Problem Definition

- Number of machines
- Processing times for \( n \) jobs
- Precedence Graph \( G \)
- \( k = \) minimal number of jobs to schedule
- Makespan / Universal Deadline \( D \)

<table>
<thead>
<tr>
<th>Job</th>
<th>( p_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>6</td>
</tr>
<tr>
<td>f</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ k = 5 \]
Problem Definition

- Number of machines
- Processing times for $n$ jobs
- Precedence Graph $G$
- $k = \text{minimal number of jobs to schedule}$
- Makespan / Universal Deadline $D$

### Parameterized Complexity of Partial Scheduling

$$t = 0$$

**Jobs**

<table>
<thead>
<tr>
<th>Job</th>
<th>$p_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>6</td>
</tr>
<tr>
<td>f</td>
<td>3</td>
</tr>
</tbody>
</table>

**Diagram:**

- $t = 8$
- $k = 5$

**Timeline:**

- $t = 0$
- $t = 8$

- Jobs: a, c, f
- Jobs: b, d

**Diagram:**

- Graph $G$ with nodes a, b, c, d, e, f
- Edges: a → b, b → c, c → d, d → e, e → f
Motivation

More jobs available than we want to process

- Close-horizon approach
- Other jobs can be outsourced
Parameterized Complexity

If $k = n$, i.e. we want to schedule all jobs, the problem is NP-hard*.

Can we do better if $k$ is small?

**Definition:** A problem is called *fixed-parameter tractable (FPT) in $k$* if it can be solved in time $O(f(k) \cdot n^c)$, where $n$ is the size of the input.

It is possible to prove a problem to be $W[1]$-hard $\Rightarrow$ no such algorithm exists under ETH.

*Except for some special cases*
Trichotomy of Partial Scheduling

\[ P|prec, p_j = 1|k\text{-sched}, C_{\text{max}} \]  
\[ P|r_j, prec, p_j = 1|k\text{-sched}, C_{\text{max}} \]  
\[ P|d_j, prec, p_j = 1|k\text{-sched}, C_{\text{max}} \]  
\[ P|d_j, prec|k\text{-sched}, C_{\text{max}} \]

- FPT in \( k \)
- \( W[1] \)-hard

Parameterized Complexity of Partial Scheduling
Today:

**Theorem:** The problem $P|prec, p_j = 1|k - sched, C_{\text{max}}$ is fixed-parameter tractable (FPT) in $k$.

To do so, we will give an algorithm which runs in time $O(8^k \cdot poly(n))$. 
Definitions

Notice:
• $G$ is acyclic
• $i < j$ if $i \rightarrow j$

Def: Let $A$ be a set of jobs.
\[ \text{pred}(A) = \{x \mid \exists a \in A \text{ s.t. } x \leq a\} \]
\[ \text{comp}(A) = \{x \mid \exists a \in A \text{ s.t. } x \leq a \text{ or } x \geq a\} \]

Let $A = \{c, d\}$, then:
• $\text{pred}(A) = \{a, c, d\}$
• $\text{comp}(A) = \{a, c, d, e, f\}$

Precendence Constraints Graph $G$
**Definitions**

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

Ex. of antichains in $G$

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

Corresponding $\text{pred}(A)$

- $\Rightarrow \{a, b, c, d\}$
- $\Rightarrow \{a, b, c\}$
- $\Rightarrow \{a, b, c, d, f\}$

No antichain:

- × $\{b, f\}$
- × $\{a, e\}$

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:

- antichain of graph $G$
- $\text{pred}(A)$
- possible set of jobs to schedule

Precendence Constraints Graph $G$
Dynamic Program

\[ S(A, t) = \begin{cases} 
  \text{true}, & \text{if } \text{pred}(A) \text{ can be done before or at } t, \\
  \text{false}, & \text{else}
\end{cases} \]

If \( S(A, D) = \text{true} \), for some \( A \) with \( |\text{pred}(A)| \geq k \), then return YES.

For \( t = 0 \)

\[ S(A, 0) = \begin{cases} 
  \text{true}, & \text{if } A = \emptyset, \\
  \text{false}, & \text{else}
\end{cases} \]
Dynamic Program

\[ S(A, t) = \begin{cases} 
  \text{true}, & \text{if } \text{pred}(A) \text{ can be done before or at } t, \\
  \text{false}, & \text{else} 
\end{cases} \]

Only \( a \in A \) can be processed at time \( t \).

So, \( \binom{|A|}{m} \leq 2^{|A|} \) possibilities.

