Combinatorial Perpetual Scheduling

Kevin Schewior (University of Cologne)

joint work with Mirabel Mendoza, Arturo Merino, and Mads Anker Nielsen

schedulingseminar.com
March 18, 2026

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Bamboo Garden Trimming
or: Perpetual Maintenance Scheduling

Model: [Gasieniec et al., SOFSEM'17]

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.

Bamboo Garden Trimming
or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.

$n = 3$

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

$n = 3$

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

$n = 3$

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM’17]

$n = 3$

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).

$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

Bamboo Garden Trimming

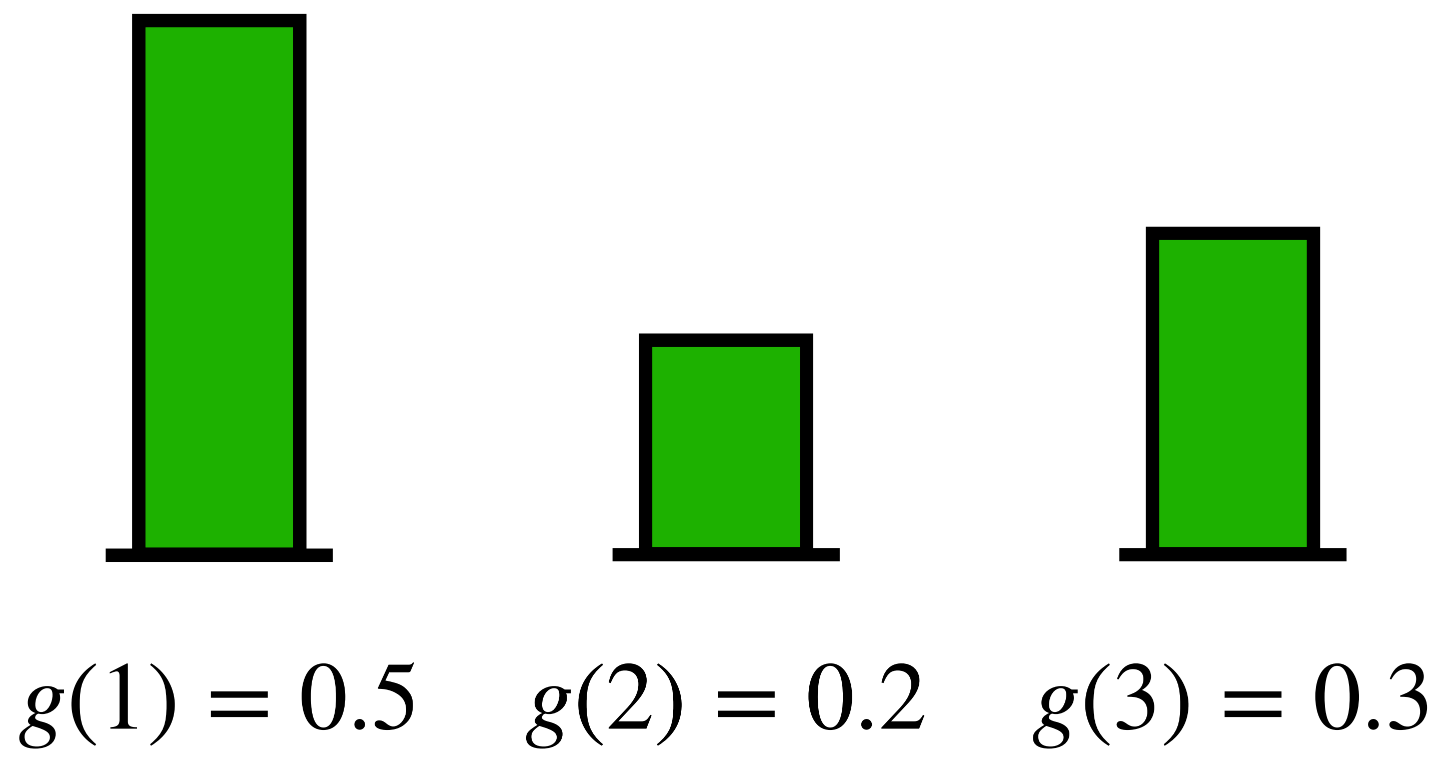
or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).

$n = 3$



Bamboo Garden Trimming

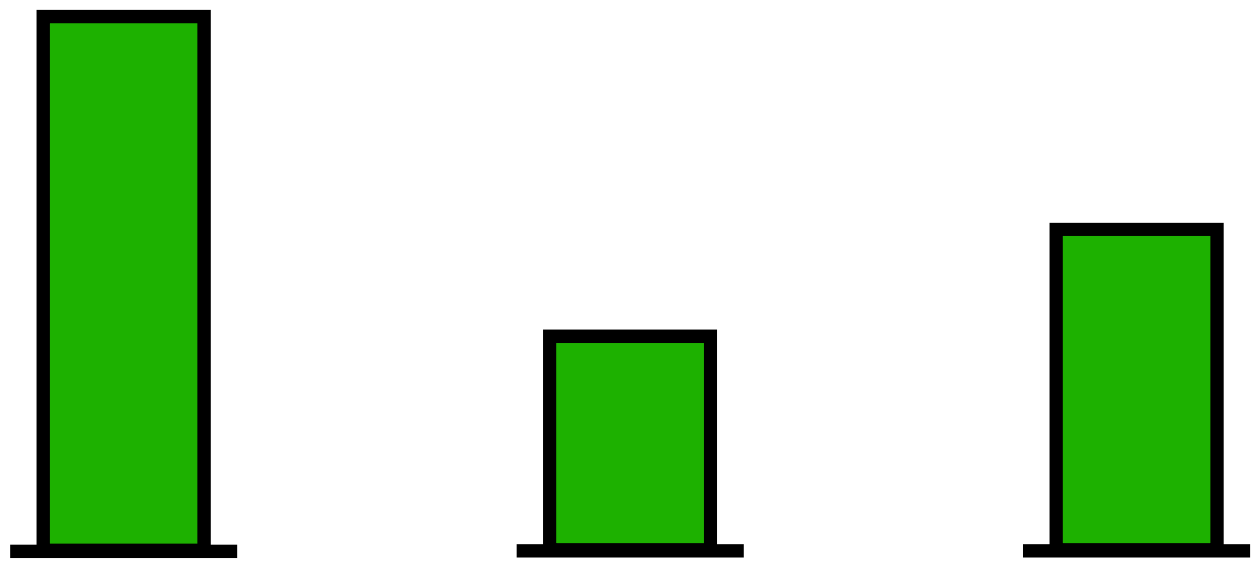
or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
    - Each bamboo *grows* by $g(e)$ (constant over time).
    - The height of some bamboo may be *cut* down to 0.

