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

Jesper Nederlof

UU

Céline Swennenhuis

TU/e

Karol Węgrzycki

MPI

Department of Mathematics and Computer Science, Eindhoven University of Technology

$P3|prec, p_j = 1|C_{\max}$

$$P3 \left| prec, p_j = 1 \right| C_{\max}$$

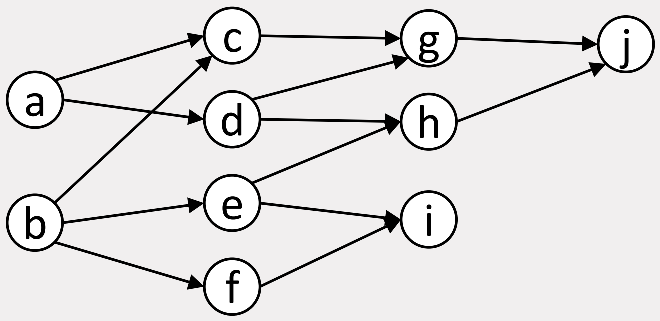
3 identical
parallel machines

$P3|prec, p_j = 1|C_{\max}$

Given:

- n jobs of length 1

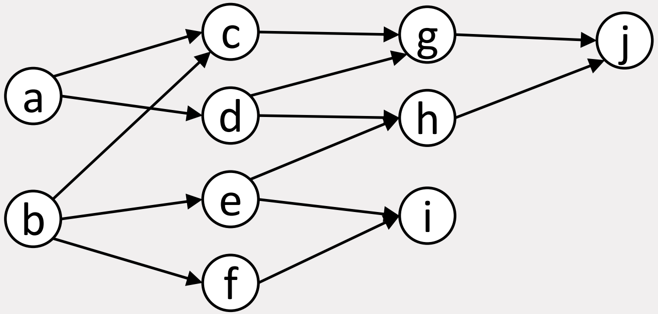
$P3|prec, p_j = 1|C_{\max}$



Given:

- n jobs of length 1
- A precedence graph G

$P3|prec, p_j = 1|C_{\max}$

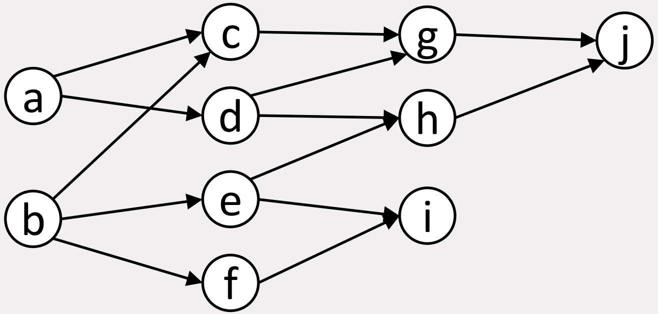


Given:

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

Q: Is there a schedule of makespan T?

$P3|prec, p_j = 1|C_{\max}$



Given:

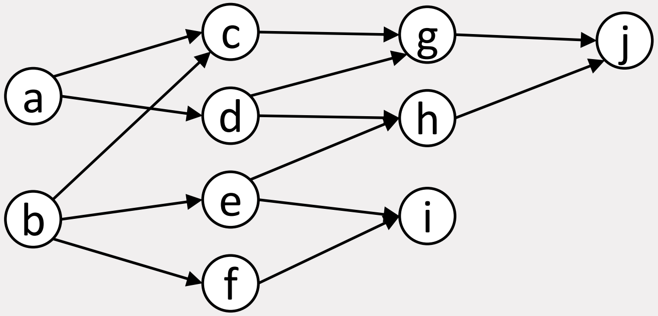
- n jobs of length 1
- A precedence graph G
- $T \in \mathbb{N}$

Q: Is there a schedule of makespan T?

time →

	1	2	3	4
1	a	c	f	i
2	b	d	g	j
3		e	h	

$P3|prec, p_j = 1|C_{\max}$



Given:

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

G defines the problem

Q: Is there a schedule of makespan T?

time

	1	2	3	4
1	a	c	f	i
2	b	d	g	j
3		e	h	

Observation:
Jobs of length one ⇒ `timeslots’

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

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

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:

$P3|prec, p_j = 1|C_{\max}$ 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.

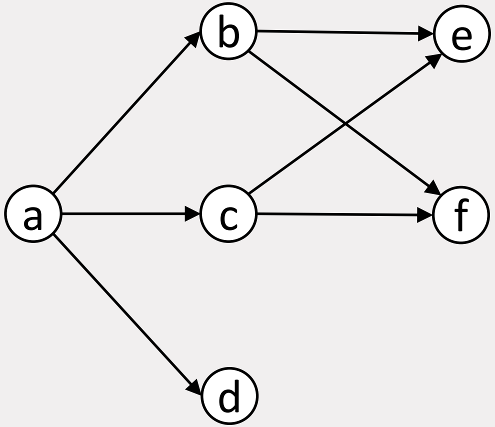
Corollary:

$P3|prec, p_j = 1|C_{\max}$ can be solved in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Two ways to explain, but main insights:

1. Use of look-up table
2. Keeping track of number of isolated vertices
3. Finding win-win strategy

Definitions

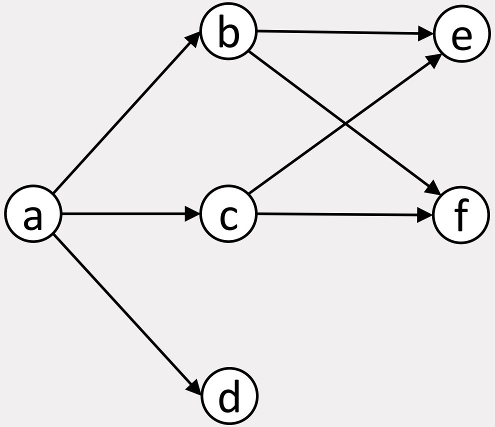


Precedence Constraints Graph G

Definitions

$G \Rightarrow$ partial order:

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



Precedence Constraints Graph G

Definitions

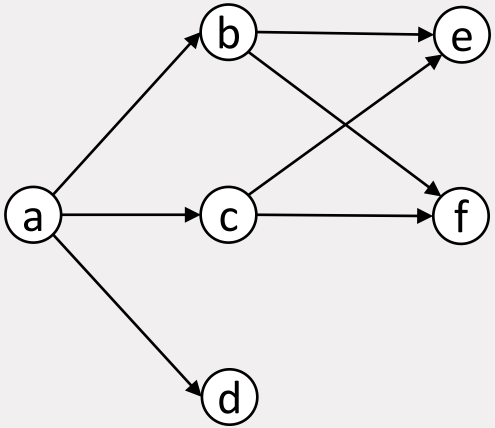
$G \Rightarrow$ partial order:

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

<u>Definitions:</u> Let A be a set of jobs.

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

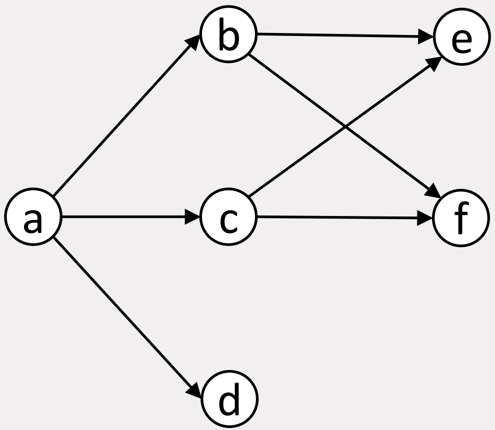
$\text{succ}[A] \;=$



Precedence Constraints Graph G

Definitions

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



Precedence Constraints Graph G

Definitions: Let A be a set of jobs.

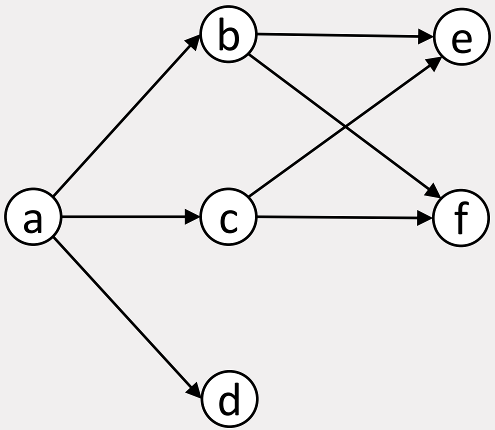
$\text{pred}[A] \;\; = \{x \mid \exists\, a \in A \; s.t. \;\; x \preccurlyeq a\}$

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

Definitions

$G \Rightarrow$ partial order:

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



Precedence Constraints Graph G

Definitions: Let A be a set of jobs.

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

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

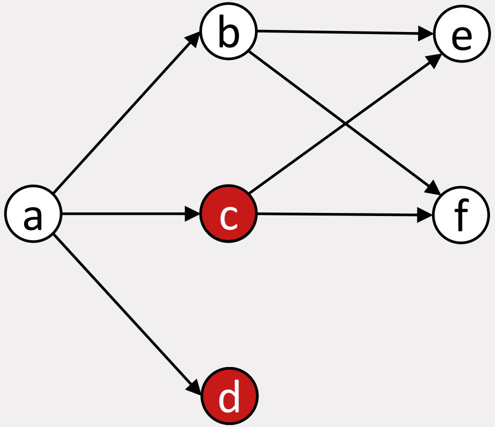
$\text{succ}[A] \ = \{x \mid \exists\, a \in A\ s.t.\ x \succcurlyeq a\}$

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

Definitions

$G \Rightarrow$ partial order:

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



Precedence Constraints Graph G

Definitions: Let A be a set of jobs.

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

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

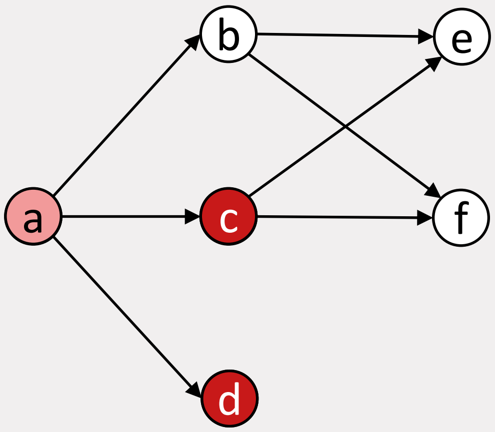
$\text{succ}[A] \;= \{x \mid \exists\, a \in A\ s.t.\ x \succcurlyeq a\}$

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

Ex. $\{c, d\}$, then:

Definitions

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



Precedence Constraints Graph G

Definitions: Let A be a set of jobs.

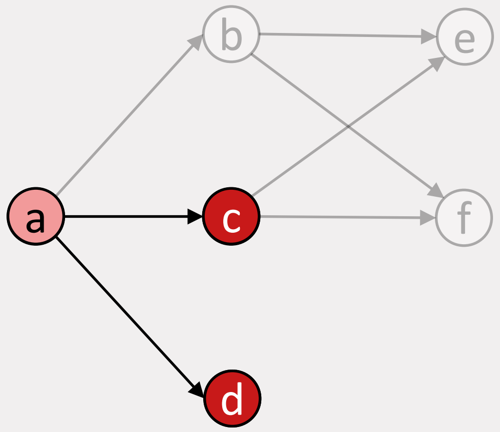
$\text{pred}[A] \;\;= \{x \mid \exists\, a \in A\ s.t.\ x \preccurlyeq a\}$
$\text{sinks}(A) = \max\{A\}$
$\text{succ}[A] \;\;= \{x \mid \exists\, a \in A\ s.t.\ x \succcurlyeq a\}$
$\text{sources}(A) = \min\{A\}$

Ex. $\{c, d\}$, then:
- $\text{pred}[\{c, d\}] = \{a, c, d\}$

Definitions

$G \Rightarrow$ partial order:

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



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

Ex. $\{c, d\}$, then:

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

Definitions

$G \Rightarrow$ partial order:

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

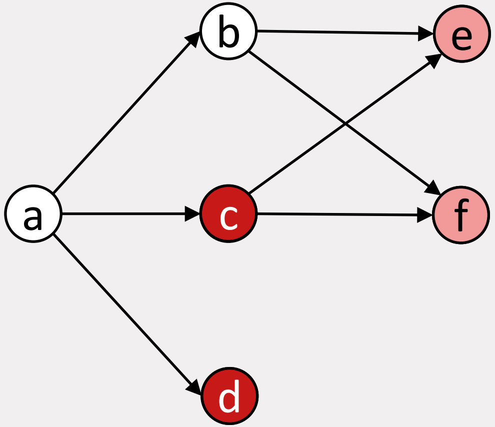
Definitions: Let A be a set of jobs.

