A Lasserre-based $(1+\varepsilon)$-approximation for Makespan Scheduling with Precedence Constraints

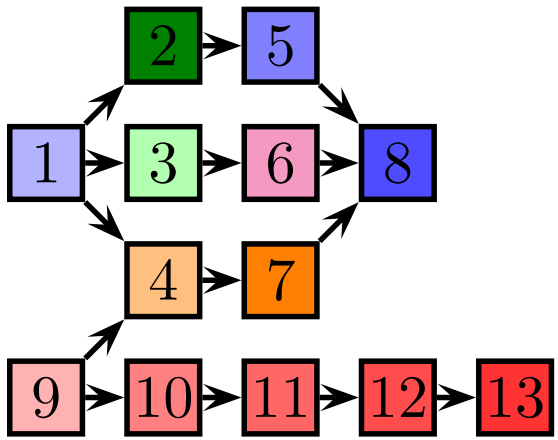
Elaine Levey and Thomas Rothvoss

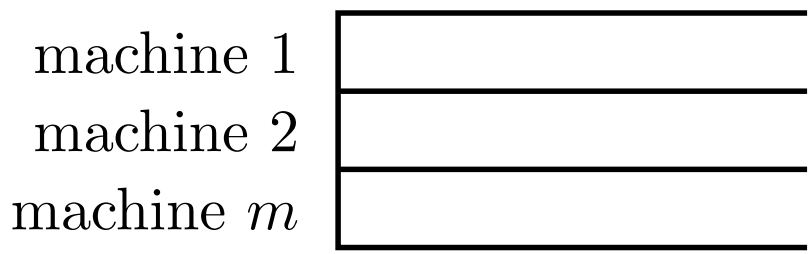
UNIVERSITY of WASHINGTON

Makespan Scheduling w. Precedence Constraints

Input:

- Unit-size jobs J with precedence constraints
- Number m of identical machines



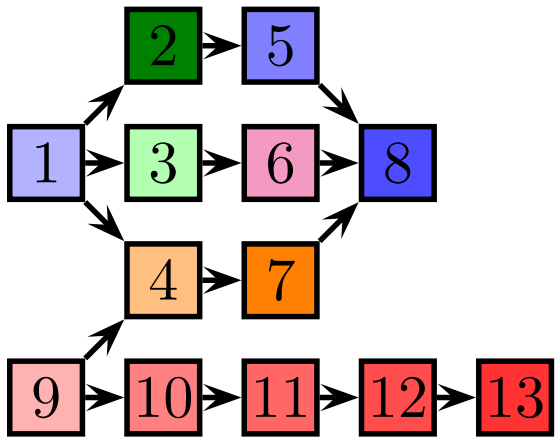


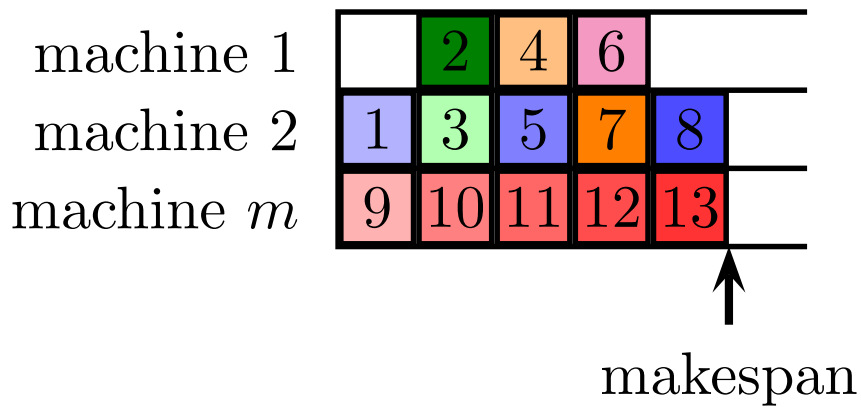
Makespan Scheduling w. Precedence Constraints

Input:

- Unit-size jobs J with precedence constraints
- Number m of identical machines

Goal: Schedule jobs as to **minimize makespan**



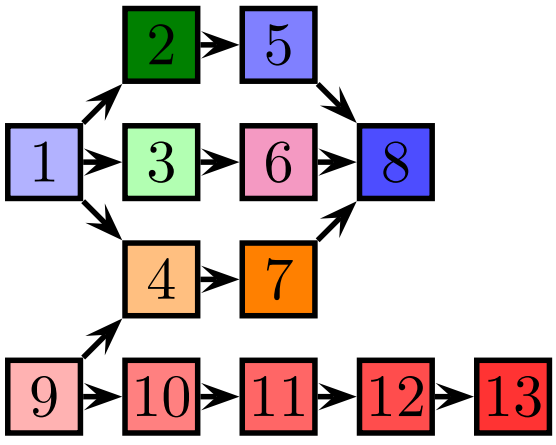


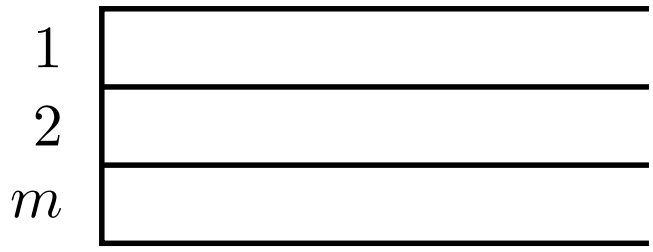
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



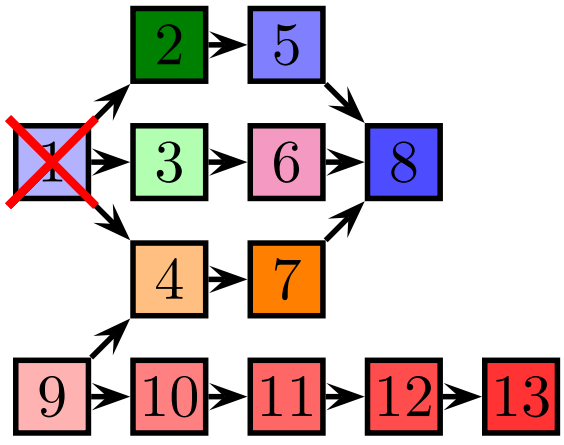


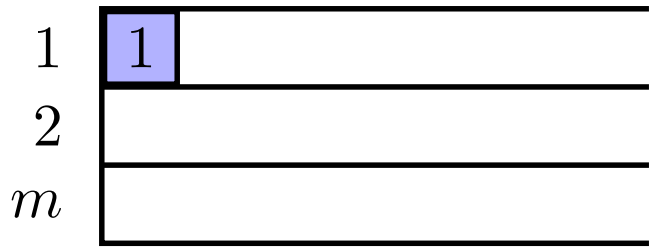
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



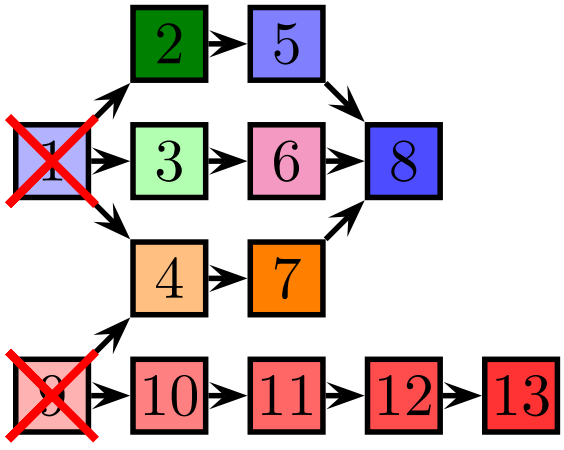


List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed

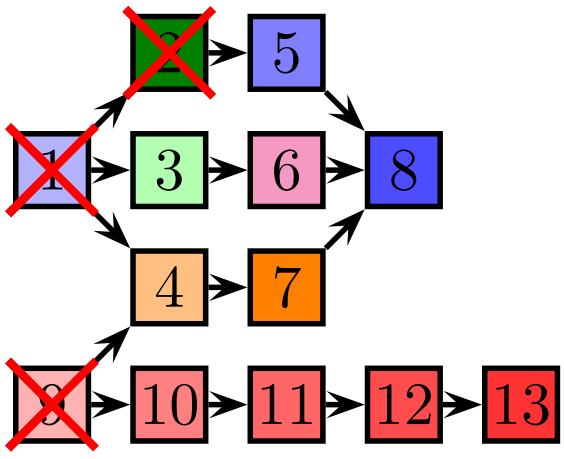


List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed

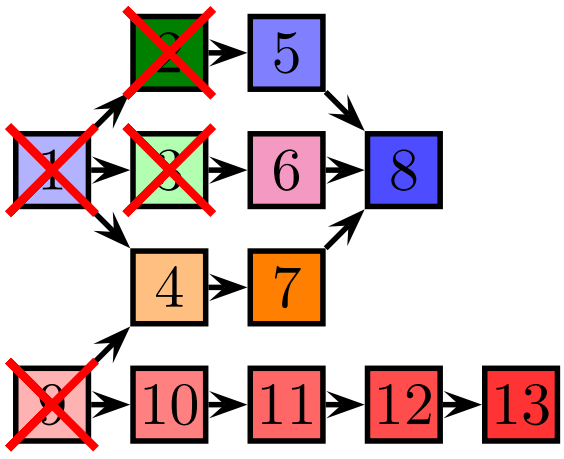


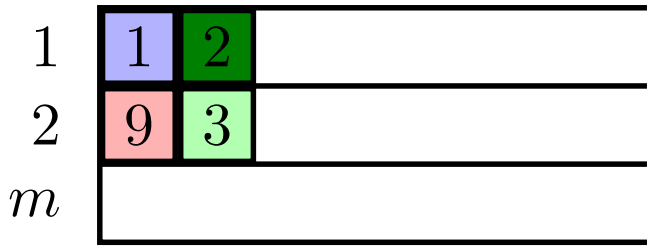
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



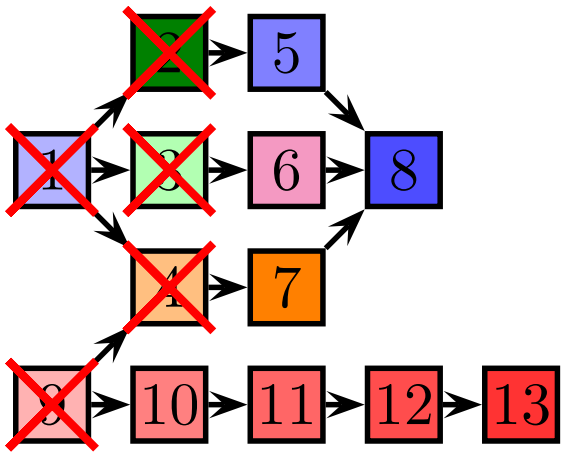


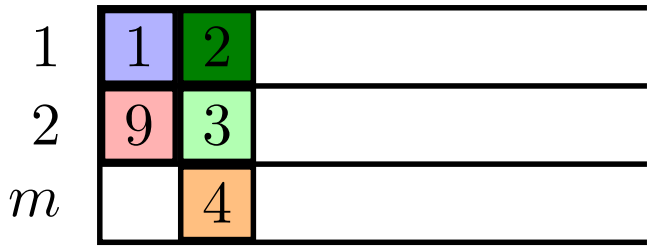
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



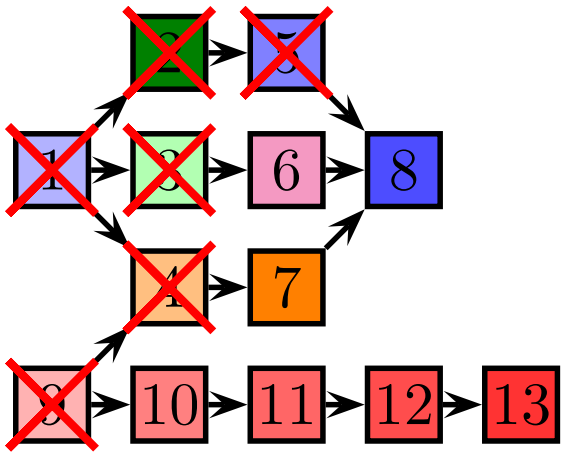


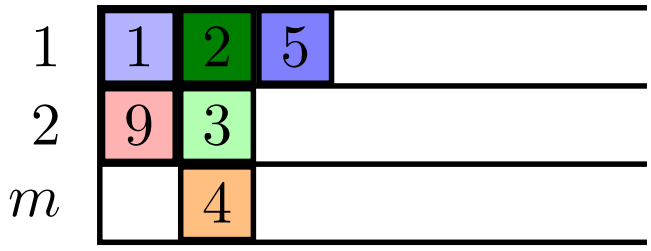
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



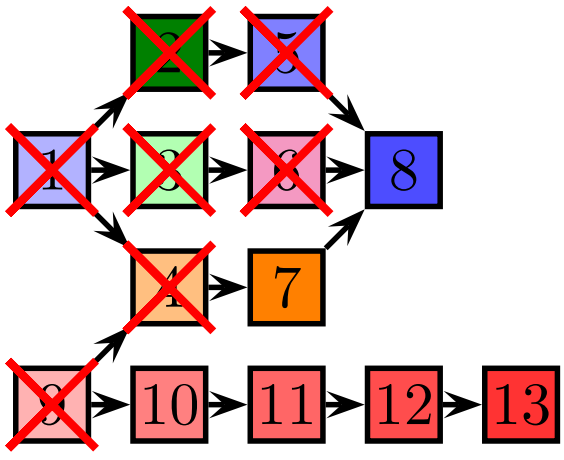


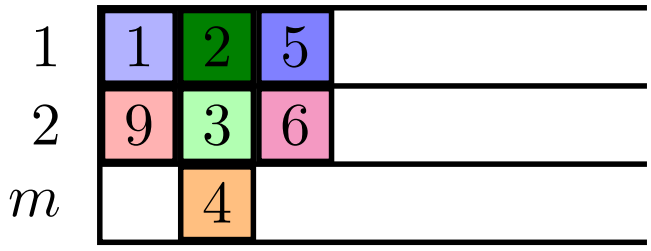
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



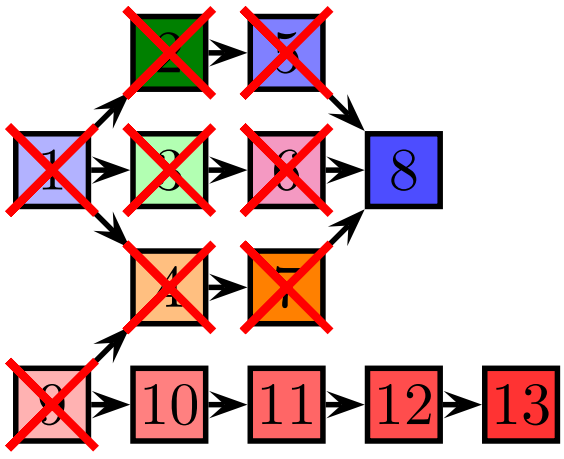


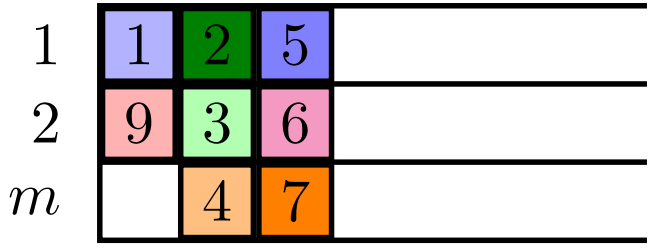
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



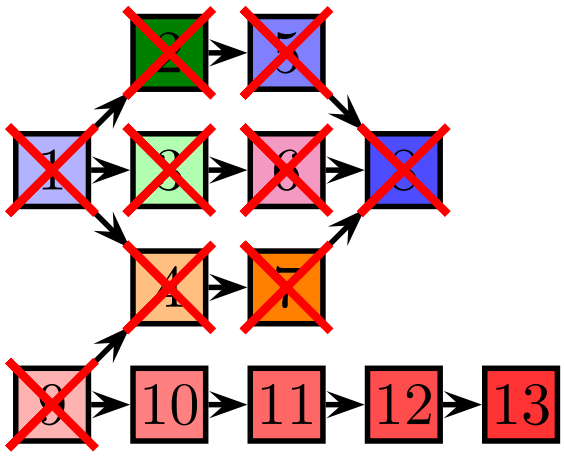


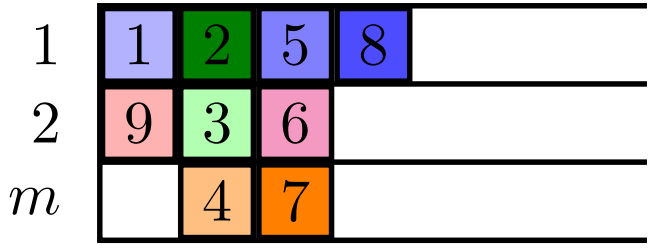
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



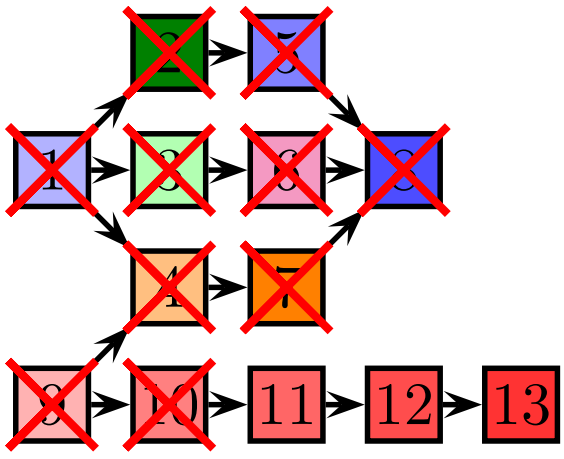


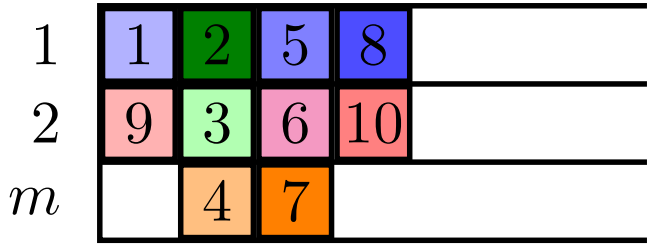
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



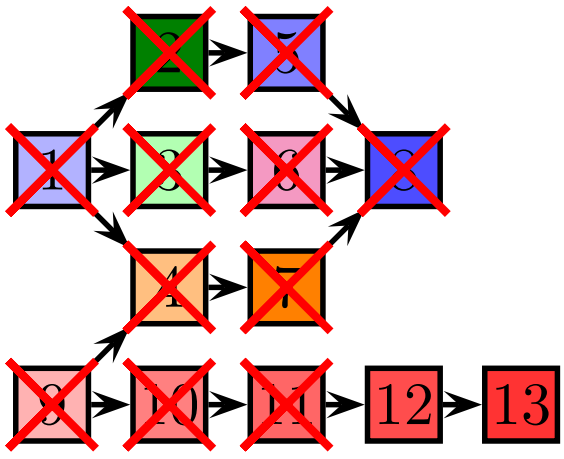


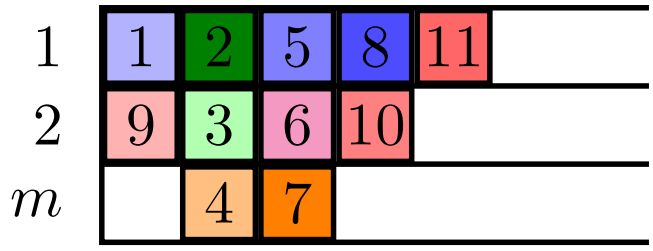
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



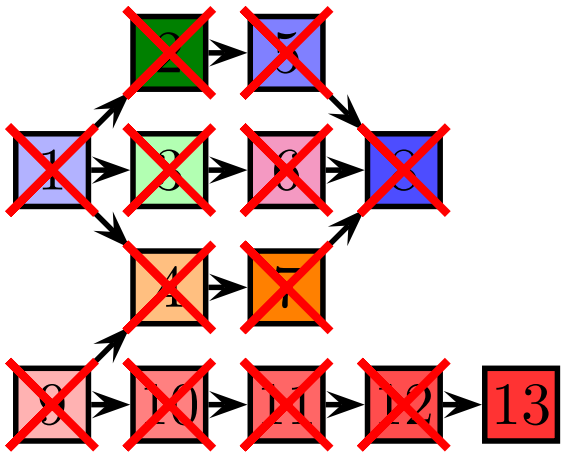


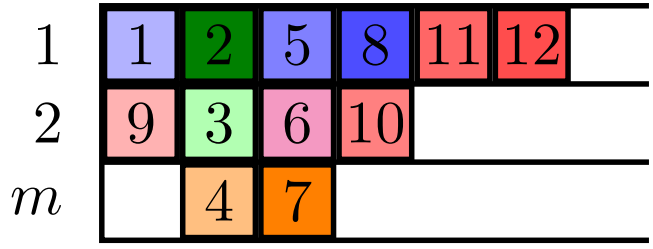
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



