TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

A Subexponential Time Algorithm for Makespan Scheduling of Unit Jobs with Precedence Constraints

Jesper Nederlof

UU

Céline Swennenhuis

TU/e

Karol Węgrzycki

Saarland University

Department of Mathematics and Computer Science, Eindhoven University of Technology


TU/e

List of Open Problems by Garey and Johnson 1979

1. Graph Isomorphism
2. Subgraph Homeomorphism
3. Graph genus
4. Chordal graph completion
5. Chromatic index
6. Spanning tree parity problem
7. Partial order dimension
8. Precedence constrained 3-processor scheduling
9. Linear Programming
10. Total unimodularity
11. Composite number
12. Minimum length triangulation

List of Open Problems by Garey and Johnson 1979

1. Graph Isomorphism
2. Subgraph Homeomorphism
3. Graph genus
4. Chordal graph completion
5. Chromatic index
6. Spanning tree parity problem
7. Partial order dimension
8. Precedence constrained 3-processor scheduling
9. Linear Programming
10. Total unimodularity
11. Composite number
12. Minimum length triangulation

List of Open Problems by Garey and Johnson 1979

1. Graph Isomorphism
2. Subgraph Homeomorphism
3. Graph genus
4. Chordal graph completion
5. Chromatic index
6. Spanning tree parity problem
7. Partial order dimension
8. Precedence constrained 3-processor scheduling
9. Linear Programming
10. Total unimodularity
11. Composite number
12. Minimum length triangulation

TU/e

List of Open Problems by Garey and Johnson 1979

1. Graph Isomorphism
2. Subgraph Homeom... hism
3. Graph genus
4. Chordal graph ...
5. Chromatic index
6. Spanning tree parity problem
7. Partial order dimension
8. Precedence constrained 3-processor scheduling
9. Linear Programming
10. Total unimodularity
11. Composite number
12. Minimum length triangulation

$2^{O\left((\log n)^3\right)}$ time
[Babai 2017]

List of Open Problems by Garey and Johnson 1979

1. Graph Isomorphism
2. Subgraph Homeom... hism
3. Graph genus
4. Chordal graph ...
5. Chromatic index
6. Spanning tree parity problem
7. Partial order dimension
8. Precedence constrained 3-processor scheduling
9. Linear Program...
10. Total unimodul...
11. Composite nu...
12. Minimum length triangulation

$2^{O\left((\log n)^3\right)}$ time
[Babai 2017]

$2^{O\left(\sqrt{n}\cdot\log n\right)}$ time
This talk

TU/e

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

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

- NP-complete[1] m = #machines given *as input*

[1]Jeffrey D. Ullman. *NP-complete scheduling problems.* Journal of Computer and System sciences, 10(3):384–393, 1975.

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

- NP-complete[1] m = #machines given *as input*

[1]Jeffrey D. Ullman. *NP-complete scheduling problems.* Journal of Computer and System sciences, 10(3):384–393, 1975.

- Poly-time solvable[2] for $m = 2$

[2]M. Fujii, T. Kasami, and K. Ninomiya. *Optimal sequencing of two equivalent processors*. SIAM Journal on Applied Mathematics, 17(4):784–789, 1969.

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

- NP-complete[1] m = #machines given *as input*

[1]Jeffrey D. Ullman. *NP-complete scheduling problems.* Journal of Computer and System sciences, 10(3):384–393, 1975.

- Poly-time solvable[2] for $m = 2$

[2]M. Fujii, T. Kasami, and K. Ninomiya. *Optimal sequencing of two equivalent processors*. SIAM Journal on Applied Mathematics, 17(4):784–789, 1969.

- ???? for $m \geq 3$ **constant** **OPEN**[3]

[3]Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

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

- NP-complete[1] m = #machines given *as input*

[1]Jeffrey D. Ullman. *NP-complete scheduling problems.* Journal of Computer and System sciences, 10(3):384–393, 1975.

- Poly-time solvable[2]   for $m = 2$

[2]M. Fujii, T. Kasami, and K. Ninomiya. *Optimal sequencing of two equivalent processors*. SIAM Journal on Applied Mathematics, 17(4):784–789, 1969.

- ????   for $m \geq 3$ **constant**   **OPEN**[3]

[3]Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

Before:

$Pm|prec, p_j = 1|C_{\max}$ can be solved in $O\left(2^n \cdot \binom{n}{m}\right)$ time.

Our Result

Our result:

$Pm|prec, p_j = 1|C_{\max}$ can be solved in $\left(1 + \frac{n}{m}\right)^{O(\sqrt{nm})}$ time.

Our Result

Our result:

$Pm|prec, p_j = 1|C_{\max}$ can be solved in $\left(1 + \frac{n}{m}\right)^{O(\sqrt{nm})}$ time.

Corollary:

$Pm|prec, p_j = 1|C_{\max}$ can be solved in $2^{O(\sqrt{n} \cdot \log n)}$ time.

TU/e

Our Result

Our result:

$Pm|prec, p_j = 1|C_{\max}$ can be solved in $\left(1 + \frac{n}{m}\right)^{O(\sqrt{nm})}$ time.

Corollary:

$Pm|prec, p_j = 1|C_{\max}$ can be solved in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key points in algorithm:

1. New decomposition of schedules
2. Use of look-up table
3. Use of Dynamic Programming in combining results

$Pm|prec, p_j = 1|C_{\max}$

$$Pm|prec, p_j = 1|C_{\max}$$

m identical parallel machines

$m = 2$

$Pm|prec, p_j = 1|C_{\max}$

Given:

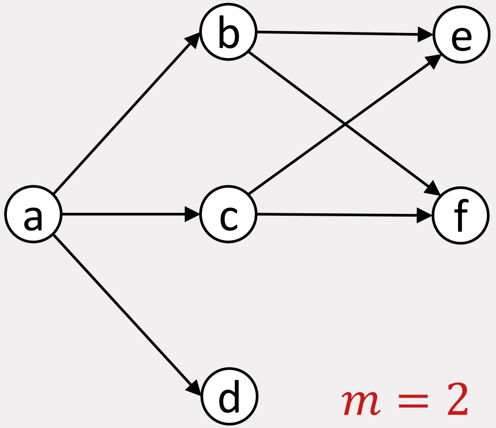
- n jobs of length 1

$m = 2$

$Pm|prec, p_j = 1|C_{\max}$

Given:

- n jobs of length 1
- A precedence graph G



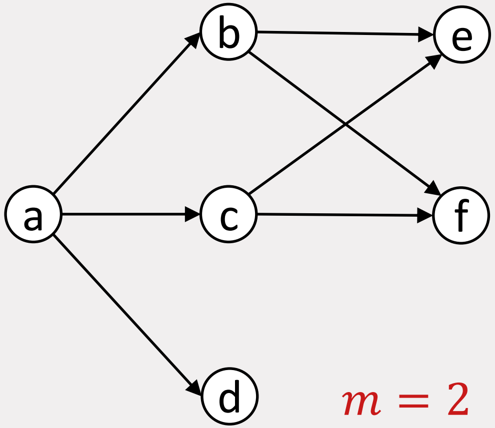
$m = 2$

$Pm|prec, p_j = 1|C_{\max}$

Given:

- n jobs of length 1
- A precedence graph G
- $T \in \mathbb{N}$

Q: Is there a schedule of makespan T?



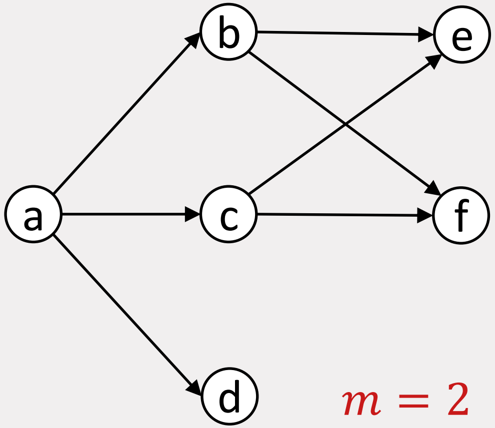
$m = 2$

$Pm|prec, p_j = 1|C_{\max}$

Given:

- n jobs of length 1
- A precedence graph G
- $T \in \mathbb{N}$

Q: Is there a schedule of makespan T?



$m = 2$

time →

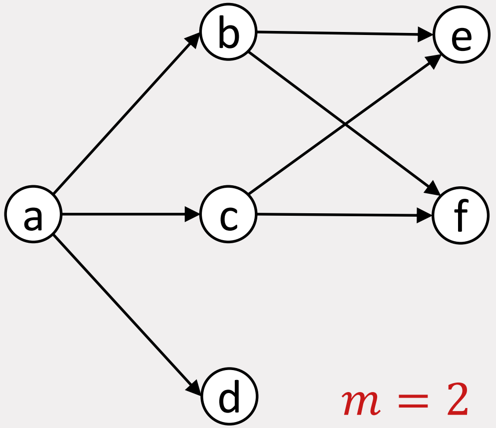
	1	2	3	4
1	a	b	d	f
2		c	e	

$Pm|prec, p_j = 1|C_{\max}$

Given:

- n jobs of length 1
- A precedence graph G
- $T \in \mathbb{N}$

Q: Is there a schedule of makespan T?