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

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

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

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

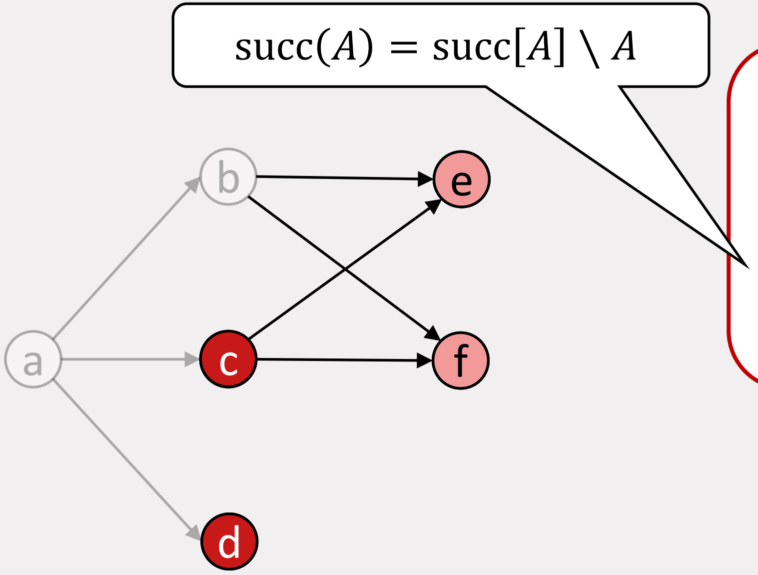


Precedence Constraints Graph G

Ex. $\{c,d\}$, then:

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

Definitions



Precedence Constraints Graph G

$G \Rightarrow$ partial order:

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

Definitions: Let A be a set of jobs.

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

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

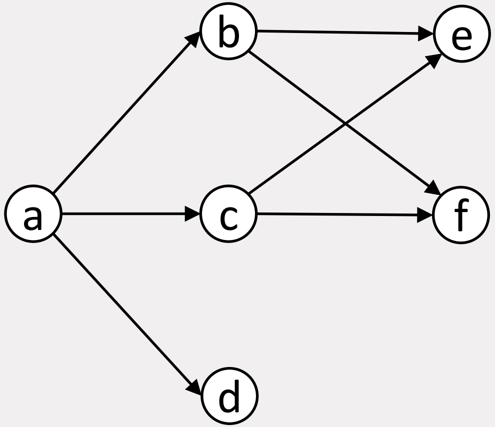
succ$[A]$ $= \{x \mid \exists\, a \in A \; s.t. \; x \succcurlyeq a\}$

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

Ex. $\{c,d\}$, then:

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

Definitions

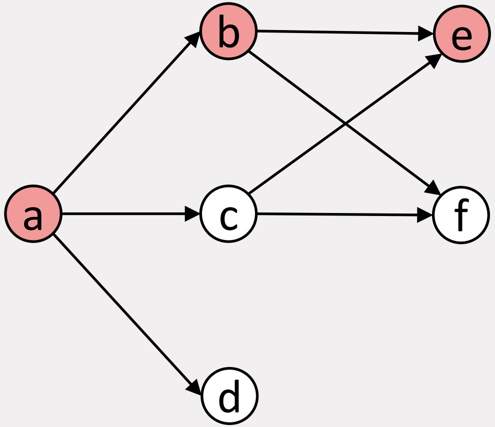


Precedence Constraints Graph G

Def: A *chain* is a set A whose elements are pairwise **comparable**.

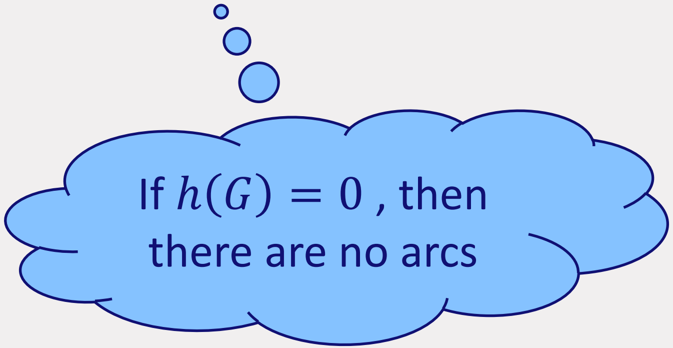
Definitions

Def: A *chain* is a set A whose elements are pairwise **comparable**.
Def: The *height* $h(G)$ is the size of the longest chain (in #arcs).

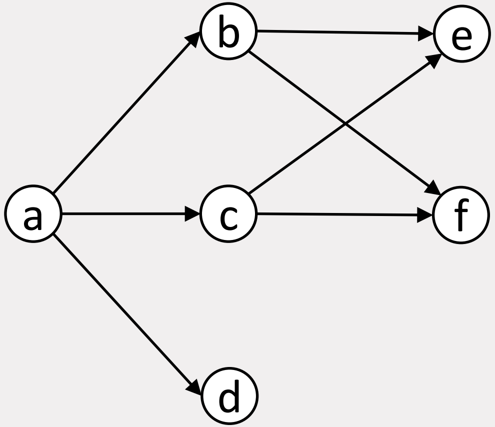


Precedence Constraints Graph G

In the example $h(G) = 2$



Definitions



Precedence Constraints Graph G

Def: A *chain* is a set A whose elements are pairwise **comparable**.
Def: The *height* $h(G)$ is the size of the longest chain (in #arcs).

In the example $h(G) = 2$

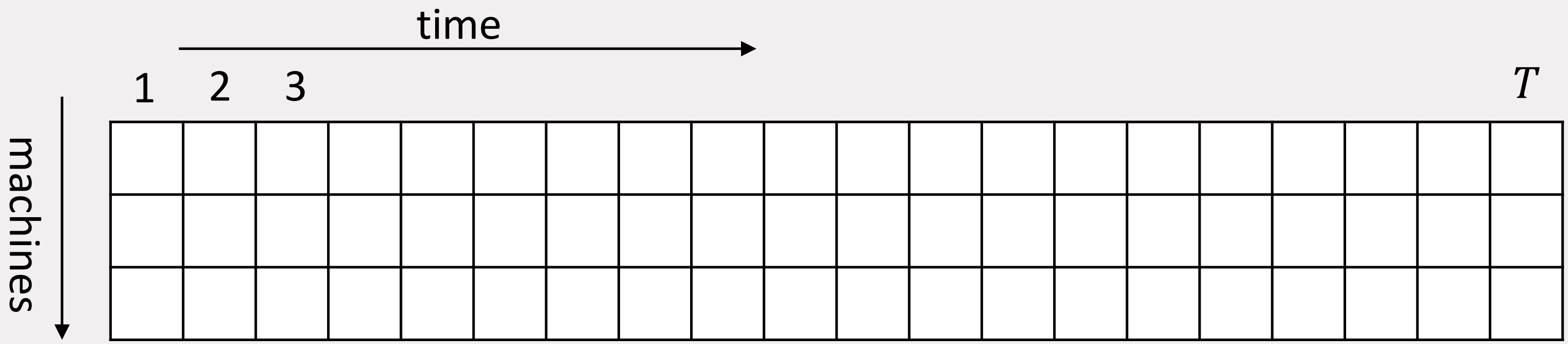
Def: An *antichain* is a set A whose elements are pairwise **incomparable**.

Examples of antichains in G

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

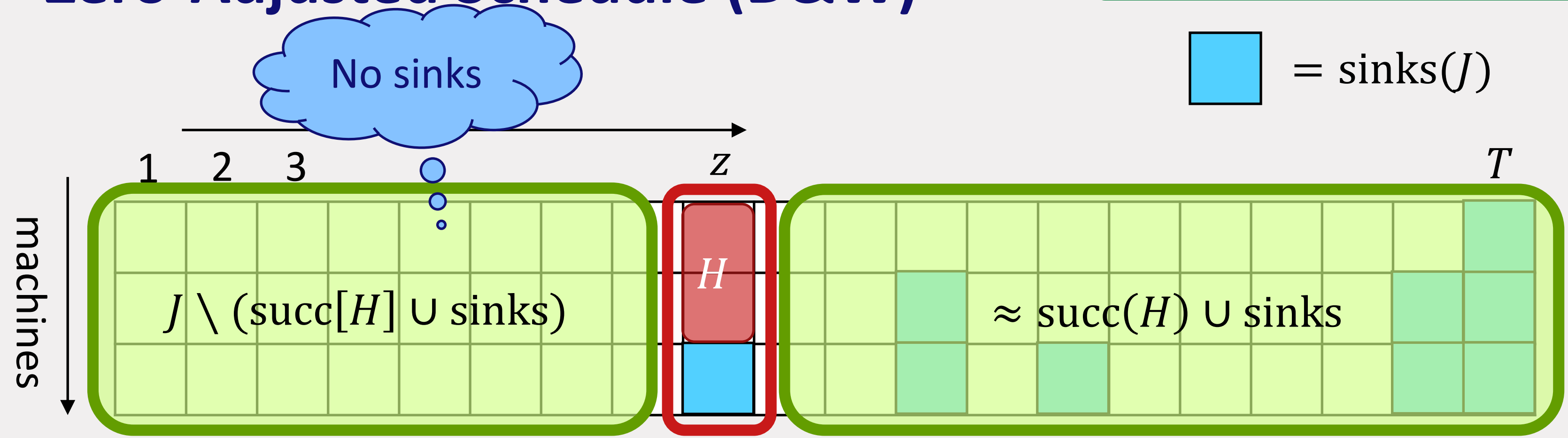
Jobs in one timeslot always form an antichain

Zero-Adjusted Schedule (D&W)



Zero-Adjusted Schedule (D&W)

Assumption: $n = 3 \cdot T$



Let $z \in [1, T]$ be the first moment with a sink.
D&W: W.m.a. Each job x **after *z*** is a sink or a successor of a job at time z.

Dolev and Warmuth

Recursive

Schedule(J):

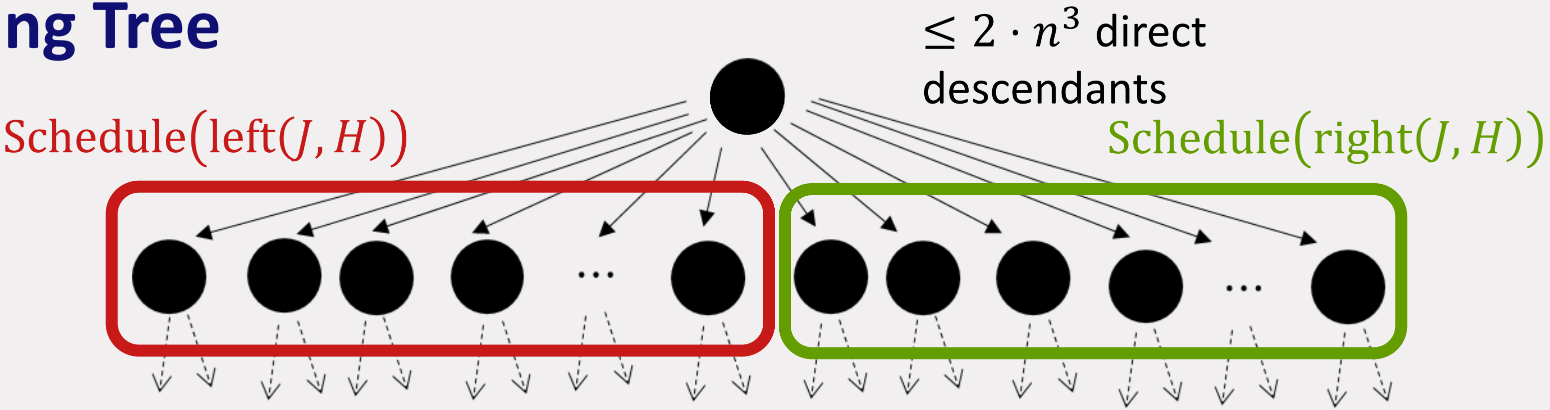
1. **if** $h(G[J]) \leq 0$ (i.e. $\text{sinks}(J) = J$) **return** $\left\lceil \frac{|J|}{3} \right\rceil$
2. **else return** $\min_{H \in \text{Sep}(J)} \{\text{Schedule}(\text{left}(J,H)) + \text{Schedule}(\text{right}(J,H)) + 1\}$