$n = 3$



$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

Bamboo Garden Trimming

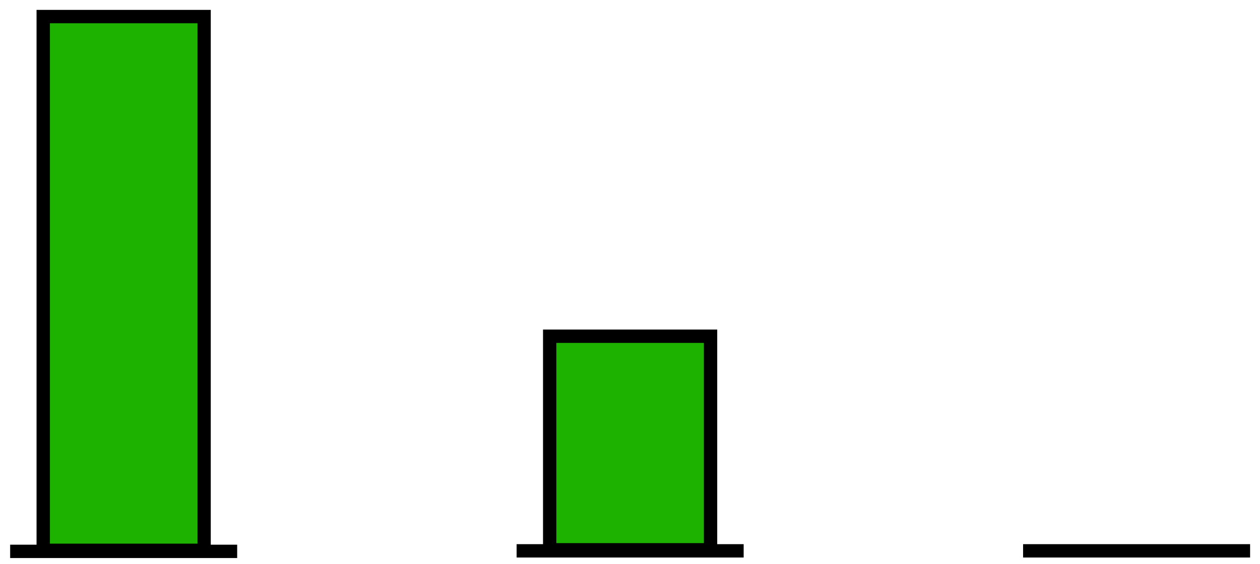
or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - The height of some bamboo may be *cut* down to 0.

$n = 3$



$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

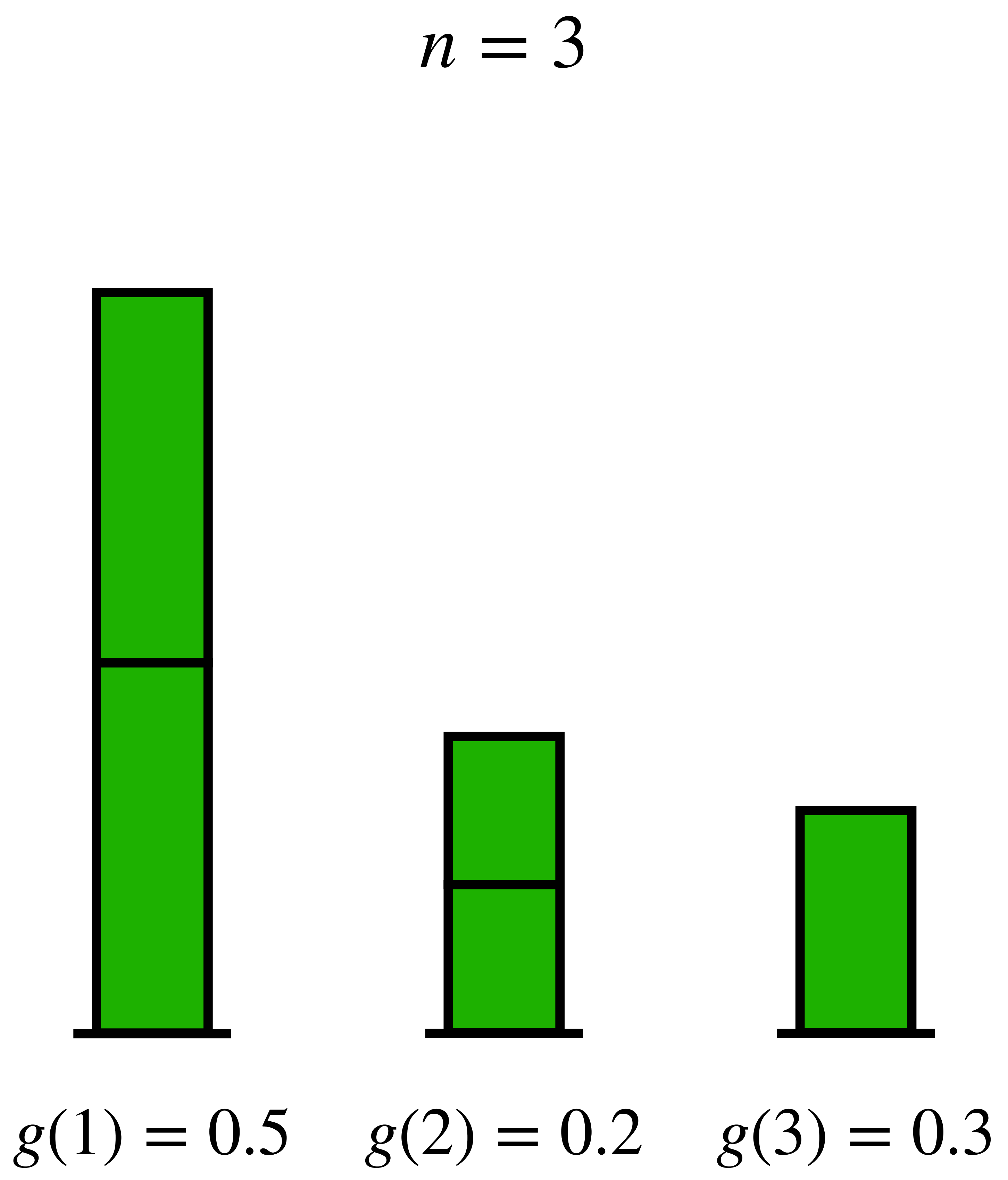
Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - The height of some bamboo may be *cut* down to 0.