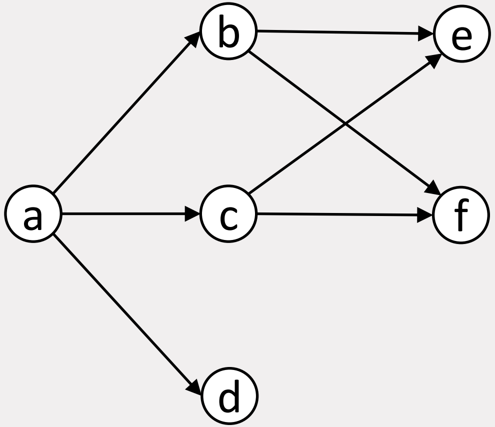


time →

	1	2	3	4
1	a	b	d	f
2		c	e	

Observation:
Jobs of length one ⇒ `timeslots’

Definitions

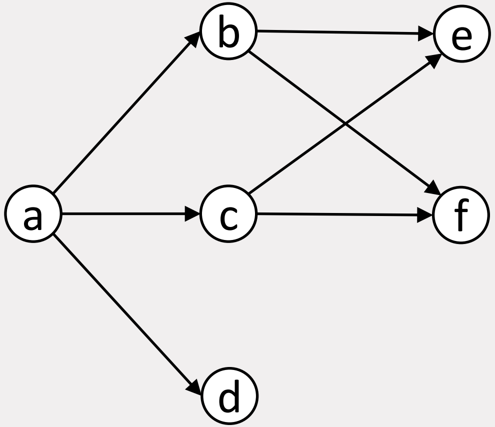


Precendence Constraints Graph G

Definitions

$G \Rightarrow$ partial order:

- $i \prec j$ if $(i, j) \in G$

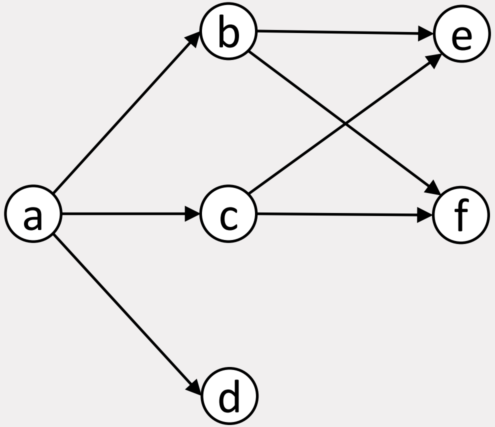


Precendence Constraints Graph G

Definitions

$G \Rightarrow$ partial order:

- $i \prec j$ if $(i,j) \in G$



Precendence Constraints Graph G

Definitions: Let A be a set of jobs.

$\text{pred}[A] \;=$

$\text{succ}[A] \;=$

Definitions

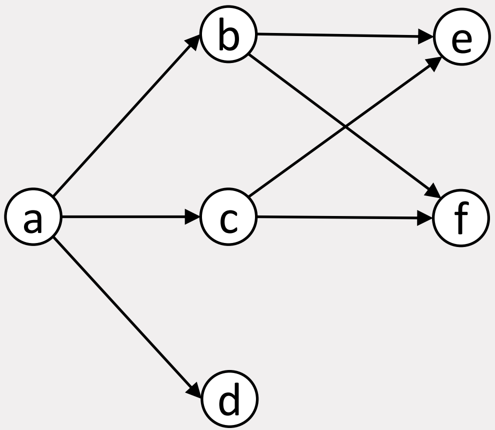
$G \Rightarrow$ partial order:

- $i \prec j$ if $(i, j) \in G$

Definitions: Let A be a set of jobs.

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

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

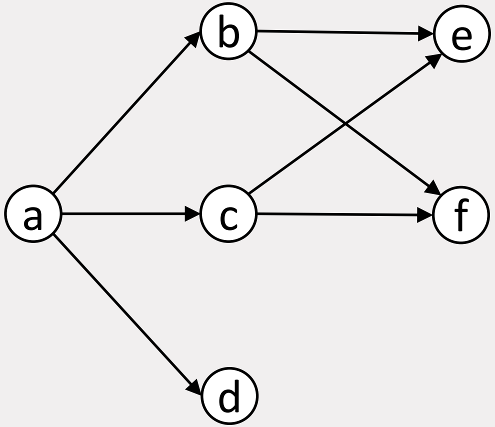


Precendence Constraints Graph G

Definitions

$G \Rightarrow$ partial order:

- $i \prec j$ if $(i,j) \in G$



Precendence Constraints Graph G

Definitions: Let A be a set of jobs.

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

sinks(A) $= \max\{A\}$

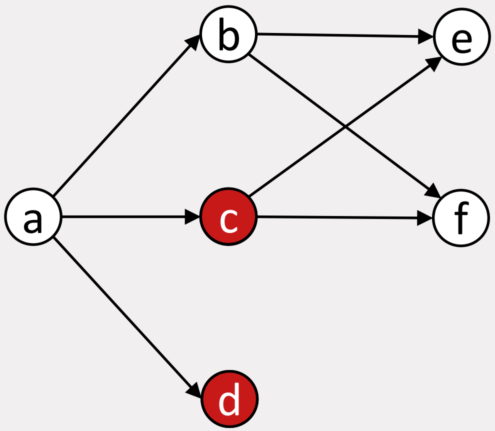
succ$[A]$ $= \{x \mid \exists\, a \in A\ s.t.\ x \succcurlyeq a\}$

sources(A) $= \min\{A\}$

Definitions

$G \Rightarrow$ partial order:

- $i \prec j$ if $(i,j) \in G$



Precendence Constraints Graph G

Definitions: Let A be a set of jobs.

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

$\text{sinks}(A) = \max\{A\}$

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

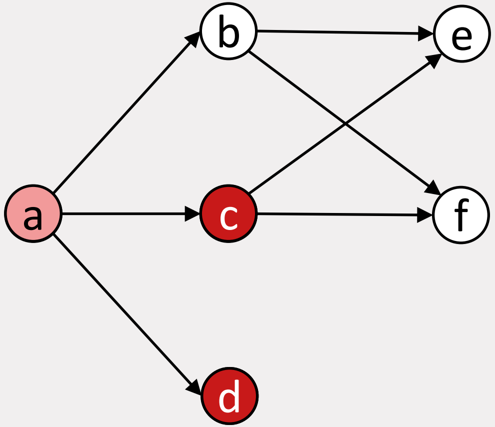
$\text{sources}(A) = \min\{A\}$

Let $A = \{c, d\}$, then:

Definitions

$G \Rightarrow$ partial order:

- $i \prec j$ if $(i, j) \in G$



Precendence Constraints Graph G

Definitions: Let A be a set of jobs.

$\text{pred}[A] = \{x \mid \exists\, a \in A\ s.t.\ x \preccurlyeq a\}$
$\text{sinks}(A) = \max\{A\}$
$\text{succ}[A] = \{x \mid \exists\, a \in A\ s.t.\ x \succcurlyeq a\}$
$\text{sources}(A) = \min\{A\}$

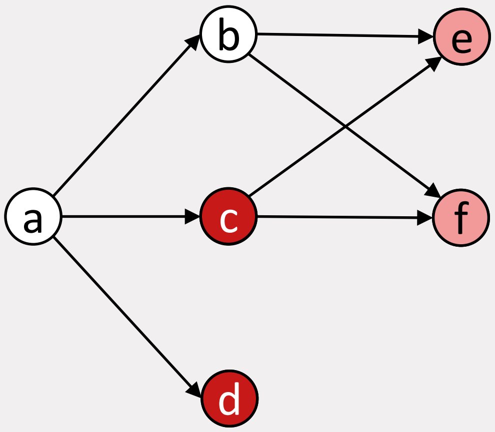
Let $A = \{c, d\}$, then:

- $\text{pred}[A] = \{a, c, d\}$
- $\text{sinks}(\{a, c, d\}) = \{c, d\} = A$

Definitions

$G \Rightarrow$ partial order:

- $i \prec j$ if $(i,j) \in G$



Precendence Constraints Graph G

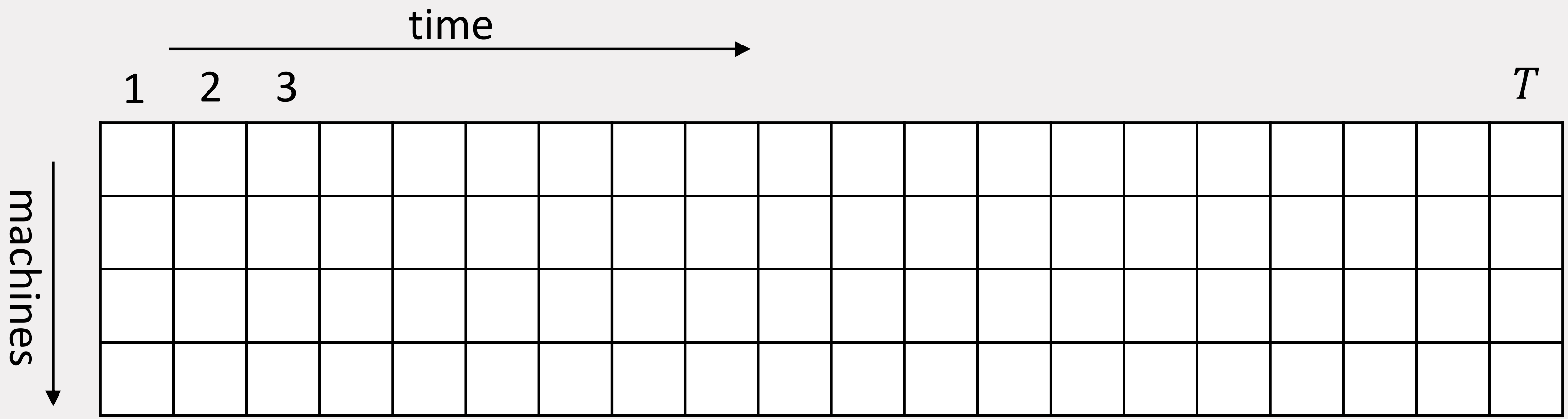
Definitions: Let A be a set of jobs.

$\text{pred}[A] = \{x \mid \exists\, a \in A \ s.t.\ x \preccurlyeq a\}$
$\text{sinks}(A) = \max\{A\}$
$\text{succ}[A] = \{x \mid \exists\, a \in A \ s.t.\ x \succcurlyeq a\}$
$\text{sources}(A) = \min\{A\}$

Let $A = \{c, d\}$, then:

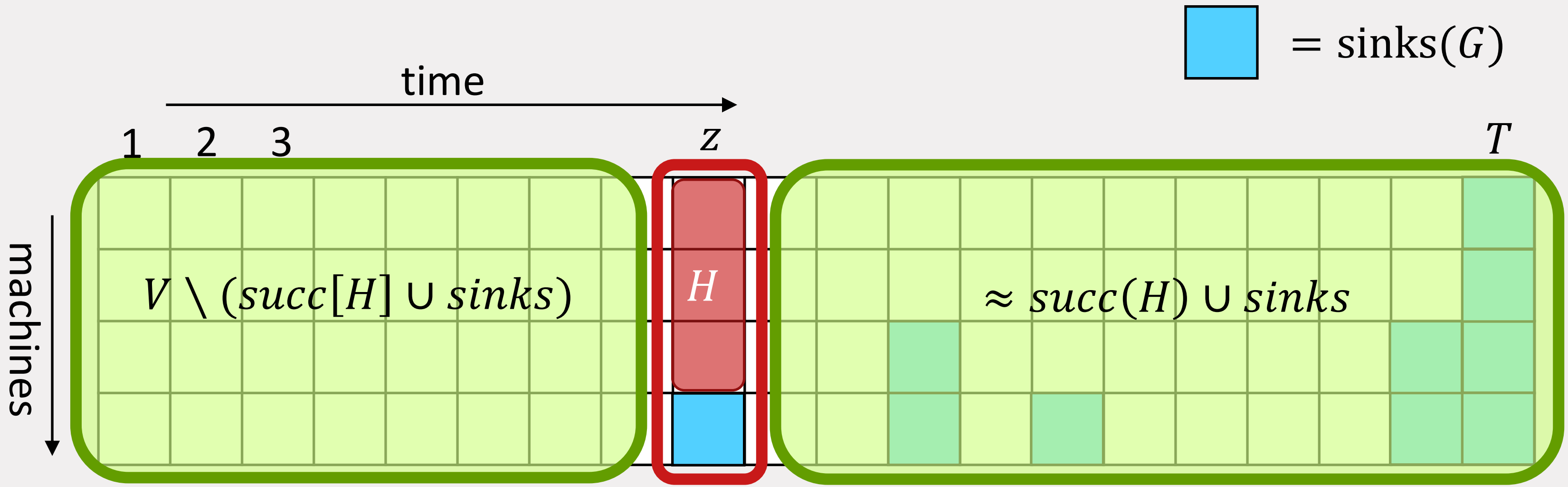
- $\text{pred}[A] = \{a, c, d\}$
- $\text{sinks}(\{a, c, d\}) = \{c, d\} = A$
- $\text{succ}[A] = \{c, d, e, f\}$
- $\text{sources}(\{c, d, e, f\}) = \{c, d\} = A$

Zero-Adjusted Schedule (D&W)



Zero-Adjusted Schedule (D&W)

Assumption: $n = m \cdot T$

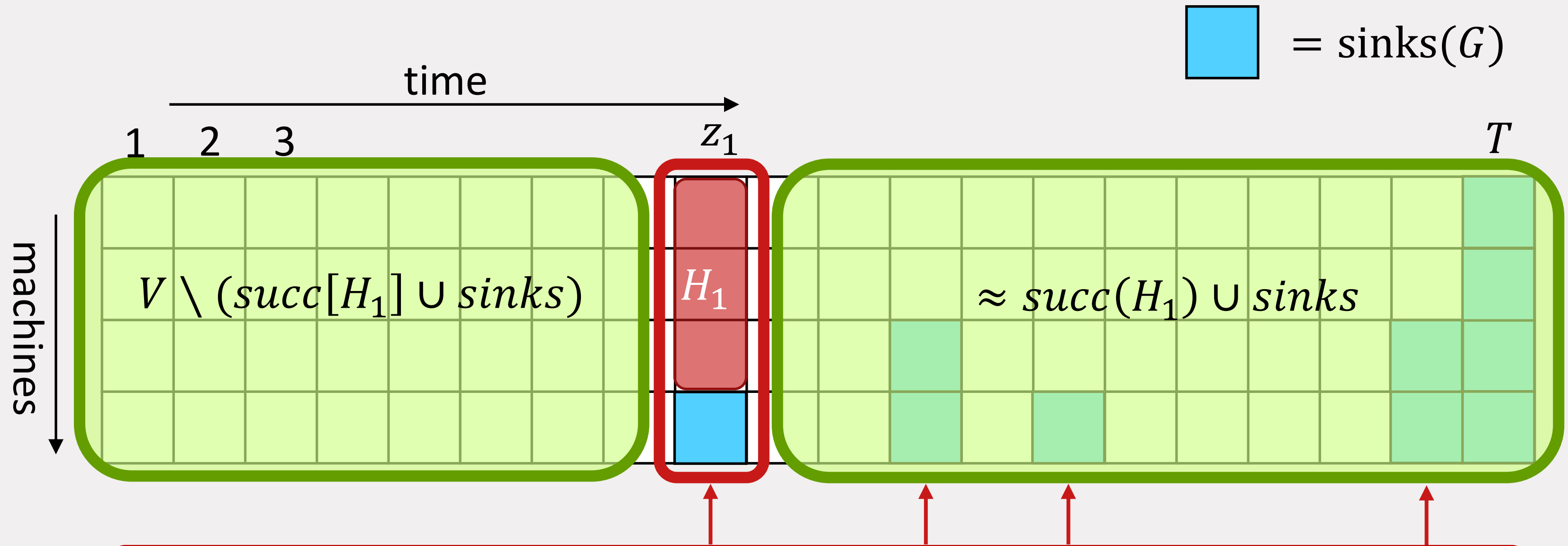


Let $z \in [1, T]$ be the first moment with a sink.

<u>D&W:</u> W.m.a. Each job x **after z** is a sink or a successor of a job at time z.