For each possibility:

Process subset at timeslot \([t - 1, t]\).

Check if leftover jobs can be finished at \( t - 1 \).

i.e. Check whether \( S(A', t - 1) = \text{true} \)
Dynamic Program

\[ S(A, t) = \begin{cases} 
  \text{true}, & \text{if } \text{pred}(A) \text{ can be done before or at } t, \\
  \text{false}, & \text{else} 
\end{cases} \]

Only \( a \in A \) can be processed at time \( t \).

So, \( \binom{|A|}{m} \leq 2^{|A|} \) possibilities.

For each possibility:

Process subset at timeslot \([t - 1, t]\).

Check if leftover jobs can be finished at \( t - 1 \).

i.e. Check whether \( S(A', t - 1) = \text{true} \)
Runtime Analysis

• Amount of $S(A, t)$ to compute:
  - $t \leq k$,
  - Number of antichains $A$ s.t. $|\text{pred}(A)| \leq k \leq \binom{n}{k} \leq n^k$,

• Computing $S(A, t)$: Trying all possible subsets of $A$ at $[t - 1, t]$
  - Assuming $|A| \leq k$, at most $2^k$ possibilities
Number of Antichains (Example)

Let $k = 3$

So at least $k^{k-1}$ possible antichains
Definition of depth

Definition: The depth of an antichain $A$ is defined as:

$$d(A) = |\text{pred}(A)| + |\min(G - \text{comp}(A))|$$

- $\text{pred}(A)$ represents the predecessors of $A$.
- $G - \text{comp}(A)$ represents the graph $G$ minus its complement.
- $\text{min}(G - \text{comp}(A))$ represents the minimum value in the graph $G$ minus its complement.

Diagram:
- Red nodes represent the antichain $A$.
- Pink nodes represent $\text{pred}(A)$.
- Blue nodes represent $\min(G - \text{comp}(A))$.
- The transformation from the left graph to the right graph illustrates the computation of $G - \text{comp}(A)$.
**Definition of depth**

**Definition:** The *depth* of an antichain $A$ is defined as:

$$d(A) = |pred(A)| + |\min(G - comp(A))|$$

**Idea:**

Only check antichains with $d(A) \leq k$.

**Theorem:** There are at most $4^k$ antichains with depth $d(A) \leq k$.

Is that enough?
Restricting Depth

Let $k = 3$

$\text{antichain } A$

$\text{pred}(A)$

$\text{min}(G - \text{comp}(A))$

$\quad d(A) = 7$

Don’t check!
Restricting Depth

Let $k = 3$

$\text{Let } k = 3$

$\text{Don’t check!}$

$d(A) = 4$

$\text{Don’t check!}$

- $\text{red} = \text{antichain } A$
- $\text{pink} = \text{pred}(A)$
- $\text{blue} = \text{min}(G - \text{comp}(A))$

Parameterized Complexity of Partial Scheduling
Restricting Depth

Let $k = 3$

$d(A) = 1$

- Red = antichain $A$
- Pink = $\text{pred}(A)$
- Blue = $\min(G - \text{comp}(A))$
Extending the Dynamic Program

Each time $S(A, t) = true$, ask:

Can we extend the given schedule to $k$ jobs before deadline $D$?

Greedily schedule available jobs between $t$ and $D$. 
Restricting Depth

Let $k = 3$
and $m = 2, D = 2$

$S(A, 1) = true$

➢ Find available jobs
➢ Greedily extend schedule
➢ Return YES
Runtime Analysis

- Amount of $S(A, t)$ to compute:
  - $t \leq k$,
  - $\#\{\text{antichains } A \text{ s.t. } |\text{pred}(A)| \leq k\} \leq \binom{n}{k} \leq n^k$,
  - $\#\{\text{antichains } A \text{ s.t. } d(A) \leq k\} \leq 4^k$

- Computing $S(A, t)$: Trying all possible subsets of $A$ at $[t - 1, t]$
  - Assuming $|A| \leq k$, at most $2^k$ possibilities
  - Expanding schedule in polynomial time

So runtime of $O(8^k \cdot \text{poly}(n))$. 
Summary / Open problems

• Partial Scheduling with parameter $k$
• Trichotomy of the parameterized complexity
• $P|prec, p_j = 1|k – sched, C_{max}$
  • Dynamic Program
  • Depth of an antichain
  • Runtime of $O(8^k \cdot poly(n))$

Open problem

• Can we improve this bound?

**Theorem:** There are at most $4^k$ antichains with depth $d(A) \leq k$. 