Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time

Jesper Nederlof  
Céline Swennenhuis  
Karol Węgrzycki

UU  
TU/e  
Saarland University

The results presented were obtained during the trimester on Discrete Optimization at Hausdorff Research Institute for Mathematics (HIM) in Bonn, Germany.

Department of Mathematics and Computer Science, Eindhoven University of Technology
$P|prec, p_j = 1|C_{\text{max}}$
Given: 

- $n$ jobs of length 1
$P|prec, p_j = 1|C_{\text{max}}$

Given:

- $n$ jobs of length 1
- $m$ (identical, parallel) machines

$m = 2$
Given:
- $n$ jobs of length 1
- $m$ (identical, parallel) machines
- A precedence graph $G$

$P|prec, p_j = 1|C_{max}$

$\text{Makespan Scheduling of Unit Jobs with Precedence Constraints in } O(1.995^n) \text{ time}$
The problem is denoted as $P|\text{prec},p_j = 1|C_{\text{max}}$.

**Given:**
- $n$ jobs of length 1
- $m$ (identical, parallel) machines
- A precedence graph $G$
- $T \in \mathbb{N}$

**Q:** Is there a schedule of makespan $T$?
\( P|\text{prec}, p_j = 1|C_{\text{max}} \)

Given:

- \( n \) jobs of length 1
- \( m \) (identical, parallel) machines
- A precedence graph \( G \)
- \( T \in \mathbb{N} \)

Q: Is there a schedule of makespan \( T \)?

Makespan Scheduling of Unit Jobs with Precedence Constraints in \( O(1.995^n) \) time
Literature overview $P|prec, p_j = 1|C_{\text{max}}$
Literature overview $P|prec, p_j = 1|C_{\text{max}}$

- NP-complete$^1$ \( m = \text{#machines given as input} \)

Literature overview $P|prec, p_j = 1|C_{\text{max}}$

- NP-complete\(^1\) $m = \# \text{machines given as input}$
  

- Poly-time solvable\(^2\) for $m = 2$
  
Literature overview \( P|\text{prec}, p_j = 1|C_{\text{max}} \)

- **NP-complete\(^1\)** \( m = \#\text{machines given as input} \)
  

- **Poly-time solvable\(^2\)** for \( m = 2 \)
  

- **???** for \( m \geq 3 \) constant \(^3\) OPEN
  
Definitions

Precendence Constraints Graph $G$
**Definitions**

\[ G \Rightarrow \text{partial order:} \]

\[ i < j \quad \text{if} \quad (i, j) \in G \]

Precedence Constraints Graph \( G \)
Definitions

\[ G \Rightarrow \text{partial order:} \]
\[ i < j \quad \text{if} \quad (i, j) \in G \]

**Definition:** Let \( A \) be a set of jobs.

\[ \text{pred}[A] = \{ x \mid \exists \, a \in A \text{ s.t. } x \preceq a \} \]

\[ \text{succ}[A] = \{ x \mid \exists \, a \in A \text{ s.t. } x \succeq a \} \]

\[ \text{sinks}(A) = \max\{A\} \]
Definitions

\( G \Rightarrow \) partial order:
- \( i < j \) if \( (i, j) \in G \)

**Definition:** Let \( A \) be a set of jobs.
- \( \text{pred}[A] = \{ x \mid \exists a \in A \text{ s.t. } x \preceq a \} \)
- \( \text{succ}[A] = \{ x \mid \exists a \in A \text{ s.t. } x \succeq a \} \)
- \( \text{sinks}(A) = \max\{A\} \)

Let \( A = \{c, d\} \), then:

Precendence Constraints Graph \( G \)
Definitions

$G \Rightarrow$ partial order:
- $i < j$ if $(i, j) \in G$

**Definition:** Let $A$ be a set of jobs.
- $\text{pred}[A] = \{x \mid \exists a \in A \text{ s.t. } x \preceq a\}$
- $\text{succ}[A] = \{x \mid \exists a \in A \text{ s.t. } x \succeq a\}$
- $\text{sinks}(A) = \max\{A\}$

Let $A = \{c, d\}$, then:
- $\text{pred}[A] = \{a, c, d\}$
Definitions

\( G \Rightarrow \) partial order:
- \( i < j \) if \( (i, j) \in G \)

**Definition:** Let \( A \) be a set of jobs.
- \( \text{pred}[A] = \{x \mid \exists a \in A \text{ s.t. } x \preceq a\} \)
- \( \text{succ}[A] = \{x \mid \exists a \in A \text{ s.t. } x \succeq a\} \)
- \( \text{sinks}(A) = \max\{A\} \)