Sink-Adjusted Schedule

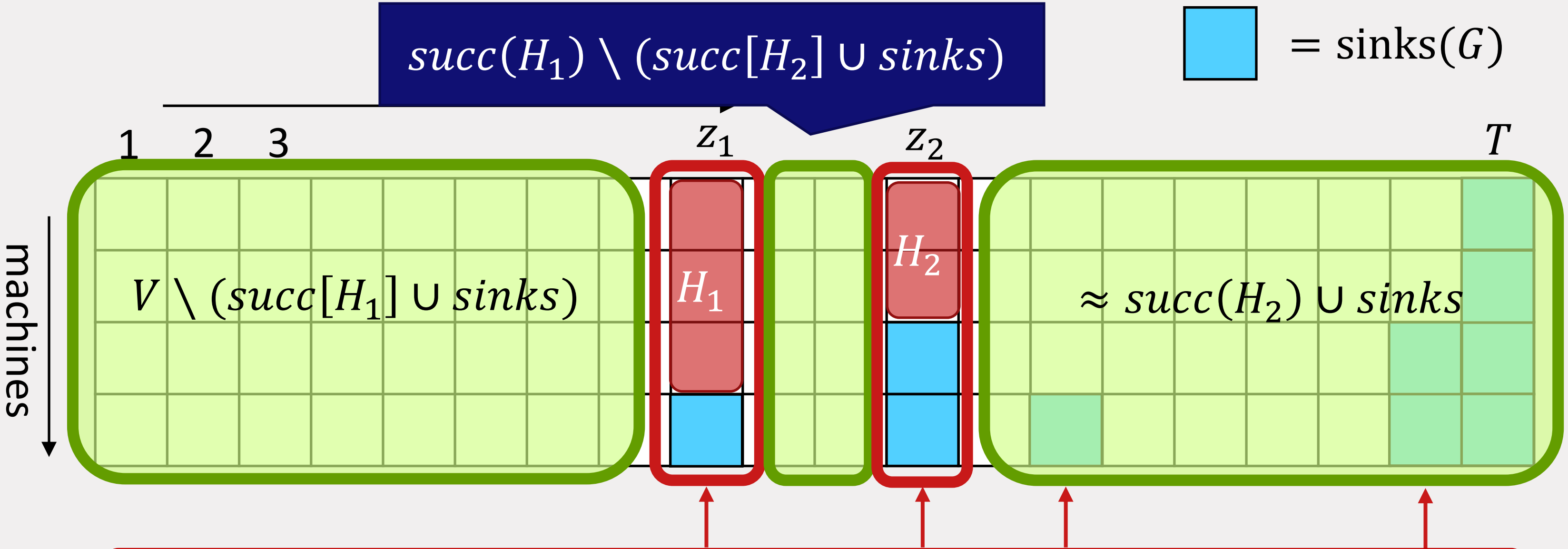
Assumption: $n = m \cdot T$



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

Sink-Adjusted Schedule

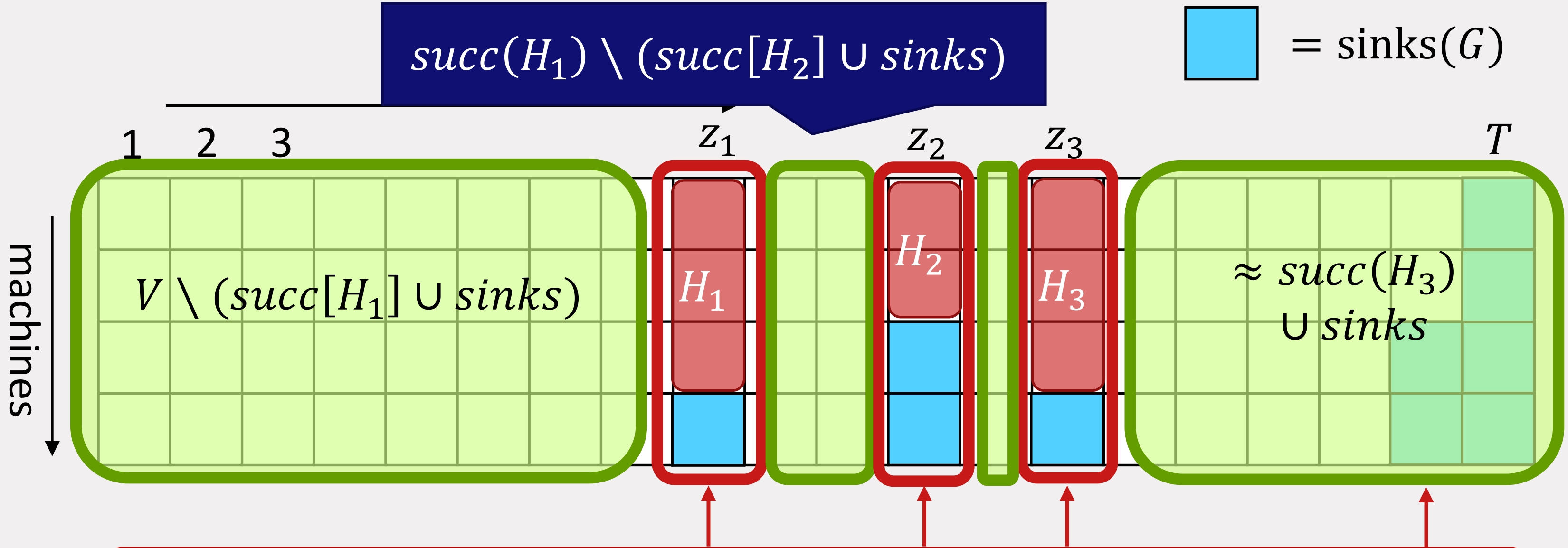
Assumption: $n = m \cdot T$



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

Sink-Adjusted Schedule

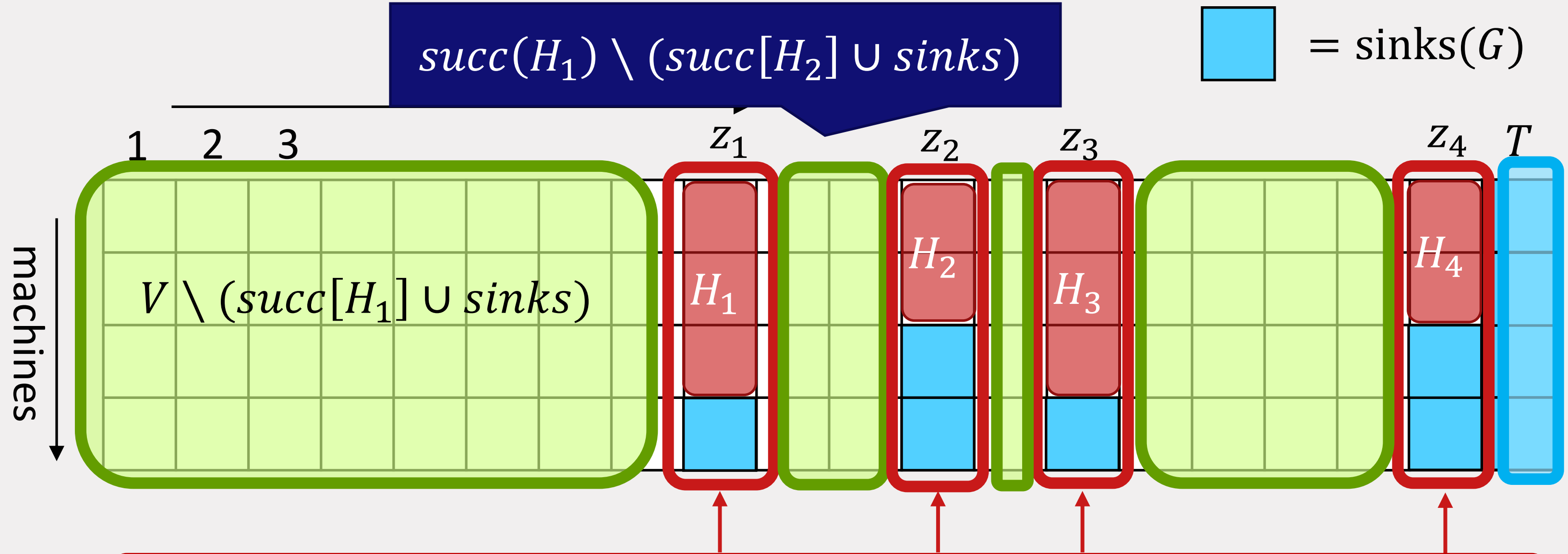
Assumption: $n = m \cdot T$



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

Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

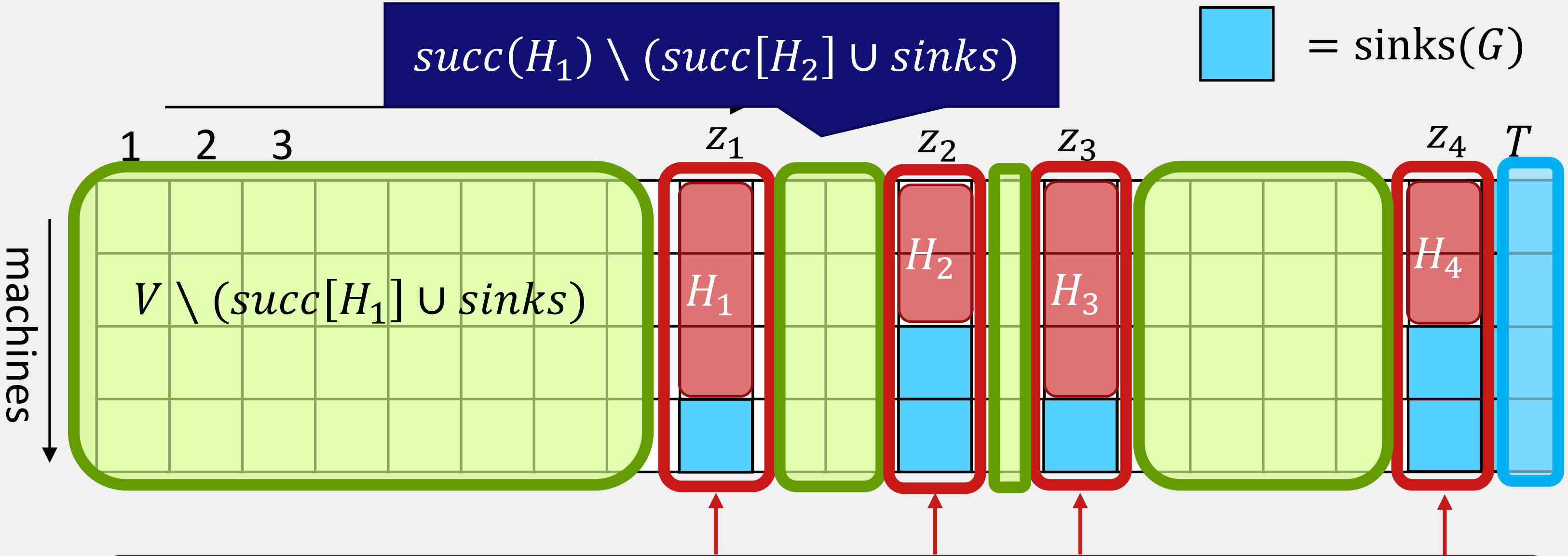


$z \in [1, T]$ is a ***sink moment*** if there are both sinks and non-sinks at time z.

