A Practical 73/50 Approximation for Contiguous Monotone Moldable Job Scheduling

Klaus Jansen Felix Ohnesorge

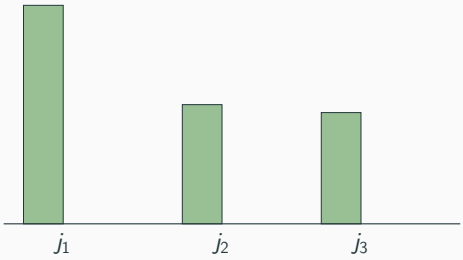
2026

Kiel University

Moldable Scheduling

Problem $(P|\text{fnct}_j|C_{max})$

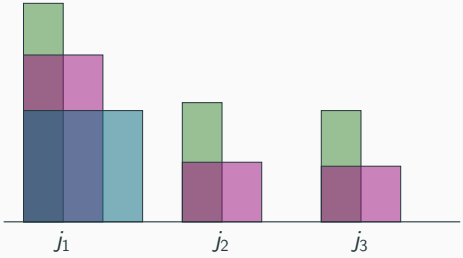
- $m \in \mathbb{N}$ *identical machines.*
- $n \in \mathbb{N}$ *jobs.*



Moldable Scheduling

Problem $(P|\text{fnct}_j|C_{max})$

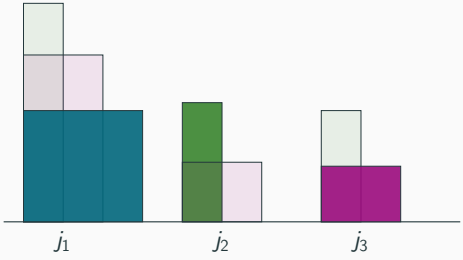
- $m \in \mathbb{N}$ *identical machines.*
- $n \in \mathbb{N}$ *jobs.*
- *Processing time* $p_j : [m] \to \mathbb{Q}_{>0}$
  - *Work function* $w_j : [m] \to \mathbb{Q}_{>0}$



Moldable Scheduling

Problem $(P|\text{fnct}_j|C_{max})$

- $m \in \mathbb{N}$ *identical machines.*
- $n \in \mathbb{N}$ *jobs.*
- *Processing time* $p_j : [m] \to \mathbb{Q}_{>0}$
  - *Work function* $w_j : [m] \to \mathbb{Q}_{>0}$
- *Objective:* **Minimize makespan**
  - $C_{max} = \max_{i \in [m]} C_i$

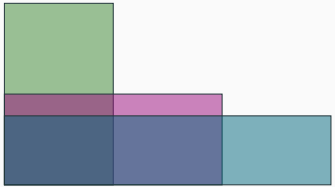


Schedule on m machines

Monotony Assumption

The **processing-time function** p_j is non-increasing, for all $j \in [n]$.

- $k \leq k' \Rightarrow t_j(k) \geq t_j(k')$.



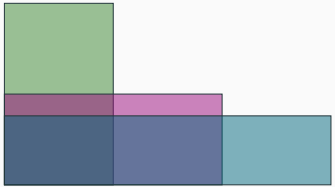
Monotony Assumption

The **processing-time function** p_j is non-increasing, for all $j \in [n]$.

- $k \leq k' \Rightarrow t_j(k) \geq t_j(k')$.

The **work function** $w_j : [m] \to \mathbb{Q}_{>0}$ is non-decreasing, for all $j \in [n]$.

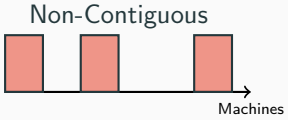
- $k \leq k' \Rightarrow w_j(k) \leq w_j(k')$.



The Contiguous Variant

Definition:

- Each job j must be assigned to a set of **adjacent** machines $\{i, i+1, \ldots, i+k-1\}$.



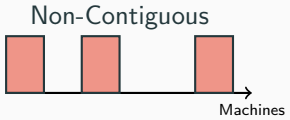
The Contiguous Variant

Definition:

- Each job j must be assigned to a set of **adjacent** machines $\{i, i+1, \ldots, i+k-1\}$.

Motivation in HPC:

- Better memory locality.
- Reduced communication overhead between processing units.
- Simplified resource management.



