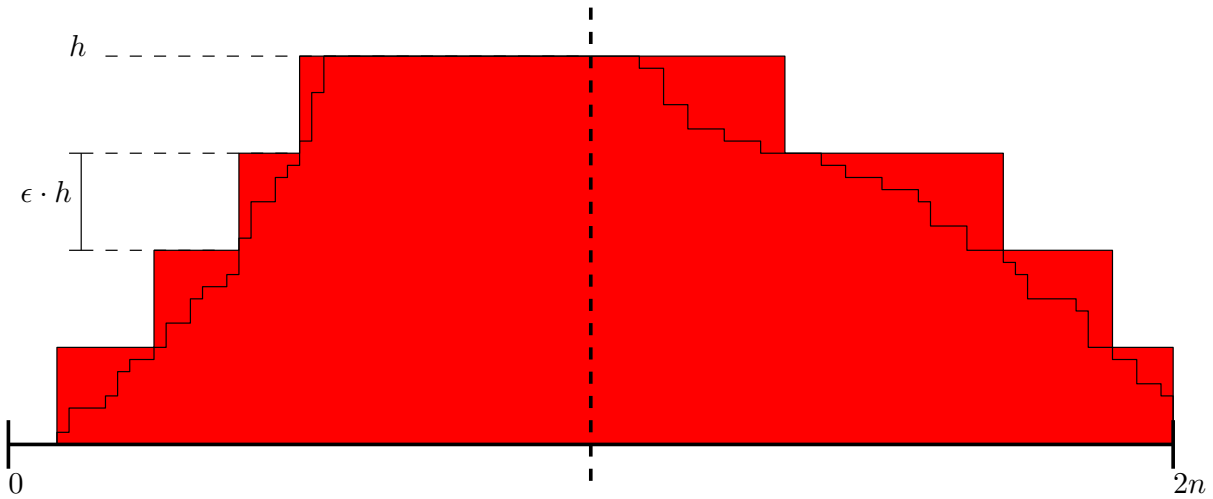


QPTAS for UFP-covering

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

group tasks with almost same size and cost $\rightsquigarrow$ $O(\log^2 n)$ groups