$\text{Sep}(J) \coloneqq \{\, H \subseteq J$ s.t.
(1) $|H| \leq 3$,
(2) H is antichain,
(3) $|H \setminus \text{sinks}(J)| < 3\}$

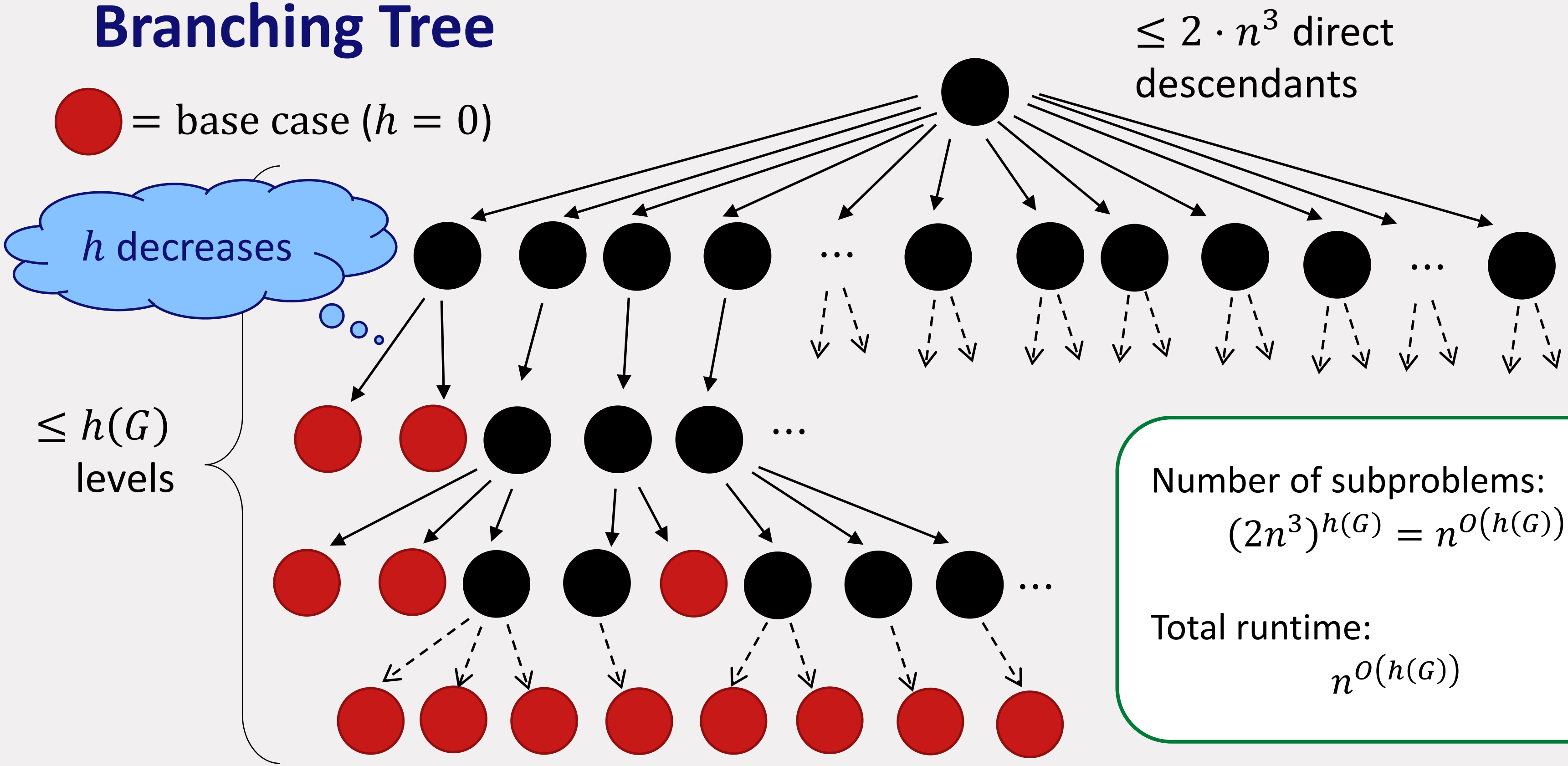
$\text{left}(J,H) \coloneqq J \setminus (\text{succ}[\text{H}] \cup \text{sinks}(J))$
$\text{right}(J,H) \coloneqq J \cap ((\text{succ}(H) \cup \text{sinks}(J)) \setminus H$

Each subproblem: height decreases by ≥ 1!

Branching Tree



Branching Tree



Number of subproblems:

$$(2n^3)^{h(G)} = n^{O(h(G))}$$

Total runtime:

$$n^{O(h(G))}$$

D&W

Schedule(J):

1. **if** $h(G[J]) \leq 0$ **return** $\left\lceil \frac{|J|}{3} \right\rceil$
2. **for each** $H \in \text{Sep}(J)$ **do:**
3. $\quad\quad \text{OPT}[\text{left}(J,H)] \coloneqq \text{Schedule}\big(\text{left}(J,H)\big)$
4. $\quad\quad \text{OPT}[\text{right}(J,H)] \coloneqq \text{Schedule}(\text{right}(J,H))$
5. $\text{OPT}[J] := \min_{H \in \text{Sep}(J)} \{\text{OPT}[\text{left}(J,H)] + \text{OPT}[\text{right}(J,H)] + 1\}$
6. Return $\text{OPT}[J]$

$\text{Sep}(J) \coloneqq \{\, H \subseteq J$ s.t.
(1) $|H| \leq 3$,
(2) H is antichain,
(3) $|H \setminus \text{sinks}(J)| < 3\}$

$\text{left}(J,H) \coloneqq J \setminus \big(\text{succ}[\text{H}] \cup \text{sinks}(J)\big)$
$\text{right}(J,H) \coloneqq J \cap \big((\text{succ}(H) \cup \text{sinks}(J)\big) \setminus H$

D&W + LookUp Table

Schedule(J):

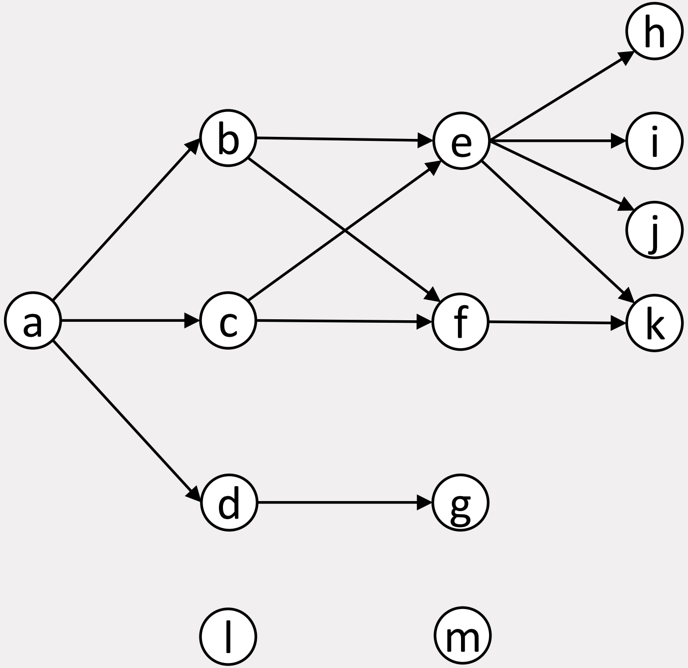
1. **return** LUT[J] if it was already set
2. **if** $h(G[J]) \leq 0$ **return** $\left\lceil \frac{|J|}{3} \right\rceil$
3. **for each** $H \in \text{Sep}(J)$ **do:**
4. $\quad\quad \text{OPT}[\text{left}(J, H)] \coloneqq \text{Schedule}\big(\text{left}(J, H)\big)$
5. $\quad\quad \text{OPT}[\text{right}(J, H)] \coloneqq \text{Schedule}(\text{right}(J, H))$
6. $\text{OPT}[J] := \min_{H \in \text{Sep}(J)} \{\text{OPT}[\text{left}(J, H)] + \text{OPT}[\text{right}(J, H)] + 1\}$
7. $\text{LUT}[J] = \text{OPT}[J]$
8. Return $\text{OPT}[J]$

$\text{Sep}(J) \coloneqq \{ H \subseteq J$ s.t.
(1) $|H| \leq 3$,
(2) H is antichain,
(3) $|H \setminus \text{sinks}(J)| < 3\}$

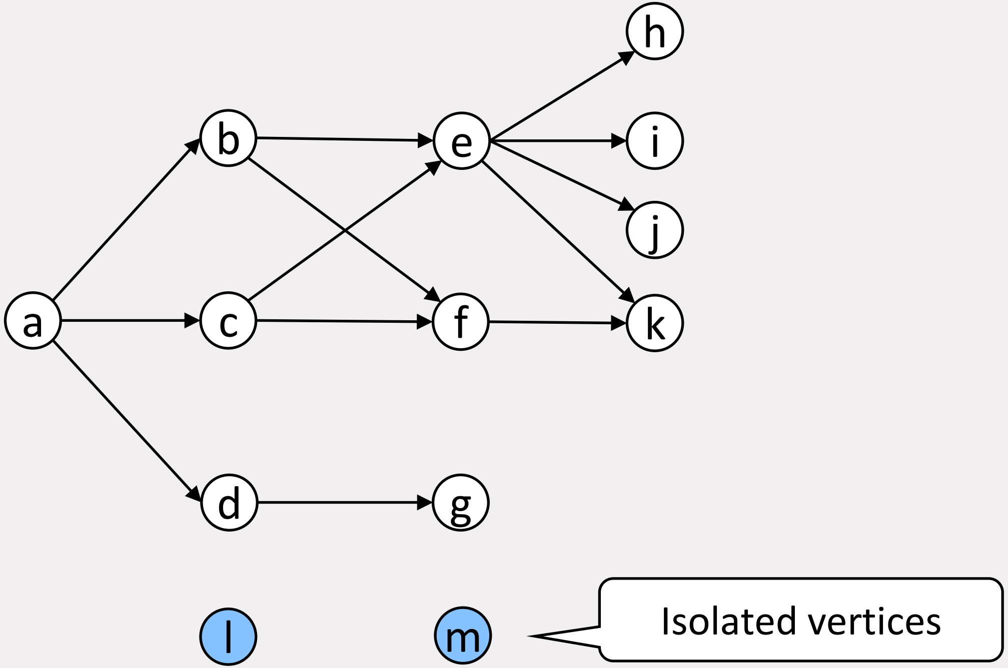
$\text{left}(J, H) \coloneqq J \setminus \big(\text{succ[H]} \cup \text{sinks}(J)\big)$
$\text{right}(J, H) \coloneqq J \cap \big((\text{succ}(H) \cup \text{sinks}(J)\big) \setminus H$

Too many different problems!

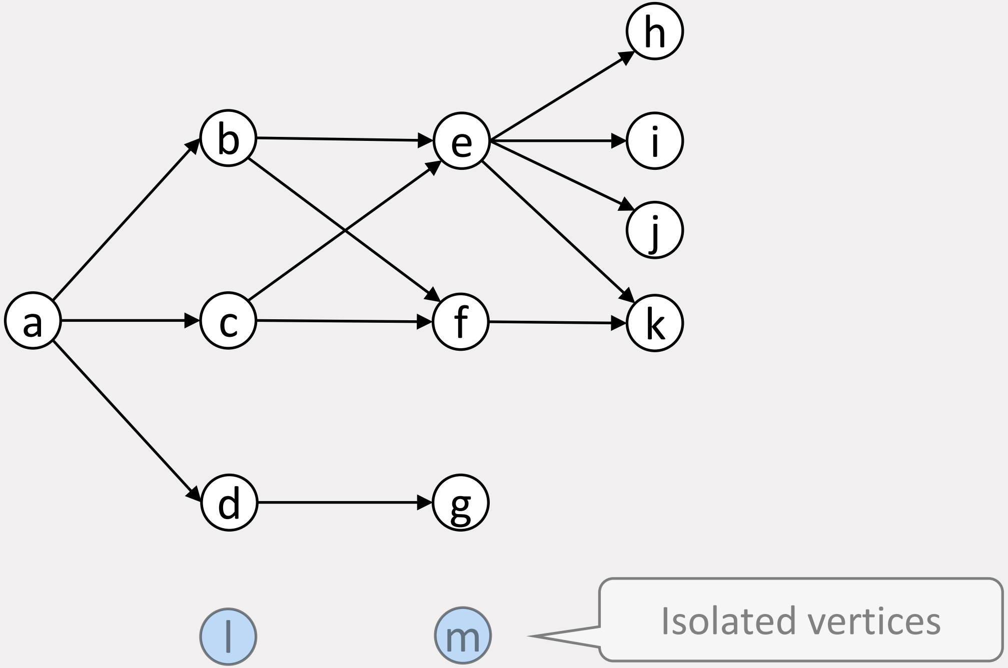
What to store?