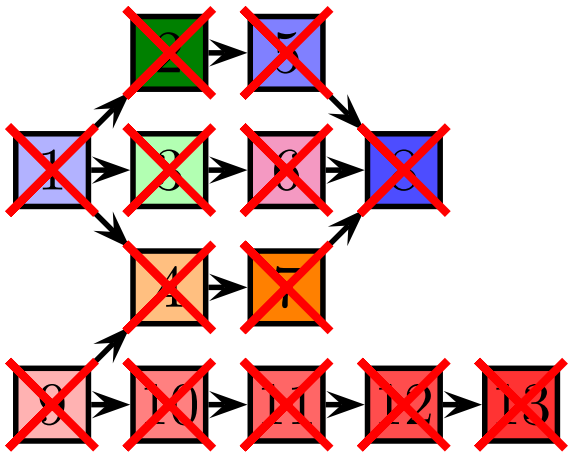


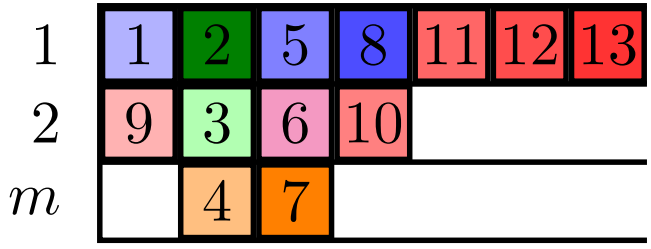
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed



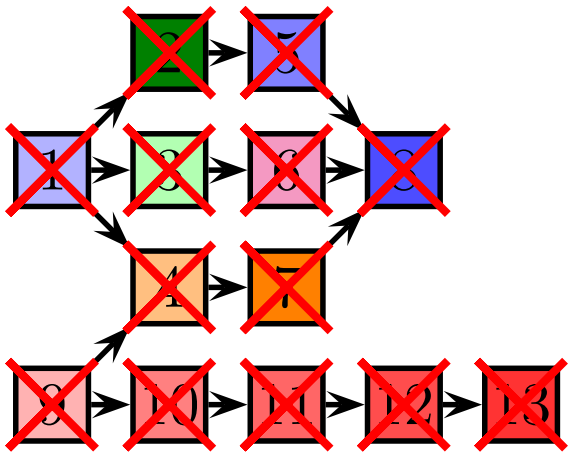


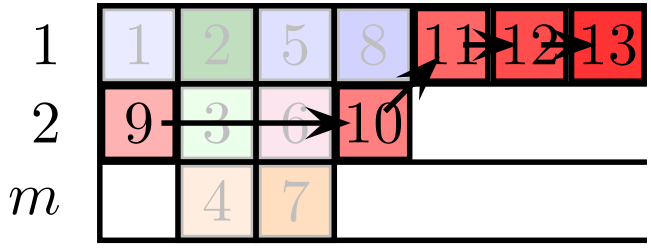
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed





Analysis:

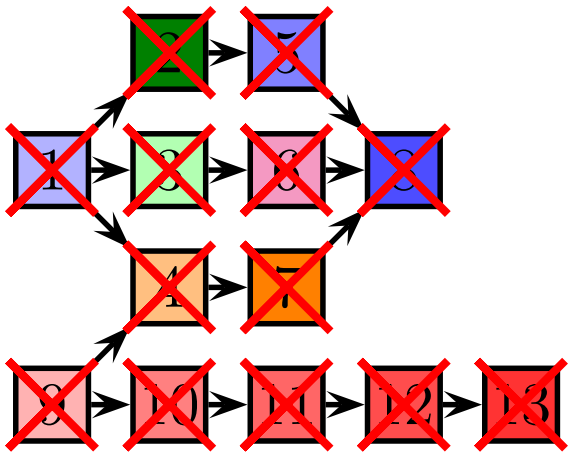
- There is a **chain** active at all non-busy times!

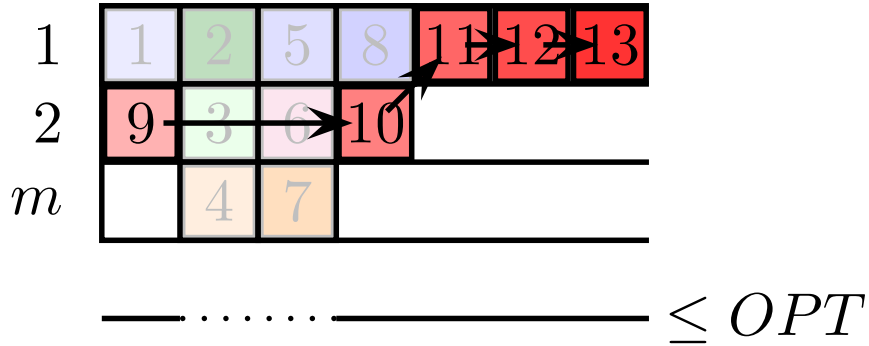
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed





Analysis:

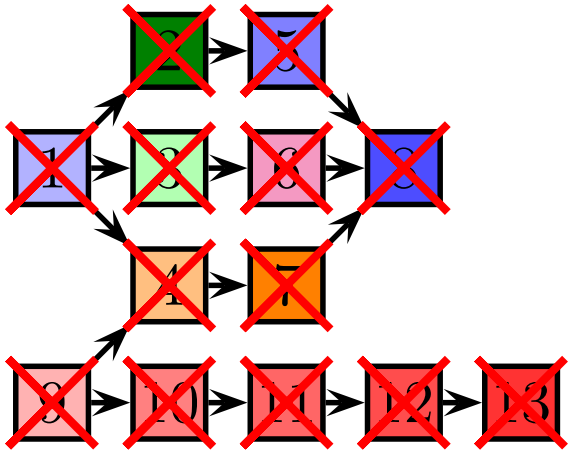
- There is a **chain** active at all non-busy times!
- Length of chain $\leq OPT$

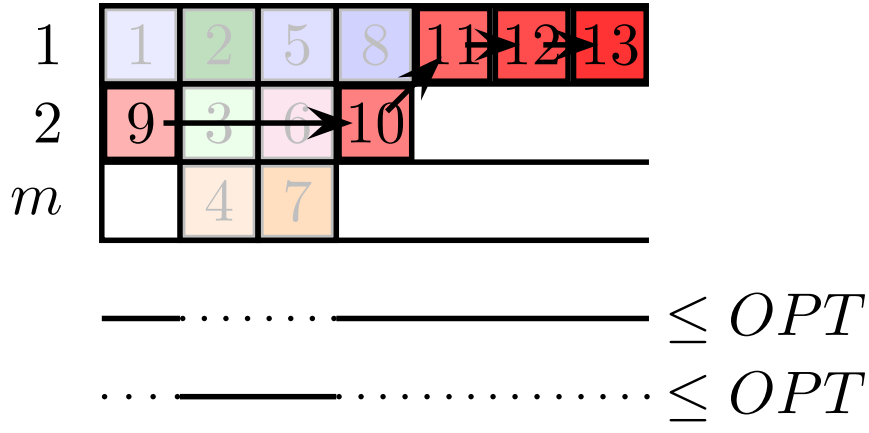
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed





Analysis:

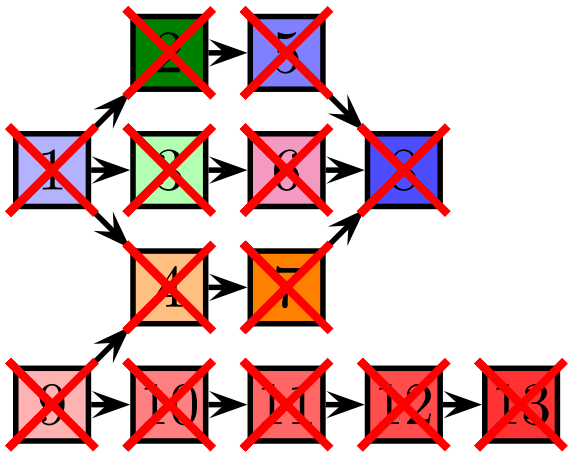
- There is a **chain** active at all non-busy times!
- Length of chain $\leq OPT$
- Length of busy times $\leq OPT$

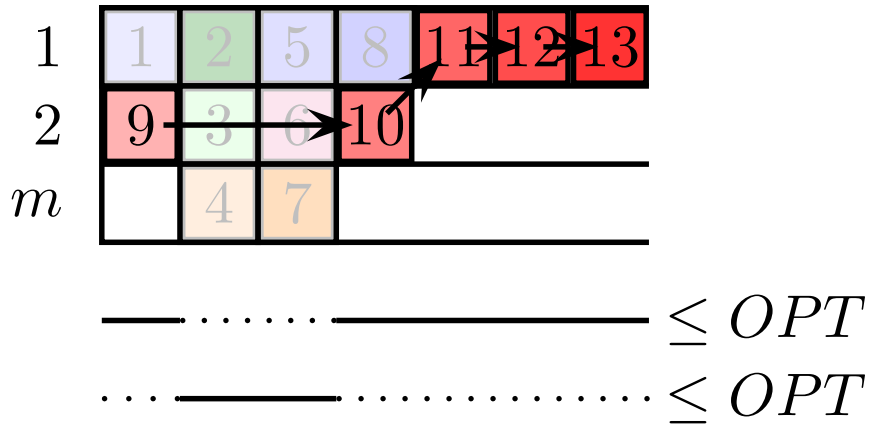
List Scheduling (Graham '66)

Algorithm:

(1) FOR slots at time $t = 1$ TO ∞ DO:

(2) Select any job that has all predecessors completed





Analysis:

- There is a **chain** active at all non-busy times!
- Length of chain $\leq OPT$
- Length of busy times $\leq OPT$

$\Rightarrow$ 2-apx ($(2 - \frac{2}{m})$ for $p_j = 1$; $2 - \frac{1}{m}$ for arbitrary p_j)

A linear programming formulation

$$
\begin{aligned}
\sum_{t=1}^{T} x_{jt} &= 1 \quad \forall j \in J \\
\sum_{j \in J} x_{jt} &\leq m \quad \forall t \in [T] \\
\sum_{t' \leq t} x_{it'} &\leq \sum_{t' \leq t+1} x_{jt'} \quad \forall i \prec j \; \forall t \in [T] \\
0 \leq x_{jt} &\leq 1 \quad \forall j \in J \; \forall t \in [T]
\end{aligned}
$$

Interpretation:

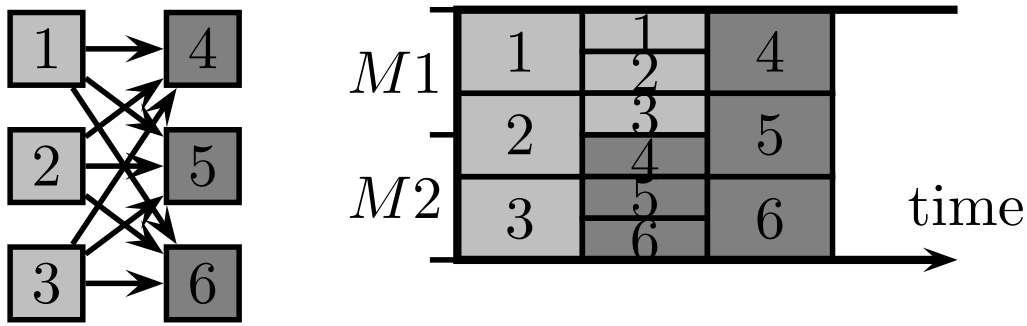
$$
x_{jt} = \begin{cases} 1 & \text{if run } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}
$$

A linear programming formulation

$$\begin{aligned}
\sum_{t=1}^{T} x_{jt} &= 1 \quad \forall j \in J \\
\sum_{j \in J} x_{jt} &\leq m \quad \forall t \in [T] \\
\sum_{t' \leq t} x_{it'} &\leq \sum_{t' \leq t+1} x_{jt'} \quad \forall i \prec j \; \forall t \in [T] \\
0 \leq x_{jt} &\leq 1 \quad \forall j \in J \; \forall t \in [T]
\end{aligned}$$

Interpretation:

$$x_{jt} = \begin{cases} 1 & \text{if run } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$



- Integrality gap: $2 - \Theta(\frac{1}{m})$

Main Theorem

Unit size job makespan minimization:

- $(2 - \Theta(\frac{1}{m}))$-apx in poly-time [Graham '66]

Main Theorem

Unit size job makespan minimization:

- $(2 - \Theta(\frac{1}{m}))$-apx in poly-time [Graham '66]
- No $(2 - \varepsilon)$-apx under variant of UGC [Svensson '10]

Main Theorem

Unit size job makespan minimization:

- $(2 - \Theta(\frac{1}{m}))$-apx in poly-time [Graham '66]
- No $(2 - \varepsilon)$-apx under variant of UGC [Svensson '10]

Open problems:

- Is $P3 \mid p_j = 1, \text{prec} \mid C_{\max}$ **NP**-hard? [Garey Johnson '79]

Main Theorem

Unit size job makespan minimization:

- $(2 - \Theta(\frac{1}{m}))$-apx in poly-time [Graham '66]
- No $(2 - \varepsilon)$-apx under variant of UGC [Svensson '10]

Open problems:

- Is $P3 \mid p_j = 1, \text{prec} \mid C_{\max}$ **NP**-hard? [Garey Johnson '79]
- Is there a PTAS for $P3 \mid p_j = 1, \text{prec} \mid C_{\max}$?
  [Part of # 1 of "Open problems in Scheduling";
  Schuurman, Woeginger '99]

Main Theorem

Unit size job makespan minimization:

- $(2 - \Theta(\frac{1}{m}))$-apx in poly-time [Graham '66]
- No $(2 - \varepsilon)$-apx under variant of UGC [Svensson '10]

Open problems:

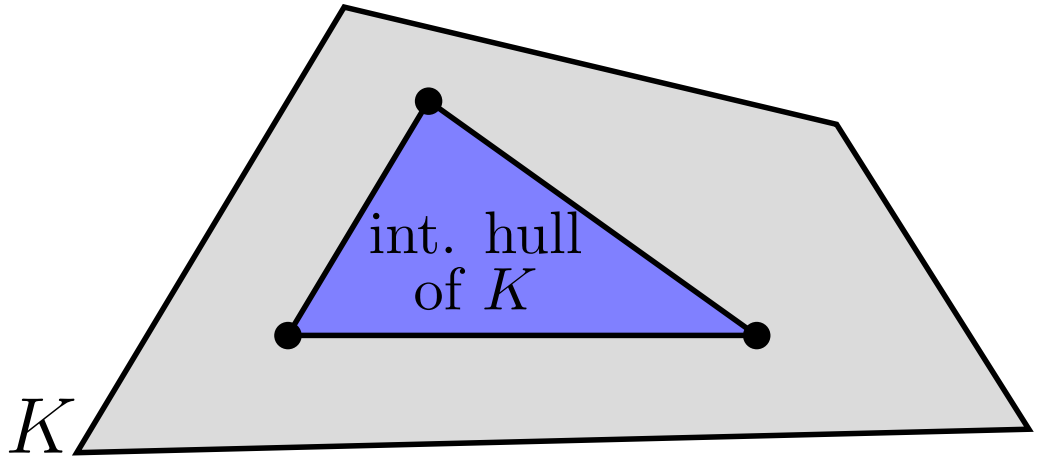
- Is $P3 \mid p_j = 1, \text{prec} \mid C_{\max}$ **NP**-hard? [Garey Johnson '79]
- Is there a PTAS for $P3 \mid p_j = 1, \text{prec} \mid C_{\max}$?
  [Part of # 1 of "Open problems in Scheduling";
  Schuurman, Woeginger '99]

Theorem (Levey, R. 15)

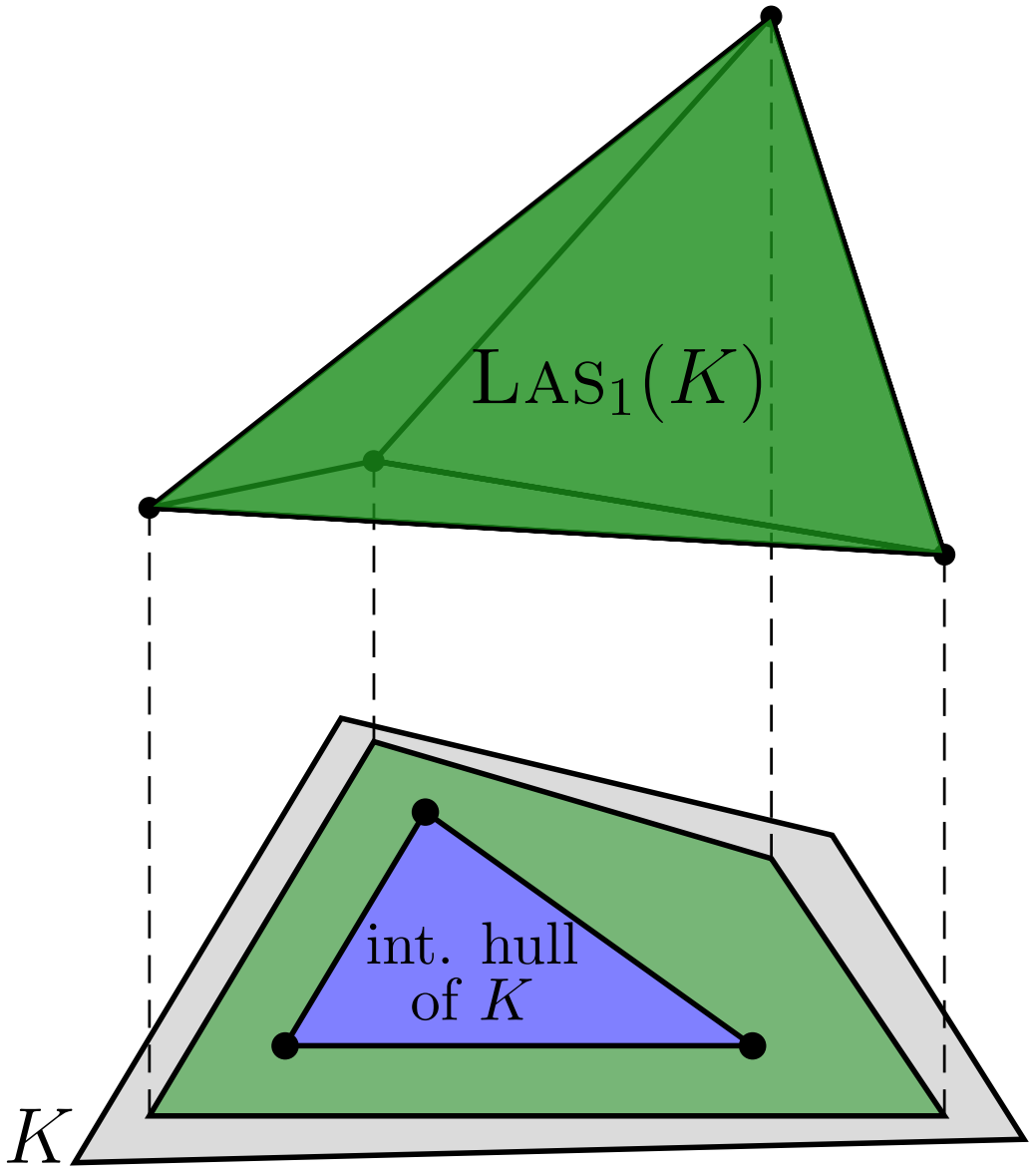
The integrality gap of LP is $1 + \varepsilon$ after $(\log n)^{O(\log \log n)}$**-Lasserre rounds** (for constants m and $\varepsilon > 0$).