Let \( A = \{c, d\} \), then:
- \( \text{pred}[A] = \{a, c, d\} \)
- \( \text{succ}[A] = \{c, d, e, f\} \)

Precendence Constraints Graph \( G \)
**Definitions**

$G \Rightarrow$ partial order:
- $i < j$ if $(i, j) \in G$

**Definition:** Let $A$ be a set of jobs.

- $\text{pred}[A] = \{x \mid \exists a \in A \text{ s.t. } x \preceq a\}$
- $\text{succ}[A] = \{x \mid \exists a \in A \text{ s.t. } x \succeq a\}$
- $\text{sinks}(A) = \max\{A\}$

Let $A = \{c, d\}$, then:
- $\text{pred}[A] = \{a, c, d\}$
- $\text{succ}[A] = \{c, d, e, f\}$
- $\text{sinks}([a, c, d]) = \{c, d\}$
Definitions

Precendence Constraints Graph $G$
Definitions

**Def:** An *antichain* is a set $A$ whose elements are pairwise incomparable.

Precendence Constraints Graph $G$
Definitions

**Def:** An *antichain* is a set $A$ whose elements are pairwise incomparable.

Ex. of antichains in $G$

✓ $\{b, c, d\}$
✓ $\{b, c\}$
✓ $\{d, f\}$

Precendence Constraints Graph $G$
Definitions

**Def:** An *antichain* is a set $A$ whose elements are pairwise incomparable.

There is a one-to-one relation between:

- $A =$ antichain of graph $G$
- $\text{pred}[A]$
- $\text{sinks}(X)$
- $X =$ possible set of jobs to schedule

Precendence Constraints Graph $G$
DP Algorithm:

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995n)$ time
DP Algorithm:

\[ T(A, t) = \begin{cases} 
  \text{true,} & \text{if } \text{pred}[A] \text{ can be processed in } t \text{ time,} \\
  \text{false,} & \text{otherwise.} 
\end{cases} \]
DP Algorithm:

\[ T(A, t) = \begin{cases} 
\text{true,} & \text{if } \text{pred}[A] \text{ can be processed in } t \text{ time,} \\
\text{false,} & \text{otherwise.} 
\end{cases} \]
DP Algorithm:

\[ T(A, t) = \begin{cases} 
\text{true}, & \text{if } \text{pred}[A] \text{ can be processed in } t \text{ time,} \\
\text{false}, & \text{otherwise.} 
\end{cases} \]

To compute \( T(A, t) \):
try all \( \binom{n}{m} \) possible jobs at time \( t \)
DP Algorithm:

\[ T(A, t) = \begin{cases} 
  \text{true,} & \text{if } \text{pred}[A] \text{ can be processed in } t \text{ time,} \\
  \text{false,} & \text{otherwise.} 
\end{cases} \]

To compute \( T(A, t) \):
try all \( \binom{n}{m} \) possible jobs at time \( t \)

Runs in \( \text{poly}(n) \cdot \#AC \cdot \binom{n}{m} \) time
**DP Algorithm:**

\[ T(A, t) = \begin{cases} 
  \text{true,} & \text{if } \text{pred}[A] \text{ can be processed in } t \text{ time,} \\
  \text{false,} & \text{otherwise.} 
\end{cases} \]

To compute \( T(A, t) \):
- try all \( \binom{n}{m} \) possible jobs at time \( t \)

Runs in \( \text{poly}(n) \cdot \#\text{AC} \cdot \binom{n}{m} \) time

\[ \Rightarrow \text{poly}(n) \cdot 2^n \cdot \binom{n}{m} \] time algorithm

Before our work:
- No \( \text{poly}(n) \cdot 2^n \) time algorithm known!
$P|\text{prec}, p_j = 1|C_{\text{max}}$

Our result:

$P|\text{prec}, p_j = 1|C_{\text{max}}$ can be solved in $O(1.995^n)$ time.
\( P|prec, p_j = 1|C_{\text{max}} \)

**Our result:**

\( P|prec, p_j = 1|C_{\text{max}} \) can be solved in \( O(1.995^n) \) time.