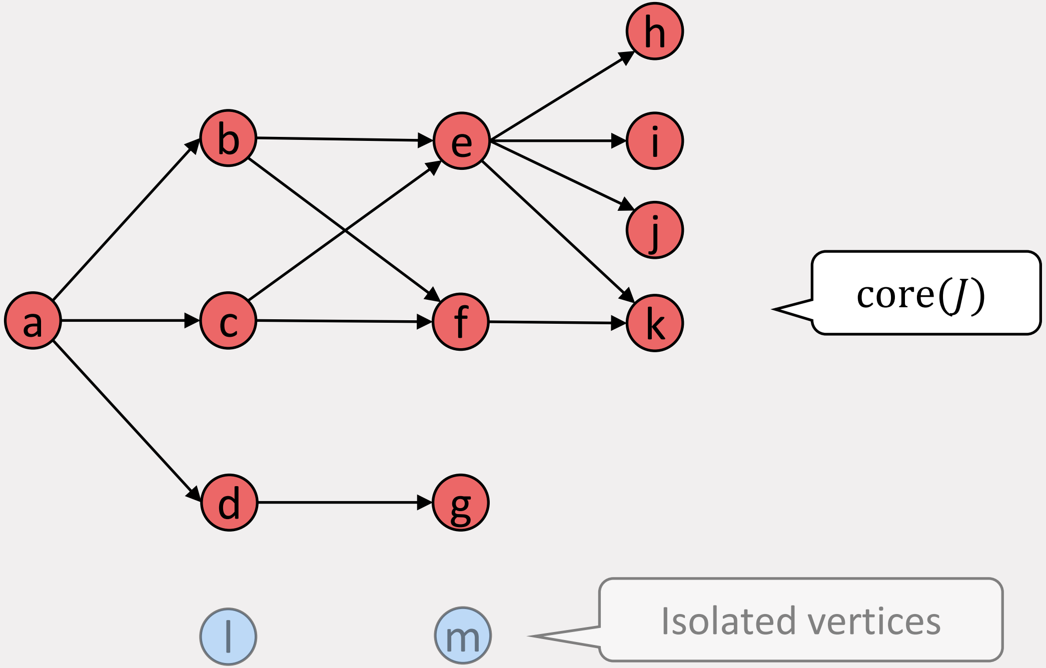
What to store?



What to store?



What to store?



D&W + LookUp Table

$\mathrm{Sep}(J) \coloneqq \{\, H \subseteq J$ s.t.
(1) $|H| \leq 3$,
(2) H is antichain,
(3) $|H \setminus \mathrm{sinks}(J)| < 3\}$

$\mathrm{left}(J,H) \coloneqq J \setminus \big(\mathrm{succ}[\mathrm{H}] \cup \mathrm{sinks}(J)\big)$
$\mathrm{right}(J,H) \coloneqq J \cap \big((\mathrm{succ}(H) \cup \mathrm{sinks}(J)) \setminus H$

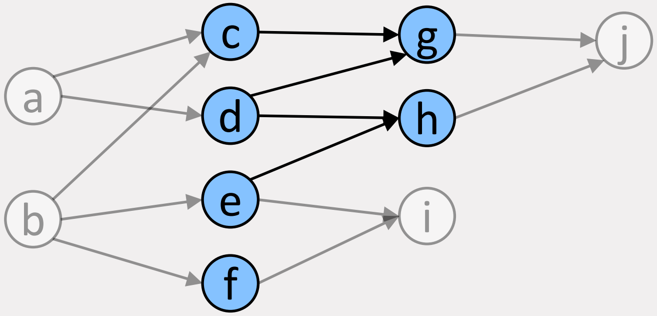
$\mathrm{Schedule}(J)$:

return $\mathrm{LUT}[\mathrm{core}(J), \#\mathrm{iso}(J)]$ if it was already set
if $J = \emptyset$ **return** 0
for each $H \in \mathrm{Sep}(J)$ **do:**
  $\mathrm{OPT}[\mathrm{left}(J,H)] \coloneqq \mathrm{Schedule}\big(\mathrm{left}(J,H)\big)$
  $\mathrm{OPT}[\mathrm{right}(J,H)] \coloneqq \mathrm{Schedule}(\mathrm{right}(J,H))$
$\mathrm{OPT}[J] := \min_{H \in \mathrm{Sep}(J)} \{\mathrm{OPT}[\mathrm{left}(J,H)] + \mathrm{OPT}[\mathrm{right}(J,H)] + 1\}$
$\mathrm{LUT}[\mathrm{core}(J), \#\mathrm{iso}(J)] = \mathrm{OPT}[J]$
Return $\mathrm{OPT}[J]$

How does this help?

Feasible Job sets

Let J be a *feasible set of jobs*.



time

	1	2	3	4
1	a	c	f	i
2	b	d	g	j
3		e	h	

Feasible Job sets

Let J be a *feasible set of jobs*.

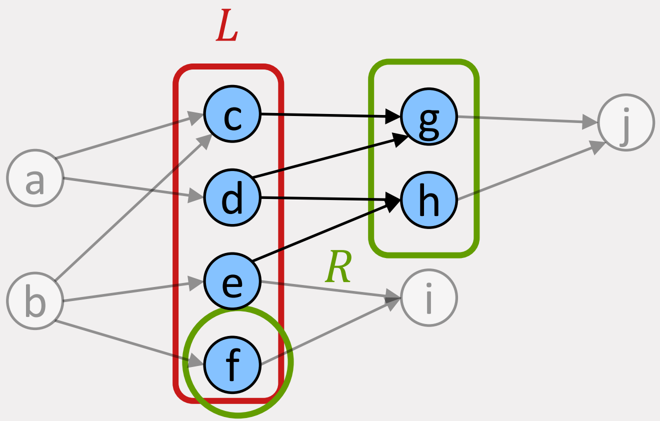
Jobs J can be described as

$$J = \mathrm{succ}[L] \cap \mathrm{pred}[R]$$

where

L = minimal elements = sources of J

R = maximal elements = sinks of J



time

	1	2	3	4
1	a	c	f	i
2	b	d	g	j
3		e	h	

Feasible Job sets

Let J be a *feasible set of jobs*.

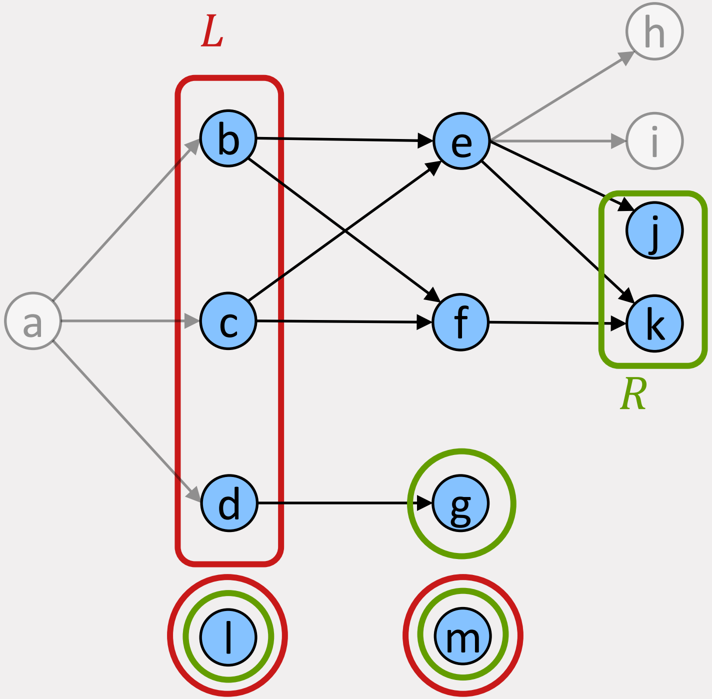
Jobs J can be described as

$$J = \mathrm{succ}[L] \cap \mathrm{pred}[R]$$

where

L = minimal elements = sources of J

R = maximal elements = sinks of J



Feasible Job sets

Let J be a *feasible set of jobs*.

~~Jobs J~~ $\mathbf{core}(J)$ can be described as

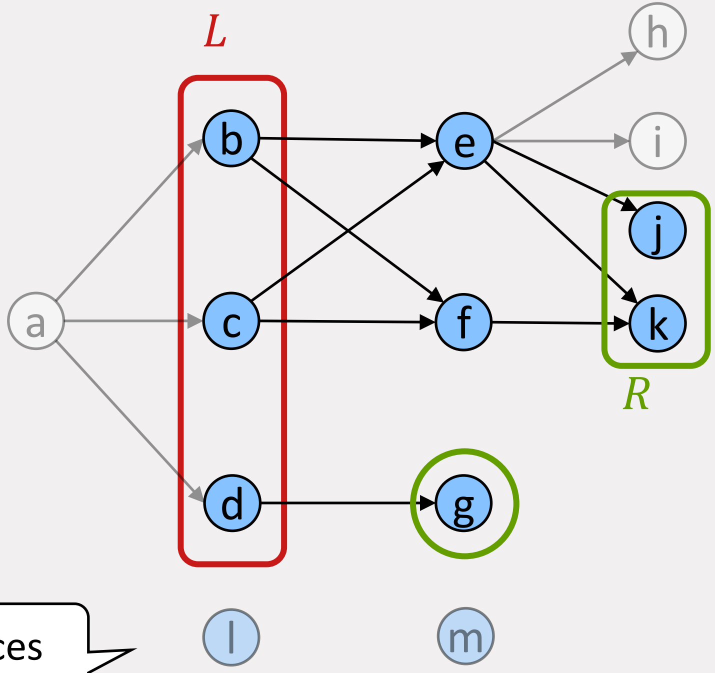
$$\cancel{J}\ \mathbf{core}(J) = \mathrm{succ}[L] \cap \mathrm{pred}[R]$$

where

L = minimal elements = sources of J

R = maximal elements = sinks of J

2 isolated vertices



Feasible Job sets

Let J be a *feasible set of jobs*.