- Running time $n^{(\log n)^{O(\log \log n)}}$
- Sherali-Adams suffices

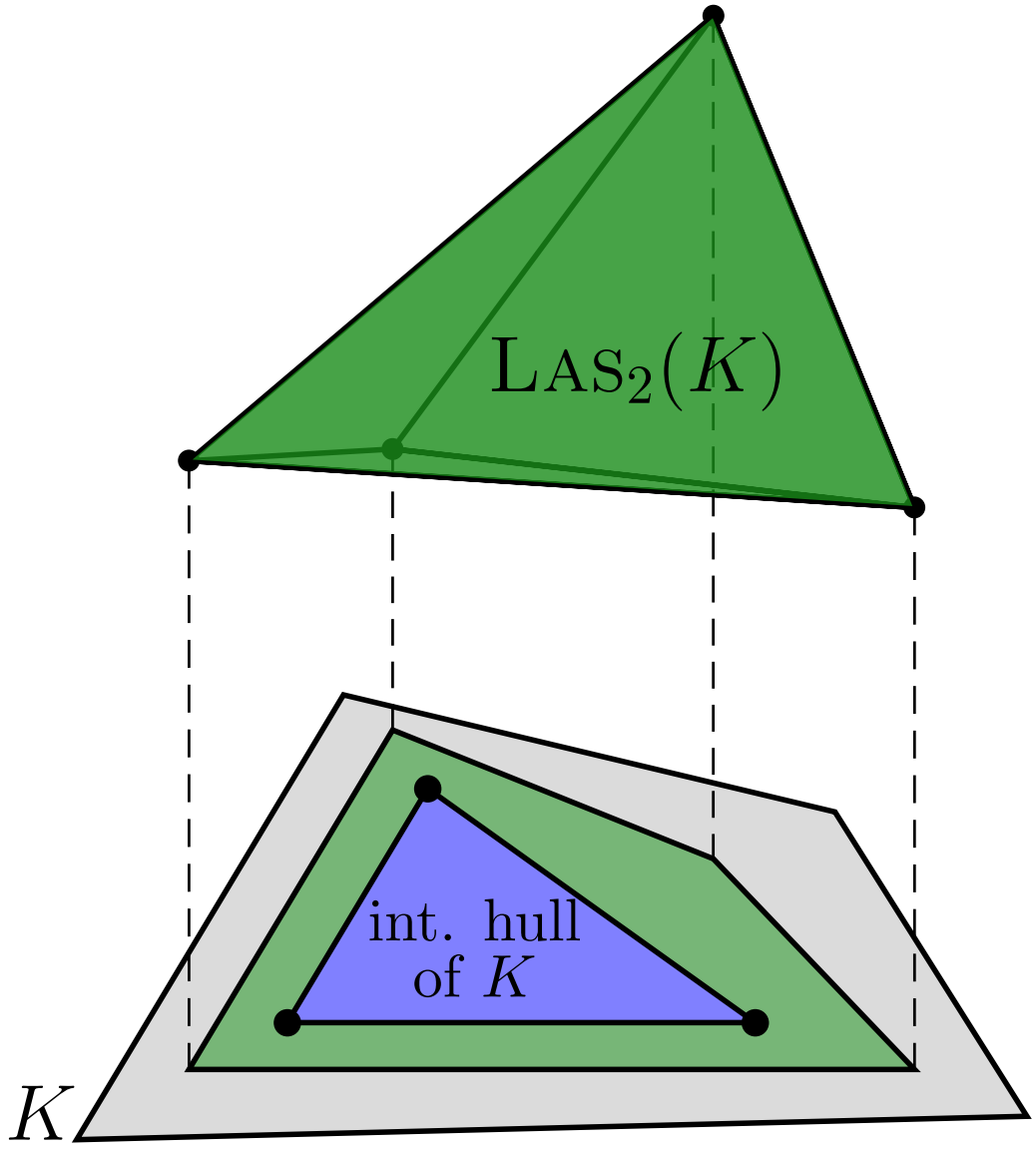
Lift-and-project methods



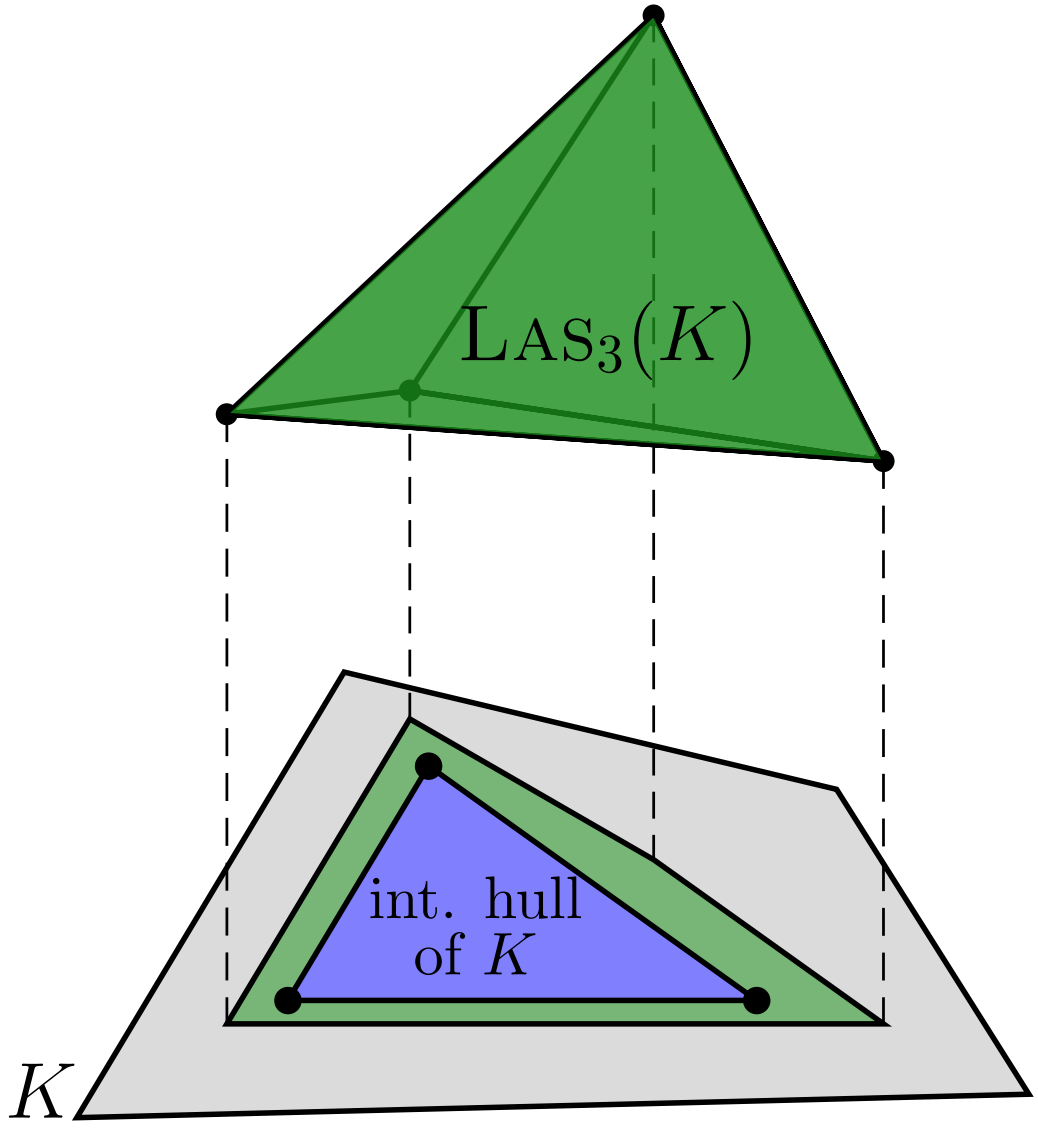
Lift-and-project methods



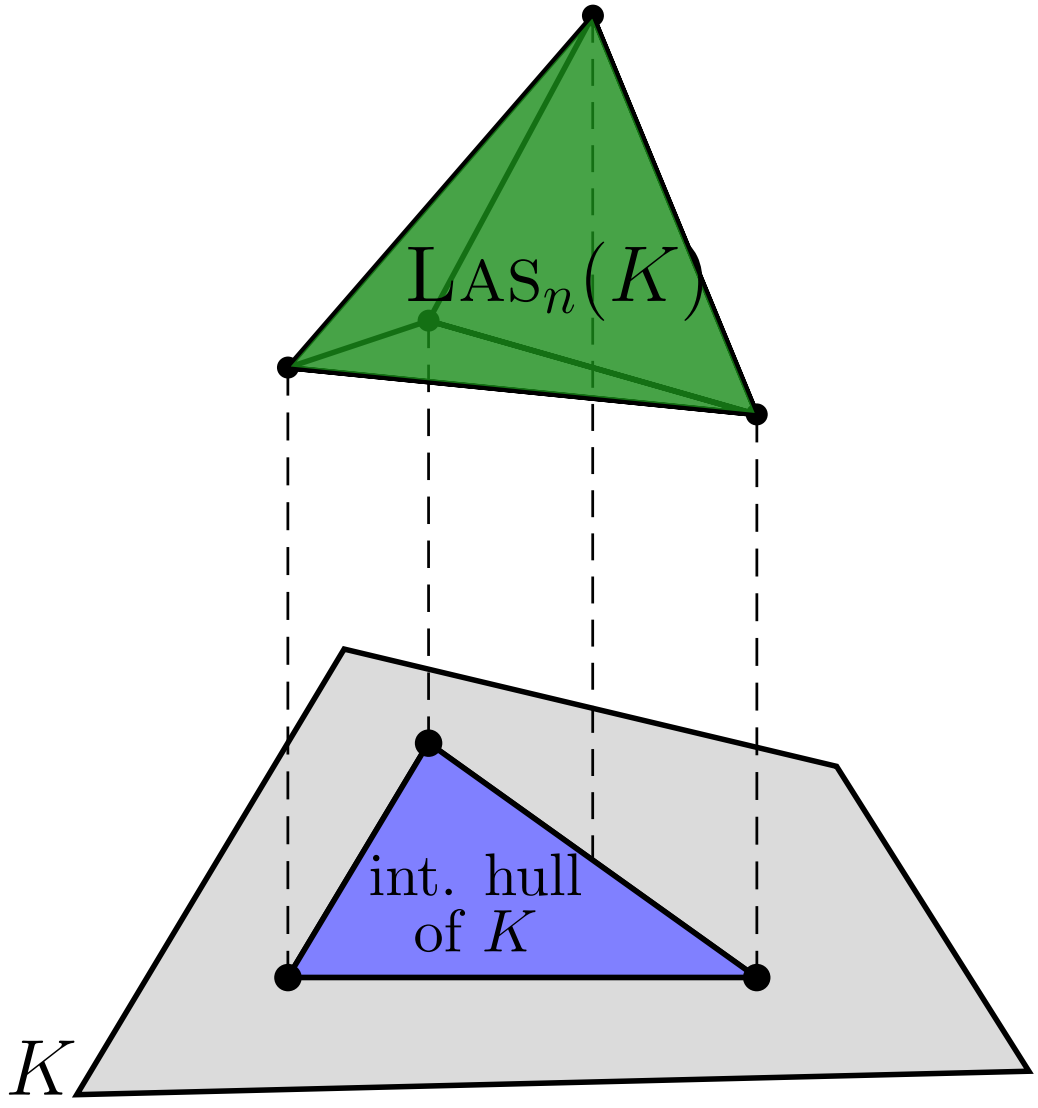
Lift-and-project methods



Lift-and-project methods



Lift-and-project methods



Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

- $y_{\{i\}} = x_i$

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

- $y_{\{i\}} = x_i$
- $y_\emptyset = 1$

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

- $y_{\{i\}} = x_i$
- $y_\emptyset = 1$

Round-t Lasserre relaxation

$$\begin{aligned} (y_{I \cup J})_{|I|,|J| \leq t} &\succeq 0 \\ \left(\textstyle\sum_{i \in [n]} A_{\ell i} y_{I \cup J \cup \{i\}} - b_\ell y_{I \cup J} \right)_{|I|,|J| \leq t} &\succeq 0 \quad \forall \ell \in [m] \\ y_\emptyset &= 1 \end{aligned}$$

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

- $y_{\{i\}} = x_i$
- $y_\emptyset = 1$

moment matrix

Round-t Lasserre relaxation

$$\begin{aligned} \boxed{(y_{I \cup J})_{|I|,|J| \leq t}} &\succeq 0 \\ \left(\textstyle\sum_{i \in [n]} A_{\ell i} y_{I \cup J \cup \{i\}} - b_\ell y_{I \cup J} \right)_{|I|,|J| \leq t} &\succeq 0 \quad \forall \ell \in [m] \\ y_\emptyset &= 1 \end{aligned}$$

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

- $y_{\{i\}} = x_i$
- $y_\emptyset = 1$

slack moment matrix

moment matrix

Round-t Lasserre relaxation

$$\begin{aligned} \left(y_{I \cup J}\right)_{|I|,|J| \leq t} &\succeq 0 \\ \left(\sum_{i \in [n]} A_{\ell i} y_{I \cup J \cup \{i\}} - b_\ell y_{I \cup J}\right)_{|I|,|J| \leq t} &\succeq 0 \quad \forall \ell \in [m] \\ y_\emptyset &= 1 \end{aligned}$$

Round-t Lasserre relaxation

- Given: $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$.
- Introduce variables for $I \subseteq [n]$, $|I| \leq t$

$$y_I \equiv \bigwedge_{i \in I} (x_i = 1)$$

- $y_{\{i\}} = x_i$
- $y_\emptyset = 1$

slack moment matrix

moment matrix

Round-t Lasserre relaxation

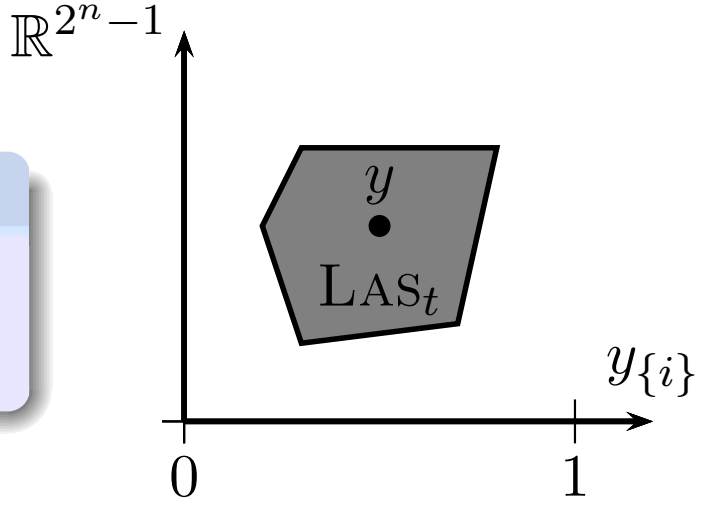
$$\begin{aligned} \left(y_{I \cup J}\right)_{|I|,|J| \leq t} &\succeq 0 \\ \left(\sum_{i \in [n]} A_{\ell i} y_{I \cup J \cup \{i\}} - b_\ell y_{I \cup J}\right)_{|I|,|J| \leq t} &\succeq 0 \quad \forall \ell \in [m] \\ y_\emptyset &= 1 \end{aligned}$$

- Solvable in time $n^{O(t)} m^{O(1)}$

Inducing on one variable

Lemma

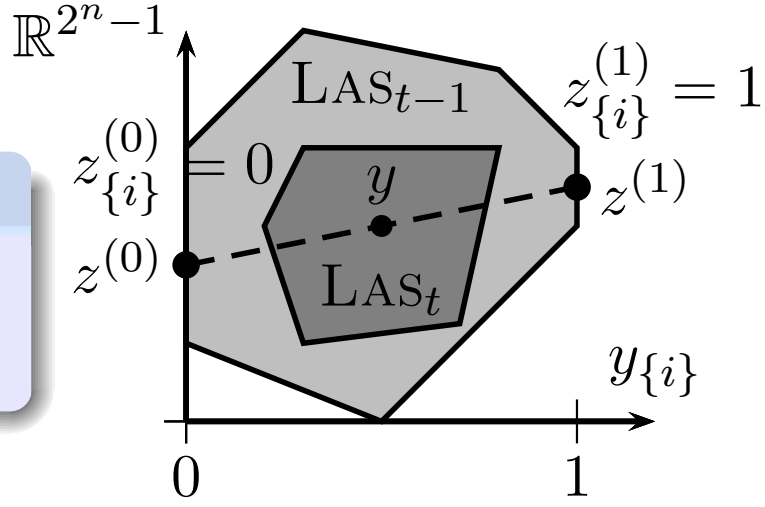
For $y \in \text{LAS}_t(K)$, $t \geq 1$, $i \in [n]$,
$y \in conv\{z \in \text{LAS}_{t-1}(K) \mid z_i \in \{0,1\}\}$.



Inducing on one variable

Lemma

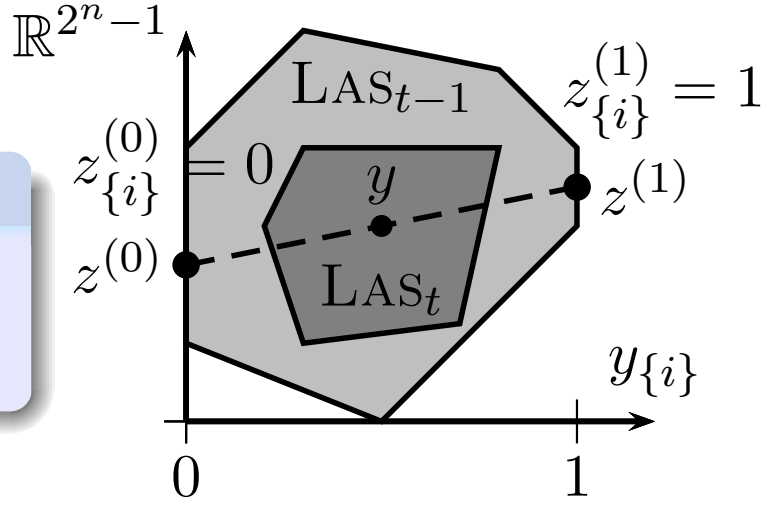
For $y \in \text{LAS}_t(K)$, $t \geq 1$, $i \in [n]$,
$y \in conv\{z \in \text{LAS}_{t-1}(K) \mid z_i \in \{0,1\}\}$.



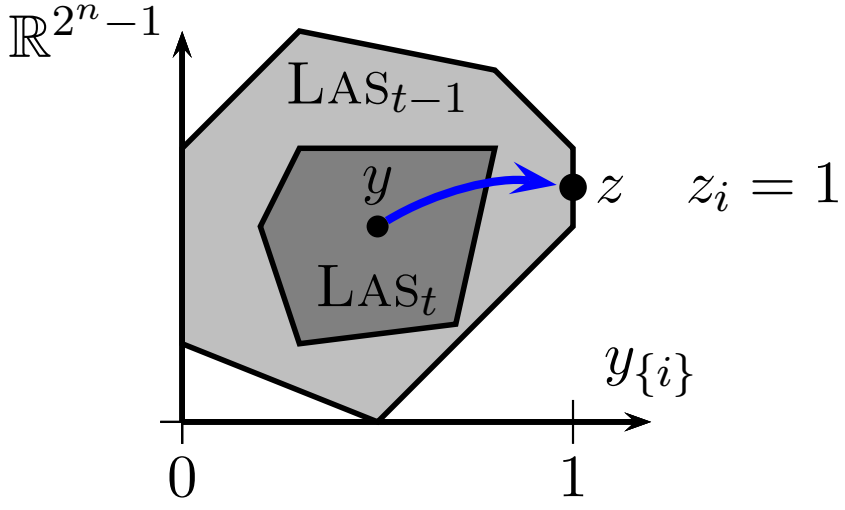
Inducing on one variable

Lemma

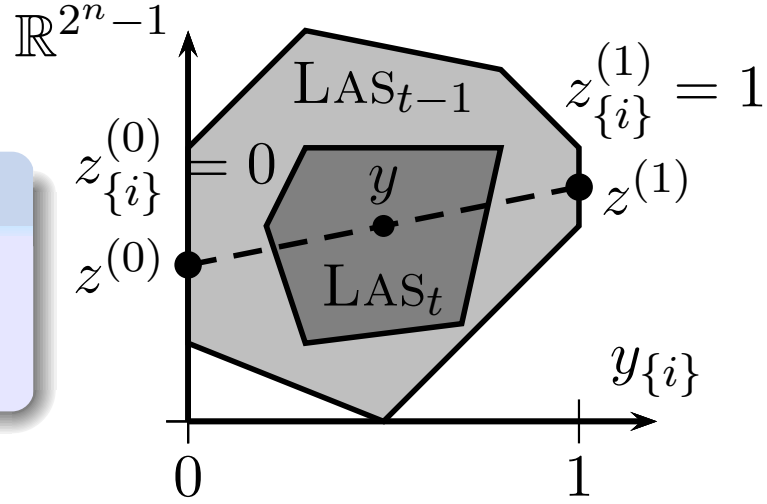
For $y \in \text{LAS}_t(K)$, $t \geq 1$, $i \in [n]$,
$y \in conv\{z \in \text{LAS}_{t-1}(K) \mid z_i \in \{0,1\}\}$.