Related Work - Overview

Ratio, Condition	**Complexity**	**Author (Year)**
$1+\varepsilon$	PTAS	Jansen, Thöle (2008)
$1+\epsilon,\ m > 8\frac{n}{\varepsilon}$	$O(n\log(m)(\log(m)+\log(\frac{1}{\varepsilon})))$	Jansen, Land (2018)
$\frac{5}{4}+\varepsilon$	$(nm)^{f(\frac{1}{\varepsilon})}$	Jansen, Rau (2021)
$1+\epsilon,\ m > 8\frac{n}{\varepsilon}$	$O(n\log^2(\frac{1}{\varepsilon}+\frac{\log(\varepsilon m)}{\varepsilon}))$	Grage, Jansen, O. (2023)

Related Work - Overview

Ratio, Condition	Complexity	Author (Year)
$1+\varepsilon$	PTAS	Jansen, Thöle (2008)
$1+\epsilon,\ m > 8\frac{n}{\varepsilon}$	$O(n\log(m)(\log(m)+\log(\frac{1}{\varepsilon})))$	Jansen, Land (2018)
$\frac{5}{4}+\varepsilon$	$(nm)^{f(\frac{1}{\varepsilon})}$	Jansen, Rau (2021)
$1+\epsilon,\ m > 8\frac{n}{\varepsilon}$	$O(n\log^2(\frac{1}{\varepsilon}+\frac{\log(\varepsilon m)}{\varepsilon}))$	Grage, Jansen, O. (2023)

Problem with the PTAS

For $\varepsilon = 0.5$, the PTAS has a running time of roughly $\boldsymbol{nm^{2.4\cdot 10^{17}}}$.

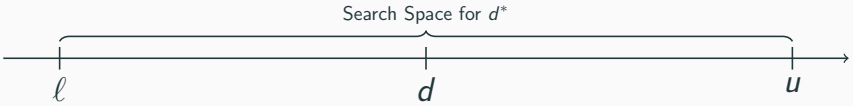
Related Work - Overview of Algorithms Polynomial in m, n, and $\frac{1}{\varepsilon}$

Ratio	**Complexity**	**Author (Year)**
$3/2 + \epsilon$	$O(nm \log(1/\varepsilon))$	Mounié, Rapine, Trystram (2007)
$3/2 + \epsilon$	$O(\frac{n}{\varepsilon^2} \log m(\frac{\log m}{\epsilon} + \log^3(\varepsilon m)))$	Jansen, Land (2018)
$3/2 + \epsilon$	$O(n \log^2(\frac{1+\log(\varepsilon m)}{\varepsilon}) + \dots)$	Grage, Jansen, O. (2023)
$3/2$	$O(nm \log(nm))$	Wu, Zhang, Chen (2023)
$1.4593 + \epsilon$	$O(nm \log(1/\varepsilon))$	**This Work**

Dual-Approximation Framework - Hochbaum & Shmoys (1985)

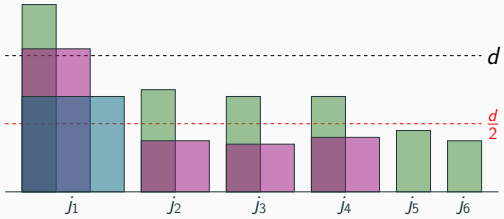
1. Find bounds on optimum: $\ell \leq d^* \leq u$.
2. Binary search on makespan guesses d:
   a) Either decide no schedule with makespan $\leq d$ exists,
   b) or build schedule with makespan $\leq \lambda d$.

$\Rightarrow$ Yields a $(\lambda + \varepsilon)$-approximation with $O(\log \frac{1}{\varepsilon})$ search steps.



Previous Work - Mounié et al. (Outline)

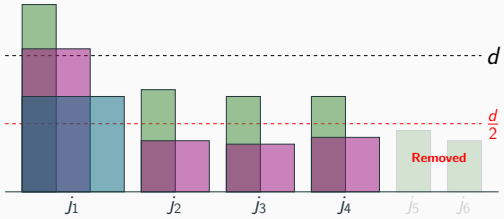
Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{d}{2}\}$



Previous Work - Mounié et al. (Outline)

Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{d}{2}\}$

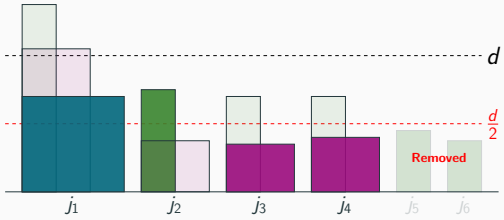
1. Remove small jobs J_S.

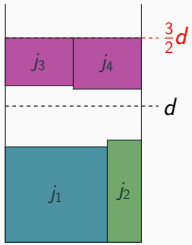


Previous Work - Mounié et al. (Outline)

Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{d}{2}\}$

1. Remove small jobs J_S.
2. Find allotment for big jobs $J_B = J \setminus J_S$.
   - using a **0/1-Knapsack** Problem.

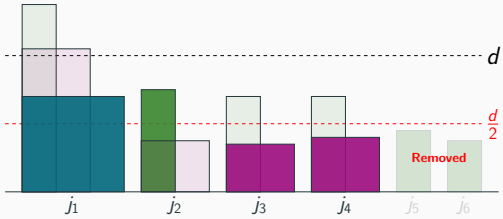


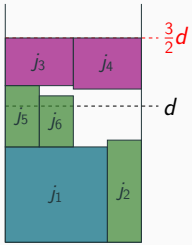


Previous Work - Mounié et al. (Outline)

Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{d}{2}\}$

1. Remove small jobs J_S.
2. Find allotment for big jobs $J_B = J \setminus J_S$.
   - using a **0/1-Knapsack** Problem.
3. Schedule small jobs greedily.