TU/e

Sink-Adjusted Schedule

Assumption: $n = m \cdot T$

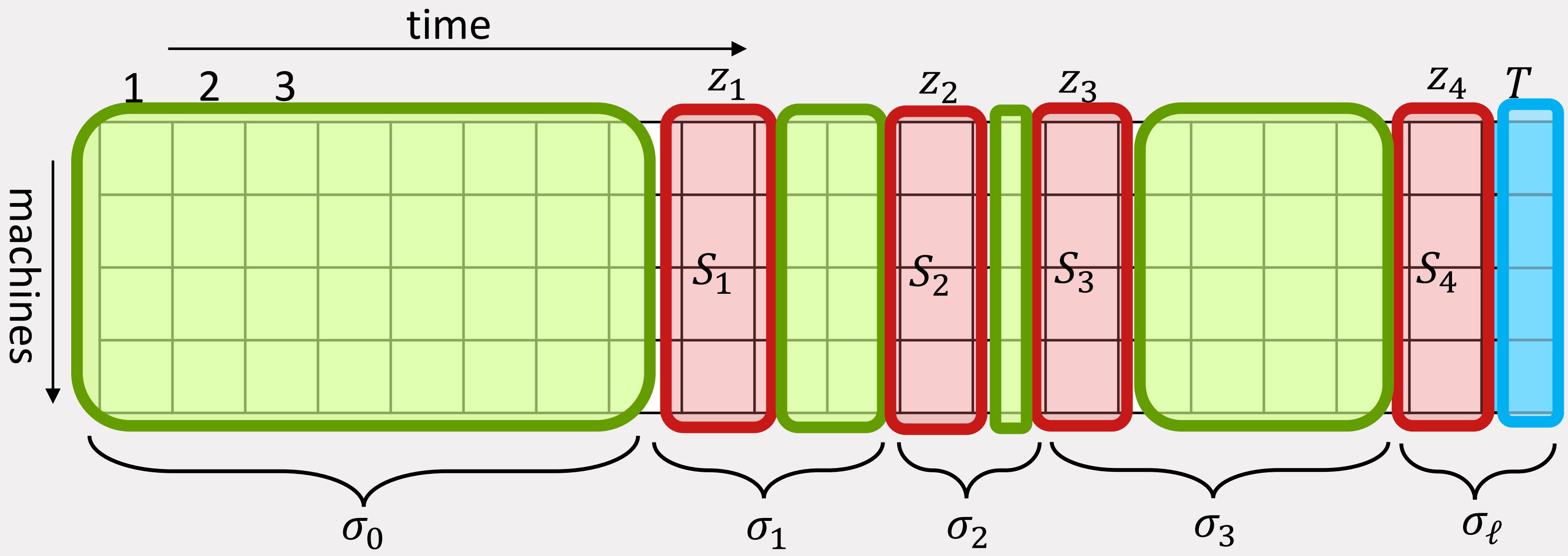


$z \in [1, T]$ is a ***sink moment*** if there are both sinks and non-sinks at time z.

No sinks in

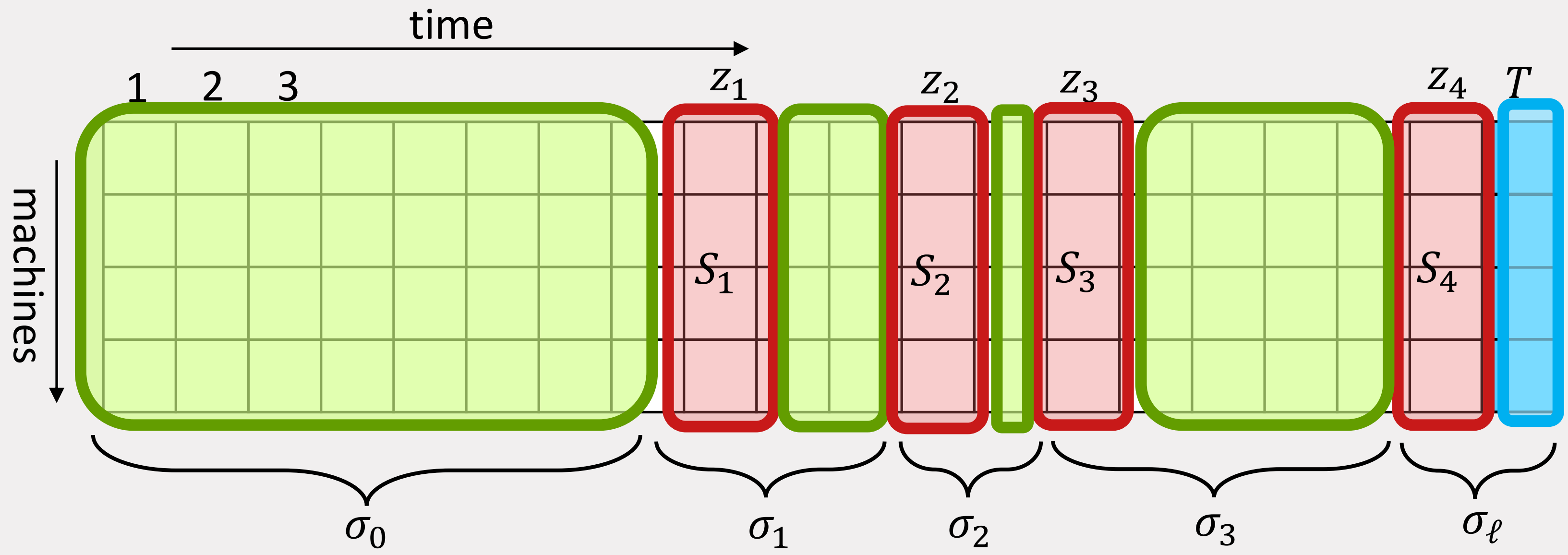
Sink-Adjusted Schedule

Assumption: $n = m \cdot T$



Sink-Adjusted Schedule

Assumption: $n = m \cdot T$



$$\sigma_i = \sigma(S_i, S_{i+1})$$

Summary of Algorithm

Input: X a set of jobs, m

Output: minimal makespan of X

1. For each $S_1, S_2 \in \binom{X}{\leq m}$ do:
   a) Compute recursively makespan of $X \cap \sigma(S_1, S_2)$
2. Find a minimal combination of these subschedules
3. Return found schedule

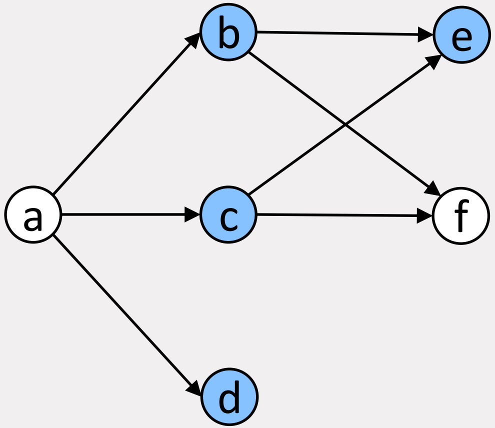
Simple analysis: $2^n\ poly(n)$ time…?

Only n^{2m} subproblems

Use of DP in $\approx n^{2m}$ time

Describing subschedules

Let σ be a *subschedule*.



time

	1	2	3	4
1	a	b	d	f
2		c	e	

Describing subschedules

Let σ be a *subschedule*.