*Proof:* Combination of three algorithms
$P|\text{prec}, p_j = 1|C_{\text{max}}$
$P|\text{prec}, p_j = 1|C_{\text{max}}$

- DP Algorithm
  - Dynamic Programming
  - $\text{poly}(n) \cdot \#AC \cdot \binom{n}{m}$ time
$P|\text{prec}, p_j = 1|C_{\text{max}}$

- DP Algorithm
  - Dynamic Programming
  - $\text{poly}(n) \cdot \#AC \cdot \binom{n}{m}$ time

- Fast Subset Convolution Algorithm
  - Non-trivial application of Björklund, Husfeldt and Koivisto (SICOMP 2009)
  - $\text{poly}(n) \cdot (\#AC + 2^{n-m})$ time
$P|prec, p_j = 1|C_{\text{max}}$

- **DP Algorithm**
  - Dynamic Programming
  - $poly(n) \cdot \#AC \cdot \binom{n}{m}$ time

- **Fast Subset Convolution Algorithm**
  - Non-trivial application of Björklund, Husfeldt and Koivisto (SICOMP 2009)
  - $poly(n) \cdot (\#AC + 2^{n-m})$ time

- **Fixed-parameter Tractable Algorithm**
  - Parameterized by *vertex cover of comparability graph*
  - $poly(n) \cdot 169^{|C|}$ time
$P|\text{prec}, \text{p}_j = 1|C_{\text{max}}$

- DP Algorithm
  - Dynamic Programming
  - $\text{poly}(n) \cdot \#AC \cdot \binom{n}{m}$ time

- Fast Subset Convolution Algorithm
  - Non-trivial application of Björklund, Husfeldt and Koivisto (SICOMP 2009)
  - $\text{poly}(n) \cdot (\#AC + 2^{n-m})$ time

- Fixed-parameter Tractable Algorithm
  - Parameterized by vertex cover of comparability graph
  - $\text{poly}(n) \cdot 169|C|$ time

Fast enough when $|C| < n/7.5$
$P|prec, p_j = 1|C_{\text{max}}$

- DP Algorithm
  - Dynamic Programming
  - $poly(n) \cdot \#AC \cdot \binom{n}{m}$ time

- Fast Subset Convolution Algorithm
  - Non-trivial application of Björklund, Husfeldt and Koivisto (SICOMP 2009)
  - $poly(n) \cdot (\#AC + 2^{n-m})$ time

- Fixed-parameter Tractable Algorithm
  - Parameterized by vertex cover of comparability graph
  - $poly(n) \cdot 169|C|^{\lceil|C|/n\rceil}$ time

Fast enough when $|C| \geq n/7.5$

Dilworth’s Theorem

$\#AC \leq 1.94^n$

Fast enough when $|C| < n/7.5$
$P|prec, p_j = 1|C_{\text{max}}$

- DP Algorithm
  - Dynamic Programming
  - $poly(n) \cdot \#AC \cdot \binom{n}{m}$ time

- Fast Subset Convolution Algorithm
  - Non-trivial application of Björklund, Husfeldt and Koivisto (SICOMP 2009)
  - $poly(n) \cdot (\#AC + 2^{n-m})$ time

- Fixed-parameter Tractable Algorithm
  - Parameterized by vertex cover of comparability graph
  - $poly(n) \cdot 169|C|$ time

Fast enough when $|C| \geq n/7.5$

Dilworth’s Theorem

$\#AC \leq 1.94^n$

Fast enough when $|C| < n/7.5$
What is vertex cover of comparability graph?
What is vertex cover of comparability graph?

Let $G = (V, A)$ be input. Then $G^{\text{comp}} := (V, E)$ where $(v, w) \in E$ if $v$ and $w$ are comparable.
What is vertex cover of comparability graph?

Let $G = (V, A)$ be input. Then $G^{\text{comp}} := (V, E)$ where $(v, w) \in E$ if $v$ and $w$ are comparable.
What is vertex cover of comparability graph?

Let $G = (V, A)$ be input. Then $G^{\text{comp}} := (V, E)$ where $(v, w) \in E$ if $v$ and $w$ are comparable.

$$C := \text{smallest vertex cover of } G^{\text{comp}}$$
What is vertex cover of comparability graph?

Let $G = (V, A)$ be input. Then $G^{\text{comp}} := (V, E)$ where $(v, w) \in E$ if $v$ and $w$ are comparable.

$$C := \text{smallest vertex cover of } G^{\text{comp}}$$
What is vertex cover of comparability graph?

Let $G = (V,A)$ be input. Then $G_{comp} := (V,E)$ where $(v,w) \in E$ if $v$ and $w$ are comparable.

$C := $ smallest vertex cover of $G_{comp}$

**Claim:** $V \setminus C$ is an antichain.
Sink-Adjusted Schedule

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

$z \in [1, T]$ is a sink moment if there are both sinks and non-sinks at time $z$. 

sink($G$)
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

$\text{Assumption: } n = m \cdot T$

$= \text{sink}(G)$

$z \in [1, T]$ is a sink moment if there are both sinks and non-sinks at time $z$. 