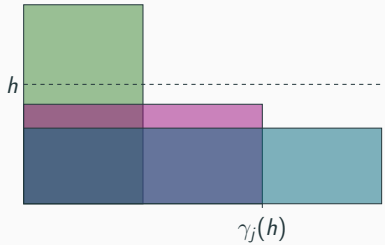




Canonical Number of Machines

Definition
$\gamma_j : \mathbb{Q}_{>0} \to [m]$, such that:

- $\gamma_j(h)$ is the minimal number of machines to execute job j in time h.



Previous Work - Mounié et al.

Theorem (Mounié et al.)
Given a partition of tasks $S_1 \sqcup S_2 = J_B$ s.t.

1. $\sum_{j \in S_1} w_j(\gamma_j(d)) + \sum_{j \in S_2} w_j(\gamma_j(d/2)) \leq md$
2. $\sum_{j \in S_1} \gamma_j(d) \leq m$

one can compute a schedule with length **at most** $\frac{3}{2}d$ in time $O(mn)$.

Previous Work - Mounié et al.

Theorem (Mounié et al.)
Given a partition of tasks $S_1 \sqcup S_2 = J_B$ s.t.

1. $\sum_{j \in S_1} w_j(\gamma_j(d)) + \sum_{j \in S_2} w_j(\gamma_j(d/2)) \leq md$
2. $\sum_{j \in S_1} \gamma_j(d) \leq m$

one can compute a schedule with length **at most** $\frac{3}{2}d$ in time $O(mn)$.

- If such partition does not exist, neither does a schedule with makespan at most d.

Previous Work - Mounié et al.

Theorem (Mounié et al.)
Given a partition of tasks $S_1 \sqcup S_2 = J_B$ s.t.

1. $\sum_{j \in S_1} w_j(\gamma_j(d)) + \sum_{j \in S_2} w_j(\gamma_j(d/2)) \leq md$
2. $\sum_{j \in S_1} \gamma_j(d) \leq m$

one can compute a schedule with length **at most** $\frac{3}{2}\boldsymbol{d}$ in time $O(mn)$.

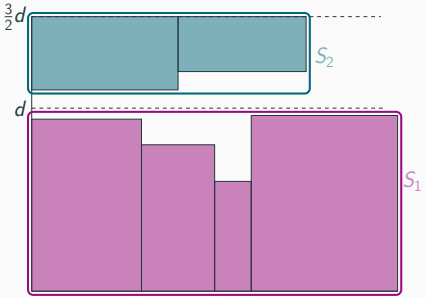
- If such partition does not exist, neither does a schedule with makespan at most d.
- Finding partition is a **0/1-Knapsack Problem**.

Previous Work - Mounié et al.

Theorem (Mounié et al.)
Given a partition of tasks $S_1 \sqcup S_2 = J_B$ s.t.

1. $\sum_{j \in S_1} w_j(\gamma_j(d)) + \sum_{j \in S_2} w_j(\gamma_j(d/2)) \leq md$
2. $\sum_{j \in S_1} \gamma_j(d) \leq m$

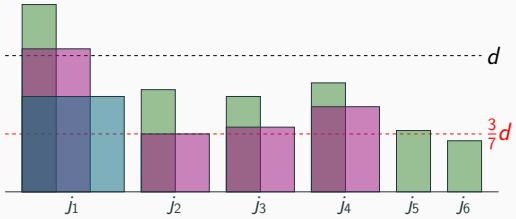
one can compute a schedule with length at most $\frac{3}{2}d$ in time $O(mn)$.



Our Algorithm

Our Algorithm (Outline)

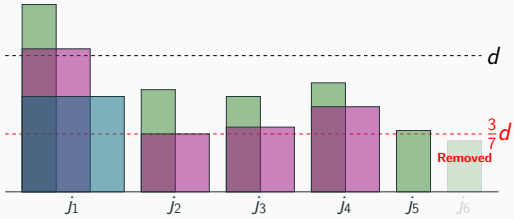
Revised Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{3}{7}d\}$



Our Algorithm (Outline)

Revised Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{3}{7}d\}$

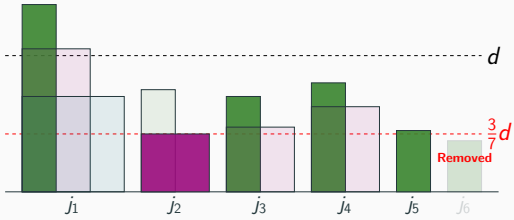
1. Remove small jobs J_S.