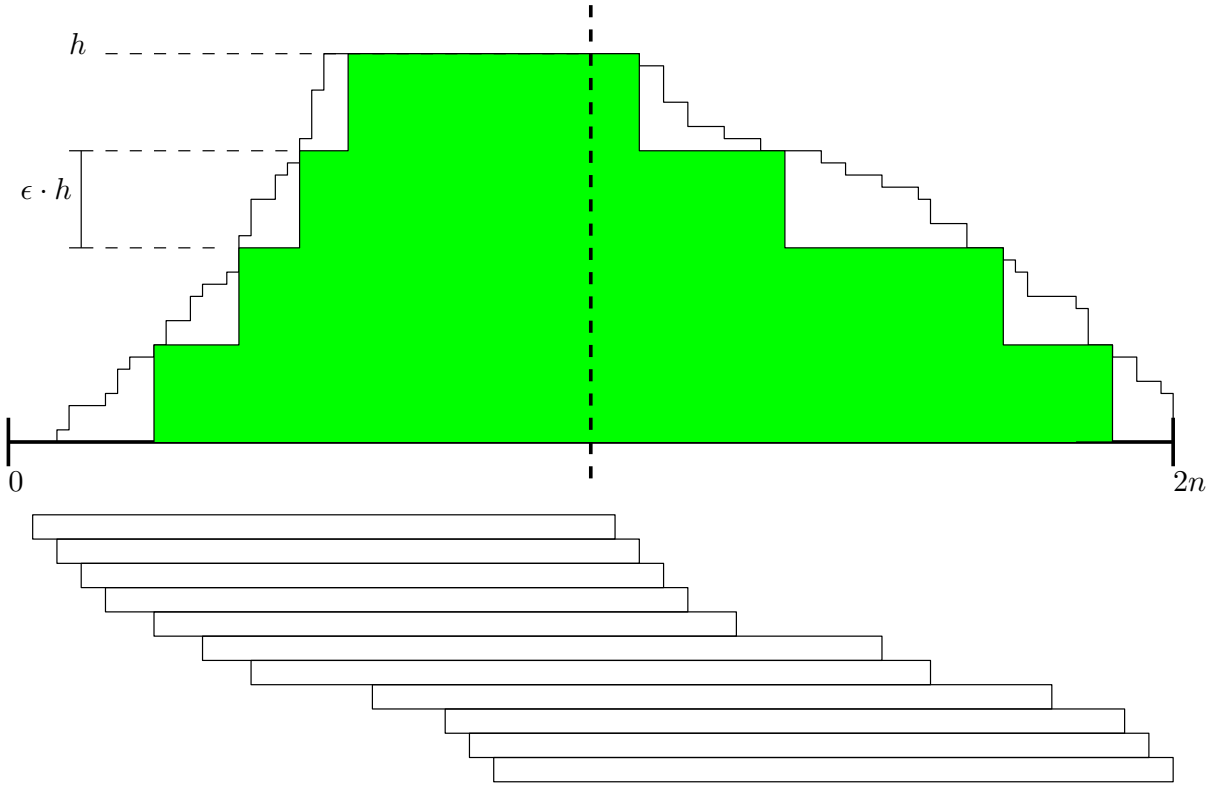


QPTAS for UFP-covering

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

group tasks with almost same size and cost $\rightsquigarrow$ $O(\log^2 n)$ groups