~~Jobs J~~ can be described as
$\mathbf{core}(J)$

$$\mathbf{succ}(L)$$

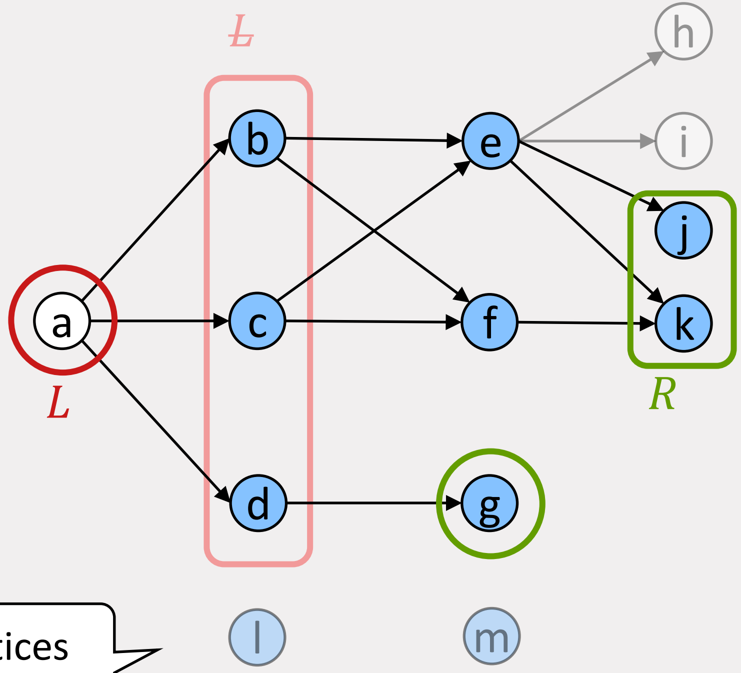
$$\cancel{J} = \cancel{\mathrm{succ}[L]} \cap \mathrm{pred}[R]$$