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

$z \in [1, T]$ is a **sink moment** if there are both sinks and non-sinks at time $z$. 

$\approx \text{succ}(H_1) \cup \text{sinks}$
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

$z \in [1, T]$ is a sink moment if there are both sinks and non-sinks at time $z$. 

$V \setminus (\text{succ}[H_1] \cup \text{sinks}) = \text{sink}(G)$
Sink-Adjusted Schedule

Assumption: \( n = m \cdot T \)

\[ z \in [1, T] \text{ is a } sink \text{ moment} \text{ if there are both sinks and non-sinks at time } z. \]
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

$V \setminus (\text{succ}[H_1] \cup \text{sinks})$

$z \in [1, T]$ is a **sink moment** if there are both sinks and non-sinks at time $z$. 

$T$ is a sink moment if there are both sinks and non-sinks at time $z$. 

$z_1, z_2, z_3, z_4$
Sink-Adjusted Schedule

Assumption: \( n = m \cdot T \)

\( \text{sink}(G) = \)

\( z \in [1, T] \) is a \textbf{sink moment} if there are both sinks and non-sinks at time \( z \).

No sinks in
Middle-Adjusted Schedule

Assumption: $n = m \cdot T$

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time
Middle-Adjusted Schedule

Assumption: $n = m \cdot T$
Middle-Adjusted Schedule

Assumption: \( n = m \cdot T \)

Makespan Scheduling of Unit Jobs with Precedence Constraints in \( O(1.995^n) \) time
Middle-Adjusted Schedule

Assumption: $n = m \cdot T$

$|C^L| \approx \frac{|C|}{2}$ \hspace{1cm} $|C^R| \approx \frac{|C|}{2}$
Middle-Adjusted Schedule

Assumption: \( n = m \cdot T \)

\[
|C^L| \approx \frac{|C|}{2} \quad \quad \quad |C^R| \approx \frac{|C|}{2}
\]
Middle-Adjusted Schedule

Assumption: \( n = m \cdot T \)

Undecided \( U \) := \( V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R]) \)
Middle-Adjusted Schedule

Assumption: $n = m \cdot T$

Undecided $U := V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
- $U$ are sources in $R$
Algorithm

Assumption: $n = m \cdot T$

Undecided $U := V \setminus (pred[C^L] \cup succ[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
- $U$ are sources in $R$
Algorithm

\[ \text{Slots for } sinks(C^L) \text{ and } sources(C^R) \]

Not-yet assigned jobs

Makespan Scheduling of Unit Jobs with Precedence Constraints in \( O(1.995^n) \) time
Algorithm

Find partition of $U$: $U^L$ and $U^R$

Slots for $\text{sources}(C^L)$ and $\text{sinks}(C^R)$

Not-yet assigned jobs
Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time

Assumption: $n = m \cdot T$

$|C^L| \approx \frac{|C|}{2}$

$|C^R| \approx \frac{|C|}{2}$
Algorithm

Assumption: $n = m \cdot T$

<table>
<thead>
<tr>
<th>$C^L$</th>
<th>$pred[C^L]$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U^L$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$C^R$</th>
<th>$succ[C^R]$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U^R$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$|C^L| \approx \frac{|C|}{2}$

$|C^R| \approx \frac{|C|}{2}$

Guesses: $13^{|C|} \Rightarrow O(poly(n) \cdot 169^{|C|})$ time.
Conclusion

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time
Conclusion

Main result:

\( P | prec, p_j = 1 | C_{\text{max}} \) in \( O(1.995^n) \) time.
Conclusion

Main result:

\[ P|\text{prec}, p_j = 1|C_{\text{max}} \text{ in } O(1.995^n) \text{ time.} \]

Key idea’s:

- Tradeoff between \#AC’s and \(|C|\).
- \[ O(poly(n) \cdot 169|C|) \text{ time algorithm} \]
Conclusion

Main result:

\[ P_{\text{prec}, p_j = 1} | C_{\text{max}} \text{ in } O(1.995^n) \text{ time.} \]

Key idea’s:

• Tradeoff between \#AC’s and \(|C|\).
• \( O(\text{poly}(n) \cdot 169|C|) \) time algorithm

Future Research:

\[ P3_{\text{prec}, p_j = 1} | C_{\text{max}} \text{ in } 2^o(n) \text{ time?} \]
Conclusion

Main result:

\( P|\text{prec}, p_j = 1|C_{\text{max}} \text{ in } O(1.995^n) \text{ time.} \)

Key idea’s:

- Tradeoff between \#AC’s and \(|C|\).
- \( O(poly(n) \cdot 169^{|C|}) \text{ time algorithm} \)

Future Research:

\( P3|\text{prec}, p_j = 1|C_{\text{max}} \text{ in } 2^o(n) \text{ time?} \)
Size of $C$ vs $\#AC’s$

Claim: $|C| \geq n/7.5 \Rightarrow \#AC’s \leq 1.94^n$

Proof. $|C| \geq n/7.5 \Rightarrow \text{largest AC } \leq n - \frac{n}{7.5} = \alpha n$ with $\alpha = \left(1 - \frac{1}{7.5}\right)$.