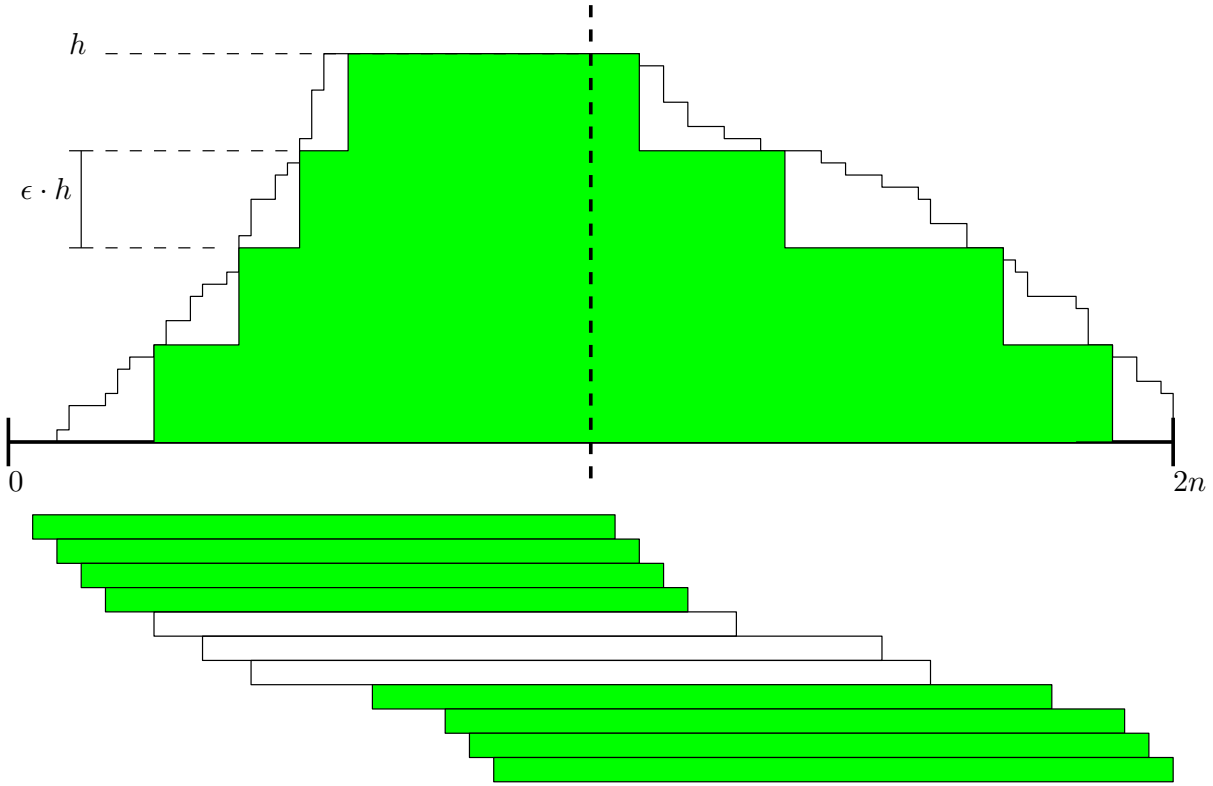


QPTAS for UFP-covering

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

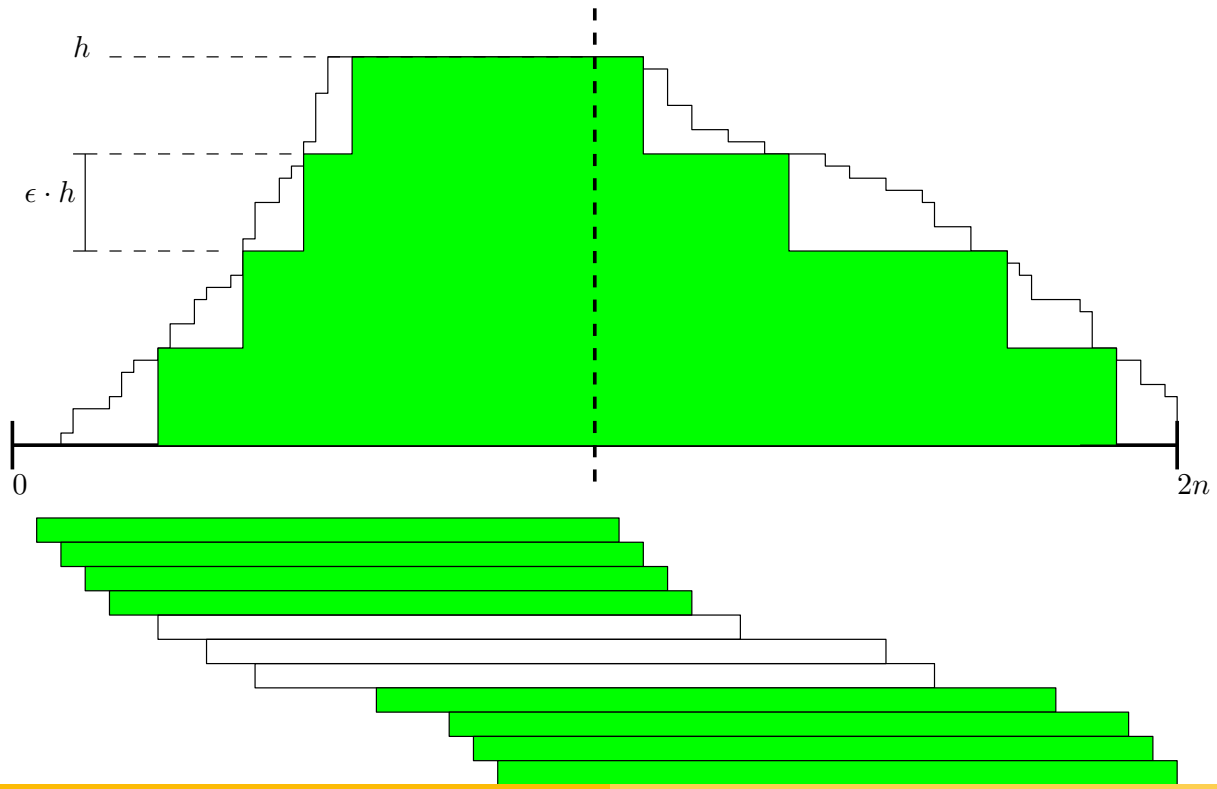
group tasks with almost same size and cost $\rightsquigarrow$ $O(\log^2 n)$ groups



Analysis

Running time

- guessing for one group: $n^{O(1/\varepsilon)}$ possibilities
- $O(\log^2 n)$ groups: $n^{O(\frac{1}{\varepsilon}\log^2 n)}$ possibilities
- recursion depth $O(\log n) \rightsquigarrow$ running time of $n^{O(\frac{1}{\varepsilon}\log^3 n)}$ overall



Analysis

Running time

- guessing for one group: $n^{O(1/\varepsilon)}$ possibilities
- $O(\log^2 n)$ groups: $n^{O(\frac{1}{\varepsilon}\log^2 n)}$ possibilities
- recursion depth $O(\log n) \rightsquigarrow$ running time of $n^{O(\frac{1}{\varepsilon}\log^3 n)}$ overall

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

Analysis

Running time

- guessing for one group: $n^{O(1/\varepsilon)}$ possibilities
- $O(\log^2 n)$ groups: $n^{O(\frac{1}{\varepsilon}\log^2 n)}$ possibilities
- recursion depth $O(\log n) \rightsquigarrow$ running time of $n^{O(\frac{1}{\varepsilon}\log^3 n)}$ overall

Theorem

There is a QPTAS for UFP-covering if task-sizes are in a polynomial range.

Corollary

There is a $(e+\varepsilon)$-approximation algorithm with quasi-polynomial running time for GSP, if the job-lengths are in a polynomial range.