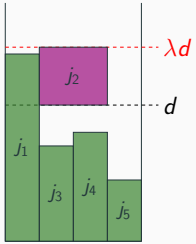


Our Algorithm (Outline)

Revised Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{3}{7}d\}$

1. Remove small jobs J_S.
2. Find allotment for big jobs $J_B = J \setminus J_S$ into **3 sets**.
   - using a **Multiple-Choice-Knapsack** Problem.

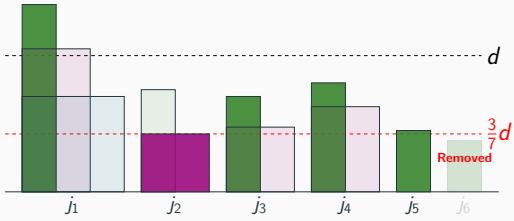


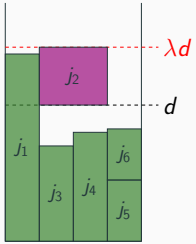


Our Algorithm (Outline)

Revised Definition: Small Jobs
$J_S := \{j \in J \mid t_j(1) \leq \frac{3}{7}d\}$

1. Remove small jobs J_S.
2. Find allotment for big jobs $J_B = J \setminus J_S$ into **3 sets**.
   - using a **Multiple-Choice-Knapsack** Problem.
3. Schedule small jobs greedily.





This Work - The 3-Shelf Approach

Main Theorem (This Work)
Given a partition of tasks $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \mathcal{C}_3 = J_B$ s.t.

1. $\sum_{j\in\mathcal{C}_1} w_j(\gamma_j(d)) + \sum_{j\in\mathcal{C}_2} w_j((\gamma_j(\frac{4}{7}d)) + \sum_{j\in\mathcal{C}_3} w_j((\gamma_j(\frac{3}{7}d)) \leq md$
2. $\sum_{j\in\mathcal{C}_1} \gamma_j(d) + \sum_{j\in\mathcal{C}_2} \gamma_j(\frac{4}{7}d) \cdot \frac{1}{2} \leq m$

one can compute a schedule with length **at most** $\frac{73}{50}\boldsymbol{d}$ in time $O(mn)$.

This Work - The 3-Shelf Approach

Main Theorem (This Work)
Given a partition of tasks $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \mathcal{C}_3 = J_B$ s.t.

1. $\sum_{j\in\mathcal{C}_1} w_j(\gamma_j(d)) + \sum_{j\in\mathcal{C}_2} w_j((\gamma_j(\frac{4}{7}d)) + \sum_{j\in\mathcal{C}_3} w_j((\gamma_j(\frac{3}{7}d)) \leq md$
2. $\sum_{j\in\mathcal{C}_1} \gamma_j(d) + \sum_{j\in\mathcal{C}_2} \gamma_j(\frac{4}{7}d) \cdot \frac{1}{2} \leq m$

one can compute a schedule with length **at most** $\frac{73}{50}\boldsymbol{d}$ in time $O(mn)$.

- If such partition does not exist, neither does a schedule with makespan at most d.

This Work - The 3-Shelf Approach

Main Theorem (This Work)
Given a partition of tasks $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \mathcal{C}_3 = J_B$ s.t.

1. $\sum_{j\in\mathcal{C}_1} w_j(\gamma_j(d)) + \sum_{j\in\mathcal{C}_2} w_j((\gamma_j(\frac{4}{7}d)) + \sum_{j\in\mathcal{C}_3} w_j((\gamma_j(\frac{3}{7}d)) \leq md$
2. $\sum_{j\in\mathcal{C}_1} \gamma_j(d) + \sum_{j\in\mathcal{C}_2} \gamma_j(\frac{4}{7}d) \cdot \frac{1}{2} \leq m$

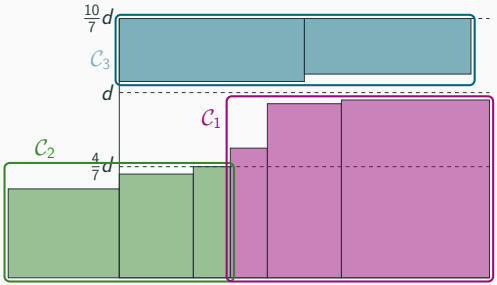
one can compute a schedule with length **at most** $\frac{73}{50}\boldsymbol{d}$ in time $O(mn)$.

- If such partition does not exist, neither does a schedule with makespan at most d.
- Finding partition is a **Multiple-Choice-Knapsack Problem**.

Main Theorem (This Work)
Given a partition of tasks $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \mathcal{C}_3 = J_B$ s.t.