$$\mathbf{core}(J)$$

where

L = minimal elements = sources of J

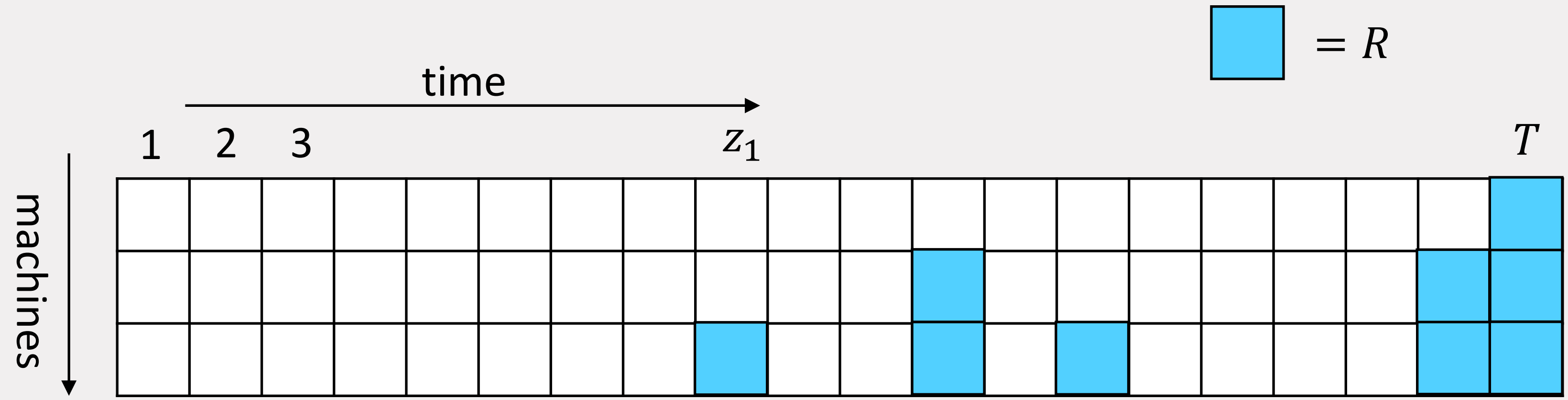
R = maximal elements = sinks of J



2 isolated vertices

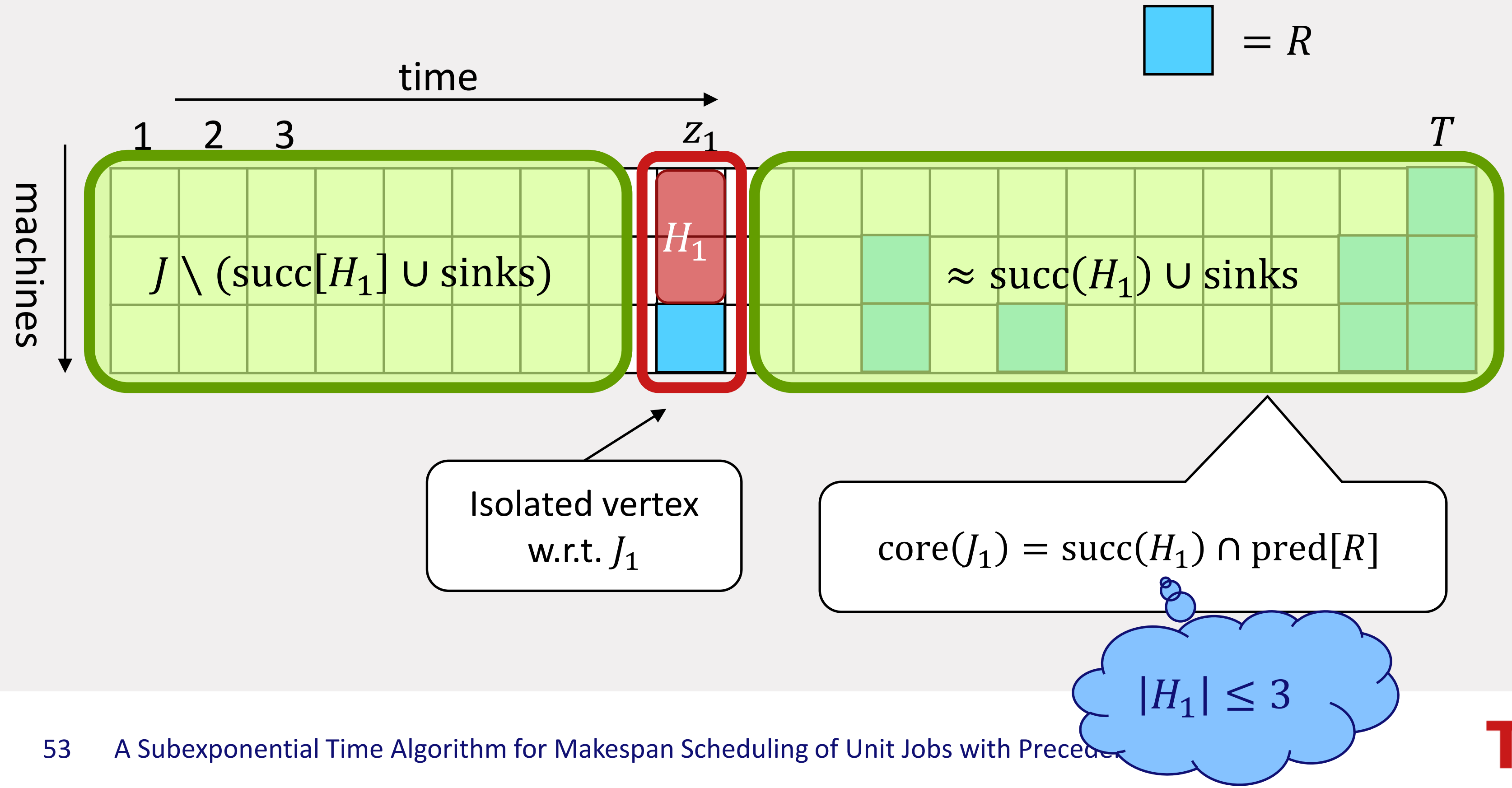
Going to the right

Assumption: $n = 3 \cdot T$



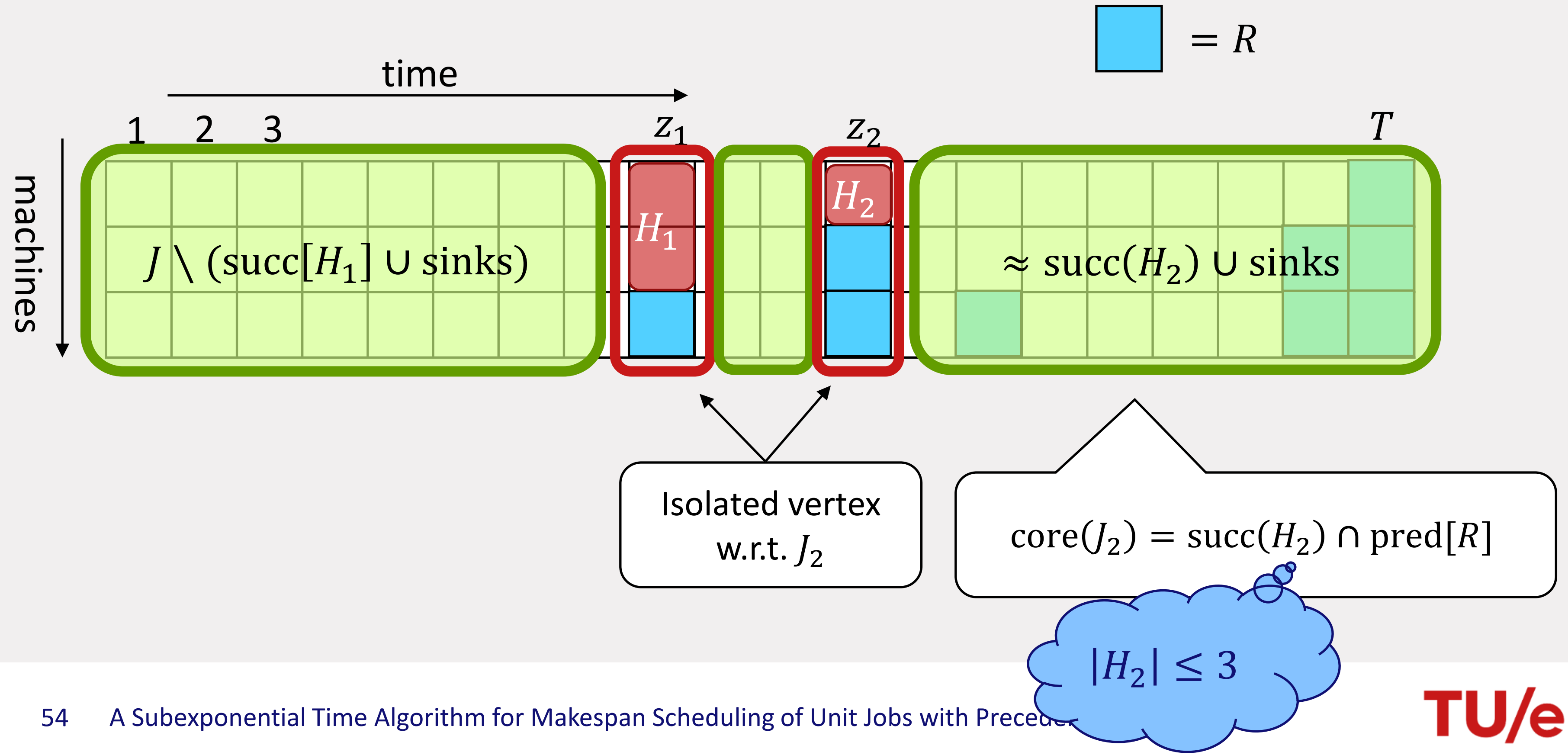
Going to the right

Assumption: $n = 3 \cdot T$



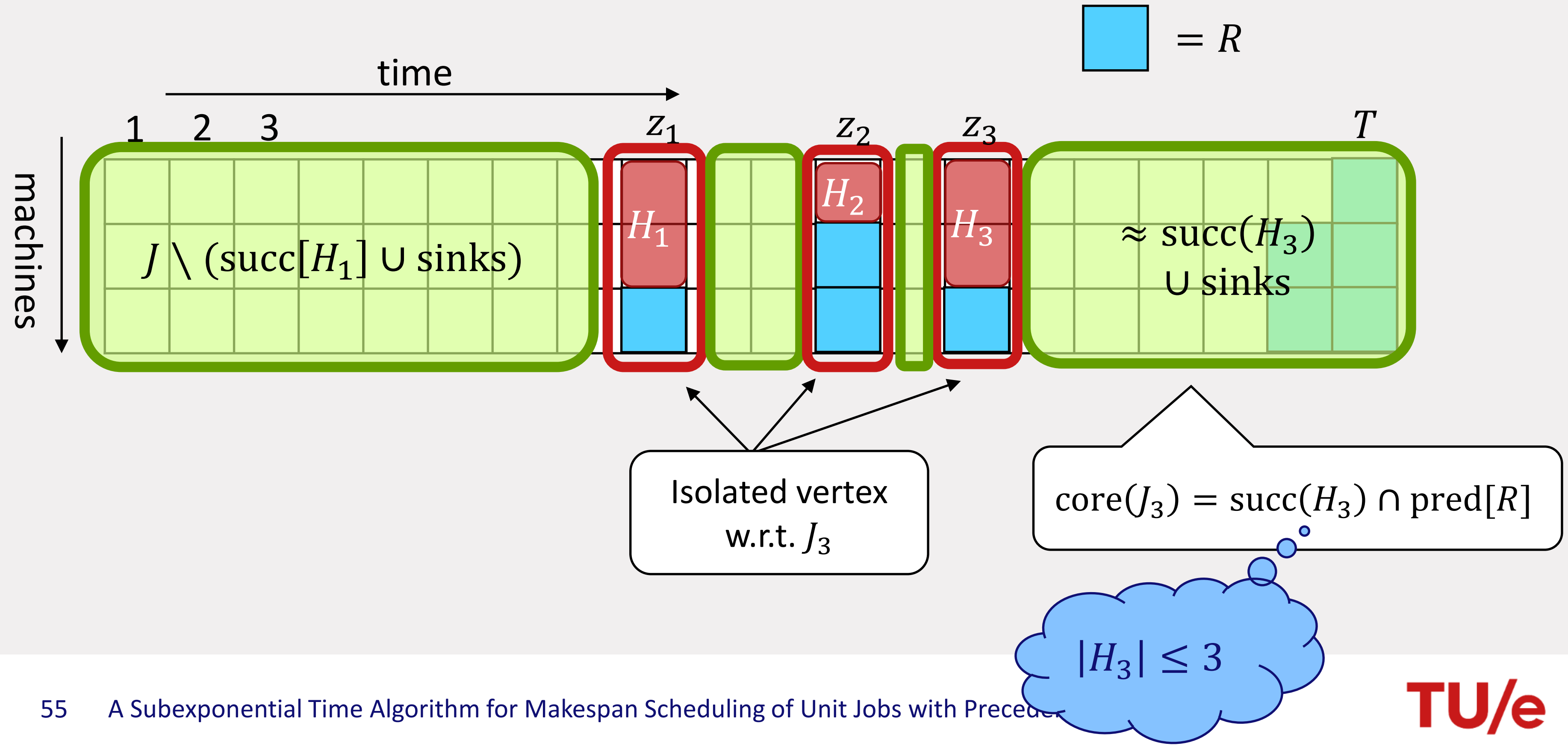
Going to the right

Assumption: $n = 3 \cdot T$



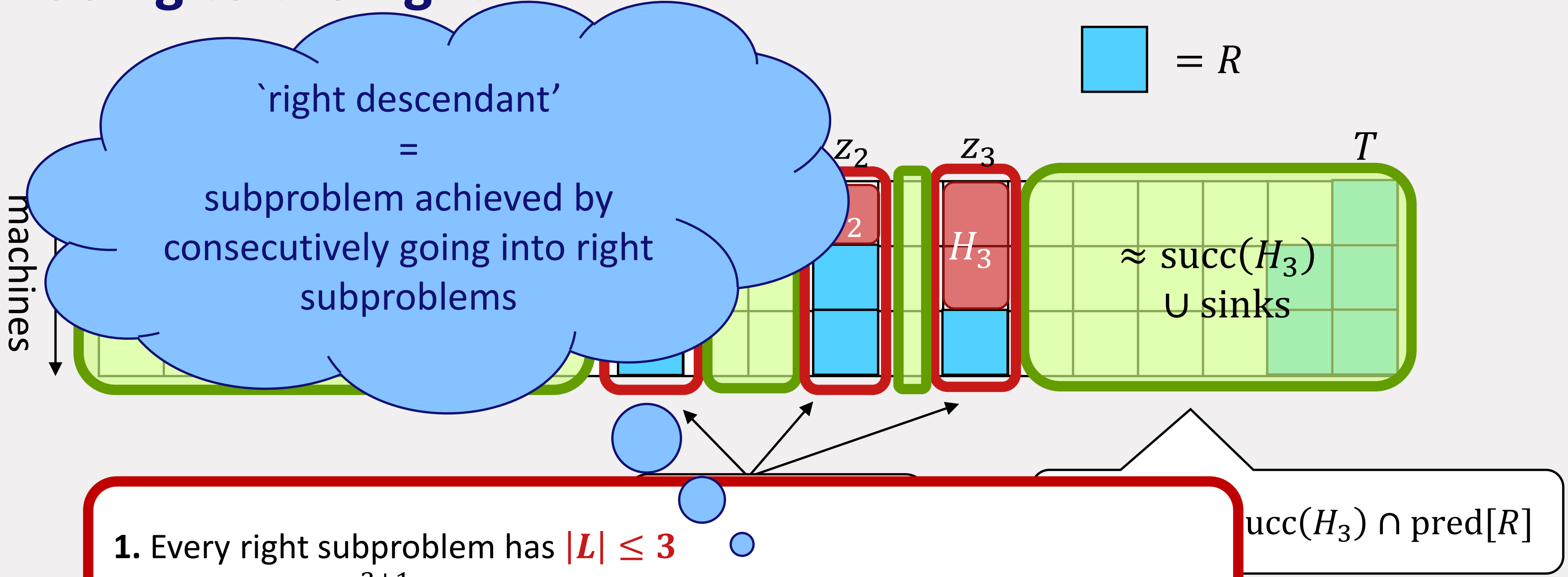
Going to the right

Assumption: $n = 3 \cdot T$



Going to the right

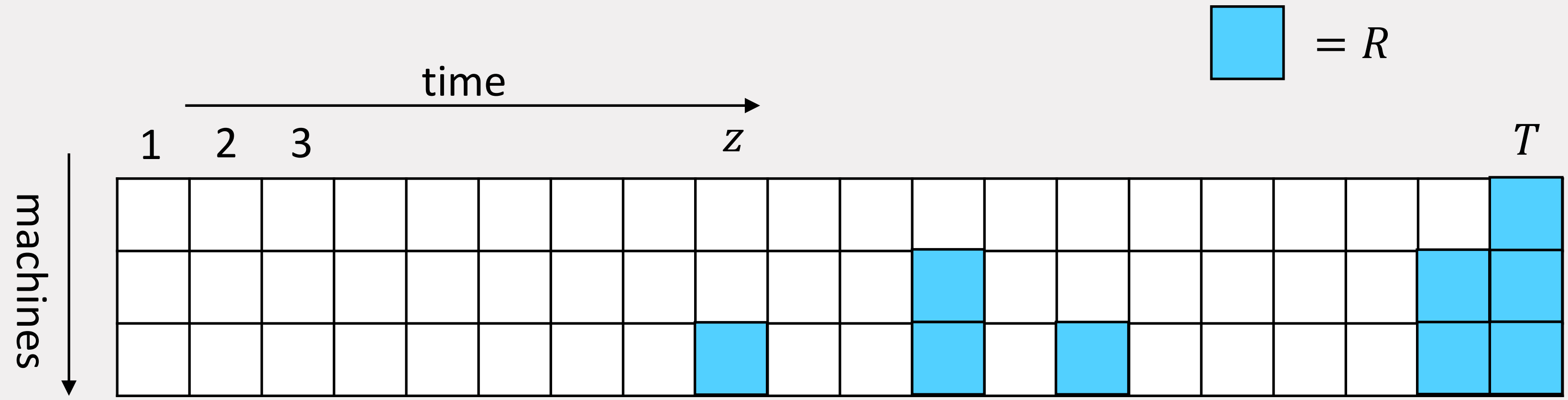
Assumption: $n = 3 \cdot T$



1. Every right subproblem has $|L| \leq 3$
2. There are $\leq n^{3+1}$ different **`right descendants'**

Going to the left

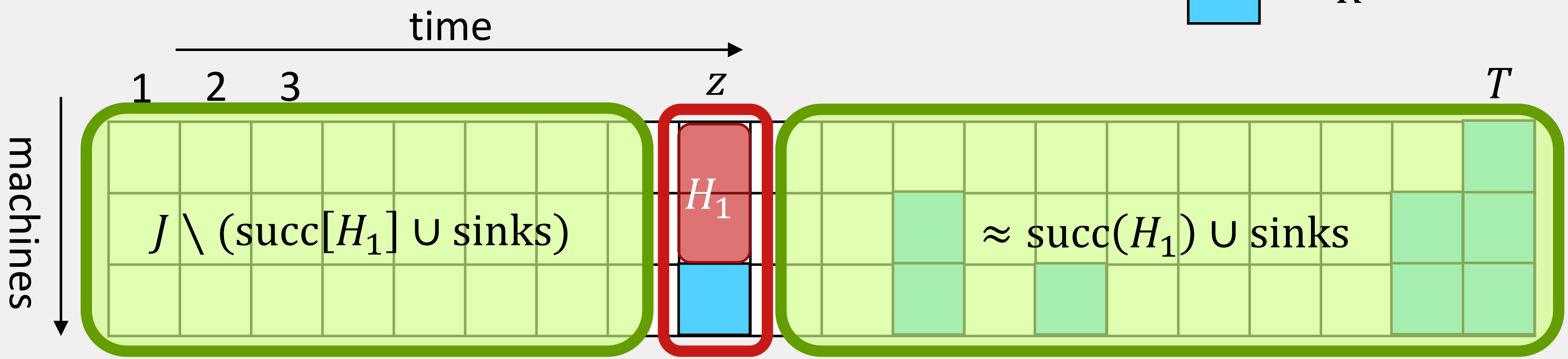
Assumption: $n = 3 \cdot T$



Going to the left

Assumption: $n = 3 \cdot T$

$= R$



core $= \mathbf{succ}(\boldsymbol{L}) \cap \mathrm{pred}[R_{\mathrm{new}}]$

1. Every left subproblem has $|\boldsymbol{L}| \leq \mathbf{3}$
2. Problem size decreases by $|\boldsymbol{R}|$

Win-Win strategy

Invariant: $J = (\mathrm{succ}(L) \cap \mathrm{pred}[R]) \cup k$ isolated vertices, $|L| \leq 3$

$\leq n^3$

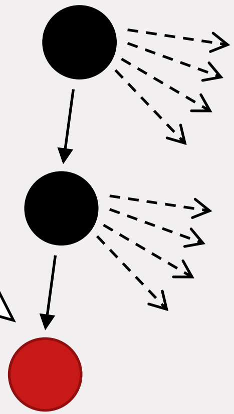
Win-win strategy

$\leq n$

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

TU/e

Branching Tree

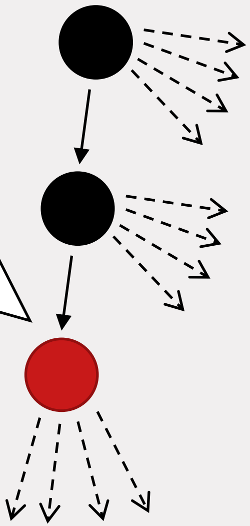


Either already in Lookup Table:
- Base case

$= |R| \leq \sqrt{n}$

$= |R| > \sqrt{n}$

Branching Tree



Either already in Lookup Table:
- Base case

Or not yet in Lookup Table:

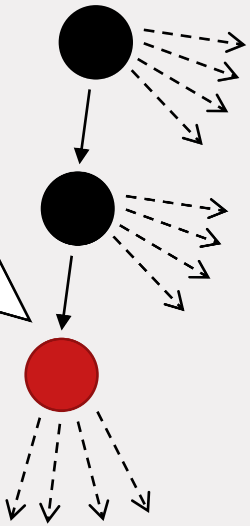
$= |R| \leq \sqrt{n}$

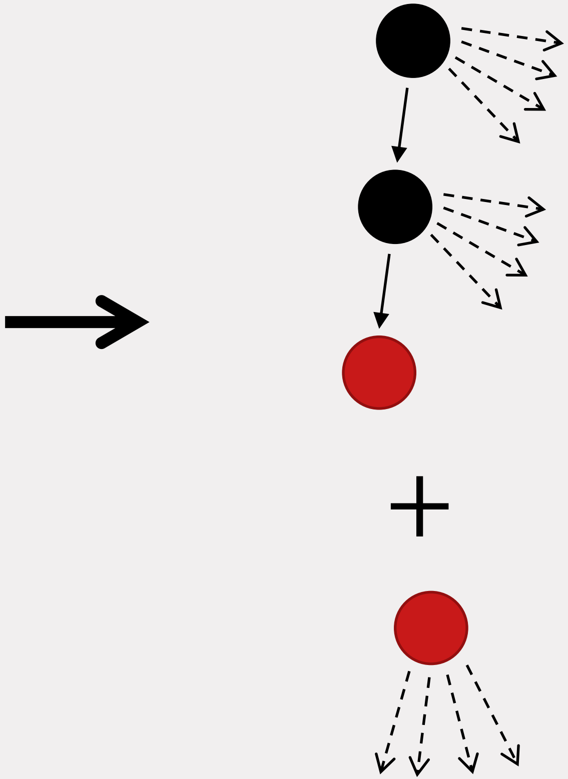
$= |R| > \sqrt{n}$

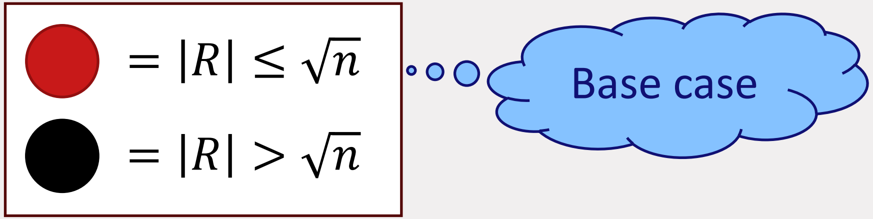
Branching Tree

Either already in Lookup Table:
- Base case

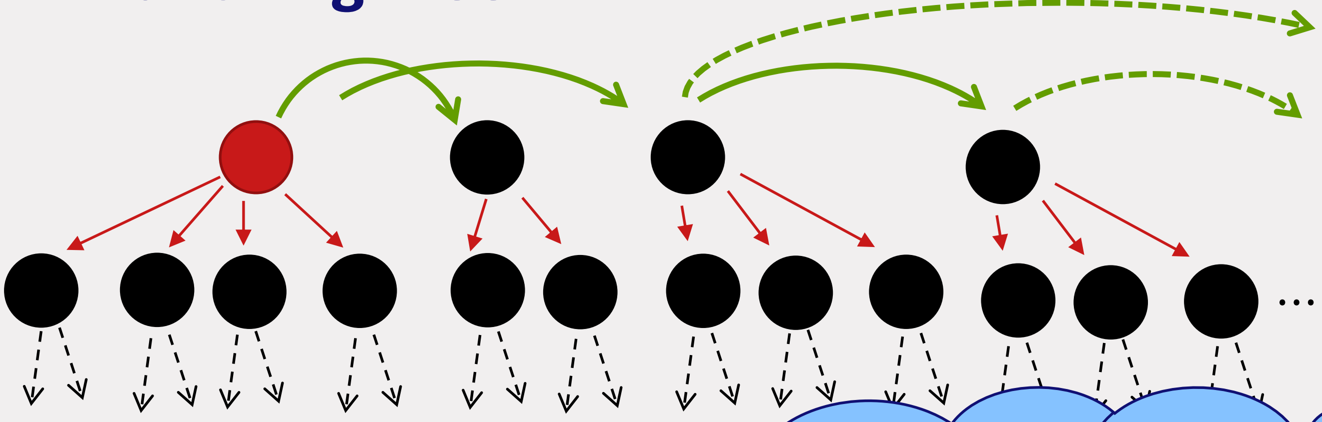
Or not yet in Lookup Table:
- View as its `own tree'
- $\Rightarrow n^{\sqrt{n}}$ such trees







Branching Tree

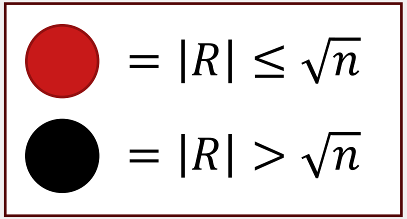


$\leq n^4$ right descendants

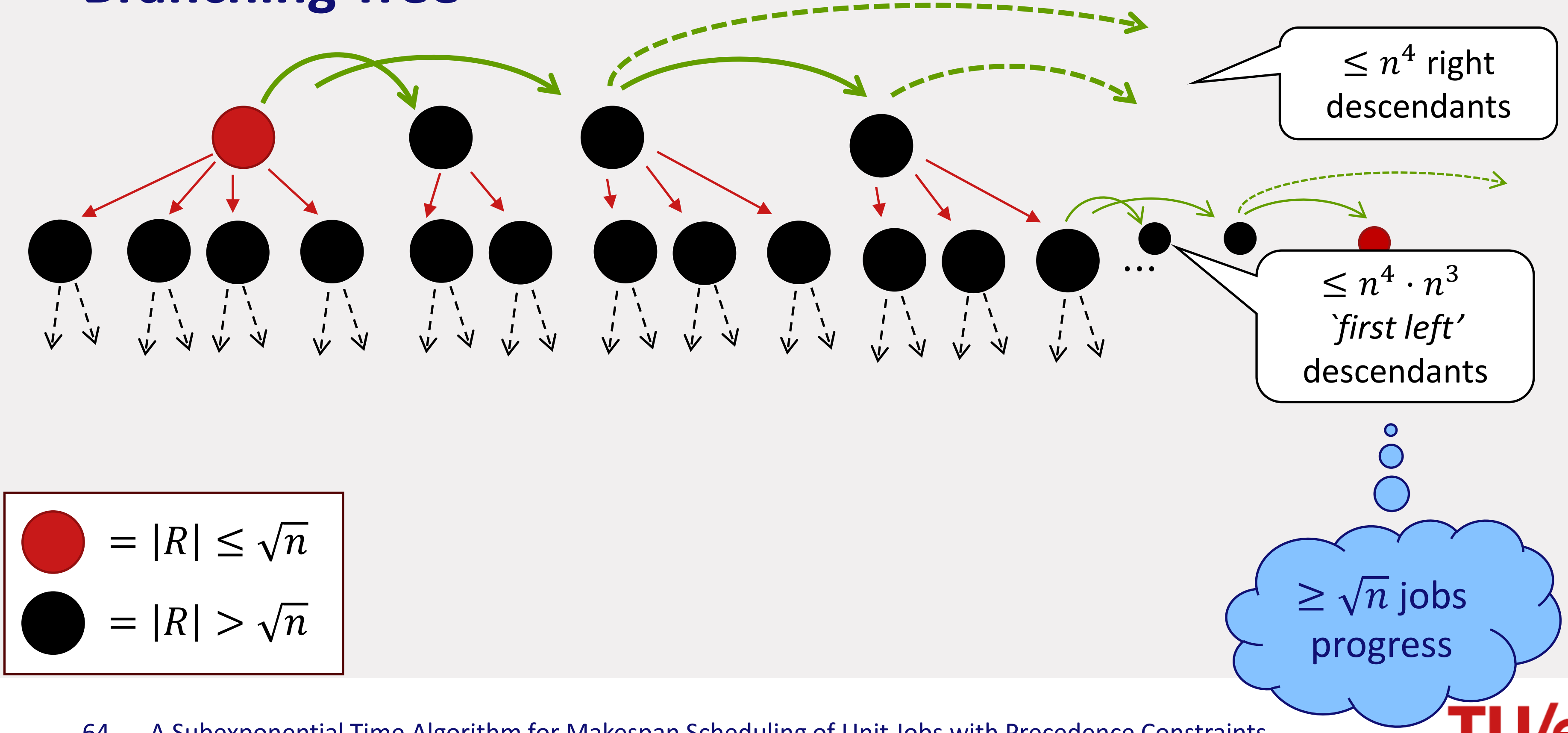
$\leq n^4 \cdot n^3$ `first left' descendants

`first left descendant'
=
subproblem achieved by consecutively going into right subproblems, then **once left**

