Unrelated Machines Interval Scheduling

Danny Hermelin, **Yuval Itzhaki**, Dvir Shabtay, Hendrik Molter

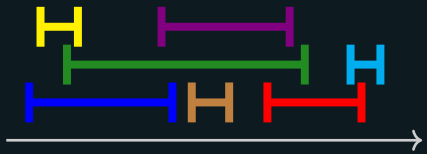
JIT Scheduling

- In **Just in Time** scheduling jobs are completed *exactly* on their deadlines.

JIT Scheduling

אוניברסיטת בן-גוריון בנגב
جامعة بن غوريون في النقب
Ben-Gurion University of the Negev

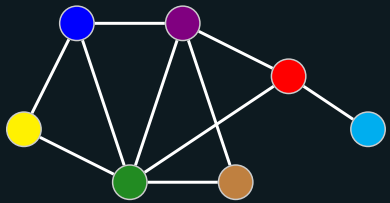
- In **Just in Time** scheduling jobs are completed *exactly* on their deadlines.
- Equivalently, every job has a predetermined interval.

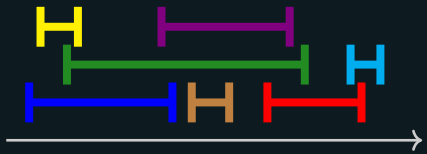


JIT Scheduling

אוניברסיטת בן-גוריון בנגב
جامعة بن غوريون في النقب
Ben-Gurion University of the Negev

- In **Just in Time** scheduling jobs are completed *exactly* on their deadlines.
- Equivalently, every job has a predetermined interval.
- Jobs intervals intersect are in *conflict*.

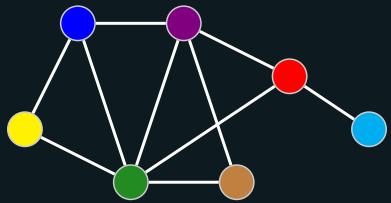




JIT Scheduling

אוניברסיטת בן-גוריון בנגב
جامعة بن غوريون في النقب
Ben-Gurion University of the Negev

- In **Just in Time** scheduling jobs are completed *exactly* on their deadlines.
- Equivalently, every job has a predetermined interval.
- Jobs intervals intersect are in *conflict*.
- The conflict-graph is an **interval graph**.



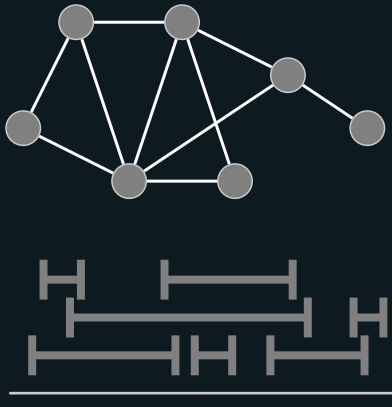
Interval Scheduling Problems

Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.



Interval Scheduling Problems

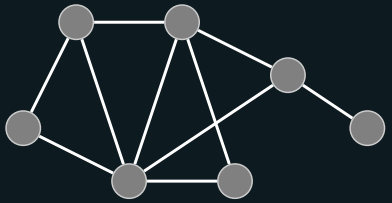
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

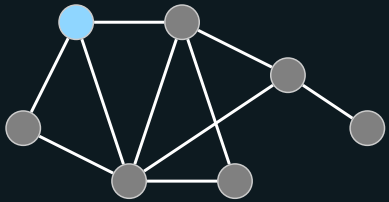
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

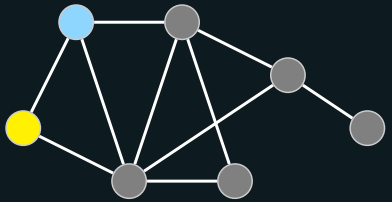
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

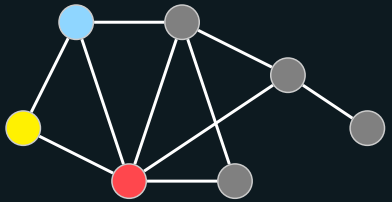
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