1. $\sum_{j\in\mathcal{C}_1} w_j(\gamma_j(d)) + \sum_{j\in\mathcal{C}_2} w_j((\gamma_j(\frac{4}{7}d)) + \sum_{j\in\mathcal{C}_3} w_j((\gamma_j(\frac{3}{7}d)) \leq md$
2. $\sum_{j\in\mathcal{C}_1} \gamma_j(d) + \sum_{j\in\mathcal{C}_2} \gamma_j(\frac{4}{7}d) \cdot \frac{1}{2} \leq m$

one can compute a schedule with length at most $\frac{73}{50}d$ in time $O(mn)$.

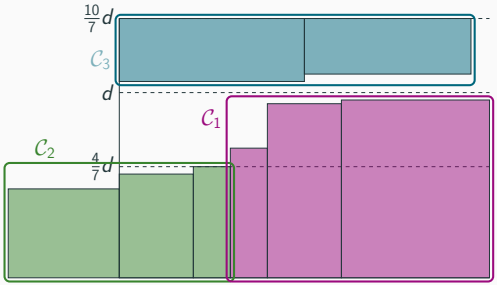


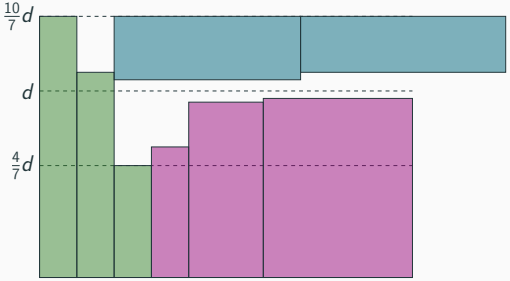
This Work - The 3-Shelf Approach

Main Theorem (This Work)
Given a partition of tasks $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \mathcal{C}_3 = J_B$ s.t.

1. $\sum_{j\in\mathcal{C}_1} w_j(\gamma_j(d)) + \sum_{j\in\mathcal{C}_2} w_j((\gamma_j(\frac{4}{7}d)) + \sum_{j\in\mathcal{C}_3} w_j((\gamma_j(\frac{3}{7}d)) \leq md$
2. $\sum_{j\in\mathcal{C}_1} \gamma_j(d) + \sum_{j\in\mathcal{C}_2} \gamma_j(\frac{4}{7}d) \cdot \frac{1}{2} \leq m$

one can compute a schedule with length at most $\frac{73}{50}d$ in time $O(mn)$.

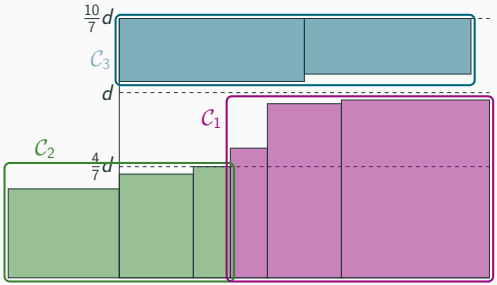


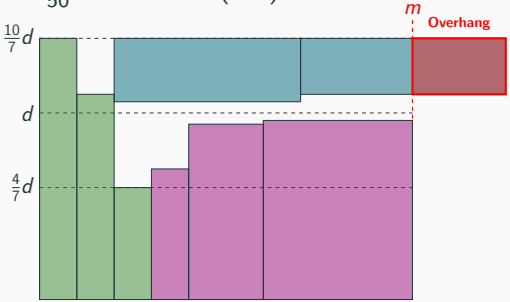


Main Theorem (This Work)
Given a partition of tasks $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \mathcal{C}_3 = J_B$ s.t.

1. $\sum_{j\in\mathcal{C}_1} w_j(\gamma_j(d)) + \sum_{j\in\mathcal{C}_2} w_j((\gamma_j(\frac{4}{7}d)) + \sum_{j\in\mathcal{C}_3} w_j((\gamma_j(\frac{3}{7}d)) \leq md$
2. $\sum_{j\in\mathcal{C}_1} \gamma_j(d) + \sum_{j\in\mathcal{C}_2} \gamma_j(\frac{4}{7}d) \cdot \frac{1}{2} \leq m$

one can compute a schedule with length at most $\frac{73}{50}d$ in time $O(mn)$.

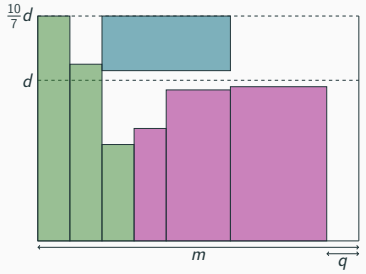




Repairing the Schedule

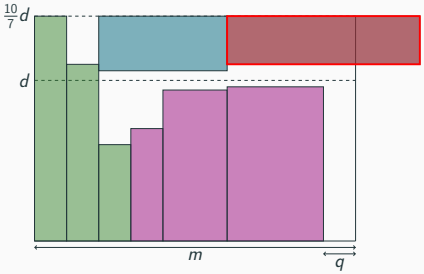
Case 1: Schedule is feasible

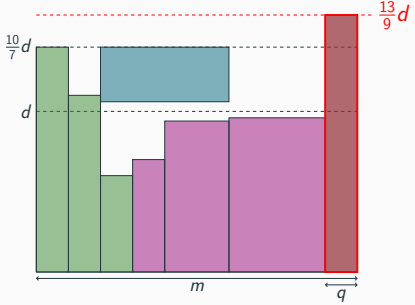
Guaranteed ratio of $\frac{10}{7}$. Everything fits within the machine bounds.