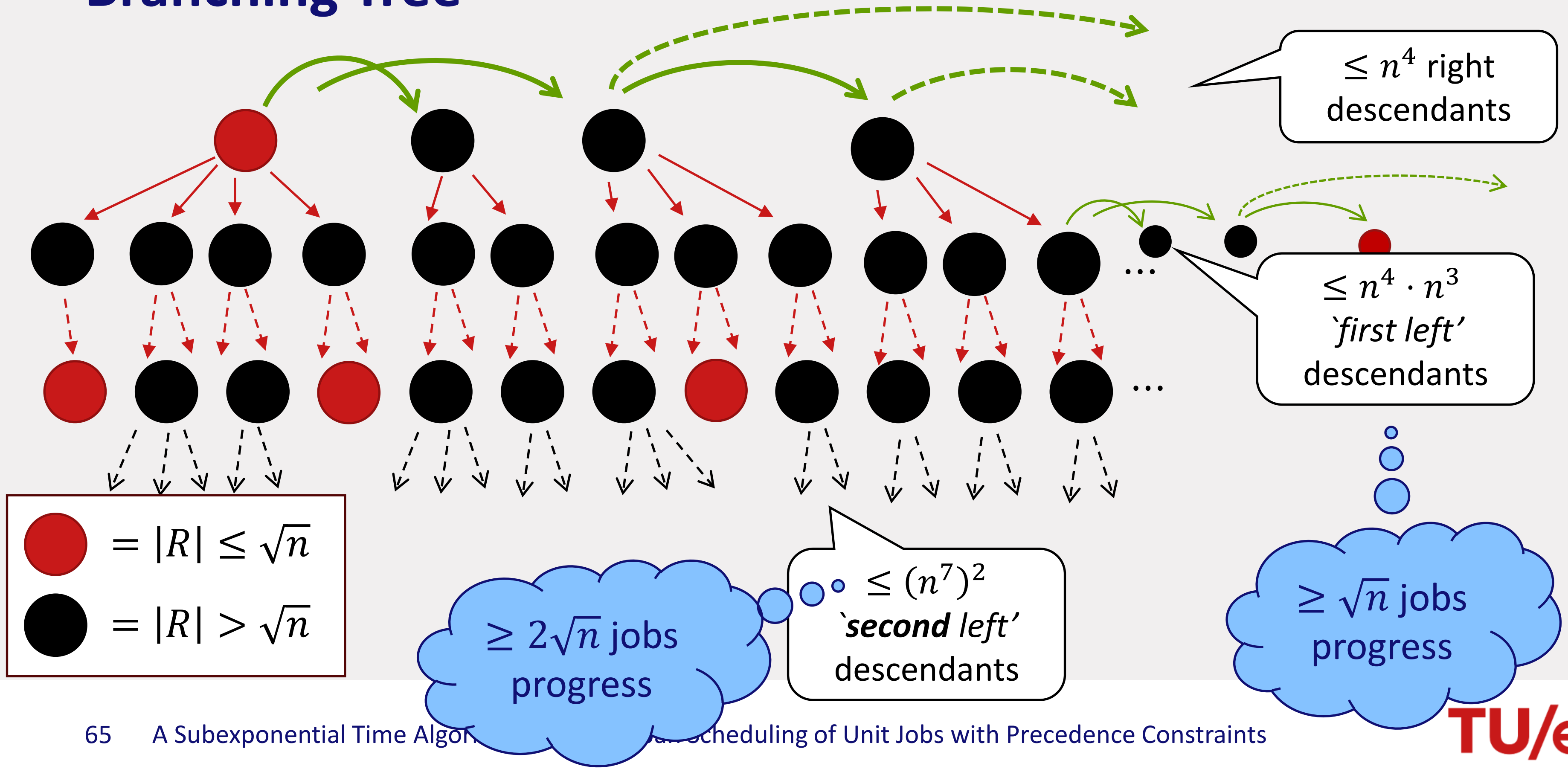
$\geq \sqrt{n}$ jobs progress



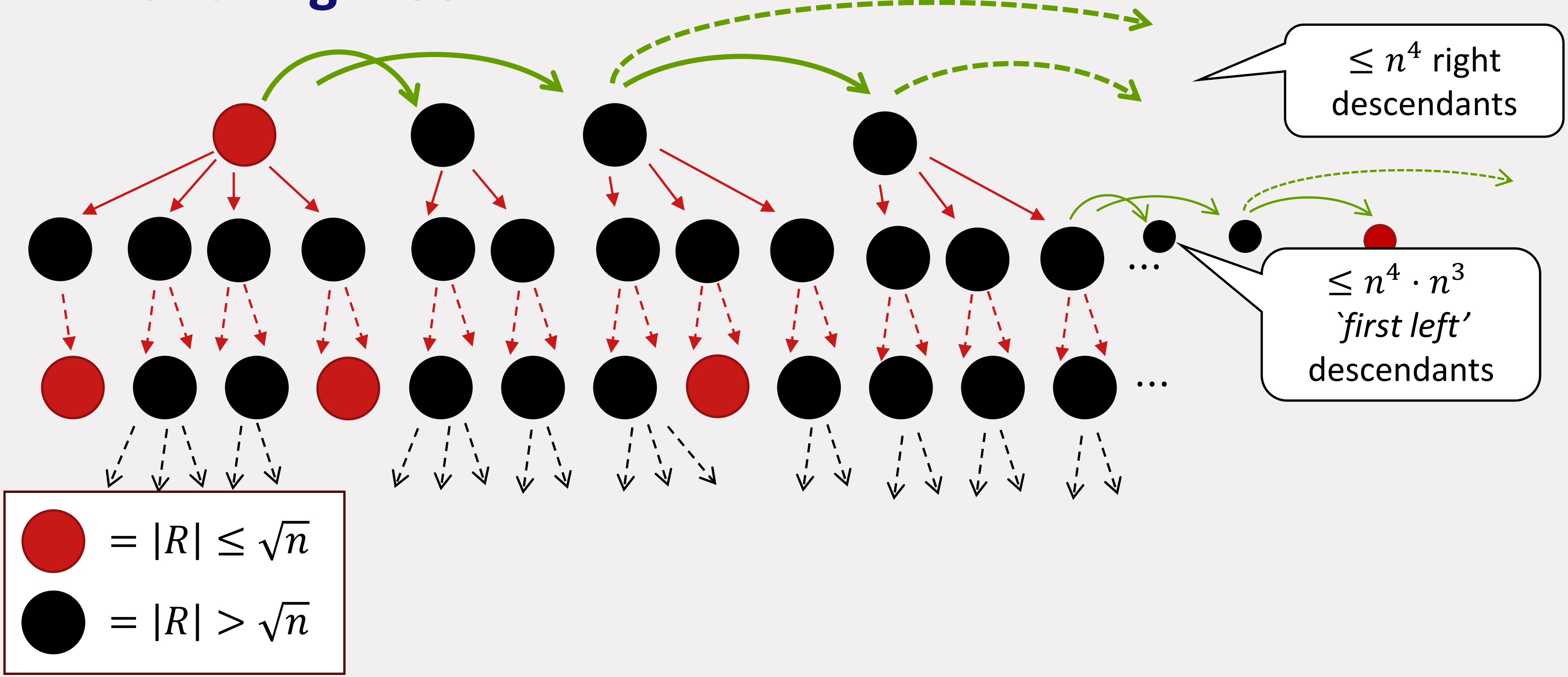
Branching Tree



Branching Tree

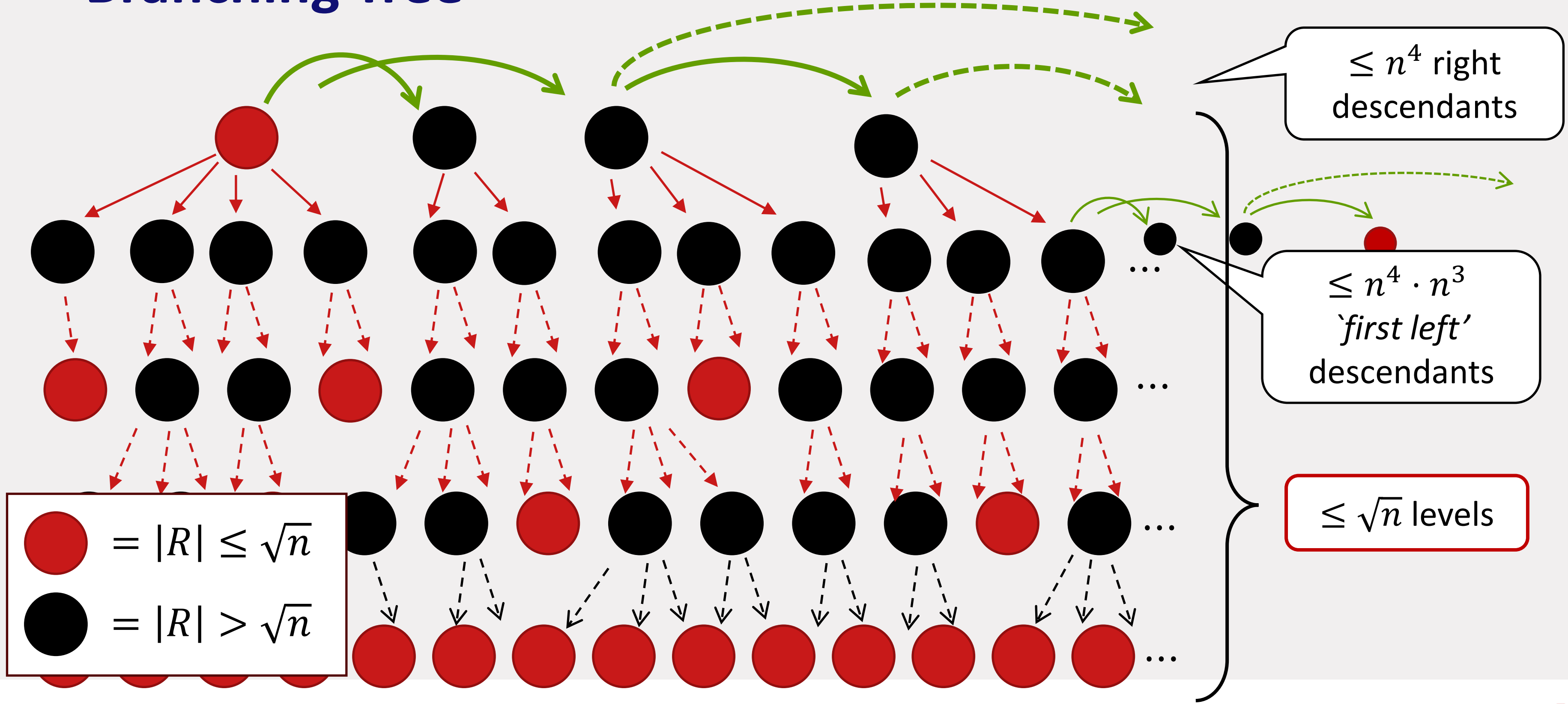


Branching Tree

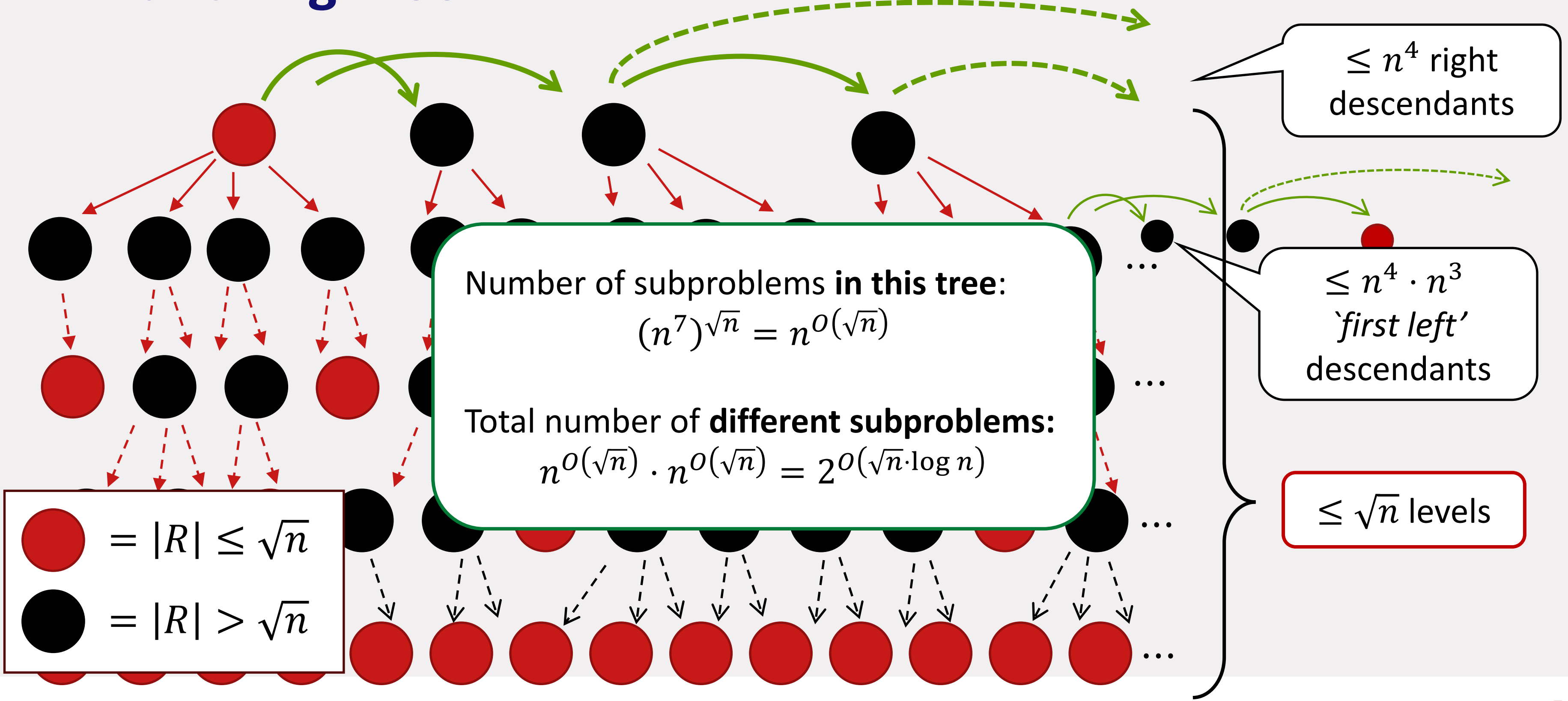


TU/e

Branching Tree



Branching Tree



Algorithm

Schedule(J):

1. **return** $\mathrm{LUT}[\mathrm{core}(J), \#\mathrm{iso}(J)]$ if it was already set
2. **if** $J = \emptyset$ **return** 0
3. **for each** $H \in \mathrm{Sep}(J)$ **do:**
4. $\qquad \mathrm{OPT}[\mathrm{left}(J,H)] \coloneqq \mathrm{Schedule}\big(\mathrm{left}(J,H)\big)$
5. $\qquad \mathrm{OPT}[\mathrm{right}(J,H)] \coloneqq \mathrm{Schedule}(\mathrm{right}(J,H))$
6. $\mathrm{OPT}[J] := \min_{H \in \mathrm{Sep}(J)} \{\mathrm{OPT}[\mathrm{left}(J,H)] + \mathrm{OPT}[\mathrm{right}(J,H)] + 1\}$
7. $\mathrm{LUT}[\mathrm{core}(J), \#\mathrm{iso}(J)] = \mathrm{OPT}[J]$
8. Return $\mathrm{OPT}[J]$

$\mathrm{Sep}(J) \coloneqq \{\, H \subseteq J$ s.t.
(1) $|H| \leq 3$,
(2) H is antichain,
(3) $|H \setminus \mathrm{sinks}(J)| < 3\}$

$\mathrm{left}(J,H) \coloneqq J \setminus \big(\mathrm{succ}[\mathrm{H}] \cup \mathrm{sinks}(J)\big)$
$\mathrm{right}(J,H) \coloneqq J \cap \big((\mathrm{succ}(H) \cup \mathrm{sinks}(J)\big) \setminus H$

Only $2^{O(\sqrt{n}\cdot\log n)}$ different problems encountered

TU/e

Corollaries

Our result:

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

Corollary 1

$Pm|prec, p_j = 1|C_{\max}$ can be solved in subexponential time whenever $m = o(n)$.

Corollary 2

$\boldsymbol{P}|prec, p_j = 1|C_{\max}$ can be solved in $1.997^n \cdot poly(n)$ time.

Conclusion

Conclusion

Main result:

$P3|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n}\cdot\log n)}$ time.

Conclusion

Main result:

$P3|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key idea's:

1. Use of look-up table
2. Keeping track of core + # isolated vertices
3. Finding win-win strategy using number of sinks

Conclusion

Main result:

$P3|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key idea's:

1. Use of look-up table
2. Keeping track of core + # isolated vertices
3. Finding win-win strategy using number of sinks

Future Research:

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

Conclusion

Main result:

$P3|prec, p_j = 1|C_{\max}$ in $2^{O(\sqrt{n} \cdot \log n)}$ time.

Key idea's:

1. Use of look-up table
2. Keeping track of core + # isolated vertices
3. Finding win-win strategy using number of sinks

Future Research:

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

Thanks for your attention!