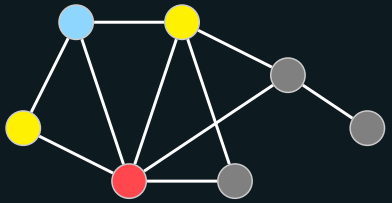
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

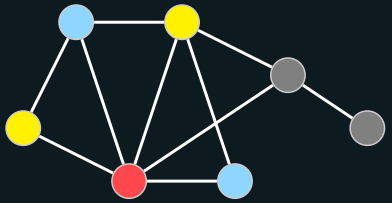
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

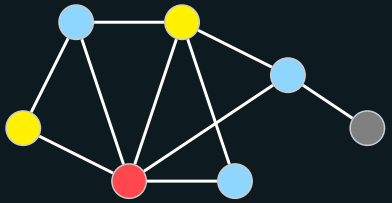
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



Interval Scheduling Problems

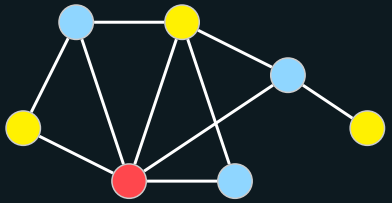
Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each job has

- a *processing time* p_j,
- a *weight* w_j,
- a *deadline* d_j

Output: Maximum weighted JIT schedule.

This problem is solvable in polynomial time!



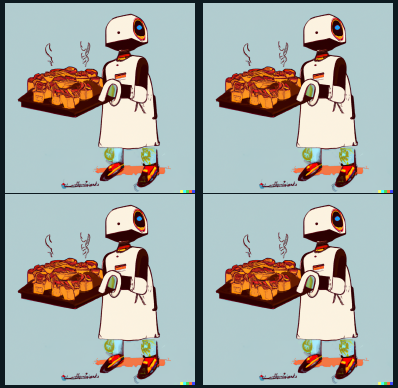
Problem Definition

Interval Scheduling on Identical Machines

Input: *m machines* and *n jobs*, each with:

- *processing time* p_j
- *weight* w_j
- *deadline* d_j

Output: Maximum weighted JIT schedule.



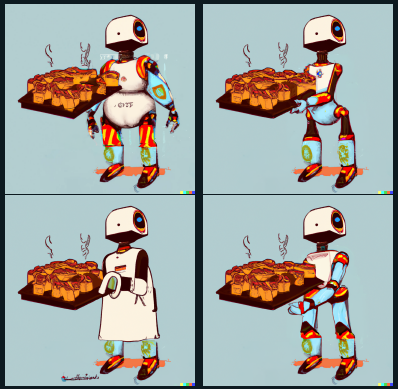
Problem Definition

Interval Scheduling on Unrelated Machines

Input: *m machines* and *n jobs*, each with:

- *processing time* $p_{i,j}$
- *weight* w_j
- *deadline* d_j

Output: Maximum weighted JIT schedule.



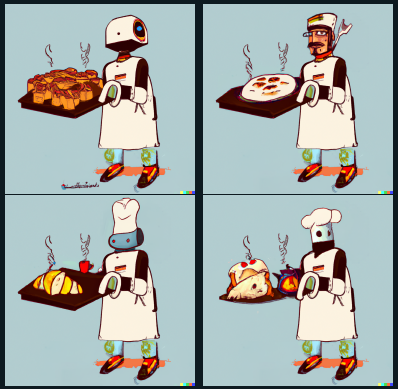
Problem Definition

Interval Scheduling on Eligible Machines

Input: *m machines* and *n jobs*, each with:

- *processing time* p_j
- *weight* w_j
- *deadline* d_j
- *eligible machine set* $M_j \subseteq [m]$

Output: Maximum weighted JIT schedule.



Related Work

Related Work

- The problem is *strongly* NP-hard. [AS87]

Related Work

- The problem is *strongly* NP-hard.
  [AS87] - No $(Wn)^{O(1)}$ algorithm.

Related Work

- The problem is *strongly* NP-hard. [AS87] - No $(Wn)^{O(1)}$ algorithm.
- Mnich and van Bevern have asked in 2018 whether the problem is fixed-parameter tractable [MvB18].