Case 2: Simple Case ($q \leq \frac{m}{6}$)

Guaranteed ratio of $\frac{13}{9}$.



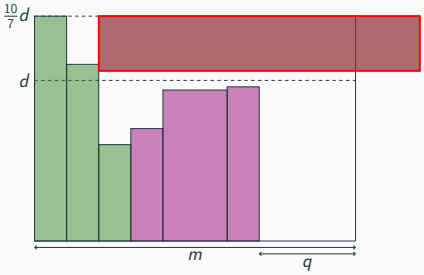


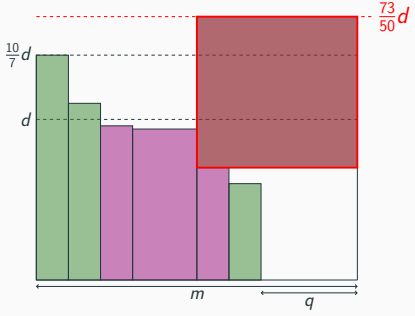
Case 3: Critical Case ($q > \frac{m}{6}$)

Guaranteed ratio of ≈ 1.4593.

Key Observation

If $q > m/6$, then $\mathcal{C}_3$ contains **exactly one job**.





Results

Our Results for Contiguous Monotone Moldable Scheduling

Our algorithm produces a **contiguous schedule** with an approximation ratio of $\approx 1.4593 + \varepsilon$.

Our Results for Contiguous Monotone Moldable Scheduling

Our algorithm produces a **contiguous schedule** with an approximation ratio of $\approx 1.4593 + \varepsilon$.

The Contiguity Gap

- We compare our contiguous makespan to the **non-contiguous lower bound**.
- Result: The makespan gap between optimal contiguous and non-contiguous schedules is **bounded by our ratio**.

Practical Results - Overview

- Implementation in Java (available on GitHub)
- Tested on a single core of an Intel i5-8400T CPU with 2GB RAM
- Two sets of tests:
  - Varying number of machines (m) with fixed number of jobs ($n = 1000$)
  - Varying number of jobs (n) with fixed number of machines ($m = 1000$)

Practical Results - Runtime

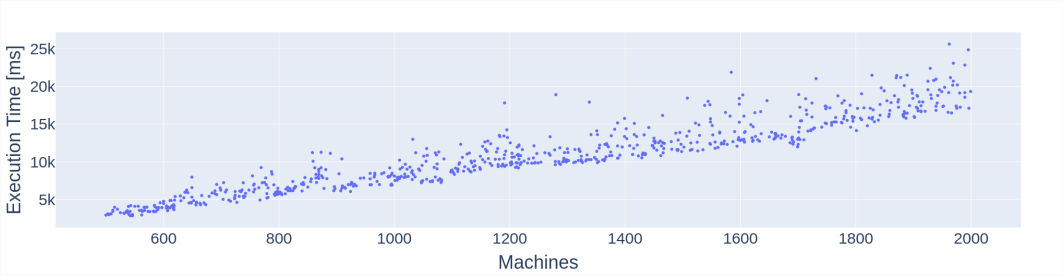


Figure 1: Execution Time of the Implementation for Varying Number of Machines.

Practical Results - Runtime

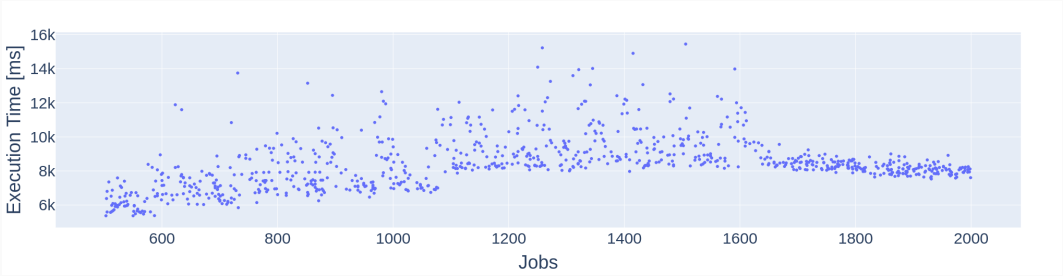


Figure 1: Execution Time of the Implementation for Varying Number of Jobs.