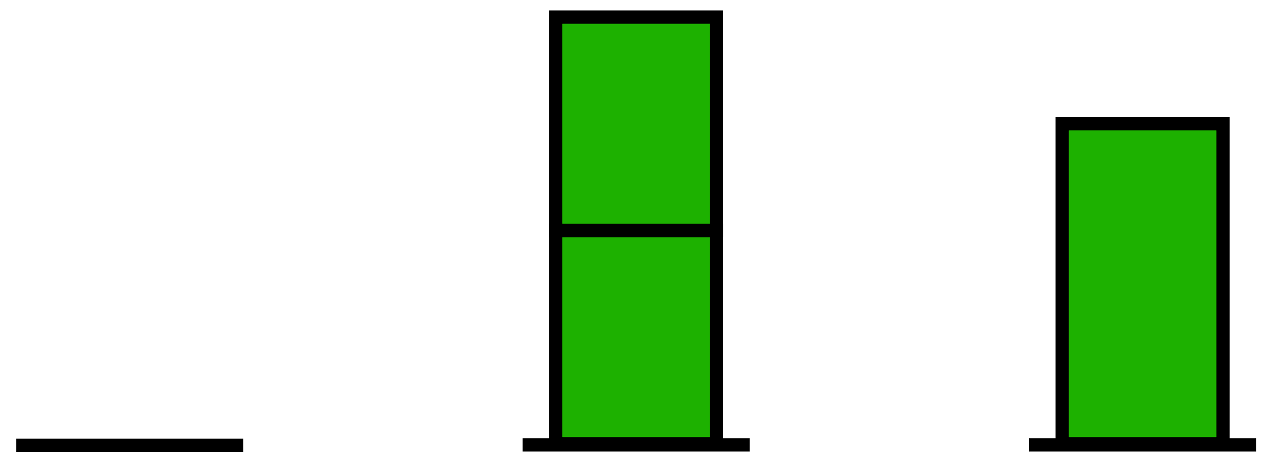
Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model: [Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
    - Each bamboo *grows* by $g(e)$ (constant over time).
    - The height of some bamboo may be *cut* down to 0.

$n = 3$



$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

Bamboo Garden Trimming

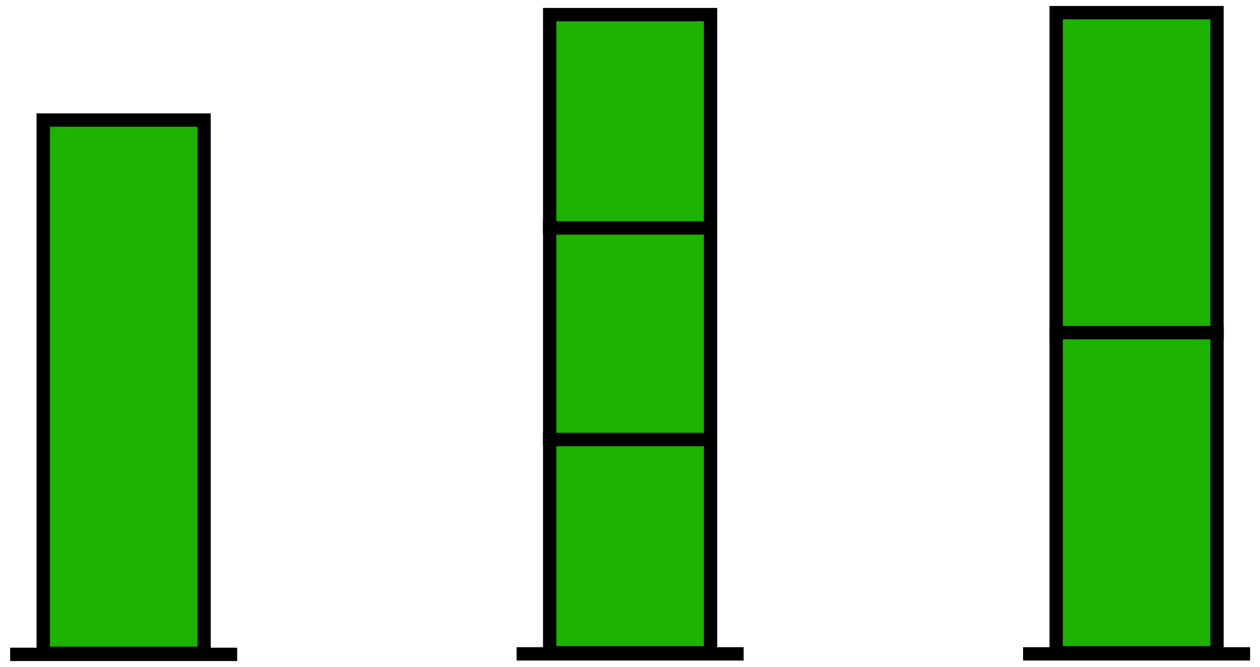
or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
    - Each bamboo *grows* by $g(e)$ (constant over time).
    - The height of some bamboo may be *cut* down to 0.