ELSEVIER

Computers & Operations Research

Volume 100, December 2018, Pages 254-261

Survey in Operations Research and Management Science

Parameterized complexity of machine scheduling: 15 open problems

Matthias Mnich [a b], René van Bevern [c d]

Arkin and Silverberg (1987) show that this problem is solvable in $O(n^2 \log n)$ time. However, they showed that the variant $P|M_j, d_j - r_j = p_j|\Sigma w_j U_j$ is NP-hard and solvable in $O(n^{m+1})$ time.

Open Problem 8

Is $P|M_j, d_j - r_j = p_j|\Sigma w_j U_j$ fixed-parameter tractable parameterized by the number of machines?

Related Work

- The problem is *strongly* NP-hard. [AS87] - No $(Wn)^{O(1)}$ algorithm.
- Mnich and van Bevern have asked in 2018 whether the problem is fixed-parameter tractable [MvB18].
- Interval Scheduling on Eligible Machines is solvable in $\mathcal{O}(n^m)$ [AS87].

ELSEVIER

Computers & Operations Research

Volume 100, December 2018, Pages 254-261

Survey in Operations Research and Management Science

Parameterized complexity of machine scheduling: 15 open problems

Matthias Mnich [a b], René van Bevern [c d]

Arkin and Silverberg (1987) show that this problem is solvable in $O(n^2 \log n)$ time. However, they showed that the variant $P|M_j, d_j - r_j = p_j|\Sigma w_j U_j$ is NP-hard and solvable in $O(n^{m+1})$ time.

Open Problem 8

Is $P|M_j, d_j - r_j = p_j|\Sigma w_j U_j$ fixed-parameter tractable parameterized by the number of machines?

Related Work

- The problem is *strongly* NP-hard. [AS87] - No $(Wn)^{O(1)}$ algorithm.
- Mnich and van Bevern have asked in 2018 whether the problem is fixed-parameter tractable [MvB18].
- Interval Scheduling on Eligible Machines is solvable in $\mathcal{O}(n^m)$ [AS87].
- Unrelated Machines Interval Scheduling is solvable in $\mathcal{O}(mn^m)$ [SV05].

ELSEVIER

Computers & Operations Research

Volume 100, December 2018, Pages 254-261

Survey in Operations Research and Management Science

Parameterized complexity of machine scheduling: 15 open problems

Matthias Mnich [a b], René van Bevern [c d]

Arkin and Silverberg (1987) show that this problem is solvable in $O(n^2 \log n)$ time. However, they showed that the variant $P|M_j, d_j - r_j = p_j|\Sigma w_j U_j$ is NP-hard and solvable in $O(n^{m+1})$ time.

Open Problem 8

Is $P|M_j, d_j - r_j = p_j|\Sigma w_j U_j$ fixed-parameter tractable parameterized by the number of machines?

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Is there $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for some parameter k?

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Is there $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for some parameter k?

Our research has produced the following findings:

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Is there $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for some parameter k?

Our research has produced the following findings:

- For $k := m$ no such algorithm exist

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Is there $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for some parameter k?

Our research has produced the following findings:

- For $k := m$ no such algorithm exist
- For $k := p_{\max}$ no such algorithm exist

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Is there $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for some parameter k?

Our research has produced the following findings:

- For $k := m$ no such algorithm exist
- For $k := p_{\max}$ no such algorithm exist
- Interval Scheduling on Eligible Machines is FPT w.r.t. $p_{\max} + m$

Obtained Results

Is Unrelated Machines Interval Scheduling *fixed-parameter tractable* (FPT)?

Is there $f(k) \cdot n^{\mathcal{O}(1)}$ time algorithm for some parameter k?