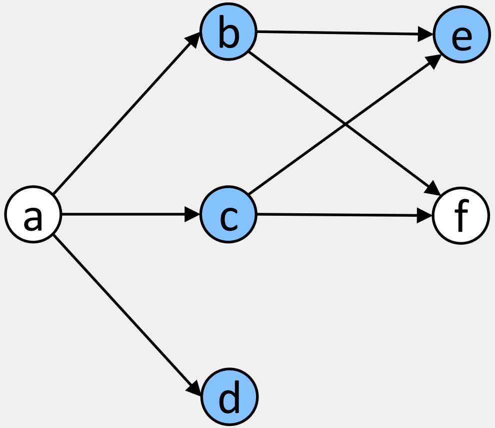
Jobs $V(\sigma)$ can be described as

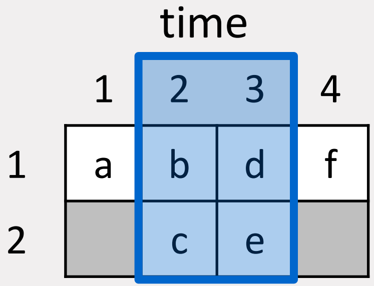
$$V(\sigma) = \mathrm{succ}[A] \cap \mathrm{pred}[B]$$

where

A = minimal elements = sources

B = maximal elements = sinks





Describing subschedules

Let σ be a *subschedule*.

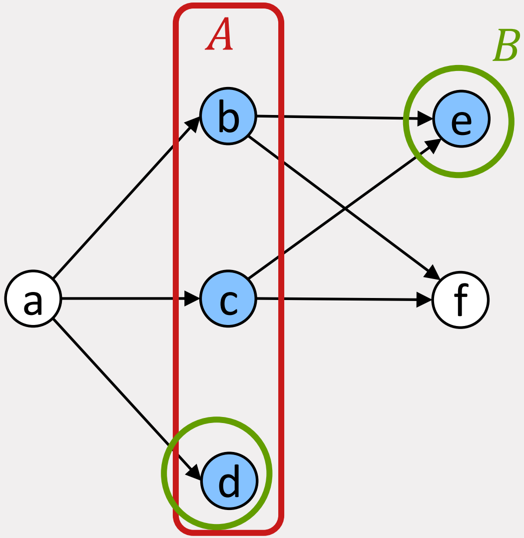
Jobs $V(\sigma)$ can be described as

$$V(\sigma) = \mathrm{succ}[A] \cap \mathrm{pred}[B]$$

where

A = minimal elements = sources

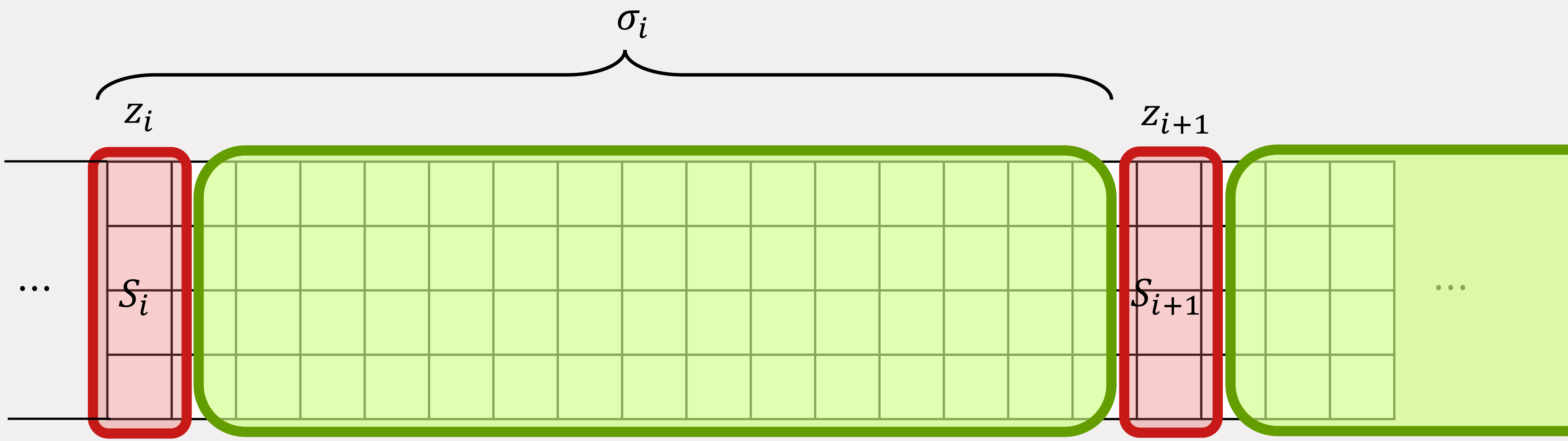
B = maximal elements = sinks



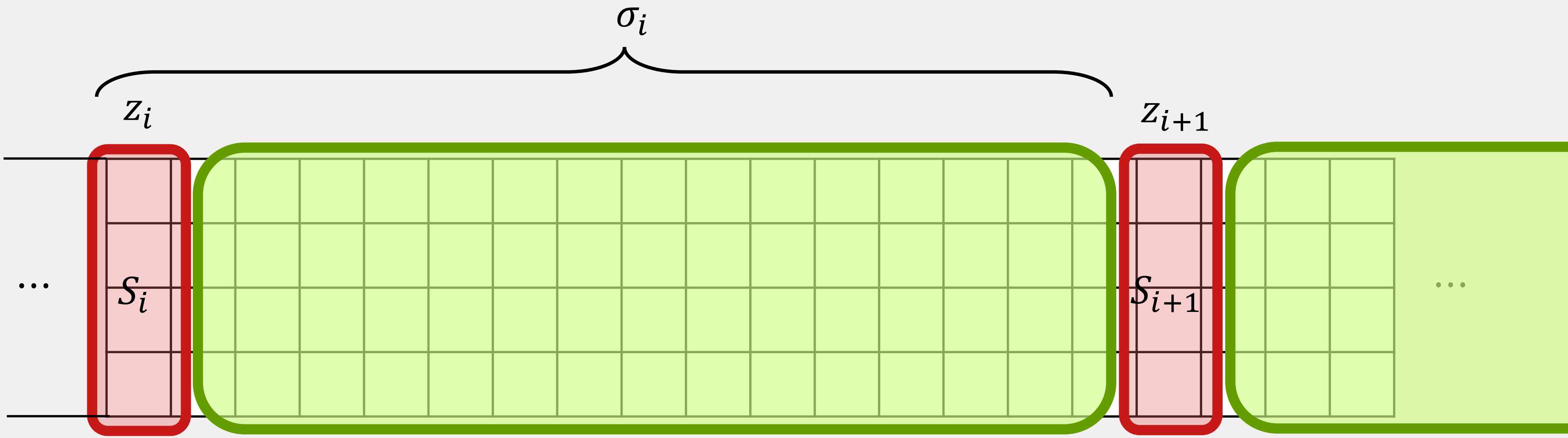
time

	1	2	3	4
1	a	b	d	f
2		c	e	

Sink-Adjusted Schedule



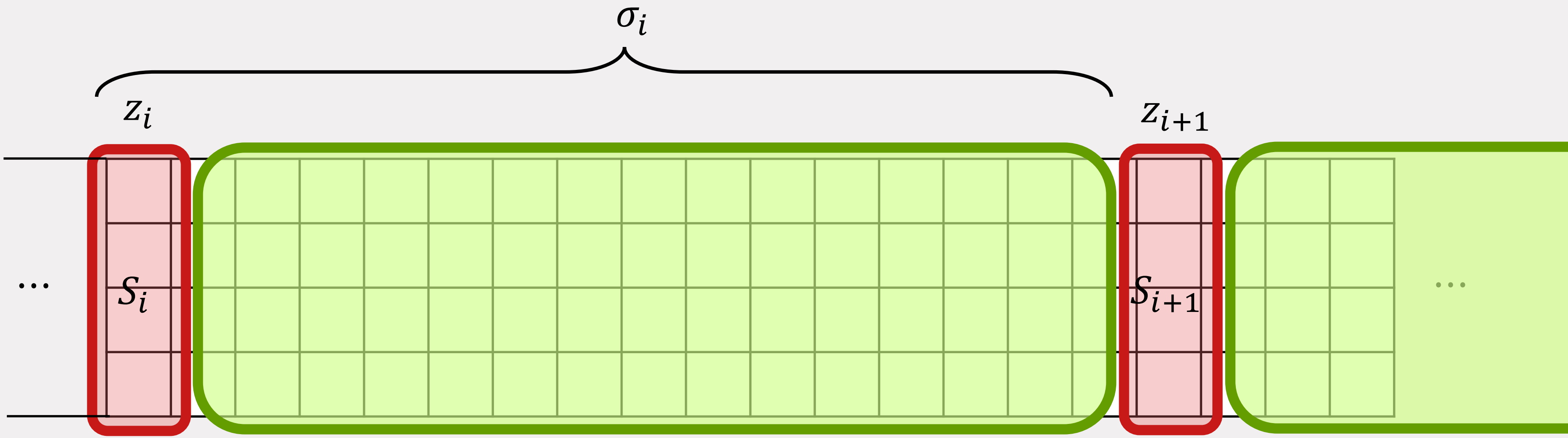
Sink-Adjusted Schedule



Lemma:

$$V(\sigma_i) \approx \mathrm{succ}[S_i] \setminus \left(\mathrm{succ}[S_{i+1}] \cup \mathrm{sinks}(G)\right)$$

Sink-Adjusted Schedule



Lemma:

$$V(\sigma_i) \approx \text{succ}[S_i] \setminus \left(\text{succ}[S_{i+1}] \cup \text{sinks}(G)\right)$$
$$= \text{succ}[S_i] \cap \text{pred}[B]$$

Algorithm

Store subprohlem results in lookup table:

$$T(A, B) \coloneqq \text{min makespan of succ}[A] \cap \text{pred}[B]$$

Use of lookup table, how many *different* subproblems?