Operation: "Induce on $x_i = 1$"



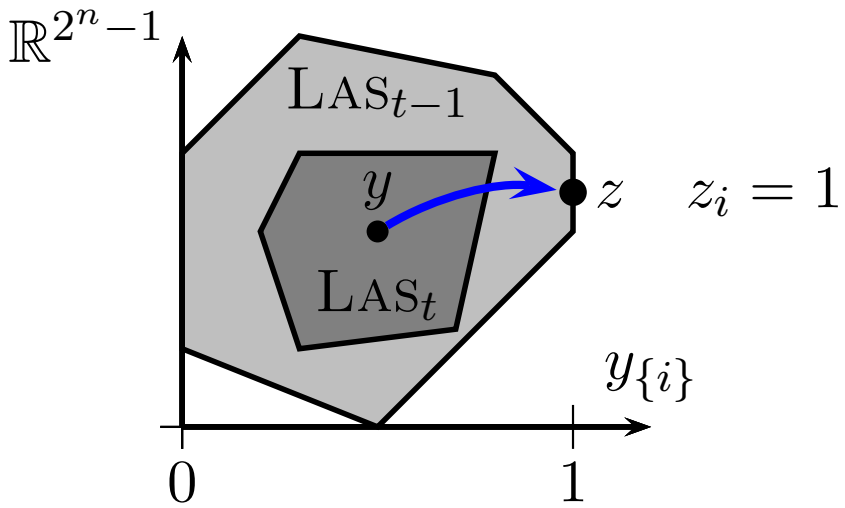
Inducing on one variable



Lemma

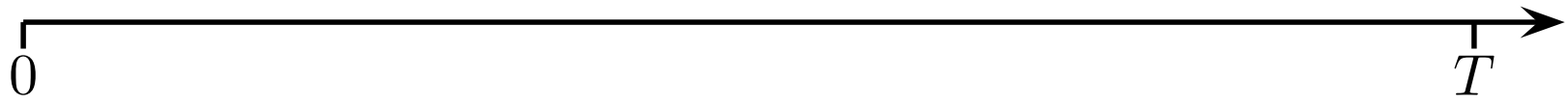
For $y \in \text{LAS}_t(K)$, $t \geq 1$, $i \in [n]$,
$y \in conv\{z \in \text{LAS}_{t-1}(K) \mid z_i \in \{0,1\}\}$.

Operation: "Induce on $x_i = 1$"

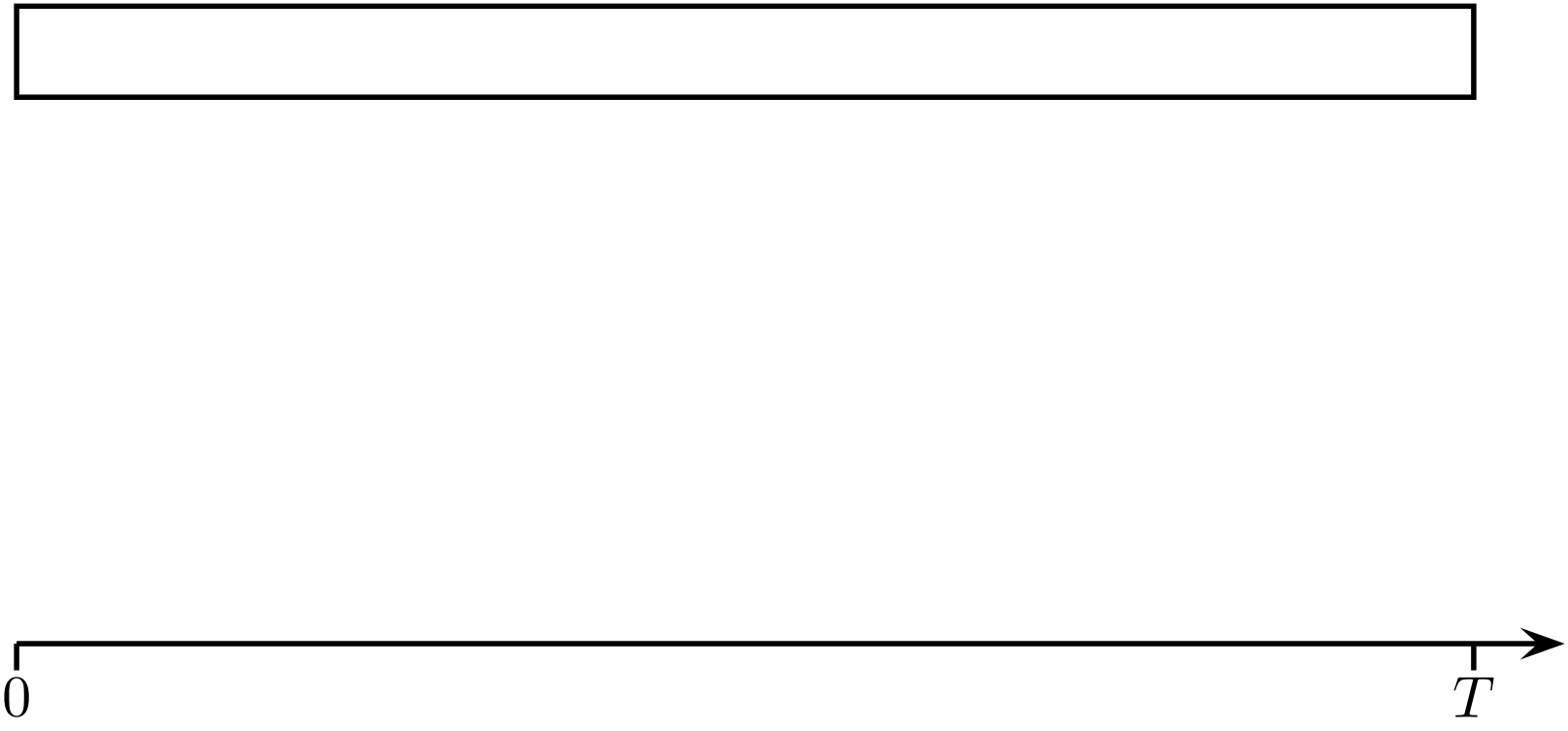


- **Observation:** Conditioning only shrinks support!

Partition

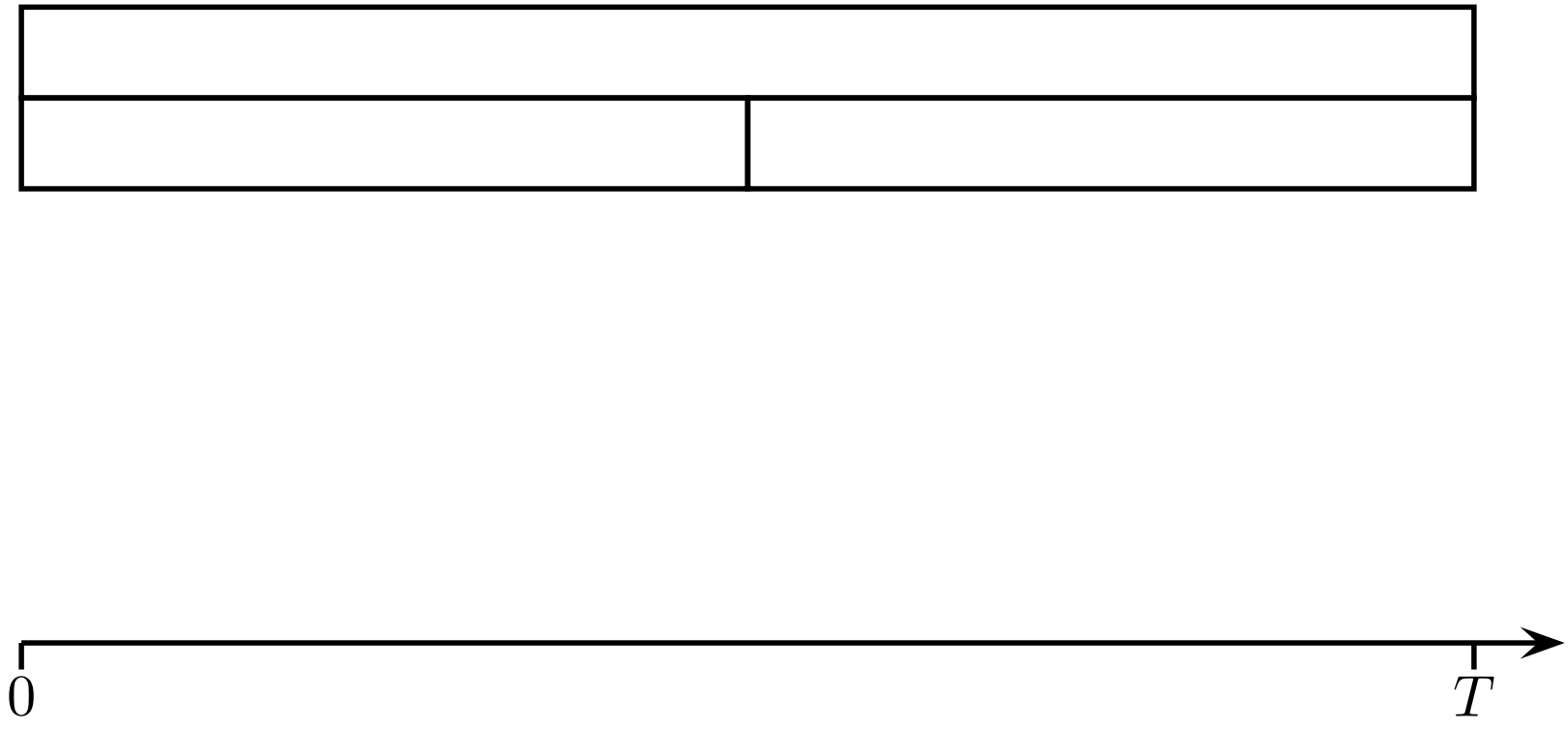


Partition



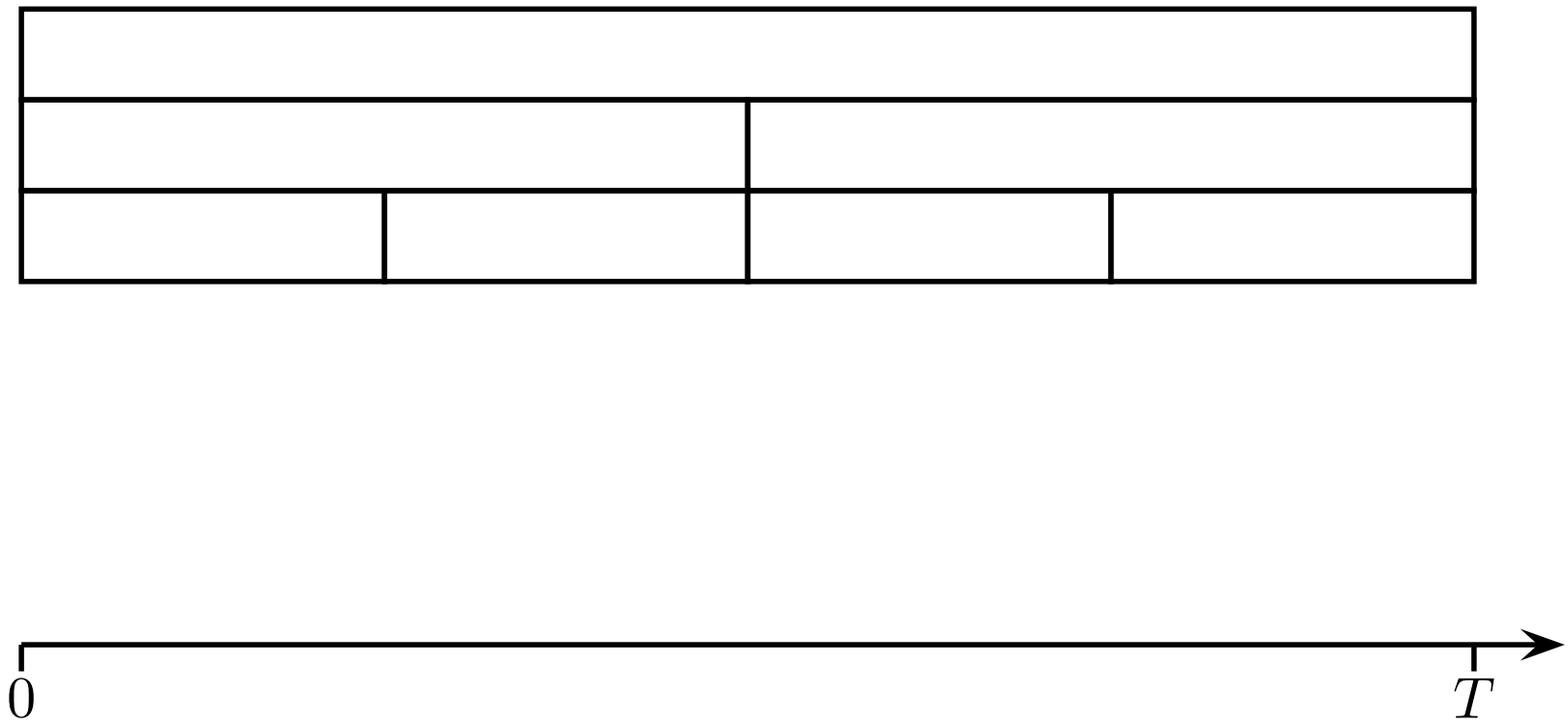
- Partition time horizon into binary family

Partition



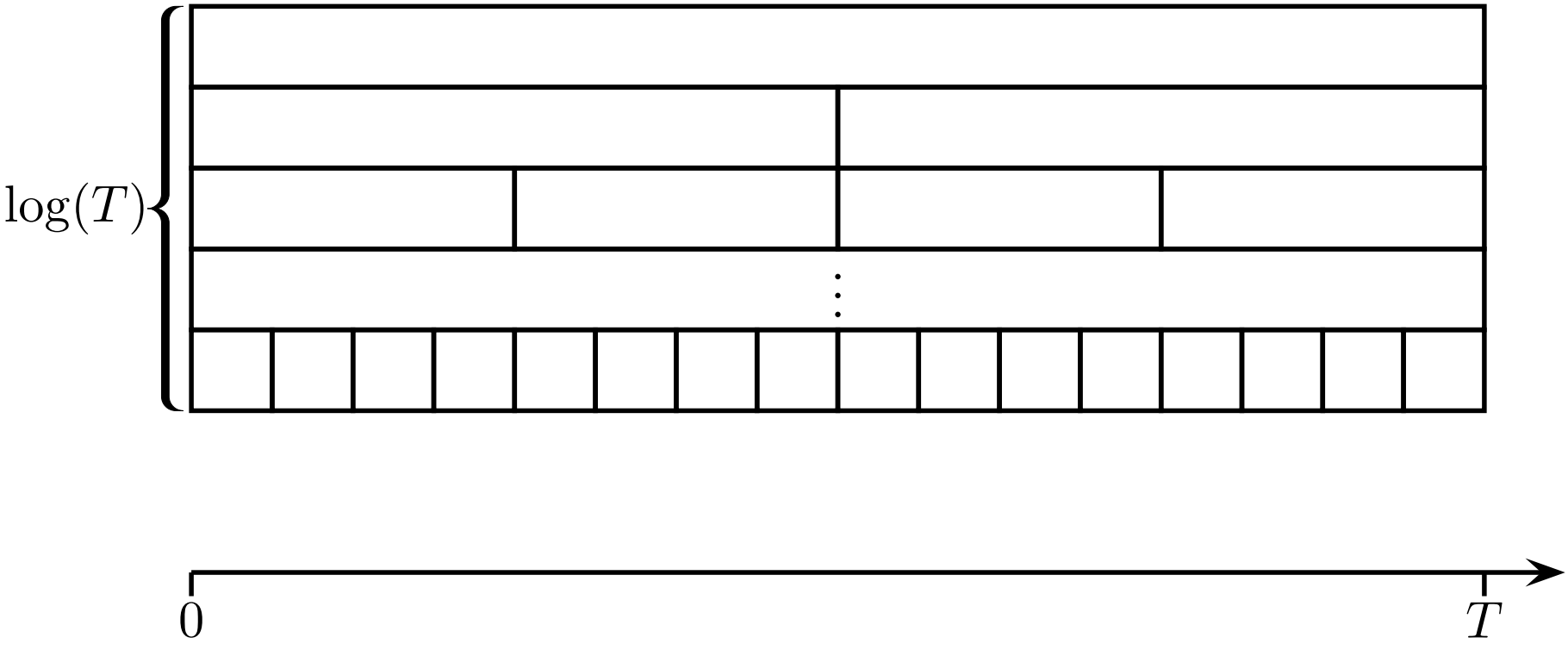
- Partition time horizon into binary family

Partition



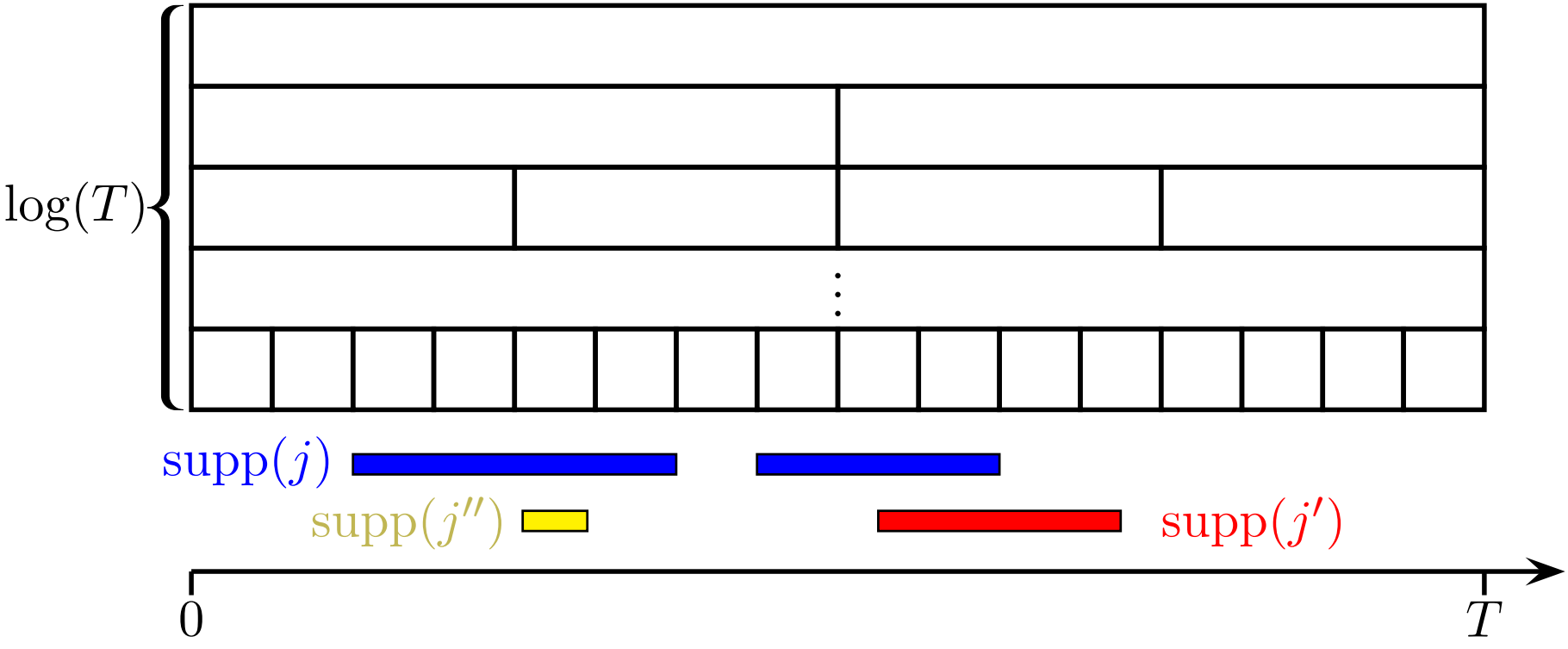
- Partition time horizon into binary family

