Parallel Machine Scheduling
with a Single Resource per Job

T. Janssen\textsuperscript{1} C. Swennenhuis\textsuperscript{2} A. Bitar\textsuperscript{3} T. Bosman\textsuperscript{4}
S. Dauzère-Pérès\textsuperscript{5} D. Gijswijt\textsuperscript{1} L. van Iersel\textsuperscript{1} C. Yugma\textsuperscript{5}

\textsuperscript{1}Delft Institute of Applied Mathematics, Delft University of Technology
\textsuperscript{2}Department of Mathematics and Computer Science, Eindhoven University of Technology
\textsuperscript{3}Decision Brain, Montpelleir
\textsuperscript{4}Department of Econometrics and Operations Research, Vrije Universiteit Amsterdam
\textsuperscript{5}Department of Manufacturing Sciences and Logistics, Center of Microelectronics in Provence

MAPSP, June 2019
Motivation: Semi-conductor Industry
Motivation: Semi-conductor Industry
Motivation: Semi-conductor Industry
Table of Contents

1. Problem Definition
2. SPT-available Rule
3. Ordered Resources Property
4. Practical Solution
5. Future Work
# Table of Contents

1. Problem Definition
2. SPT-available Rule
3. Ordered Resources Property
4. Practical Solution
5. Future Work
Problem Definition

Céline Swennenhuis (TU/e)

Single Resource Scheduling

MAPSP 2019
Problem Definition

\[
\sum_{j} C_j = 20 \\
\sum_{j} C_j = 22
\]
Problem Definition

\[ \sum_j C_j = 20 \]

\[ \sum_j C_j = 22 \]
Problem Definition

\[ \sum_j C_j = 22 \]

\[ \sum_j C_j = 20 \]
Problem Definition

- **Input:**
  - $m$ machines
  - $n$ jobs with processing times $p_j$
  - $k$ resources
  - jobs are partitioned into which resource they use

- **Schedule feasible**
  - When each resource is used at most once at a time

- **Objective:**
  - Minimize $\sum_j C_j$

We denote\(^1\) this problem by:

$$P|\text{partition}| \sum_j C_j$$

\(^1\)Using the notation of [Graham et al., 1979]
The complexity of $P|\text{partition}| \sum_j C_j$ remains open.
# Table of Contents

1. Problem Definition

2. SPT-available Rule

3. Ordered Resources Property

4. Practical Solution

5. Future Work
SPT-available Rule

Definition (Shortest Processing Time Rule, SPT)
The next job to be scheduled is the job with the shortest processing time.

SPT is exact for $P||\sum_j C_j$ [Smith, 1956]. However, how to deal with the resources?
SPT-available Rule

Definition (Shortest Processing Time Rule, SPT)
The next job to be scheduled is the job with the shortest processing time.

SPT is exact for $P|| \sum_j C_j$ [Smith, 1956]. However, how to deal with the resources?

Definition (SPT-available Rule)
The next job to be scheduled is the job with the shortest processing time. If its resources is not available, schedule it at the earliest opportunity.
SPT-available Rule
SPT-available Rule
SPT-available Rule
SPT-available Rule
SPT-available Rule
SPT-available Rule
SPT-available Rule

\[ \sum_{j} C_j = 20 \]
\[ \sum_{j} C_j = 22 \]

Céline Swennenhuis (TU/e)
Single Resource Scheduling
MAPSP 2019
SPT-available Rule

\[
\sum_{j} C_j = 20 \\
\sum_{j} C_j = 22
\]
SPT-available Rule

\[ \sum_j C_j = 22 \]

\[ \sum_j C_j = 20 \]
Theorem 2

The SPT-available rule gives a \((2 - \frac{1}{m})\)-approximation for \(P|\text{partition}|\sum_j C_j\).
The SPT-available rule gives a \((2 - \frac{1}{m})\)-approximation for \(P|\text{partition}|\sum_j C_j\). This bound is not tight. We can construct instances where it gives an \(\alpha\)-approximation for \(\alpha < \frac{4}{3}\).
# Table of Contents

1. Problem Definition
2. SPT-available Rule
3. Ordered Resources Property
4. Practical Solution
5. Future Work
Ordered Resources Property

Theorem 1

In any optimal schedule for $P|\text{partition}|\sum_j C_j$, all jobs sharing the same resource are processed in order of non-decreasing processing times.
Ordered Resources Property

Theorem 1

In any optimal schedule for \( P|partition| \sum_j C_j \), all jobs sharing the same resource are processed in order of non-decreasing processing times.

Outline of proof:
Theorem 1

In any optimal schedule for $P|\text{partition}|\sum_j C_j$, all jobs sharing the same resource are processed in order of non-decreasing processing times.