Algorithm

Store subproblem results in lookup table:

$$T(A, B) \coloneqq \text{min makespan of succ}[A] \cap \text{pred}[B]$$

Use of lookup table, how many *different* subproblems?

Lemma:
$V(\sigma_i) = \text{succ}[S_i] \cap \text{pred}[B]$

Algorithm

Store subproblem results in lookup table:

$$T(A, B) := \text{min makespan of succ}[A] \cap \text{pred}[B]$$

Use of lookup table, how many *different* subproblems?

Lemma:
$V(\sigma_i) = \text{succ}[S_i] \cap \text{pred}[B]$

Remark: $|S_i| \leq m$!!

TU/e

Algorithm

Store subproblem results in lookup table:

$$T(A, B) := \text{min makespan of succ}[A] \cap \text{pred}[B]$$

Use of lookup table, how many *different* subproblems?

$\Rightarrow$ at most $\binom{n}{m} \cdot \#\text{different } B$

Lemma:
$V(\sigma_i) = \text{succ}[S_i] \cap \text{pred}[B]$

Remark: $|S_i| \leq m$!!

Algorithm

Store subproblem results in lookup table:

$$T(A, B) \coloneqq \text{min makespan of } \mathrm{succ}[A] \cap \mathrm{pred}[B]$$

Use of lookup table, how many *different* subproblems?

$\Rightarrow$ at most $\binom{n}{m} \cdot \#\text{different } B$

Lemma:
$V(\sigma_i) = \mathrm{succ}[S_i] \cap \mathrm{pred}[B]$

Remark: $|S_i| \leq m$!!

Case $\lvert B\rvert \leq \sqrt{n}$	Case $\lvert B\rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's	

TU/e

Algorithm

Store subproblem results in lookup table:

$$T(A, B) \coloneqq \text{min makespan of succ}[A] \cap \text{pred}[B]$$

Use of lookup table, how many *different* subproblems?

$\Rightarrow$ at most $\binom{n}{m} \cdot \#\text{different } B$

Lemma:
$V(\sigma_i) = \text{succ}[S_i] \cap \text{pred}[B]$

Remark: $|S_i| \leq m$!!

Case $\lvert B\rvert \leq \sqrt{n}$	Case $\lvert B\rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	

TU/e

Algorithm

Store subproblem results in lookup table:

$$T(A, B) := \text{min makespan of } \mathrm{succ}[A] \cap \mathrm{pred}[B]$$

Use of lookup table, how many *different* subproblems?

$\Rightarrow$ at most $\binom{n}{m} \cdot \#\text{different } B$

Lemma:
$V(\sigma_i) = \mathrm{succ}[S_i] \cap \mathrm{pred}[B]$

Remark: $|S_i| \leq m$!!

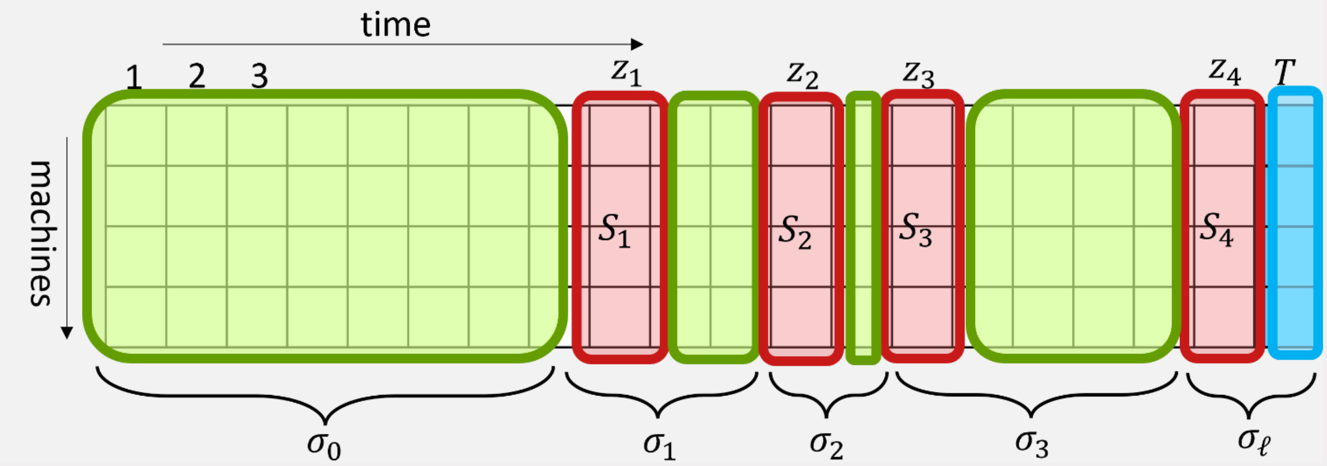
Case $\lvert B \rvert \leq \sqrt{n}$	Case $\lvert B \rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress!

TU/e

Algorithm

No sinks in [green box]

$V(\sigma) = \text{succ}[A] \cap \text{pred}[B]$

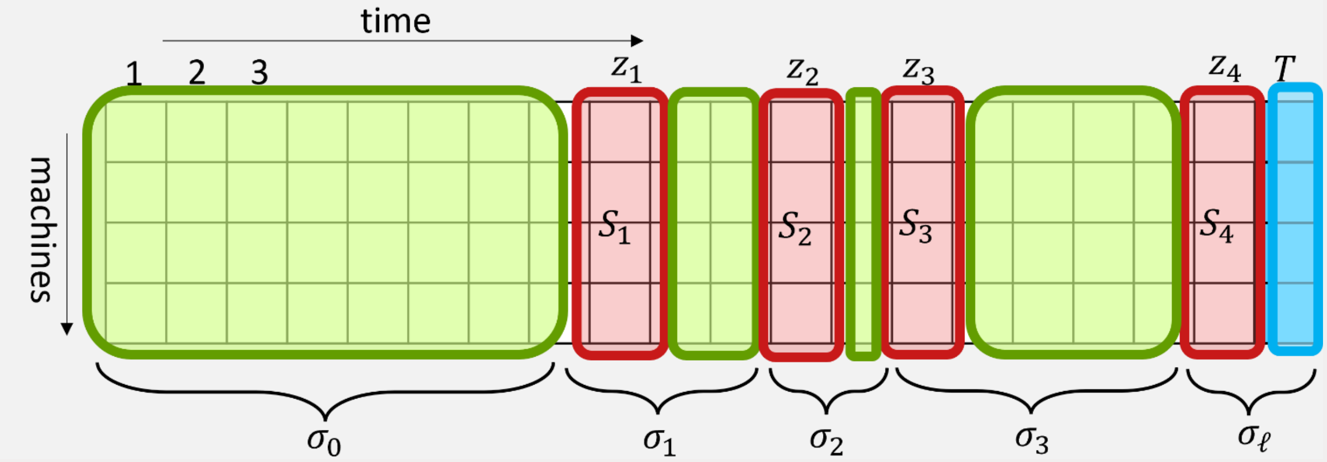


Case $\lvert B\rvert \leq \sqrt{n}$	**Case** $\lvert B\rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n}\cdot\log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress!