Partition



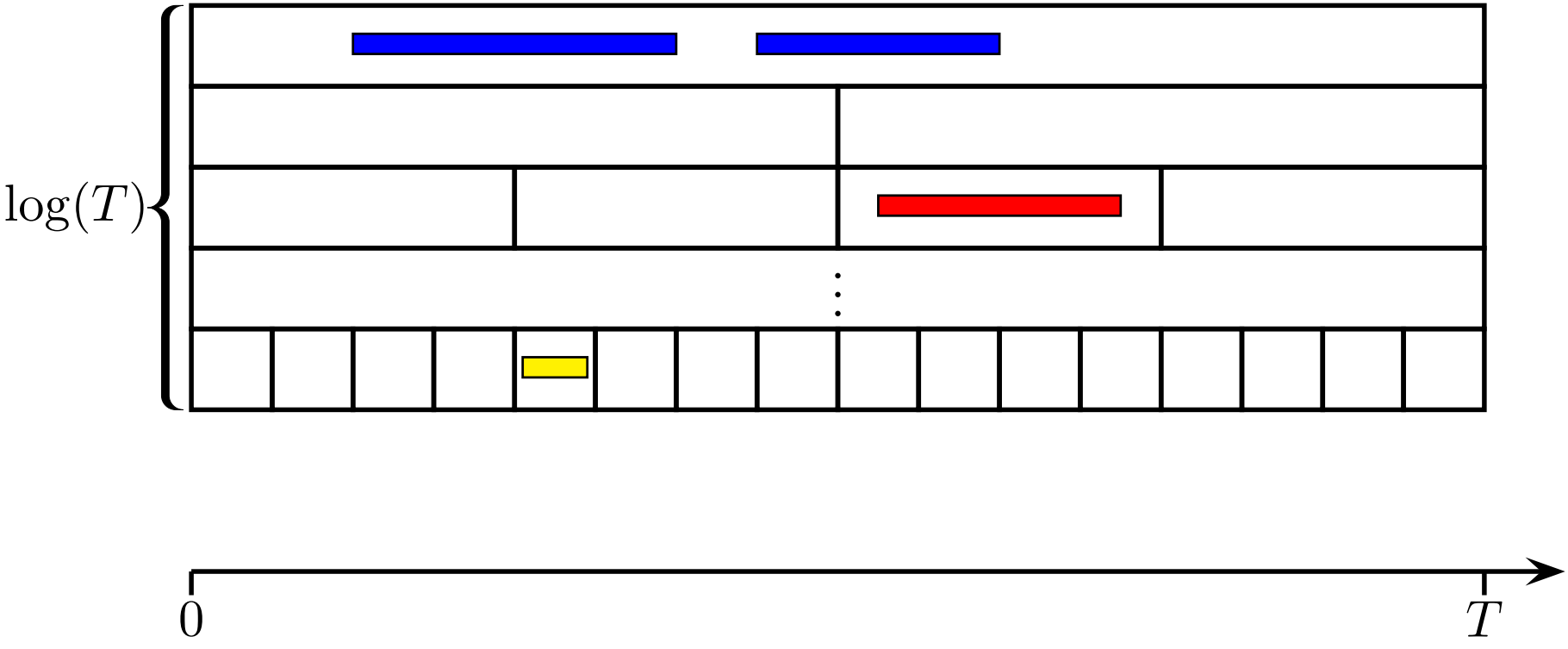
- Partition time horizon into binary family

Partition



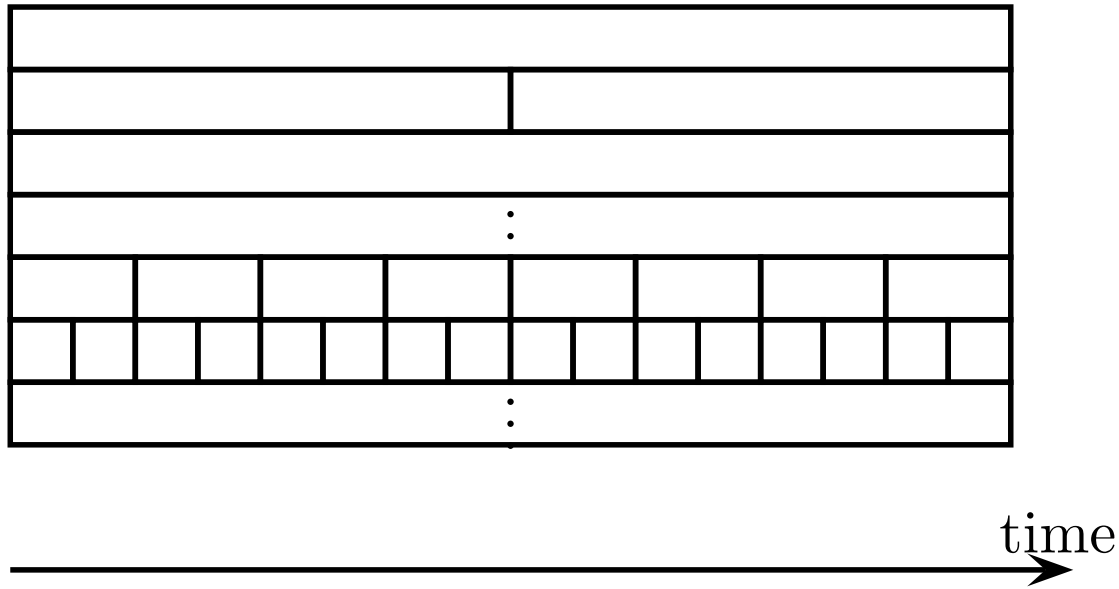
- Partition time horizon into binary family

Partition

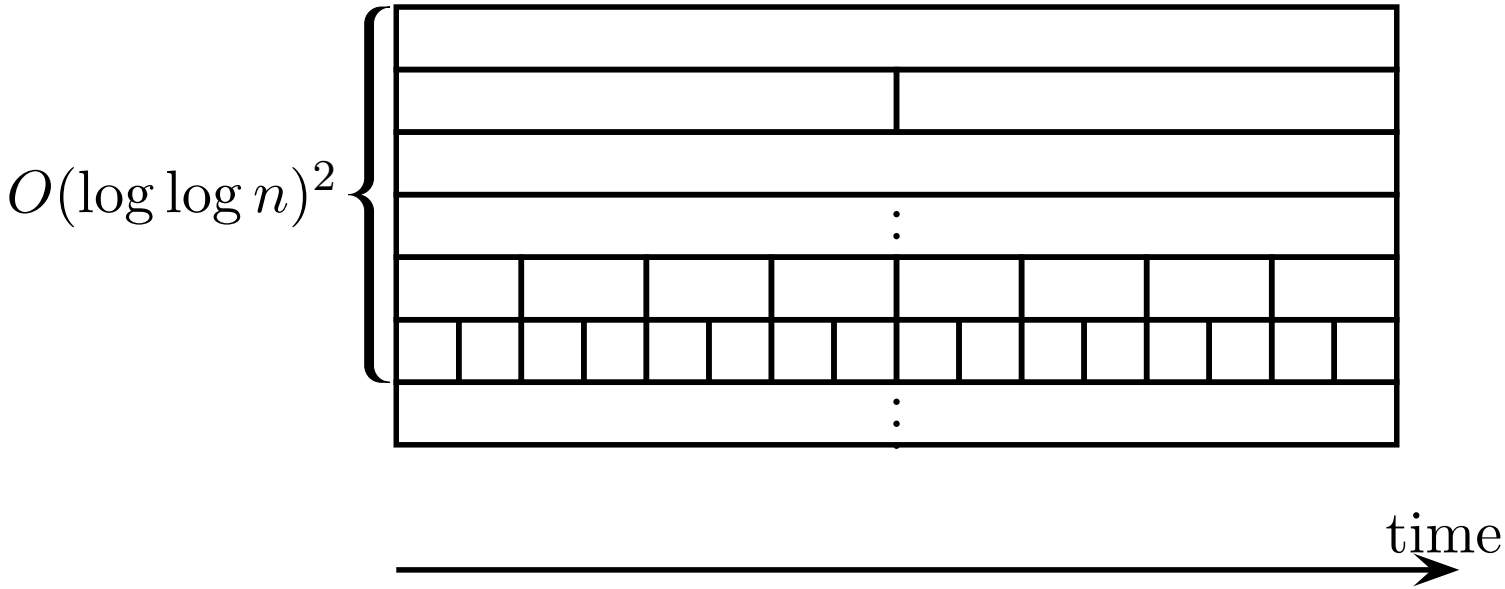


- Partition time horizon into binary family
- Assign jobs to min interval containing support

The algorithm

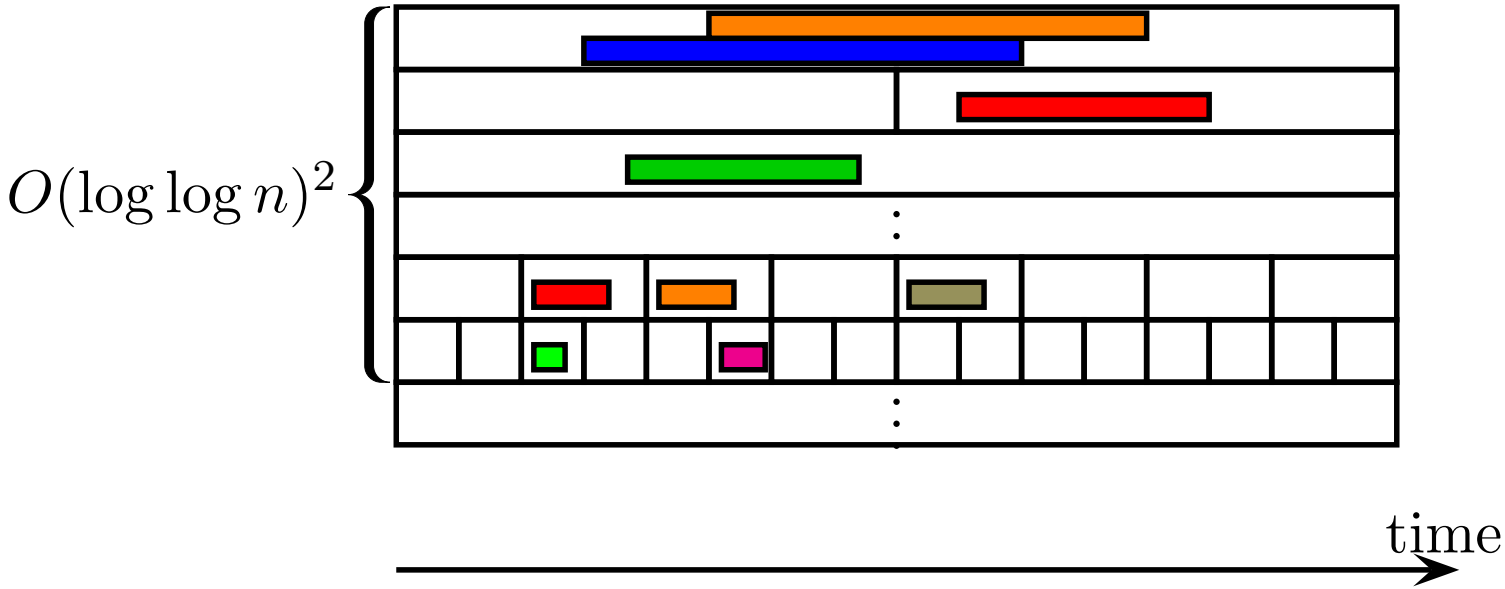


The algorithm



(1) Apply **conditioning** until max chain length of jobs in first $O(\log \log n)^2$ levels down to $\varepsilon' T$

The algorithm



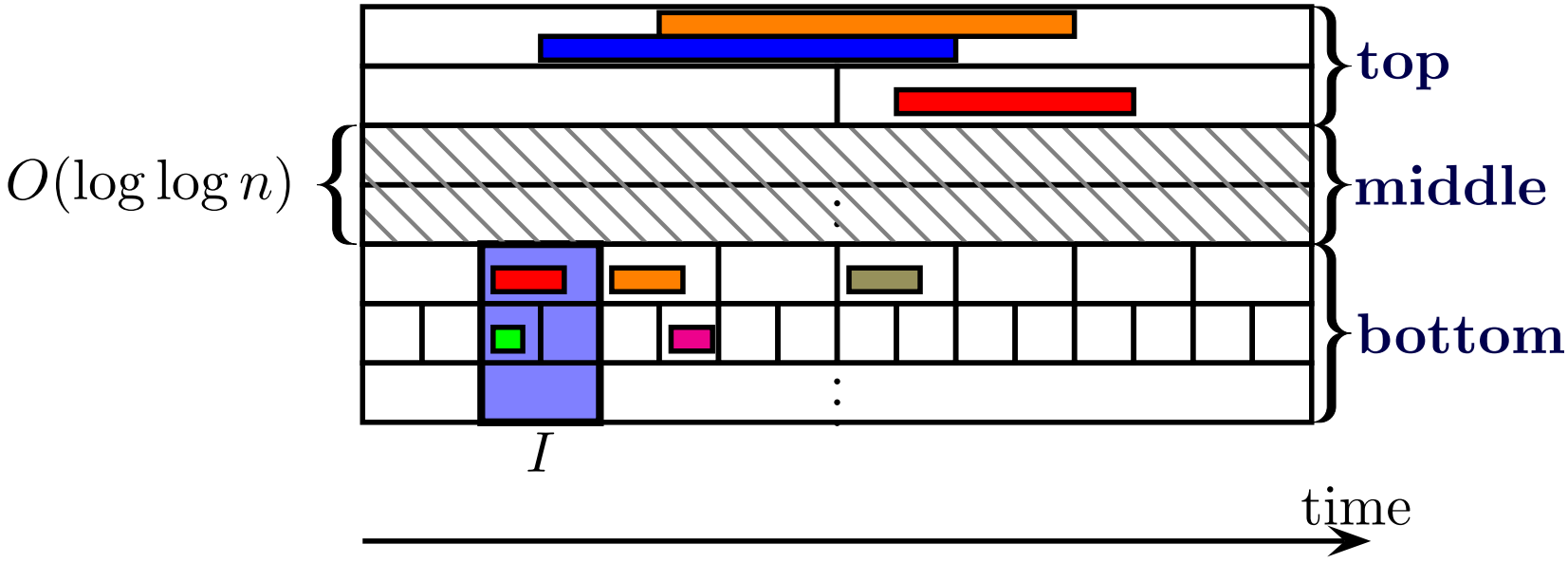
(1) Apply **conditioning** until max chain length of jobs in first $O(\log \log n)^2$ levels down to $\varepsilon' T$