Our research has produced the following findings:

- For $k := m$ no such algorithm exist
- For $k := p_{\max}$ no such algorithm exist
- Interval Scheduling on Eligible Machines is FPT w.r.t. $p_{\max} + m$

Included in the 17th International Symposium on Parameterized and Exact Computation (IPEC 2022)

Published in Journal of Computer and System Sciences (JCSS-D-23-00304)

Eligible Machines Interval Scheduling is W[1]-hard w.r.t. m

Theorem 1

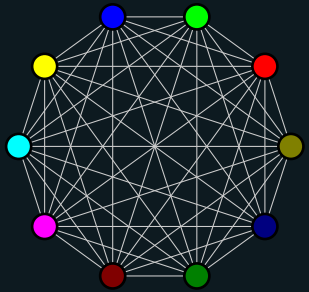
Eligible Machines Interval Scheduling *is strongly* W[1]*-hard when parameterized by* m*, the number of machines.*

Eligible Machines Interval Scheduling is W[1]-hard w.r.t. m

Theorem 1

Eligible Machines Interval Scheduling *is strongly* W[1]*-hard when parameterized by* m*, the number of machines.*

- Parameterized reduction from Multicolored k-Clique

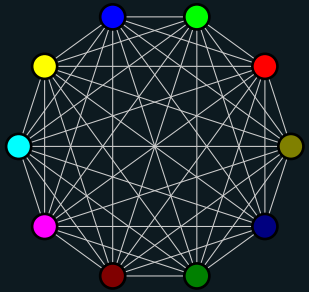


Eligible Machines Interval Scheduling is W[1]-hard w.r.t. m

Theorem 1

Eligible Machines Interval Scheduling *is strongly* W[1]*-hard when parameterized by* m*, the number of machines.*

- Parameterized reduction from Multicolored k-Clique
- Reduce such that $m \leq f(k)$

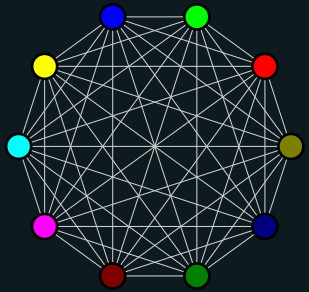


Eligible Machines Interval Scheduling is W[1]-hard w.r.t. m

Theorem 1

Eligible Machines Interval Scheduling *is strongly* W[1]*-hard when parameterized by* m*, the number of machines.*

- Parameterized reduction from Multicolored k-Clique
- Reduce such that $m \leq f(k)$
- Key ideas:
  - Map all colorful edges and vertices to jobs
  - Schedule maximum vertices and edges jobs
  - Prevent the simultaneous schedule of non-adjacent vertices



Gadget 1 - Vertex Selection Machine

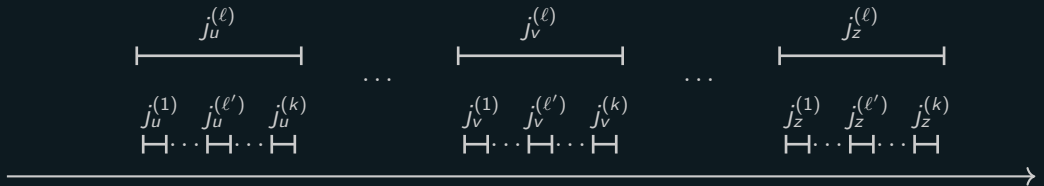


Figure 1: Illustration of the validation machine. Depicted are intervals of jobs corresponding to vertices $v, u, z \in V_\ell$ with $v <_\pi u <_\pi z$.

Gadget 2 - Edge Selection Machine

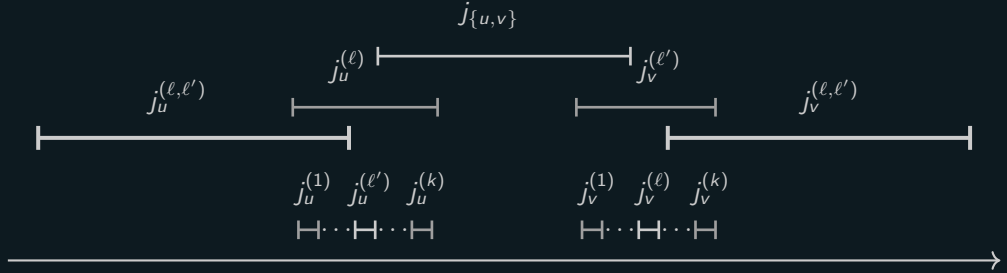


Figure 2: An illustration of the edge selection machine for color combination ℓ, ℓ' with $\ell < \ell'$. Depicted are intervals of jobs relating to $u \in V_\ell$, $v \in V_{\ell'}$, and $e = \{u, v\} \in E$. Gray intervals correspond to jobs that are not eligible on the machine.