$n = 3$



$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

Bamboo Garden Trimming

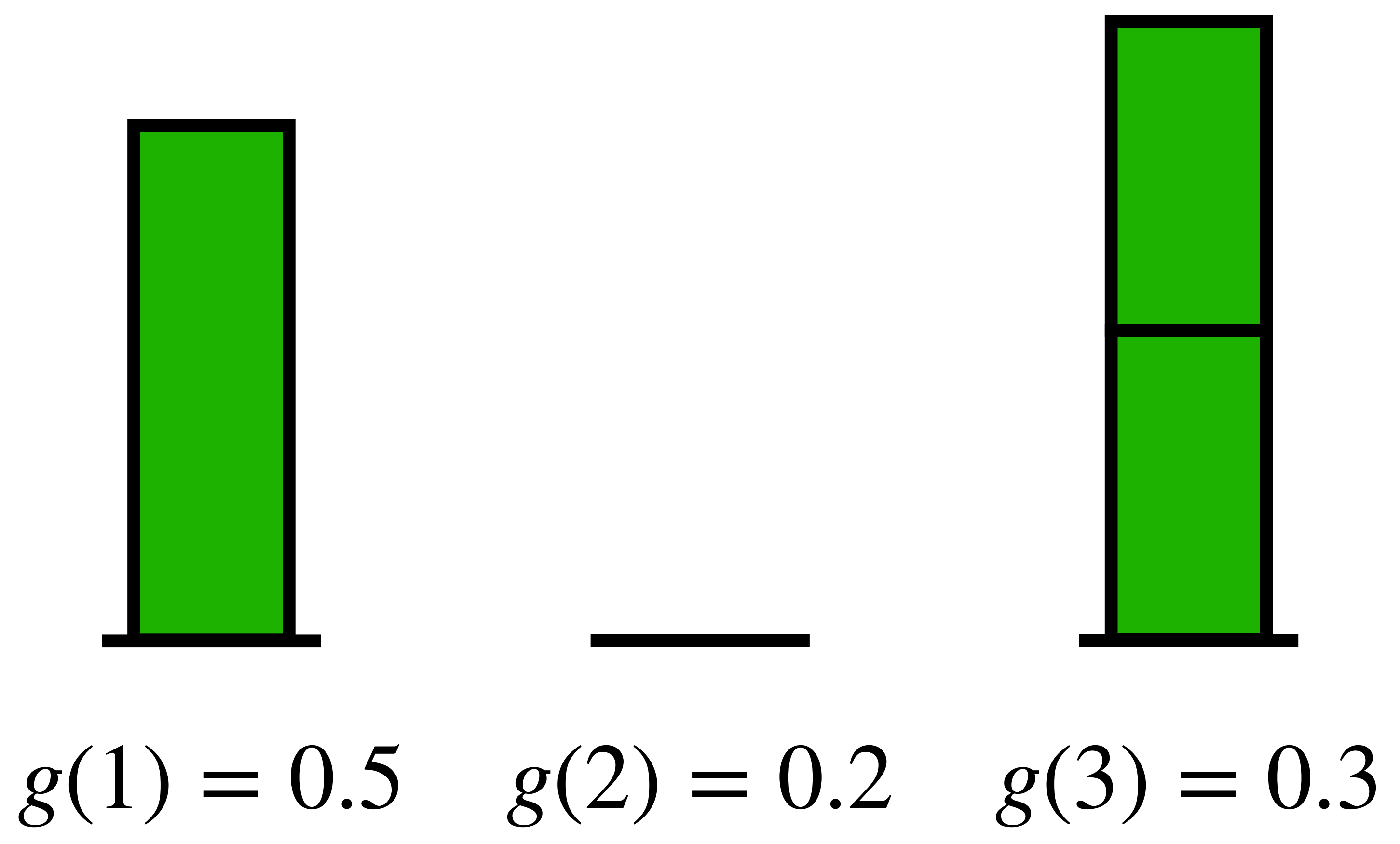
or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
    - Each bamboo *grows* by $g(e)$ (constant over time).
    - The height of some bamboo may be *cut* down to 0.

$n = 3$



Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - The height of some bamboo may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.

$n = 3$

$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - The height of some bamboo may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.

$n = 3$

$g(1) = 0.5$ $g(2) = 0.2$ $g(3) = 0.3$

Question here:

- What height can be guaranteed for all instances?