Practical Results - Approximation Ratio

d^* is the last makespan guess of the dual approximation framework. Thus:

$$d^* \leq (1+\varepsilon)\mathrm{OPT} = 1.05 \cdot \mathrm{OPT}.$$

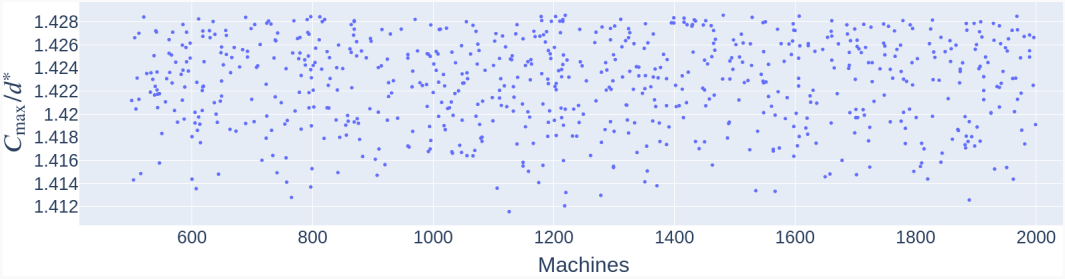


Figure 2: Approximation Ratio of the Implementation for Varying Number of Machines.

Practical Results - Approximation Ratio

d^* is the last makespan guess of the dual approximation framework. Thus:

$$d^* \leq (1+\varepsilon)\mathrm{OPT} = 1.05 \cdot \mathrm{OPT}.$$

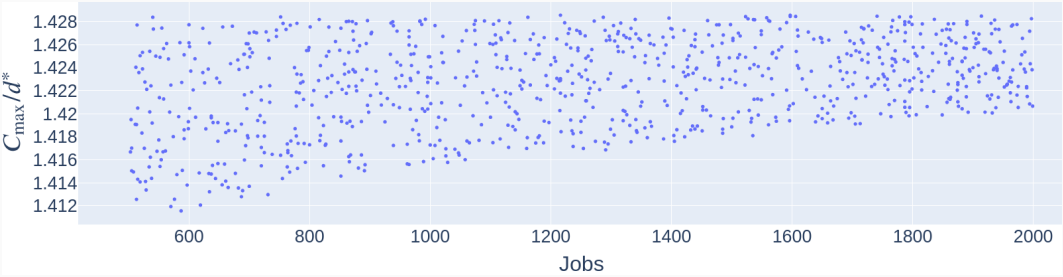


Figure 2: Achieved Approximation Ratio of the Implementation for Varying Number of Jobs.

Future Work

Future Work

Can the **makespan guarantee** be improved further?

- Is an approximation ratio of $\frac{13}{9}$ possible?

Future Work

Can the **makespan guarantee** be improved further?

- Is an approximation ratio of $\frac{13}{9}$ possible?

Can the **running time** be improved?

- With better/approximative Multiple-Choice Knapsack algorithms?
- With compression techniques from Jansen & Land (2018)?

Future Work

Can the **makespan guarantee** be improved further?

- Is an approximation ratio of $\frac{13}{9}$ possible?

Can the **running time** be improved?

- With better/approximative Multiple-Choice Knapsack algorithms?
- With compression techniques from Jansen & Land (2018)?

Can this result be applied to **other scheduling problems**?

- CPU-GPU scheduling?

Future Work

Can the **makespan guarantee** be improved further?

- Is an approximation ratio of $\frac{13}{9}$ possible?

Can the **running time** be improved?

- With better/approximative Multiple-Choice Knapsack algorithms?
- With compression techniques from Jansen & Land (2018)?

Can this result be applied to **other scheduling problems**?

- CPU-GPU scheduling?

Thank you for your attention!

Omitted Slides

The Geometric Analysis

If the algorithm fails, the load L_i on each machine i is greater than:

$$g(i) = \lambda d - \frac{dm - \frac{\lambda}{2} d(m-q)}{m-i}$$

Bounding the Work Area

We can bound the total work on Shelf 1 (W_1) using an integral over this curve:

$$W_1 > \int_0^{m-q} \max\left\{ g(i), \frac{\lambda}{2} d \right\} di$$

Adding the work of W_2, we find the minimum of this function occurs at $q = \frac{\lambda - 1}{\lambda} m$.