Algorithm

No sinks in [green box]

$V(\sigma) = \text{succ}[A] \cap \text{pred}[B]$



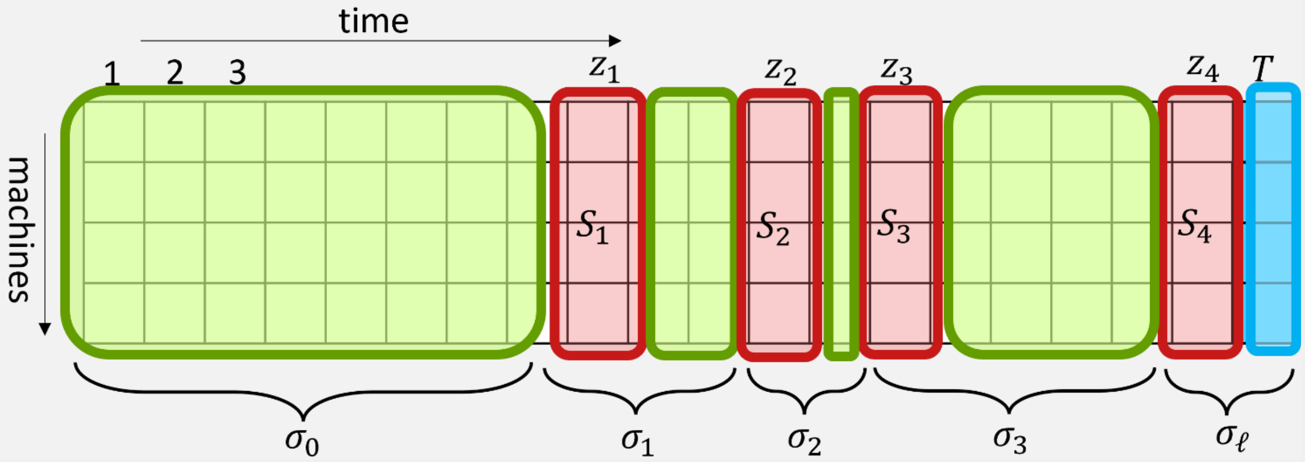
Each σ_i contains at most m sinks (and σ_ℓ is extremely easy)

Case $\lvert B \rvert \leq \sqrt{n}$	**Case** $\lvert B \rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress!

Algorithm

No sinks in [green box]

$V(\sigma) = \text{succ}[A] \cap \text{pred}[B]$ → $\text{sinks}(\sigma)$



Each σ_i contains at most m sinks (and σ_ℓ is extremely easy)

$|\sigma_i| \leq n - \sqrt{n} + m$

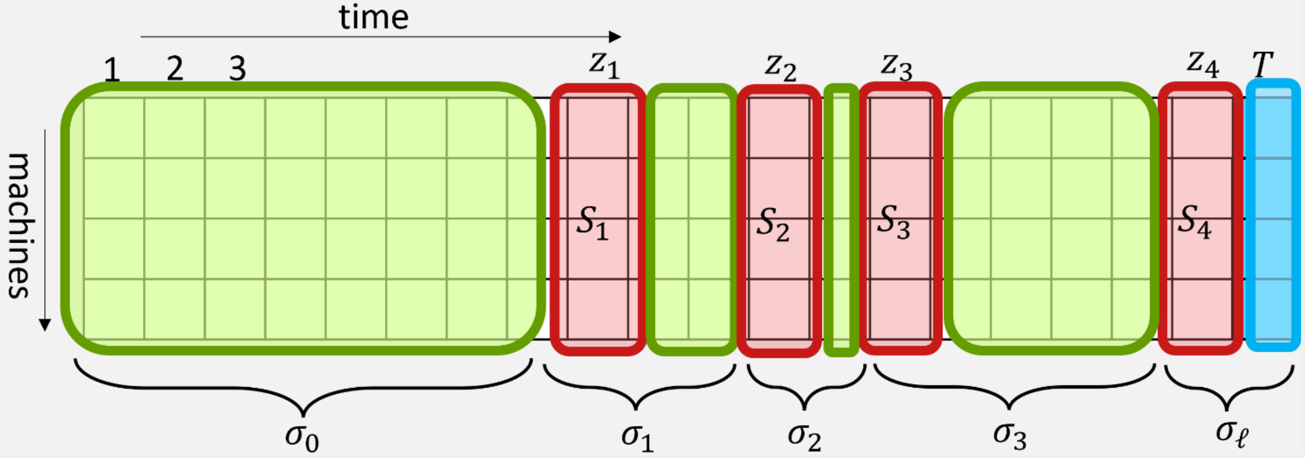
Case $\|B\| \leq \sqrt{n}$	**Case** $\|B\| > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress!

TU/e

Algorithm

No sinks in [green box]

$V(\sigma) = \text{succ}[A] \cap \text{pred}[B]$ → $\text{sinks}(\sigma)$



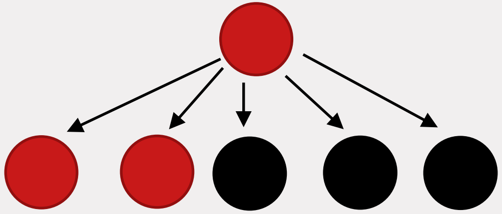
Each σ_i contains at most m sinks (and σ_ℓ is extremely easy)

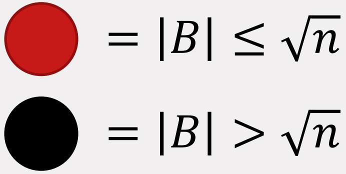
$|\sigma_i| \leq n - \sqrt{n} + m$

Case $|B| \leq \sqrt{n}$	**Case** $|B| > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress! ✓

Algorithm

$\leq n^{2m} = n^{O(1)}$
children





Case $|B| \leq \sqrt{n}$	**Case** $|B| > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress! ✓

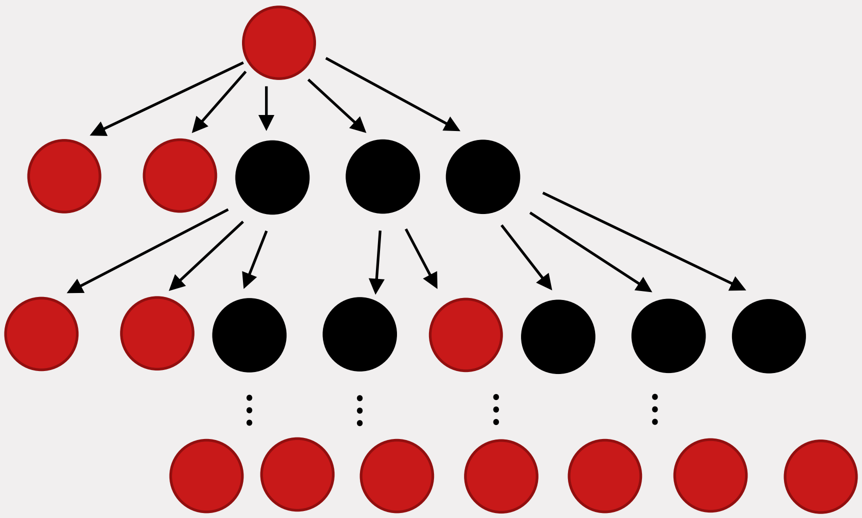
TU/e

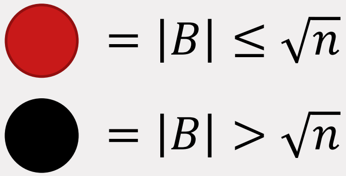
Algorithm

$\leq n^{2m} = n^{O(1)}$
children

Height $\leq \sqrt{n}$

$\leq \left(n^{O(1)}\right)^{\sqrt{n}}$
$= 2^{O(\sqrt{n} \cdot \log n)}$
nodes in tree!





Case $\lvert B\rvert \leq \sqrt{n}$	Case $\lvert B\rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress! ✓

Algorithm

Store subproblem results in lookup table:

$$T(A, B) \coloneqq \text{min makespan of succ}[A] \cap \text{pred}[B]$$

Use of lookup table, how many *different* subproblems?

$\Rightarrow$ at most $2^{O(\sqrt{n} \cdot \log n)}$ trees with $2^{O(\sqrt{n} \cdot \log n)}$ nodes

$\Rightarrow$ $2^{O(\sqrt{n} \cdot \log n)}$ *different* subproblem!

Case $\lvert B \rvert \leq \sqrt{n}$	**Case** $\lvert B \rvert > \sqrt{n}$
$\Rightarrow$ only $\binom{n}{\sqrt{n}} = 2^{O(\sqrt{n} \cdot \log n)}$ different B's ✓	In next step: make $\sqrt{n}$ jobs progress! ✓

TU/e

Summary of Algorithm

Input: *A, B, m*

Output: minimal makespan of $\mathrm{succ}[A] \cap \mathrm{pred}[B] = \mathrm{Jobs}$

1. For each $S_1, S_2 \in \binom{\mathrm{Jobs}}{\leq m}$ do:
   a) Compute recursively makespan of $\mathrm{Jobs}\ \cap (\mathrm{succ}[S_1] \setminus (\mathrm{succ}[S_2] \cup B)$
2. Find a minimal combination of these subschedules
3. Return found schedule

TU/e

Conclusion

Conclusion

Main result:

$Pm|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Conclusion

Main result:

$Pm|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key idea's:

- New decomposition of schedules
- Use of look-up table
- Use of Dynamic Programming in combining results

Conclusion

Main result:

$Pm|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key idea's:

- New decomposition of schedules
- Use of look-up table
- Use of Dynamic Programming in combining results

Future Research:

$P3|prec, p_j = 1|C_{\max}$ in quasi-polynomial time?

Conclusion

Main result:

$Pm|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key idea's:

- New decomposition of schedules
- Use of look-up table
- Use of Dynamic Programming in combining results

Future Research:

$P3|prec, p_j = 1|C_{\max}$ in quasi-polynomial time?

Thanks for your attention!