Bamboo Garden Trimming

or: Perpetual Maintenance Scheduling

Model:

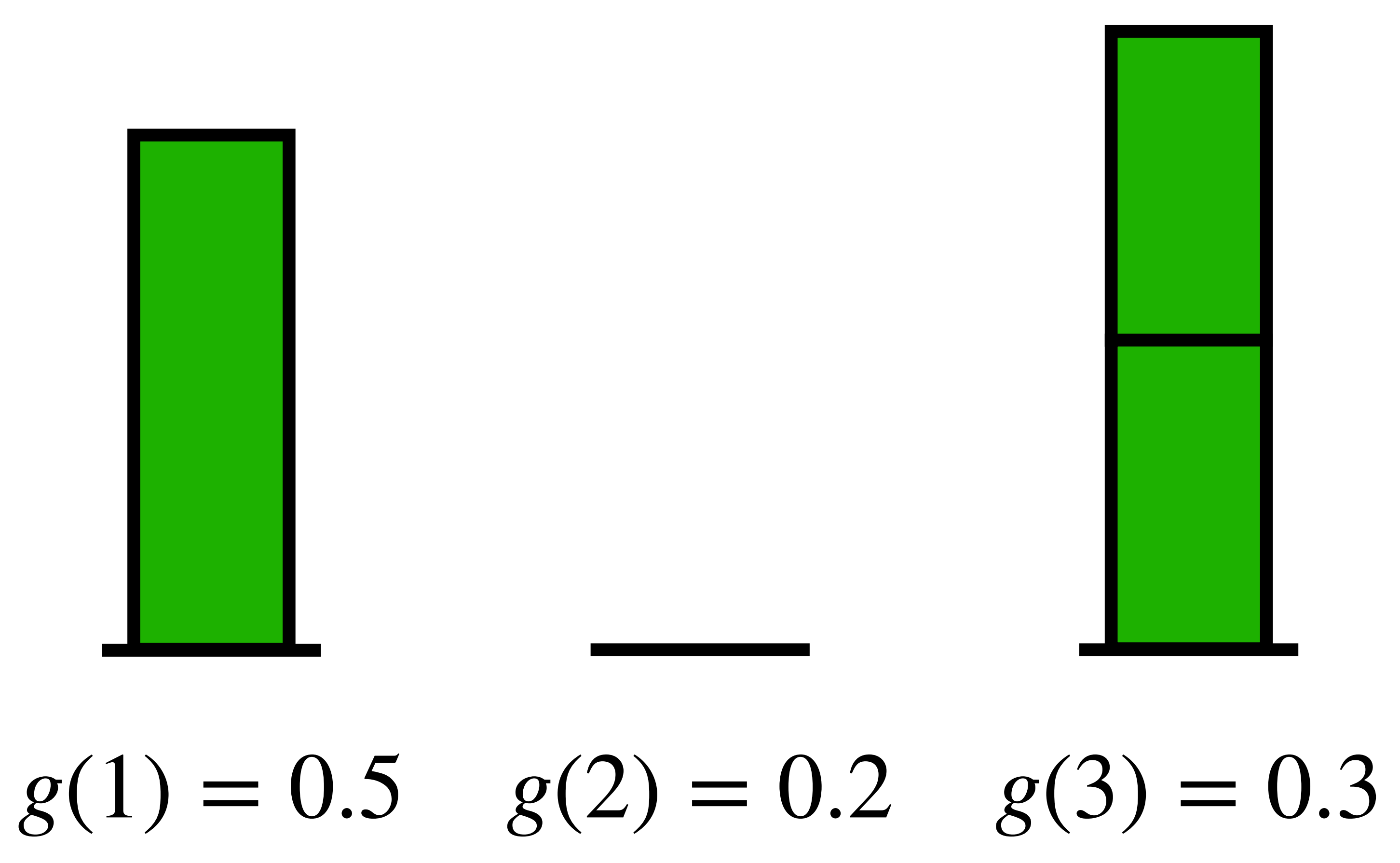
[Gasieniec et al., SOFSEM'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - The height of some bamboo may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.

Question here:

- What height can be guaranteed for all instances?
- Normalization: assume $\sum_{e=1}^{n} g(e) = 1$.

$n = 3$

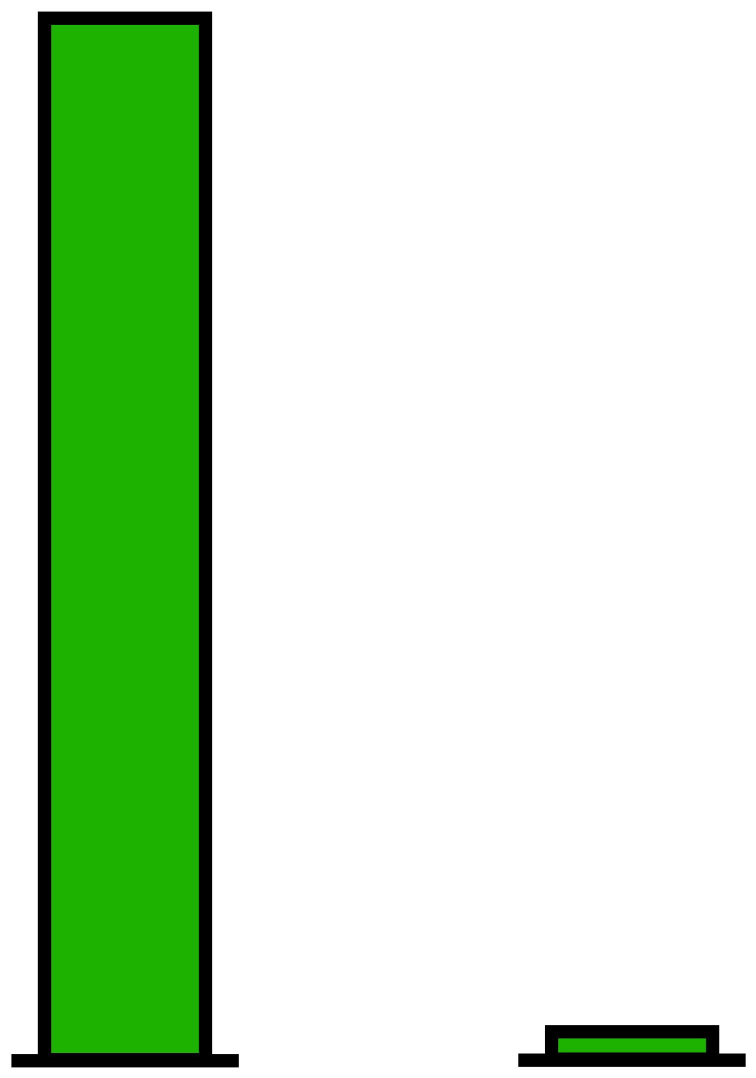


What Maximum Height can be Guaranteed?