The algorithm

(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

The algorithm

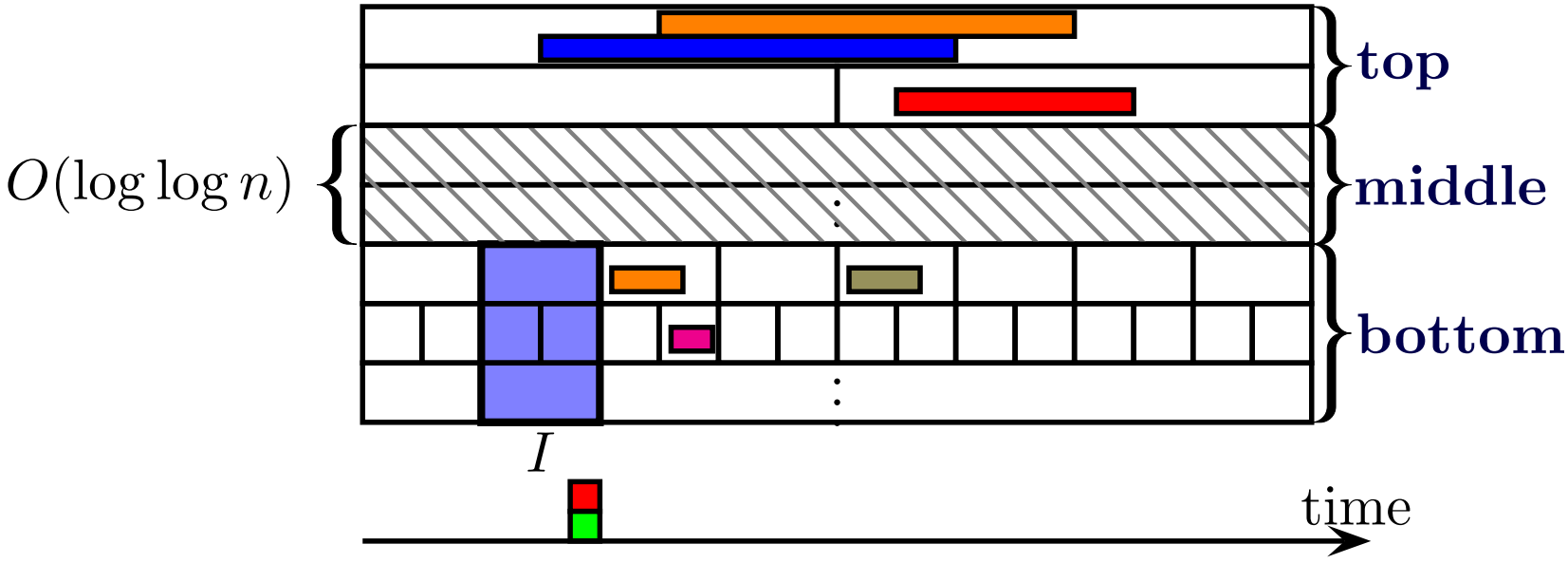


(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

The algorithm

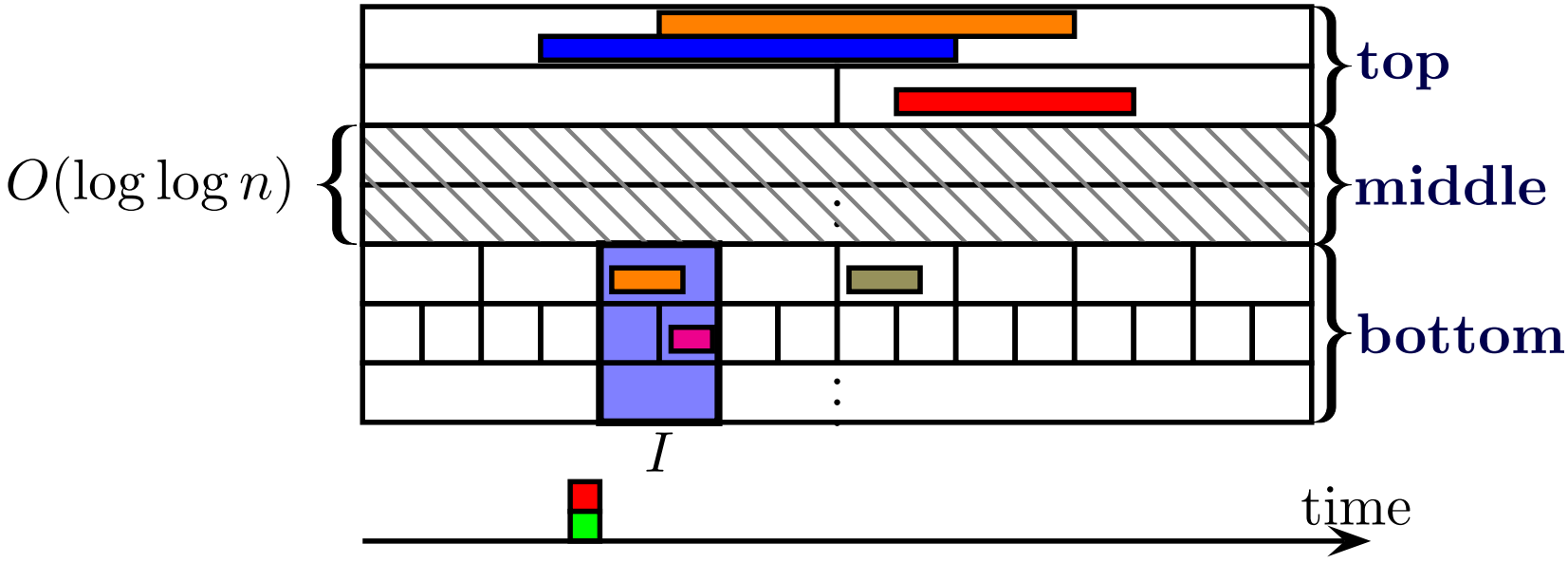


(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

The algorithm

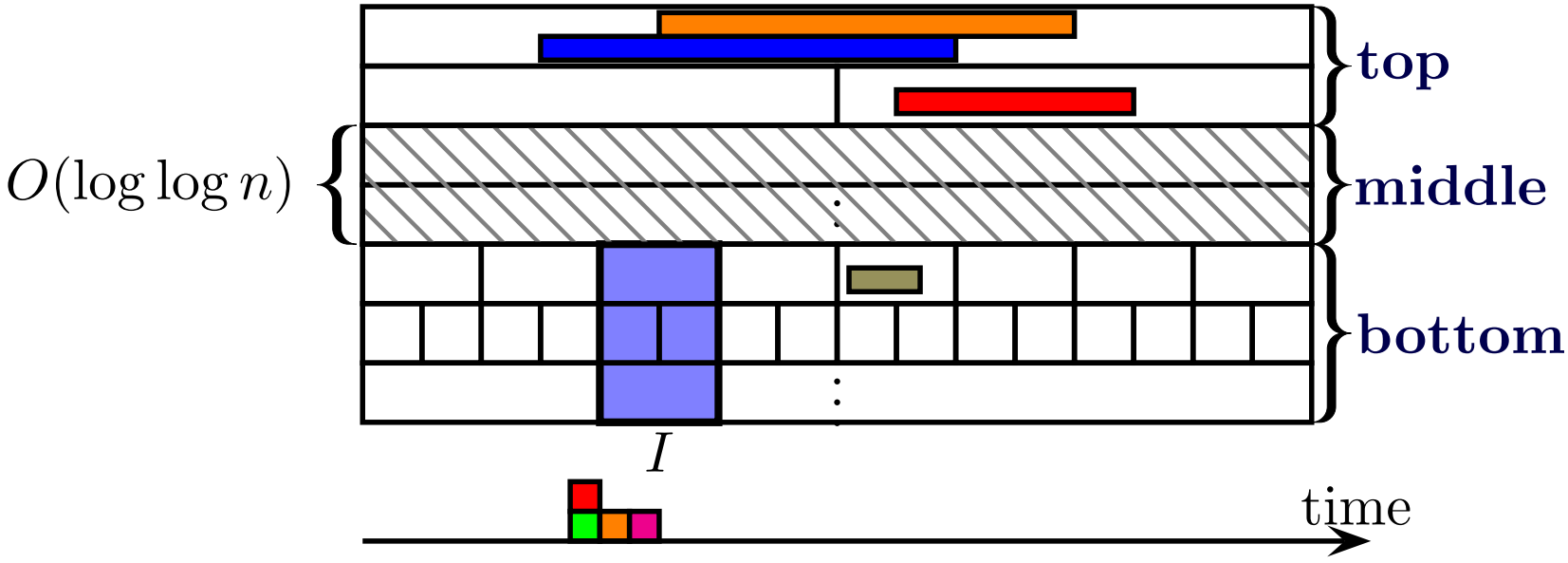


(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

The algorithm

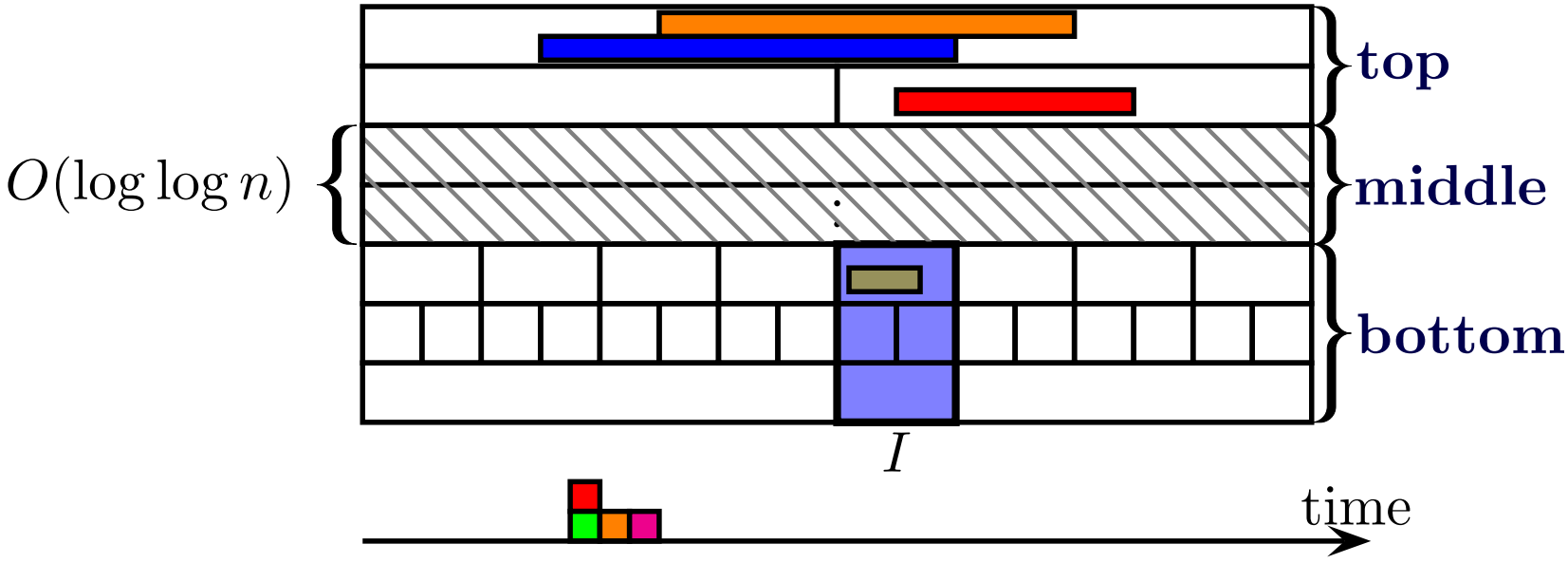


(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

The algorithm

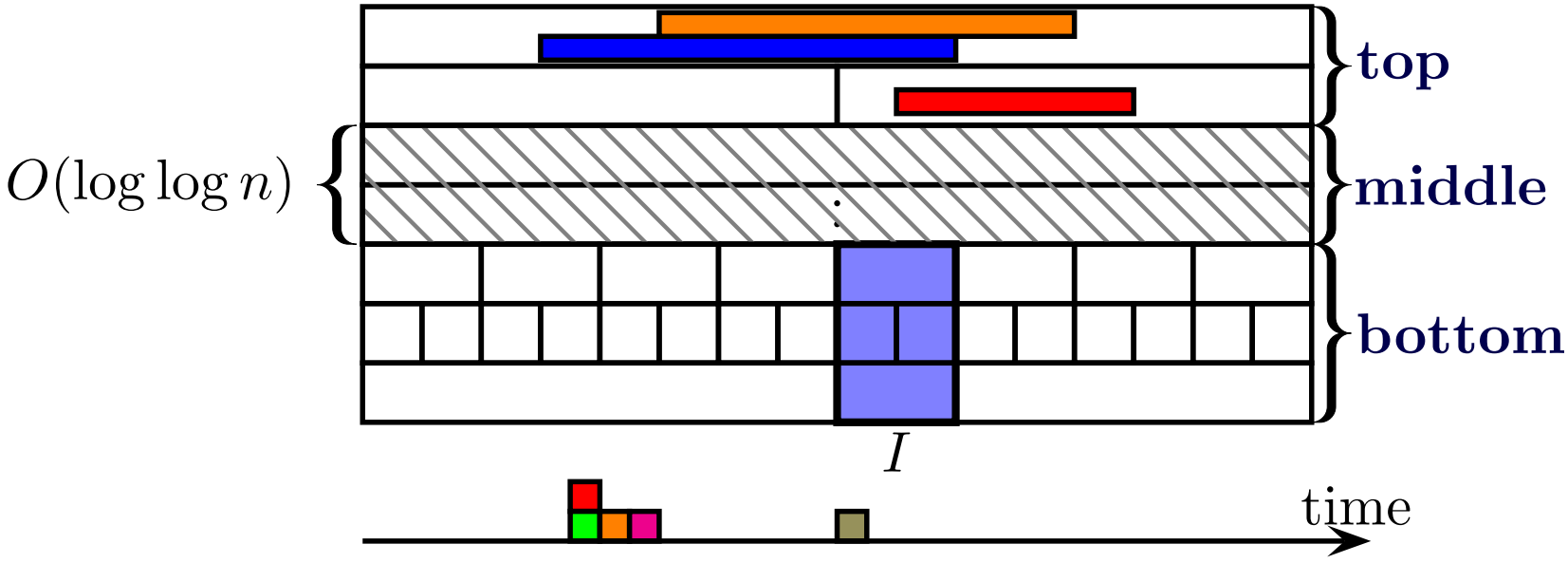


(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

The algorithm

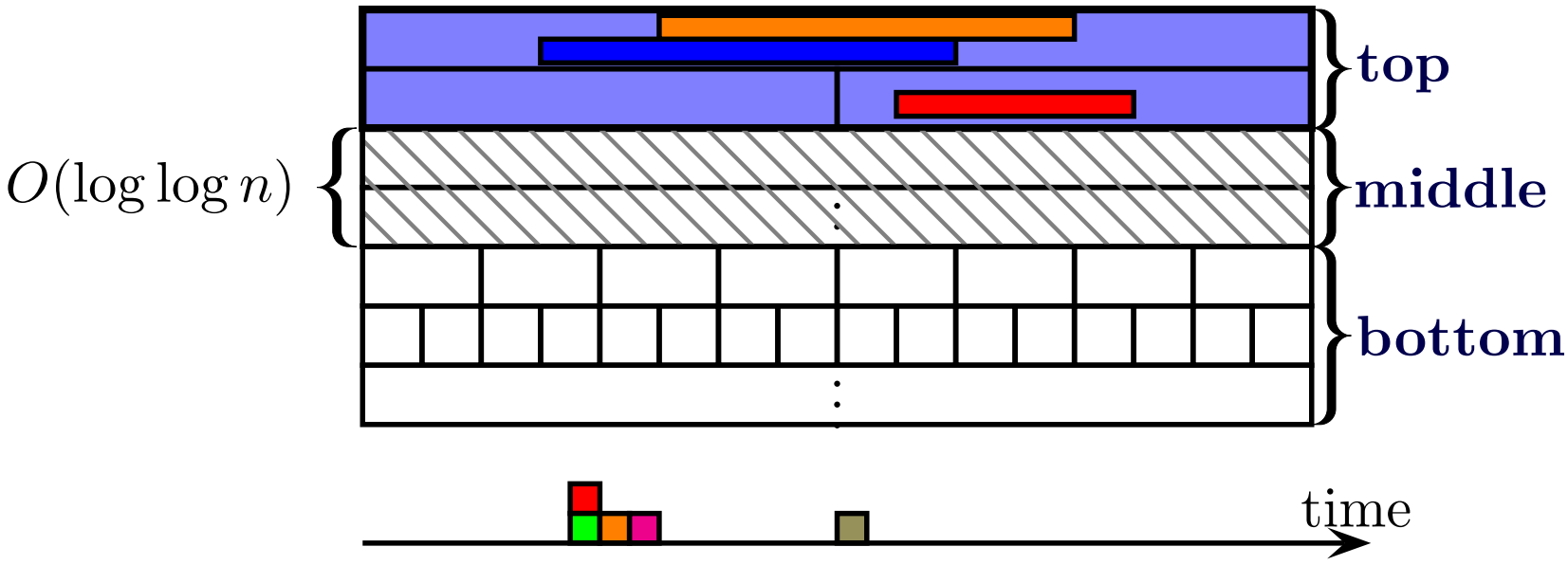


(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

The algorithm



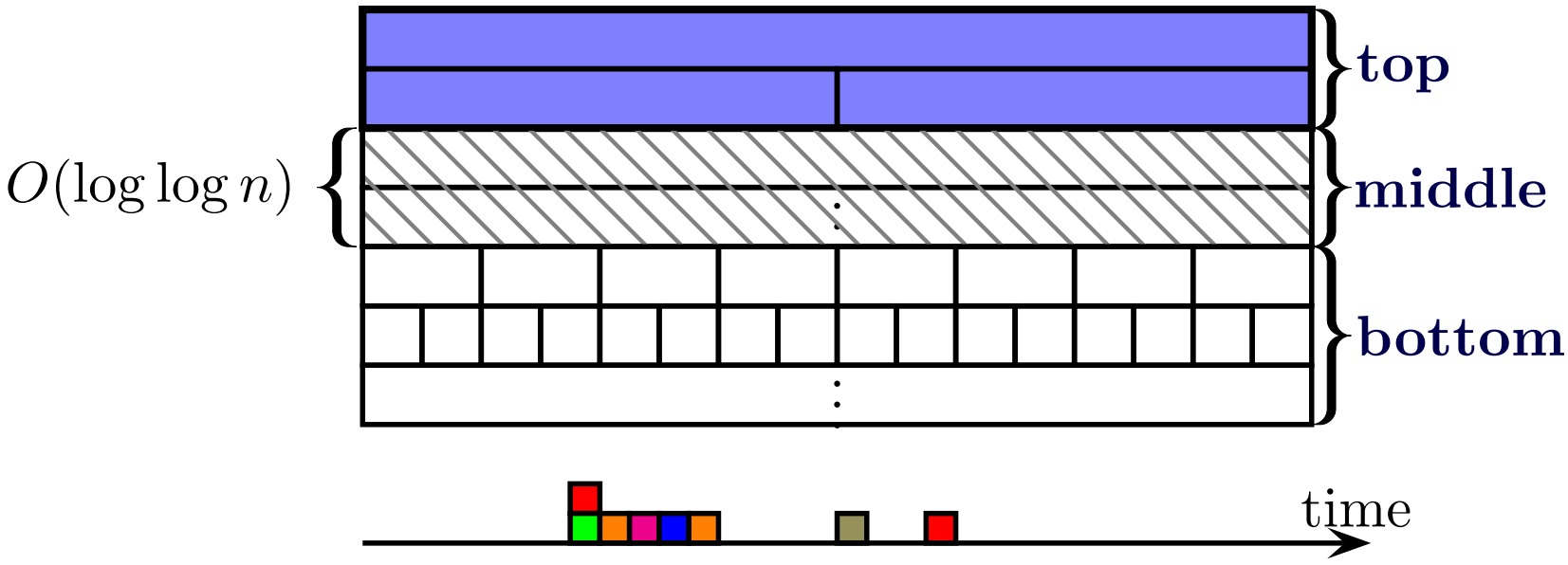
(1) Apply **conditioning** until max chain length of jobs in first $O(\log \log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log \log n)^2$ levels pick $O(\log \log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

(4) Schedule top jobs

The algorithm



(1) Apply **conditioning** until max chain length of jobs in first $O(\log\log n)^2$ levels down to $\varepsilon' T$

(2) Among first $O(\log\log n)^2$ levels pick $O(\log\log n)$ of consecutive **middle levels** and discard jobs

(3) For each interval I directly below middle levels: Recursively call algorithm I with $\{j \mid \text{supp}(j) \subseteq I\}$
$\rightarrow$ valid schedule for **bottoms jobs**

(4) Schedule top jobs

Reducing chain length

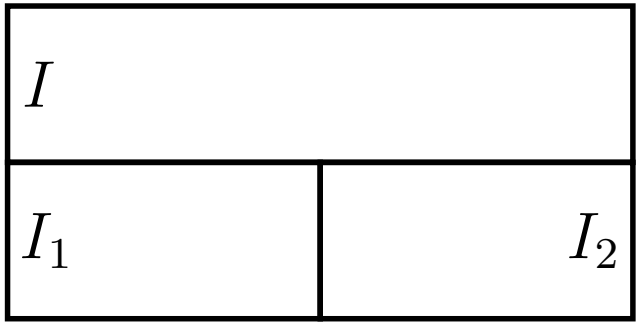
Lemma

After conditioning $2^{O(\log \log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.

Reducing chain length

Lemma

After conditioning $2^{O(\log \log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.

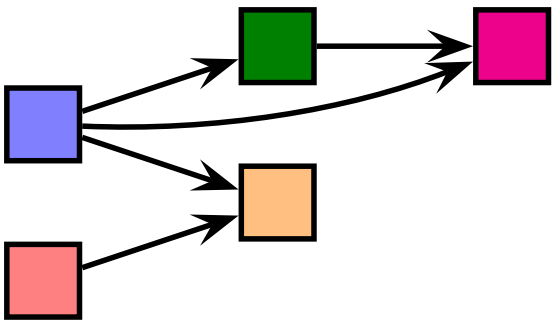


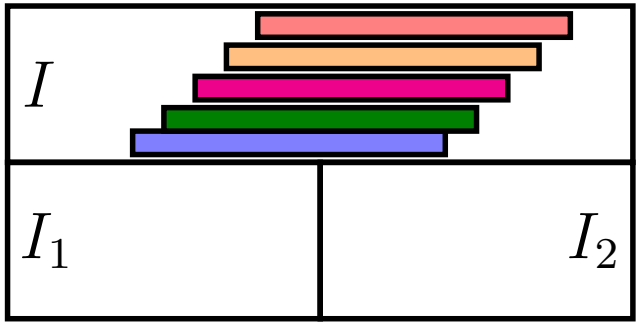
- Pick an interval I

Reducing chain length

Lemma

After conditioning $2^{O(\log \log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



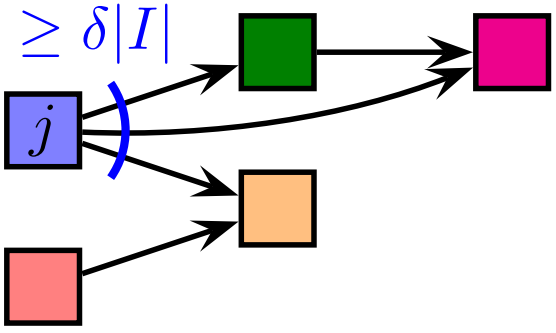


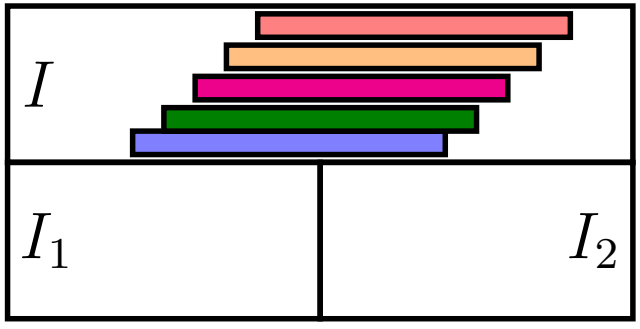
- Pick an interval I and consider jobs assigned to it.