Generalize machine with varying speed

Theorem (Megow and Verschae, ICALP 2013)

There is a polynomial time $(1+\varepsilon)$*-approximation algorithm for the objective function* $\sum_{j \in J} w_j \cdot g(C_j)$.

Generalize machine with varying speed

Theorem (Megow and Verschae, ICALP 2013)

There is a polynomial time $(1+\varepsilon)$*-approximation algorithm for the objective function* $\sum_{j \in J} w_j \cdot g(C_j)$.

Theorem

There is a quasi-polynomial time $(1+\varepsilon)$*-approximation algorithm for minimizing* $\sum_{j \in J_1} w_j \cdot g_1(C_j) + \sum_{j \in J_2} w_j \cdot g_2(C_j) + \ldots + \sum_{j \in J_{\log n}} w_j \cdot g_{\log n}(C_j)$

Generalize machine with varying speed

Theorem (Megow and Verschae, ICALP 2013)

There is a polynomial time $(1+\varepsilon)$*-approximation algorithm for the objective function* $\sum_{j \in J} w_j \cdot g(C_j)$.

Theorem

There is a quasi-polynomial time $(1+\varepsilon)$*-approximation algorithm for minimizing* $\sum_{j \in J_1} w_j \cdot g_1(C_j) + \sum_{j \in J_2} w_j \cdot g_2(C_j) + \ldots + \sum_{j \in J_{\log n}} w_j \cdot g_{\log n}(C_j)$ *with up to* $O(\log n)$ *distinct release dates.*

Generalize machine with varying speed

Theorem (Megow and Verschae, ICALP 2013)

There is a polynomial time $(1+\varepsilon)$*-approximation algorithm for the objective function* $\sum_{j \in J} w_j \cdot g(C_j)$.

Theorem

There is a quasi-polynomial time $(1+\varepsilon)$*-approximation algorithm for minimizing* $\sum_{j \in J_1} w_j \cdot g_1(C_j) + \sum_{j \in J_2} w_j \cdot g_2(C_j) + \ldots + \sum_{j \in J_{\log n}} w_j \cdot g_{\log n}(C_j)$ *with up to* $O(\log n)$ *distinct release dates.*

The algorithm

- Round cost functions: $O(\log n)$ values each
- Guess $(\log n)^{O(1)}/\varepsilon$ most expensive jobs
- Remaining problem admits sparse LP

Speedup

Theorem

There is a polynomial time algorithm for GSP computing a solution

- *with optimal cost*
- *which is feasible with* $1+\varepsilon$ *speedup.*

Speedup

Theorem

There is a polynomial time algorithm for GSP computing a solution

- *with optimal cost*
- *which is feasible with* $1+\varepsilon$ *speedup.*

The algorithm

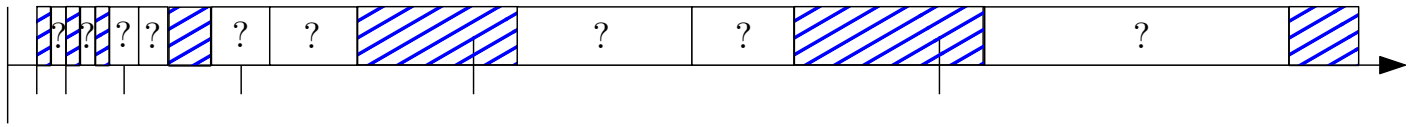
- Discretize time horizon into intervals
- Guess patterns for big jobs
- LP assigns jobs to slots of big jobs or idle space for small jobs

Speedup

Theorem

There is a polynomial time algorithm for GSP computing a solution

- *with optimal cost*
- *which is feasible with* $1+\varepsilon$ *speedup.*



The algorithm

- Discretize time horizon into intervals
- Guess patterns for big jobs
- LP assigns jobs to slots of big jobs or idle space for small jobs

Speedup

Theorem

There is a polynomial time algorithm for GSP computing a solution

- *with optimal cost*
- *which is feasible with* $1+\varepsilon$ *speedup.*

The algorithm

- Discretize time horizon into intervals
- Guess patterns for big jobs
- LP assigns jobs to slots of big jobs or idle space for small jobs

Summary

Studied setting

- Jobs with general cost functions
- One machine, identical release dates

Our results

- $(e+\varepsilon)$-approximation in quasi-polynomial time
- QPTAS for UFP-cover a.k.a. $f_j(C_j) \in \{0, c_j, \infty\}$
- QPTAS for generalized machine with varying speeds
- $(1+\varepsilon)$-speedup: schedule with optimal cost in polytime