What Maximum Height can be Guaranteed?

$g(1) = 1 - \varepsilon \quad g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

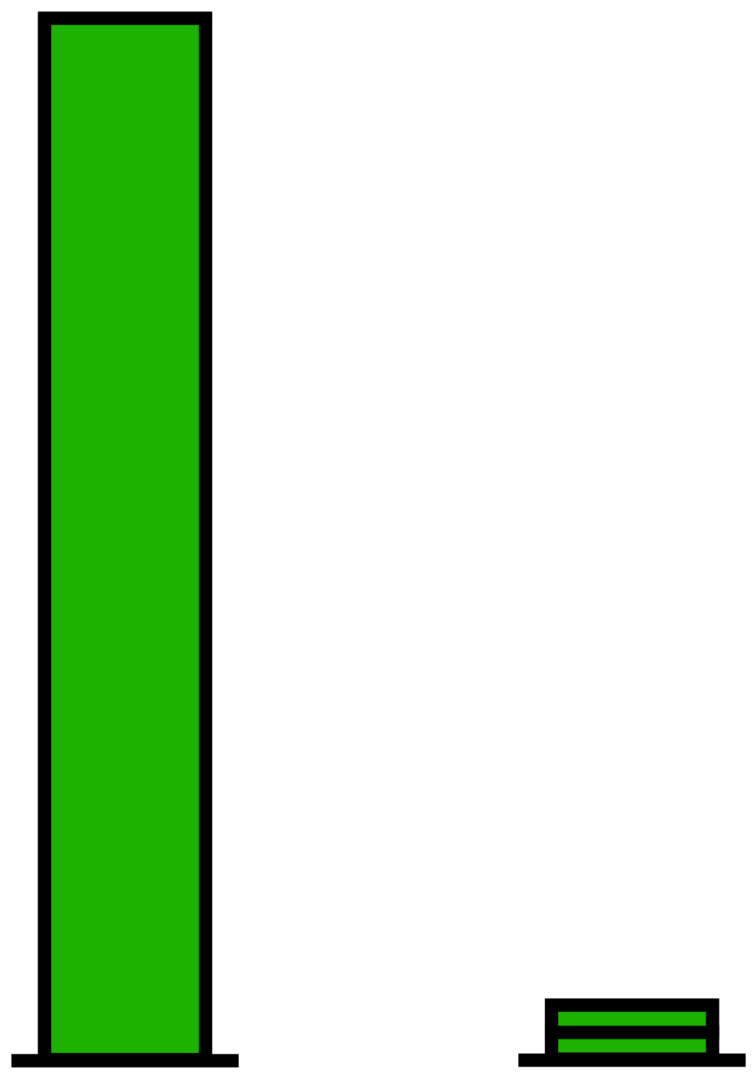


$g(1) = 1 - \varepsilon \quad g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

$g(1) = 1 - \varepsilon$ $\quad$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?



$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

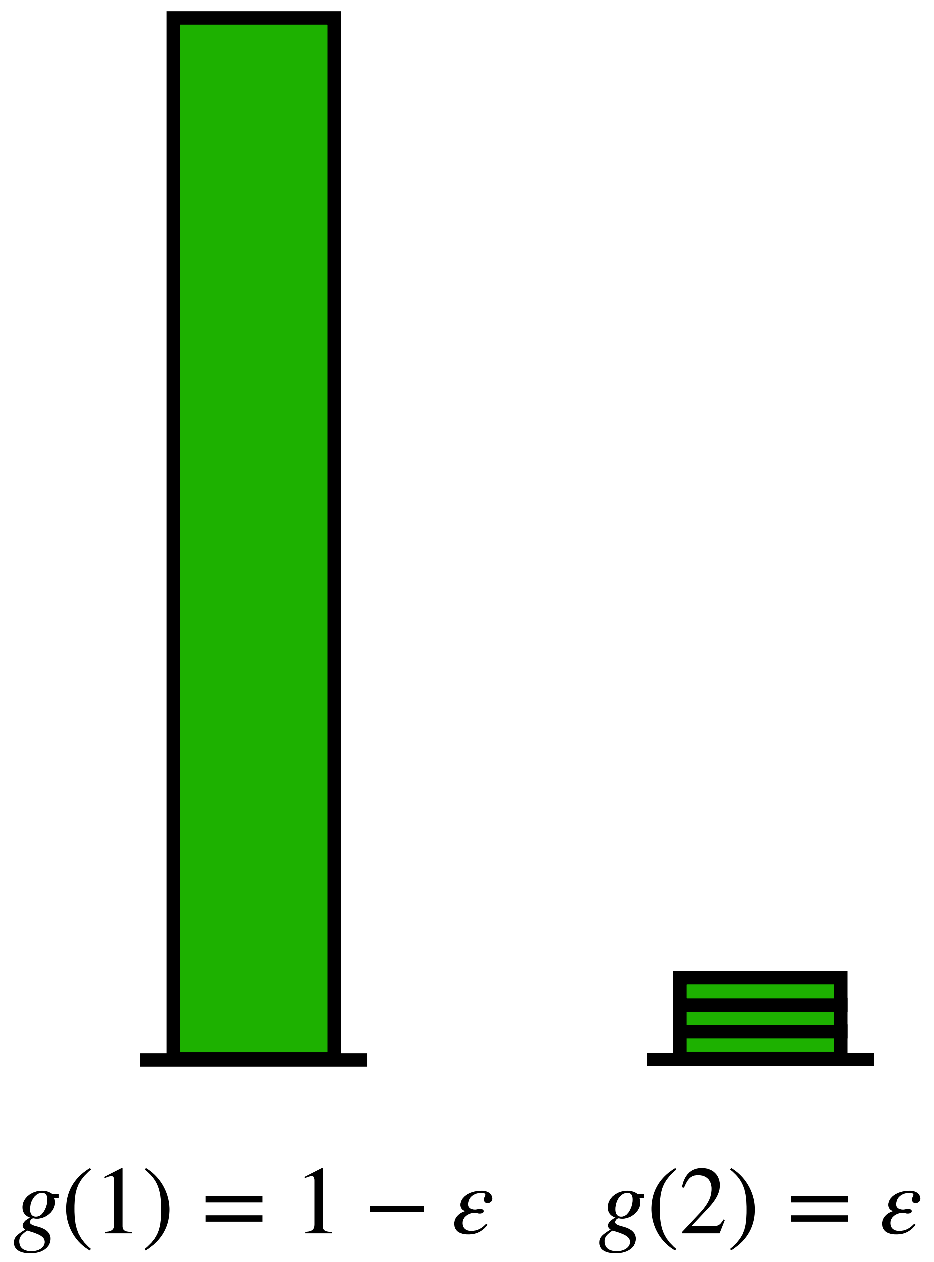
What Maximum Height can be Guaranteed?