Outline of proof:

\[ P|\text{partition}|\sum_j C_j \xrightarrow{\text{allow preemptions}} P|\text{partition,prmp}|\sum_j C_j \]
Theorem 1

In any optimal schedule for \( P|\text{partition}| \sum_j C_j \), all jobs sharing the same resource are processed in order of non-decreasing processing times.

Outline of proof:

\[ P|\text{partition}| \sum_j C_j \rightarrow \text{allow preemptions} \rightarrow P|\text{partition,prmp}| \sum_j C_j \rightarrow \text{By Lemma 1} \rightarrow P|\text{partition,prmp}| \sum_j C_j \text{ has sorted resources} \]
**Theorem 1**

In any optimal schedule for $P|\text{partition}|\sum_j C_j$, all jobs sharing the same resource are processed in order of non-decreasing processing times.

**Outline of proof:**

1. $P|\text{partition}|\sum_j C_j$  
   **allow preemptions**  
   $P|\text{partition,prmp}|\sum_j C_j$

2. By Lemma 1

3. $P|\text{partition,prmp}|\sum_j C_j$  
   has sorted resources
Lemma 1

In any optimal schedule for $P|\text{partition, prmp}|\sum_j C_j$, all jobs sharing the same resource must be processed in order of non-decreasing processing times.
Ordered Resources Property

**Lemma 1**

In any optimal schedule for $P|\text{partition}, \text{prmp} | \sum_j C_j$, all jobs sharing the same resource must be processed in order of non-decreasing processing times.
Ordered Resources Property

Lemma 1

In any optimal schedule for $P|\text{partition}, \text{prmp} | \sum j C_j$, all jobs sharing the same resource must be processed in order of non-decreasing processing times.
Lemma 1

In any optimal schedule for $P|\text{partition}, \text{prmp}| \sum_j C_j$, all jobs sharing the same resource must be processed in order of non-decreasing processing times.
Lemma 1

In any optimal schedule for $P|partition, prmp| \sum_j C_j$, all jobs sharing the same resource must be processed in order of non-decreasing processing times.
Ordered Resources Property

Theorem 1

In any optimal schedule for $P|\text{partition}|\sum_j C_j$, all jobs sharing the same resource are processed in order of non-decreasing processing times.

Outline of proof:

$P|\text{partition}|\sum_j C_j \xrightarrow{\text{allow preemptions}} P|\text{partition,prmp}|\sum_j C_j$

By Lemma 1

$P|\text{partition,prmp}|\sum_j C_j$ has sorted resources
Ordered Resources Property

Theorem 1

In any optimal schedule for $P|\text{partition}|\sum_j C_j$, all jobs sharing the same resource are processed in order of non-decreasing processing times.

Outline of proof:

1. $P|\text{partition}|\sum_j C_j$
   - allow preemptions
2. $P|\text{partition,prmp}|\sum_j C_j$
   - By Lemma 1
3. $P|\text{partition}|\sum_j C_j$
   - has sorted resources
   - remove preemptions
4. $P|\text{partition,prmp}|\sum_j C_j$
   - has sorted resources
1 Problem Definition

2 SPT-available Rule

3 Ordered Resources Property

4 Practical Solution

5 Future Work
In practice, machines are not identical (but unrelated). We proved this problem to be $NP$-hard.

- In practice, every resource (reticle) is used at most 30 times a month
  - Solve easier problem without resources
  - Make schedule feasible
- Results:
  - Gives 6.02% reduction in total completion time
  - Gives 1.97% reduction in machine usage
# Table of Contents

1. Problem Definition
2. SPT-available Rule
3. Ordered Resources Property
4. Practical Solution
5. Future Work
Future Work

- Establish the complexity of $P|\text{partition}|\sum_j C_j$
- Establish worst-case approximation ratio for SPT-available rule.
- Is $P|\text{partition}|\sum_j C_j$ fixed-parameter tractable in $k$ (the number of resources)?

Parallel Machine Scheduling
with a Single Resource per Job

T. Janssen¹  C. Swennenhuis²  A. Bitar³  T. Bosman⁴
S. Dauzère-Pérès⁵  D. Gijswijt¹  L. van Iersel¹  C. Yugma⁵

¹Delft Institute of Applied Mathematics, Delft University of Technology
²Department of Mathematics and Computer Science, Eindhoven University of Technology
³Decision Brain, Montpelleir
⁴Department of Econometrics and Operations Research, Vrije Universiteit Amsterdam
⁵Department of Manufacturing Sciences and Logistics, Center of Microelectronics in Provence

MAPSP, June 2019