Dilworth’s Theorem: $\exists$ chains $C_1, ..., C_{\alpha n}$ that partition $G$.

\[
\#AC’s = \prod_{i=1}^{\alpha n} (|C_i| + 1) \leq \left(\frac{n}{\alpha n} + 1\right)^{\alpha n} = 1.945^n
\]
Algorithm

Assumption: $n = m \cdot T$

Undecided $U := V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
Algorithm

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time

Assumption: $n = m \cdot T$

- Assume sink-adjusted

Undecided $U := V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
Algorithm

Assumption: $n = m \cdot T$

Undediced $U := V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$

- Assume sink-adjusted
Algorithm

Assumption: $n = m \cdot T$

Undecided $U \equiv V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
Algorithm

Assumption: \( n = m \cdot T \)

- Assume sink-adjusted

Assume that \( H_1, H_2, \ldots \subseteq C^L \)

- Guess jobs in \( H \) from \( C^L \)
- Reconstruct \( H_1, H_2, \ldots \)
- Reconstruct
- Distribute sinks

Undecided \( U \coloneqq V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R]) \)

- \( U \) is antichain
- \( U \) are sinks in \( L \)
Algorithm

Undecided \( U \) := \( V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R]) \)

- \( U \) is antichain
- \( U \) are sinks in \( L \)

Assumption: \( n = m \cdot T \)

- Assume sink-adjusted

Assume that \( H_1, H_2, \ldots \subseteq C^L \)

- Guess jobs in \( H \) from \( C^L \)
- Reconstruct \( H_1, H_2, \ldots \)
- Reconstruct
- Distribute sinks

TU/e

Makespan Scheduling of Unit Jobs with Precedence Constraints in \( O(1.995^n) \) time
Algorithm

Assumption: $n = m \cdot T$

- Assume sink-adjusted

Assume that $H_1, H_2, \ldots \subseteq C^L$

- Guess jobs in $H$ from $C^L$
- Reconstruct $H_1, H_2, \ldots$
- Reconstruct
- Distribute sinks

Undecided $U := V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
Algorithm

Assumption: \( n = m \cdot T \)

- Assume sink-adjusted

Assume that \( H_1, H_2, \ldots \subseteq C^L \)

- Guess jobs in \( H \) from \( C^L \)
- Reconstruct \( H_1, H_2, \ldots \)
- Reconstruct
- Distribute sinks

Undediced \( U := V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R]) \)

- \( U \) is antichain
- \( U \) are sinks in \( L \)
Algorithm

Assumption: $n = m \cdot T$

- Assume sink-adjusted

Assume that $H_1, H_2, \ldots \subseteq C^L$

- Guess jobs in $H$ from $C^L$
- Reconstruct $H_1, H_2, \ldots$
- Reconstruct $H_1, H_2, \ldots$
- Distribute sinks

Undecided $U \equiv V \setminus (\text{pred}[C^L] \cup \text{succ}[C^R])$

- $U$ is antichain
- $U$ are sinks in $L$
Algorithm

Makespan Scheduling of Unit Jobs with Precedence Constraints in $O(1.995^n)$ time
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

Claim: if we know $\cup_i H_i$, we can derive $H_1, H_2, \ldots, H_l$
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

Claim: if we know $\bigcup_i H_i$, we can derive $H_1, H_2, \ldots, H_l$
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

Claim: if we know $\bigcup_i H_i$, we can derive $H_1, H_2, ..., H_l$
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

Claim: if we know $\bigcup_i H_i$, we can derive $H_1, H_2, \ldots, H_l$
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

Claim: if we know $\bigcup_i H_i$, we can derive $H_1, H_2, ..., H_l$

⇒ for each we know which jobs are in there.

$V \setminus (\text{succ}[H_1] \cup \text{sinks})$

TU/e
Sink-Adjusted Schedule

How to use this...?

Assumption: $n = m \cdot T = \sin \left( G \right)$

Claim: if we know $\bigcup_i H_i$, we can derive $H_1, H_2, \ldots, H_l$

⇒ for each we know which jobs are in there.