$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

$g(1) = 1 - \varepsilon$ $\quad g(2) = \varepsilon$

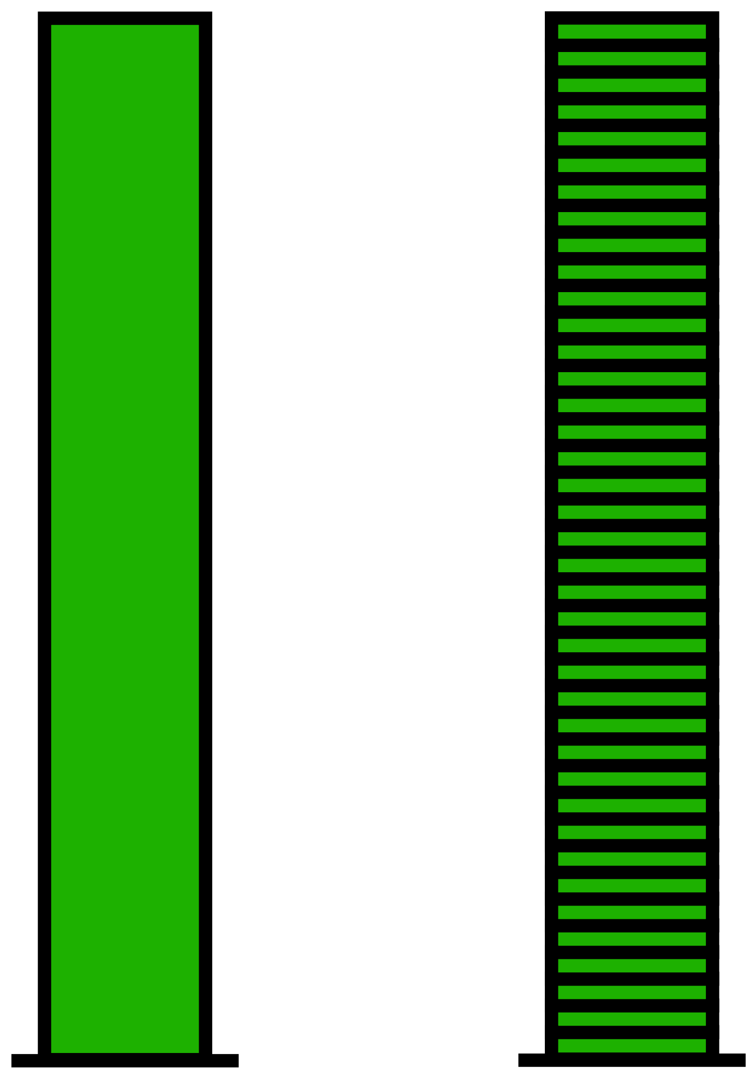
What Maximum Height can be Guaranteed?



What Maximum Height can be Guaranteed?