The Lambert W Function

Solving the integral inequality $W_1 + W_2 > md$ yields:

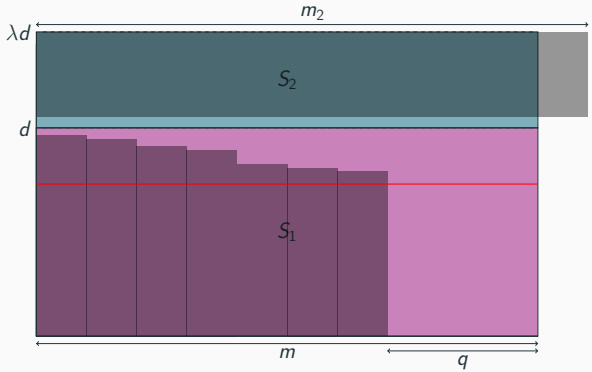
$$-3\lambda e^{-3\lambda} > -3e^{-4}$$

Which is solved using the W_{-1} branch of the Lambert W function:

$$\lambda > -\frac{1}{3} W_{-1}(-3e^{-4}) \approx 1.4593$$

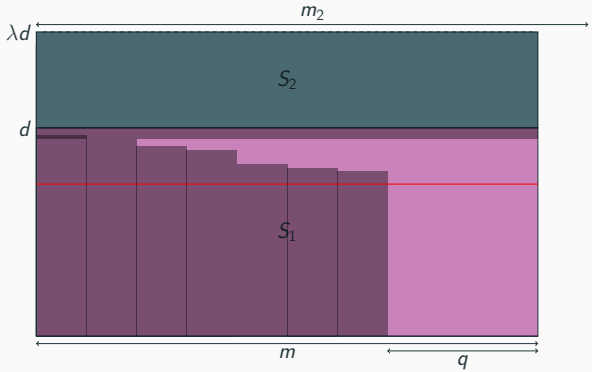
Repairing the Schedule - 2 ($q > m/6$)

- Try every possible allotment of the job in S_2.



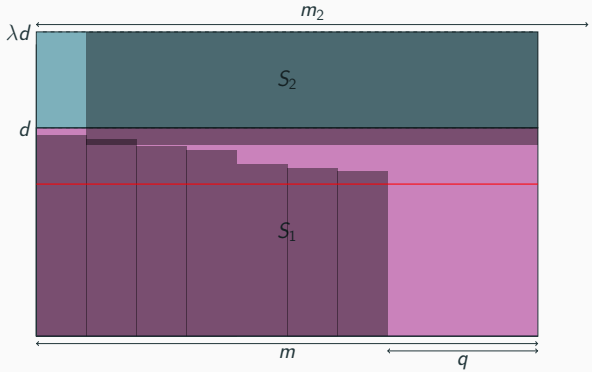
Repairing the Schedule - 2 ($q > m/6$)

- Try every possible allotment of the job in S_2.



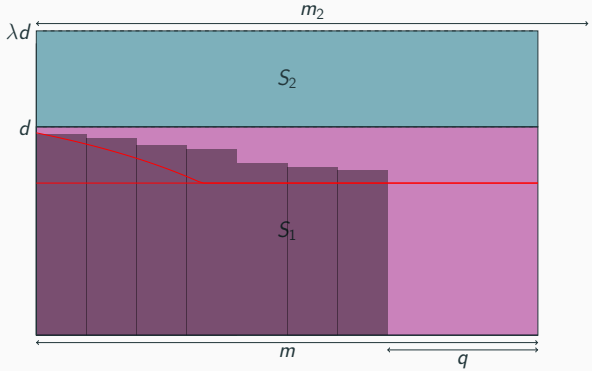
Repairing the Schedule - 2 ($q > m/6$)

- Try every possible allotment of the job in S_2.



Repairing the Schedule - 2 ($q > m/6$)

- Try every possible allotment of the job in S_2.
- If the job does not fit, the load of each machine i in S_1 is greater than $\max(\frac{\lambda d}{2}, g(i))$, with $g(i) = \lambda d - \frac{dm - \frac{\lambda}{2} d(m-q)}{m-i}$



Related Work

Ratio	Complexity	Author (Year)
$\frac{3}{2}+\epsilon$	$O(\log(1/\varepsilon)nm)$	Mounié, Rapine, Trystram (2007)
$1+\epsilon$	$O(n\log(m)(\log(m)+\log(\frac{1}{\varepsilon})))$	Jansen, Land (2018)
$\frac{3}{2}+\epsilon$	$O(\frac{n}{\varepsilon^2}\log m(\frac{\log m}{\epsilon}+\log^3(\varepsilon m)))$	
$1+\epsilon$	$O(n\log^2(\frac{1}{\varepsilon}+\frac{\log(\varepsilon m)}{\varepsilon}))$	Grage, Jansen, O. (2023)
$\frac{3}{2}+\epsilon$	$O(n\log^2(\frac{1+\log(\varepsilon m)}{\varepsilon})+\frac{n}{\varepsilon}\log(\frac{1}{\varepsilon})\log(\varepsilon m))$	
$\frac{3}{2}$	$O(nm\log(nm))$	Wu, Zhang, Chen (2023)
$\frac{73}{50}+\epsilon$	$O(\log(1/\varepsilon)nm)$	This Work