Reducing chain length

Lemma

After conditioning $2^{O(\log \log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



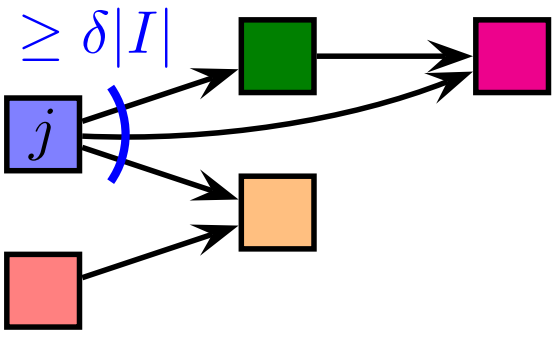


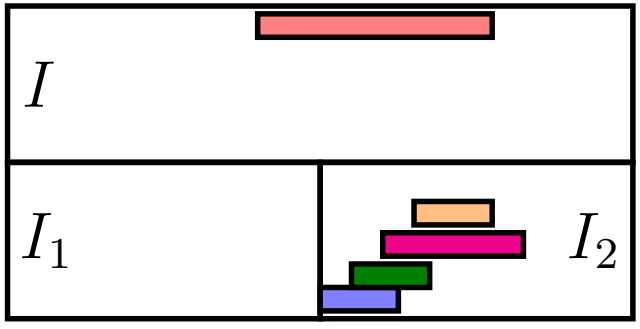
- Pick an interval I and consider jobs assigned to it.
- Consider j with $\delta|I|$ successors

Reducing chain length

Lemma

After conditioning $2^{O(\log\log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



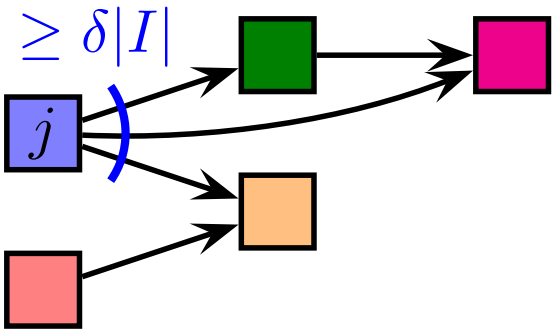


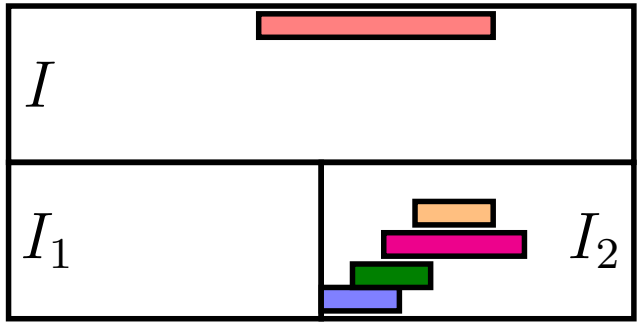
- Pick an interval I and consider jobs assigned to it.
- Consider j with $\delta|I|$ successors $\Rightarrow$ condition on $x_{j,I_2} = 1$

Reducing chain length

Lemma

After conditioning $2^{O(\log\log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



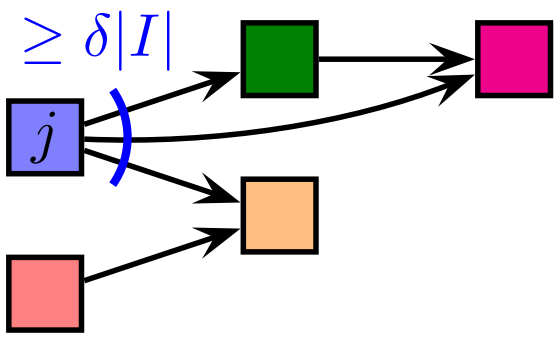


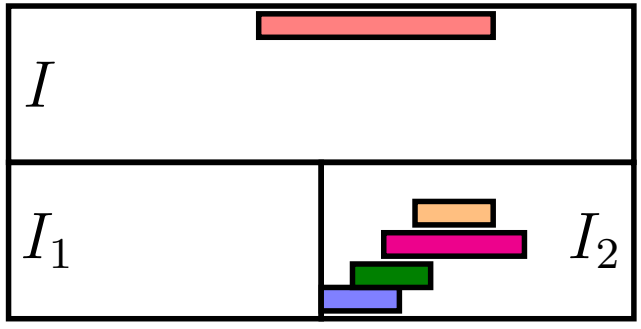
- Pick an interval I and consider jobs assigned to it.
- Consider j with $\delta|I|$ successors $\Rightarrow$ condition on $x_{j,I_2} = 1$
- Needed $\leq \frac{1}{\delta}$ times per interval

Reducing chain length

Lemma

After conditioning $2^{O(\log\log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



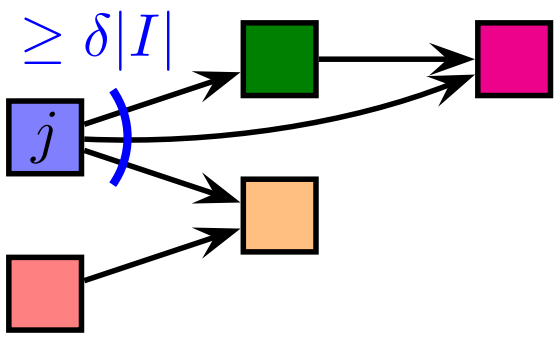


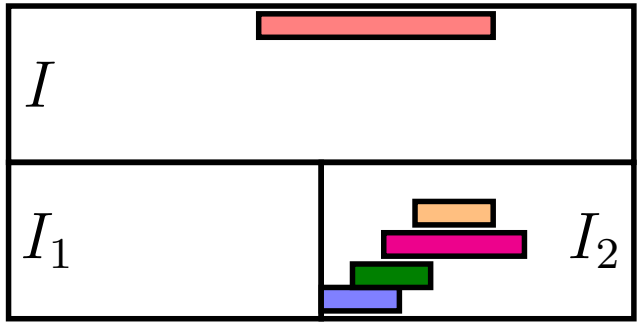
- Pick an interval I and consider jobs assigned to it.
- Consider j with $\delta|I|$ successors $\Rightarrow$ condition on $x_{j,I_2} = 1$
- Needed $\leq \frac{1}{\delta}$ times per interval; $2^{O(\log\log n)^2}$ intervals

Reducing chain length

Lemma

After conditioning $2^{O(\log\log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



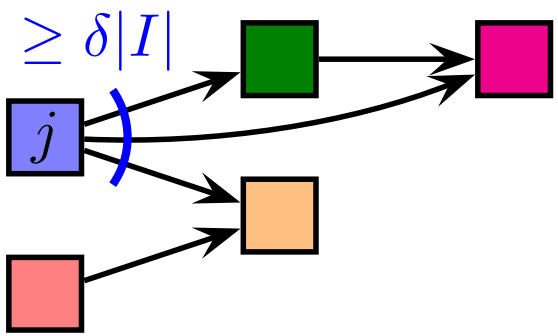


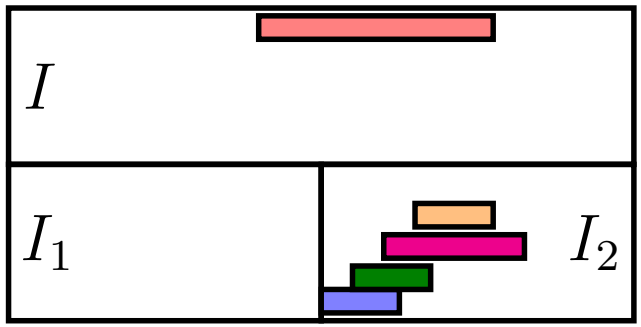
- Pick an interval I and consider jobs assigned to it.
- Consider j with $\delta|I|$ successors $\Rightarrow$ condition on $x_{j,I_2} = 1$
- Needed $\leq \frac{1}{\delta}$ times per interval; $2^{O(\log\log n)^2}$ intervals
- Loose $\leq \frac{1}{\delta} \cdot 2^{O(\log\log n)^2}$ Lasserre-rounds

Reducing chain length

Lemma

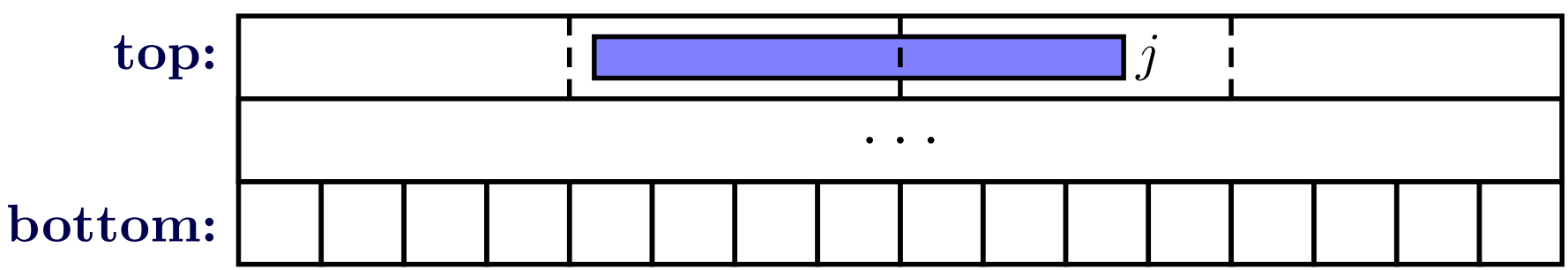
After conditioning $2^{O(\log\log n)^2} \cdot \frac{1}{\varepsilon'}$ times, **maximum length chain** among top jobs is $\leq \varepsilon' \cdot T$.



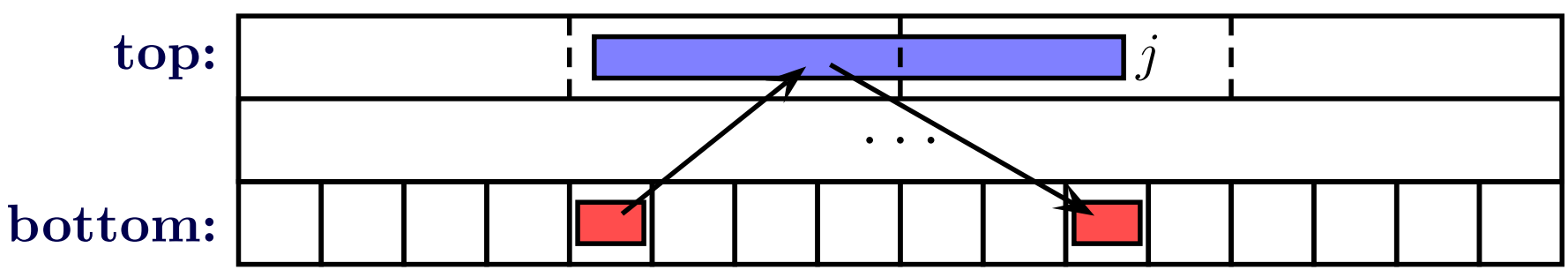


- Pick an interval I and consider jobs assigned to it.
- Consider j with $\delta|I|$ successors $\Rightarrow$ condition on $x_{j,I_2} = 1$
- Needed $\leq \frac{1}{\delta}$ times per interval; $2^{O(\log\log n)^2}$ intervals
- Loose $\leq \frac{1}{\delta} \cdot 2^{O(\log\log n)^2}$ Lasserre-rounds
- Longest chain in top jobs: $\delta T \cdot O(\log\log n)^2$ □

Scheduling top jobs (I)

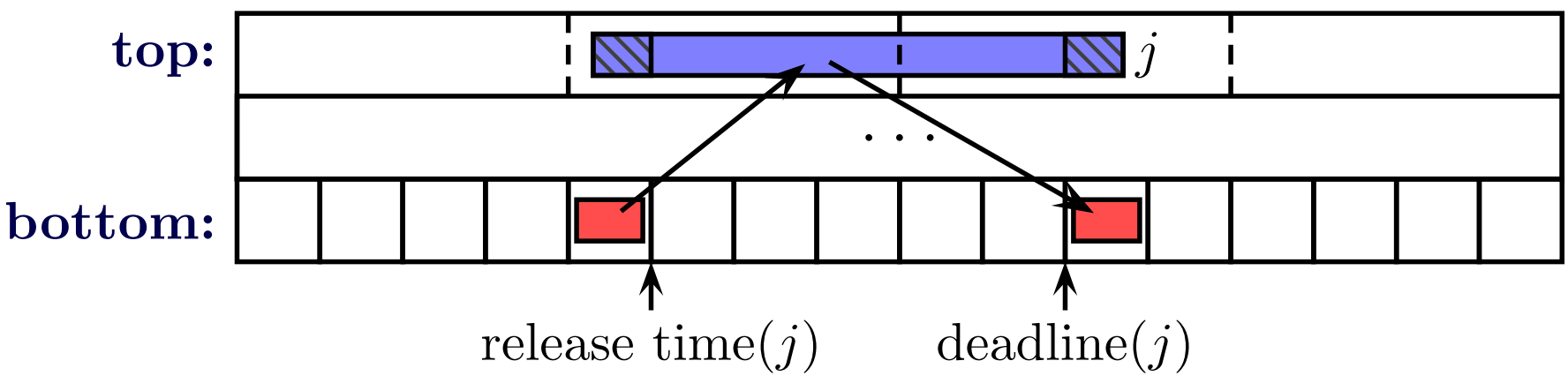


Scheduling top jobs (I)



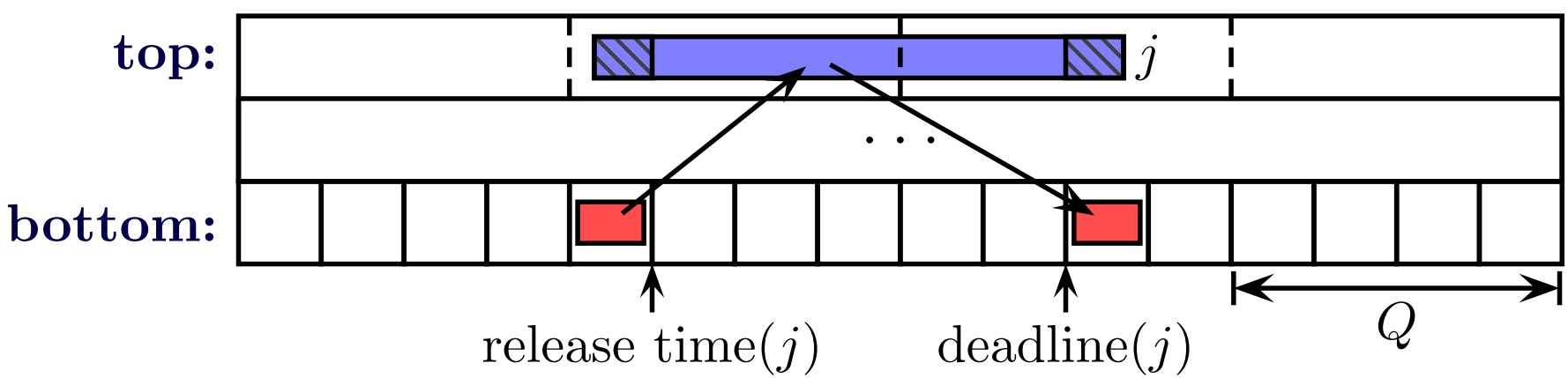
Scheduling top jobs (I)

- For top job j set
  $[\text{release time}(j), \text{deadline}(j)] :=$ fully contained bottom intervals



Scheduling top jobs (I)

- For top job j set
  $[\text{release time}(j), \text{deadline}(j)]$:= fully contained bottom intervals
- Set $Q := \text{polylog}(n)$ as top/bottom gap

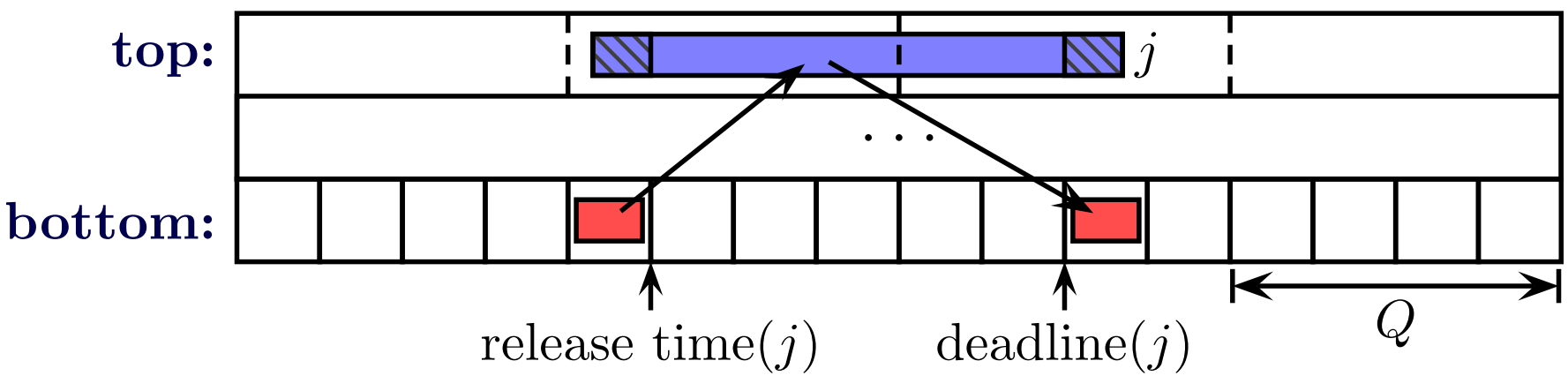


Scheduling top jobs (I)

Lemma

Can assign top jobs respecting release times/deadlines, ignoring prec. constraints & discarding $\frac{2Tm}{Q}$ jobs.

- For top job j set
  [release time(j), deadline(j)] := fully contained bottom intervals
- Set $Q :=$ polylog(n) as top/bottom gap

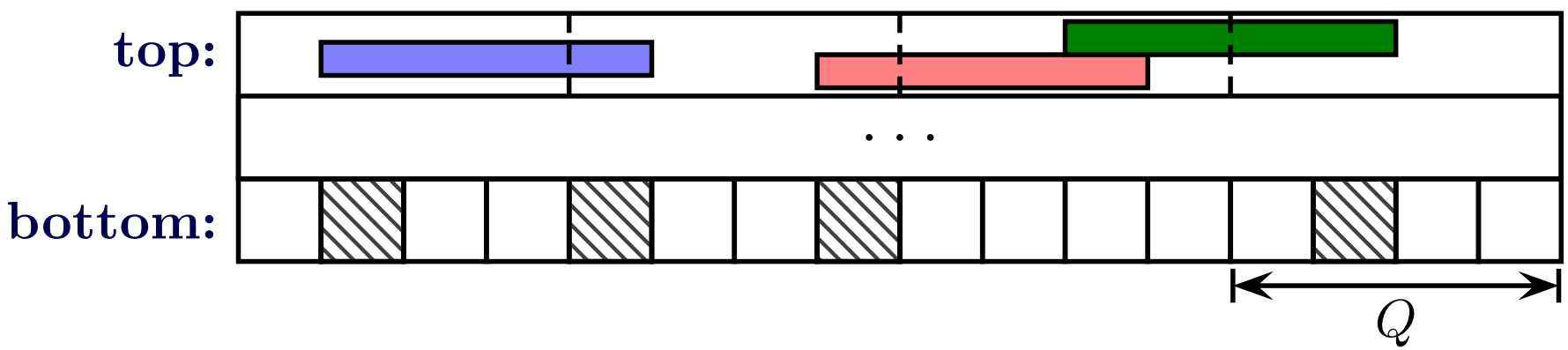


Scheduling top jobs (I)

Lemma

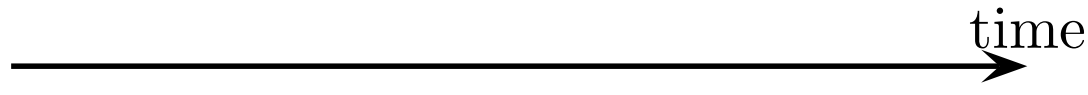
Can assign top jobs respecting release times/deadlines, ignoring prec. constraints & discarding $\frac{2Tm}{Q}$ jobs.