Eligible Machines Interval Scheduling is W[1]-hard w.r.t. $p_{\max}$

Theorem 2

Eligible Machines Interval Scheduling *is* NP*-hard even for* $p_{\max} = \mathcal{O}(1)$.

- Reduction from Exact $(3,4)$-SAT
- Key ideas:
  - Map all variables and literals to jobs
  - Schedule for every clause exactly one literal job
  - Scheduling literal jobs requires consistent variable jobs schedule
  - Jobs' processing times are bounded by a constant

Eligible Machines Interval Scheduling is W[1]-hard w.r.t. $p_{\max}$

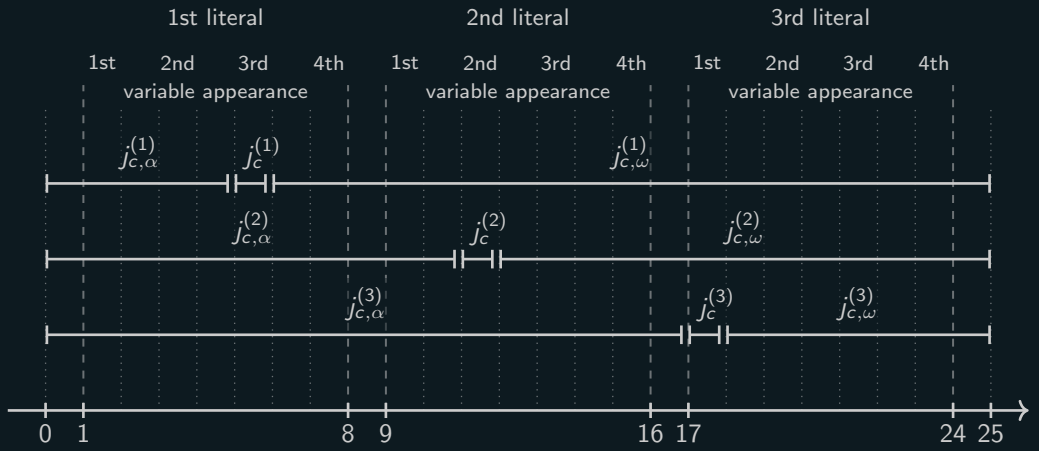


Figure 3: An illustration of the reduction when applied to the clause $c = (x \vee y \vee \neg z)$. The eligible machines of $j_c^{(1)}$ are $\{i_x^T, i_c^2, i_c^3\}$, for $j_c^{(2)}$ these are $\{i_y^T, i_c^2, i_c^3\}$, and for $j_c^{(2)}$ the eligible machines are $\{i_z^F, i_c^2, i_c^3\}$ (as z appears negated in c).

Eligible Machines Interval Scheduling is FPT w.r.t. $p_{\max} + m$

אוניברסיטת בן-גוריון בנגב
جامعة بن غوريون في النقب
Ben-Gurion University of the Negev

Theorem 3

Eligible Machines Interval Scheduling *is* FPT *with respect to the combined parameter* $p_{\max} + m$*, the longest processing time of a job and the number of machines.*

- Removal of jobs such that bounded number of jobs per time-step by $f(p_{\max}, m)$
- Dynamic program over the time-steps
- Key ideas:
  - At any time, at most m jobs can be processed
  - Some jobs are superior to others

Discussion

For m there exists no FPT algorithm for Interval Scheduling on Eligible Machines.

- Does there exists FPT algorithm when **uniform weights**?
- Does there exists FPT algorithm when **uniform processing times**?

References

[AS87] Esther M. Arkin and Ellen B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1–8, 1987.

[MvB18] Matthias Mnich and René van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, 2018.

[SV05] Shao Chin Sung and Milan Vlach. Maximizing weighted number of just-in-time jobs on unrelated parallel machines. *Journal of Scheduling*, 8(5):453–460, 2005.