- For top job j set
  [release time(j), deadline(j)] := fully contained bottom intervals
- Set $Q :=$ polylog(n) as top/bottom gap
- **Hall's Theorem:** Loss for jobs J' is $\frac{2}{Q}$-fraction of slots



EDF (1)

Setting:



EDF (1)

Setting:

- Jobs with precedence constraints

EDF (1)

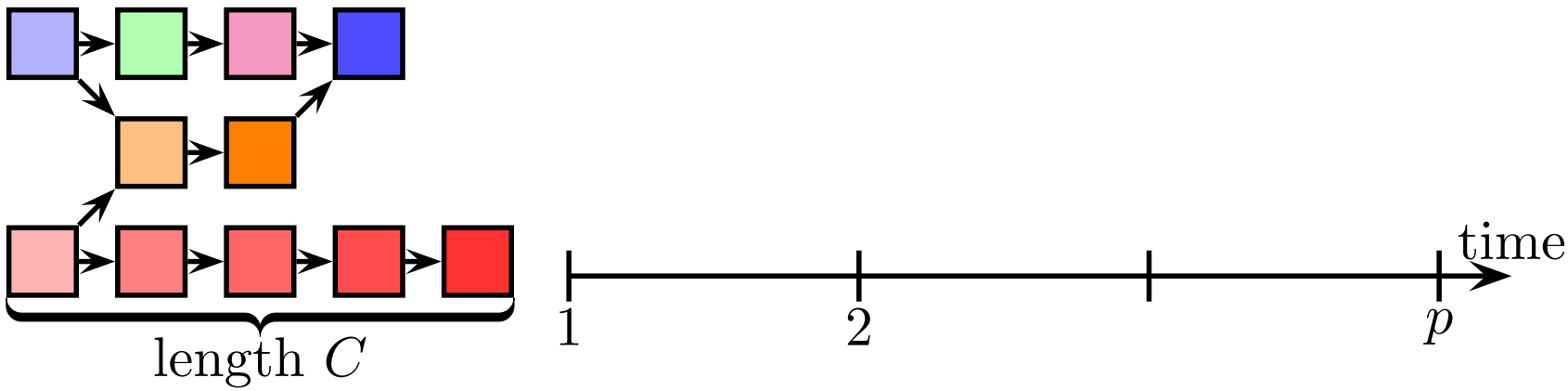
Setting:

- Jobs with precedence constraints
- Maximum chain length C

EDF (1)

Setting:

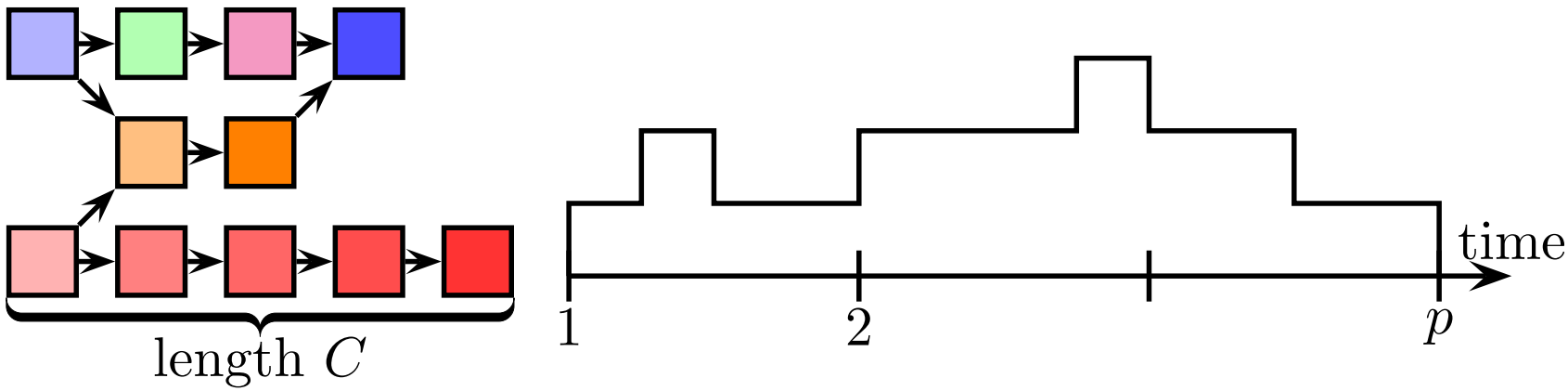
- Jobs with precedence constraints
- Maximum chain length C
- Release times, deadlines only at beginning/end of p **blocks**



EDF (1)

Setting:

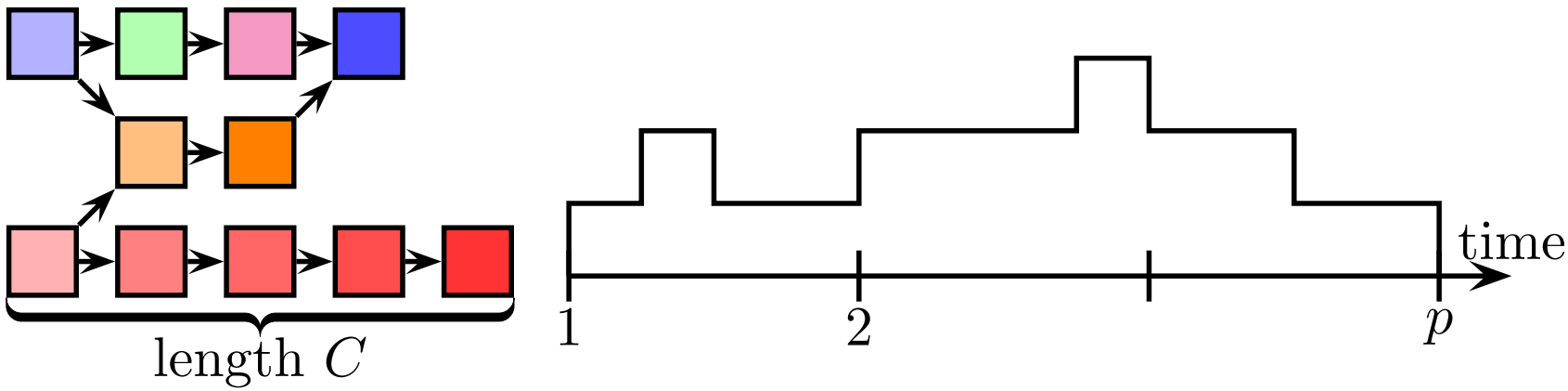
- Jobs with precedence constraints
- Maximum chain length C
- Release times, deadlines only at beginning/end of p **blocks**
- Capacity $\text{cap}(t) \in \{0, \ldots, m\}$



EDF (1)

Setting:

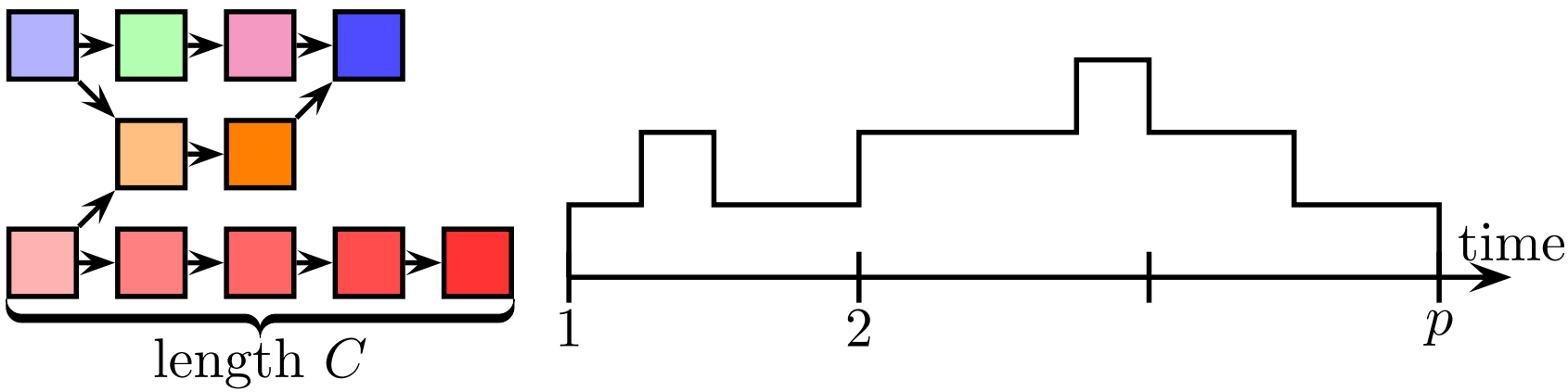
- Jobs with precedence constraints
- Maximum chain length C
- Release times, deadlines only at beginning/end of p **blocks**
- Capacity $\text{cap}(t) \in \{0, \ldots, m\}$
- There exists a schedule ignoring precedence constraints



EDF (1)

Setting:

- Jobs with precedence constraints
- Maximum chain length C
- Release times, deadlines only at beginning/end of p **blocks**
- Capacity $\text{cap}(t) \in \{0, \ldots, m\}$
- There exists a schedule ignoring precedence constraints



Earliest Deadline First:

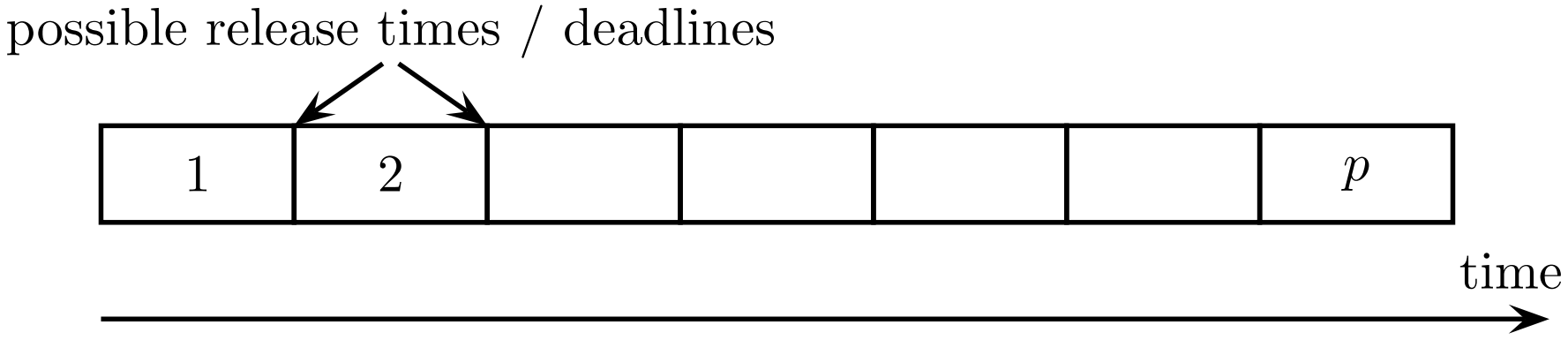
(1) FOR $t = 1$ TO T DO

(2) Process available job with minimum deadline

(3) If any job not done by deadline discard it

EDF (2)

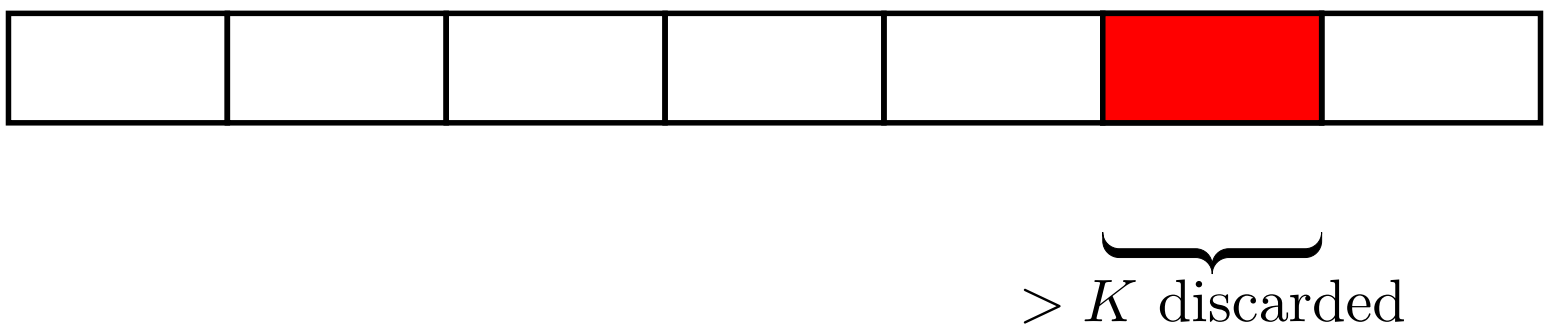
Analysis:



EDF (2)

Analysis:

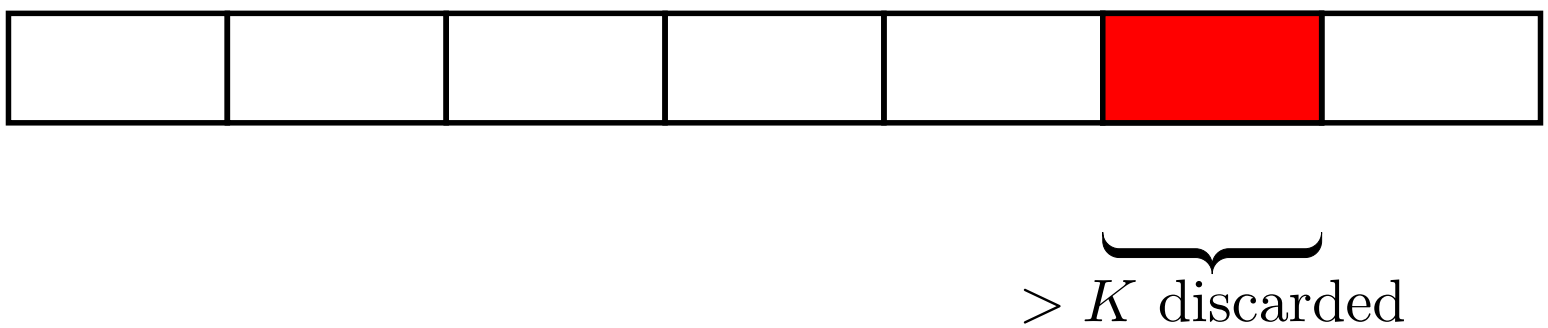
- Suppose in some interval I we discard K jobs



EDF (2)

Analysis:

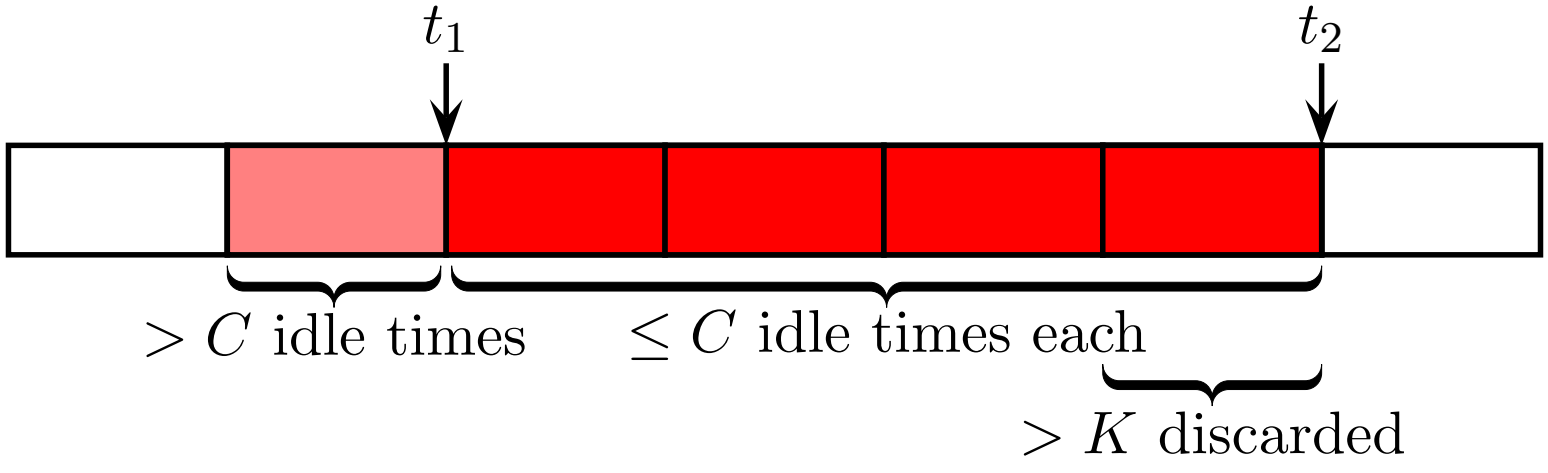
- Suppose in some interval I we discard K jobs
- Assume w.l.o.g. last discarded job had highest deadline



EDF (2)

Analysis:

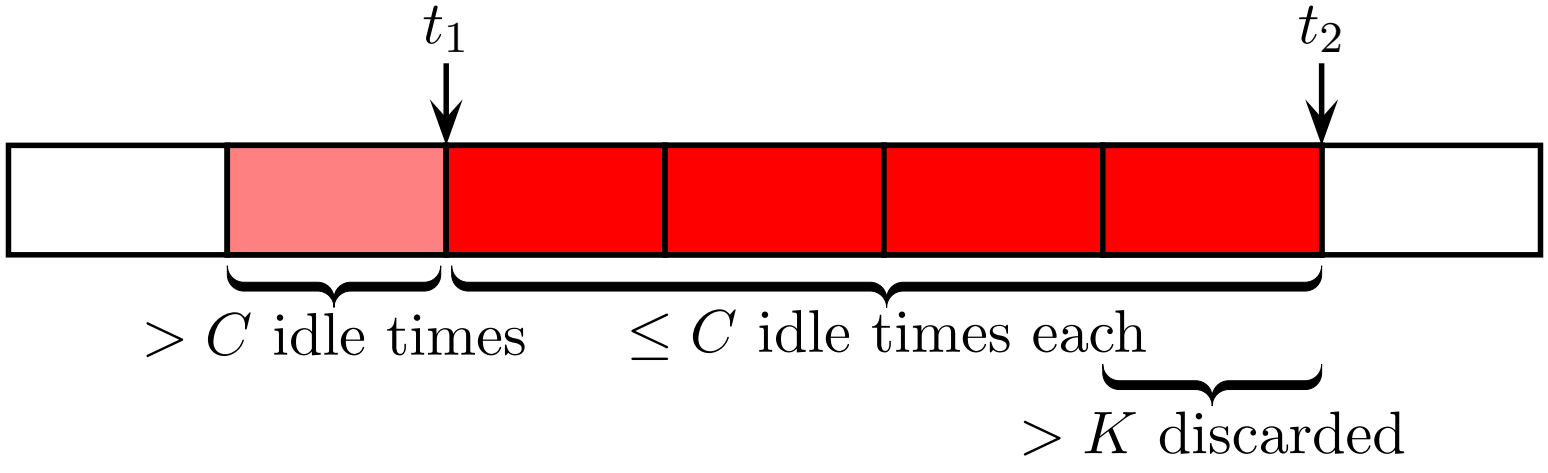
- Suppose in some interval I we discard K jobs
- Assume w.l.o.g. last discarded job had highest deadline
- Take maximum interval $[t_1, t_2] \supseteq I$ so that in each block, there are $\leq C$ idle times



EDF (2)

Analysis:

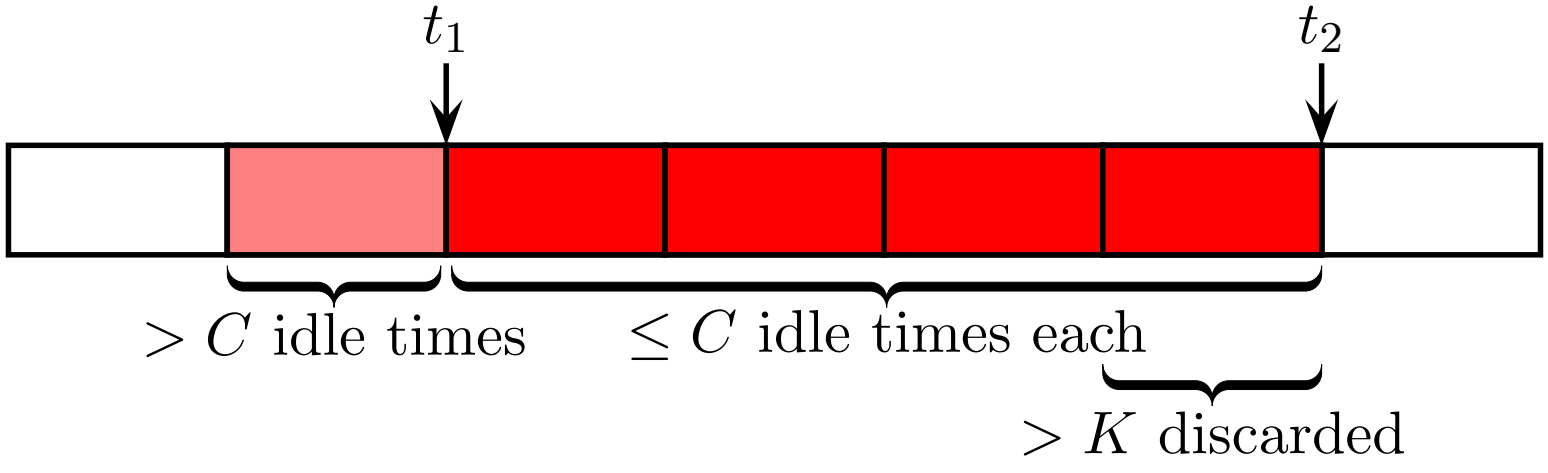
- Suppose in some interval I we discard K jobs
- Assume w.l.o.g. last discarded job had highest deadline
- Take maximum interval $[t_1, t_2] \supseteq I$ so that in each block, there are $\leq C$ idle times
- All jobs scheduled or discarded in $[t_1, t_2]$ have both release + deadline in $[t_1, t_2]$.



EDF (2)

Analysis:

- Suppose in some interval I we discard K jobs
- Assume w.l.o.g. last discarded job had highest deadline
- Take maximum interval $[t_1, t_2] \supseteq I$ so that in each block, there are $\leq C$ idle times
- All jobs scheduled or discarded in $[t_1, t_2]$ have both release + deadline in $[t_1, t_2]$.
- $K \leq$ idle slots in $[t_1, t_2] \leq pmC$ (feasibility w/o prec.-constr.)



The end

Open problems:

- Are $c(\varepsilon, m)$-Lasserre rounds enough?
- .. or at least $(\log(n))^{c(\varepsilon,m)}$
- What about $Pm \mid \text{prec} \mid C_{\max}$ (i.e. arbitrary running times)?

The end

Open problems:

- Are $c(\varepsilon, m)$-Lasserre rounds enough?
- .. or at least $(\log(n))^{c(\varepsilon,m)}$
- What about $Pm \mid \text{prec} \mid C_{\max}$ (i.e. arbitrary running times)?

Thanks for your attention