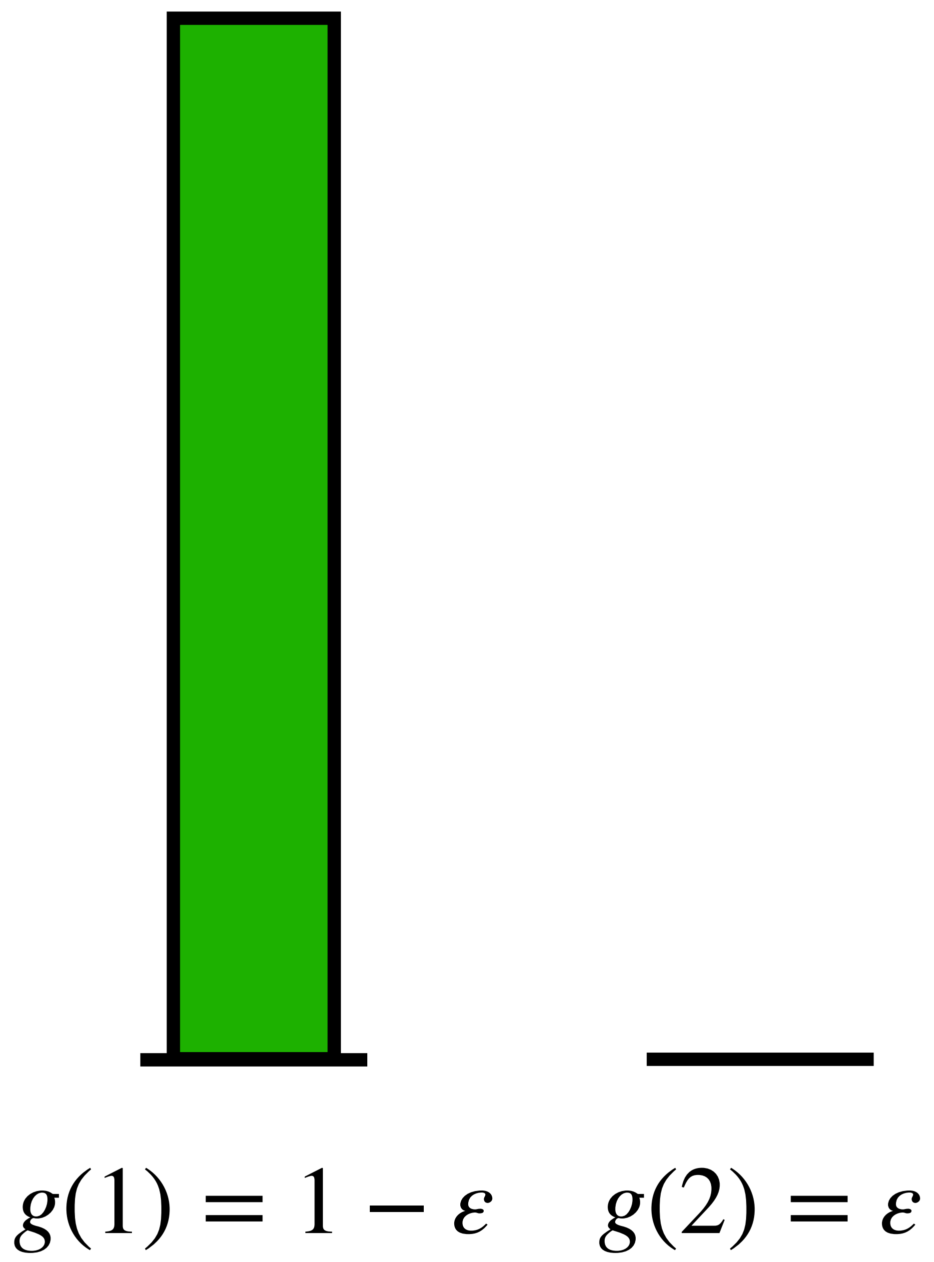
$g(1) = 1 - \varepsilon$ $\quad$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

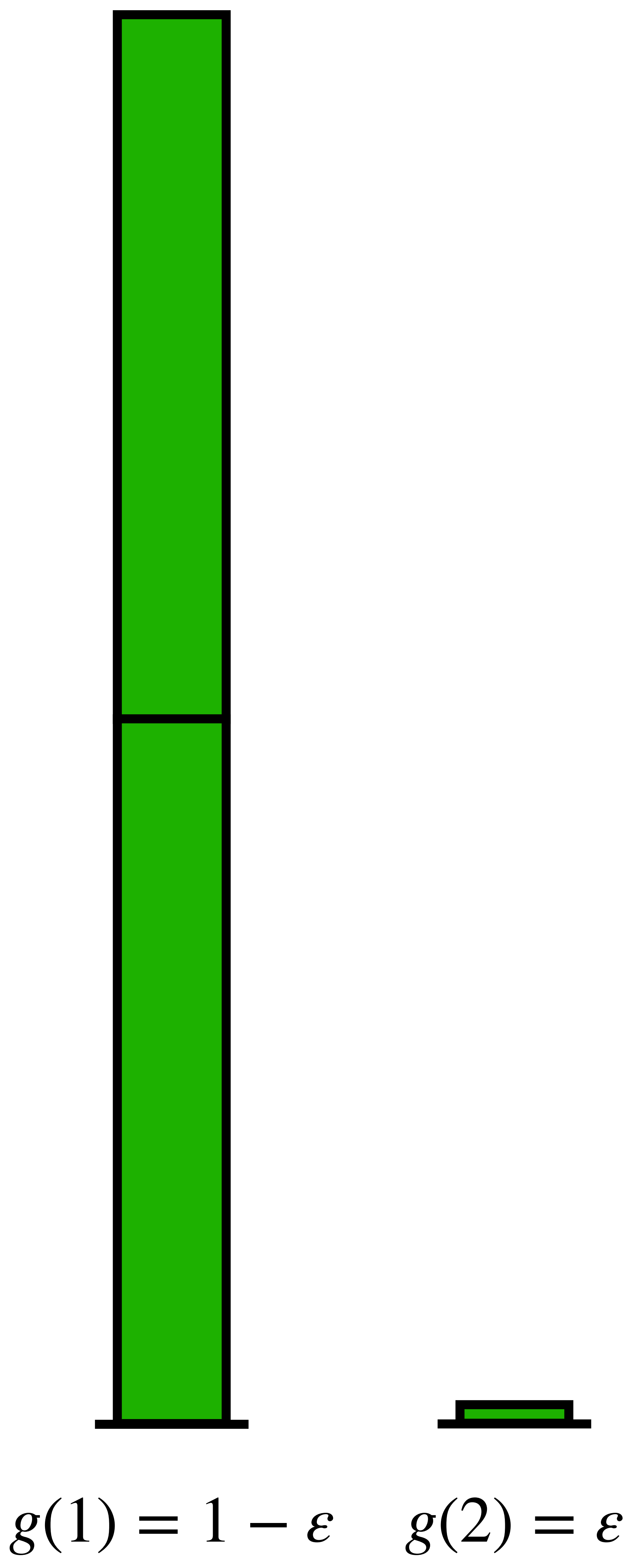


$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?



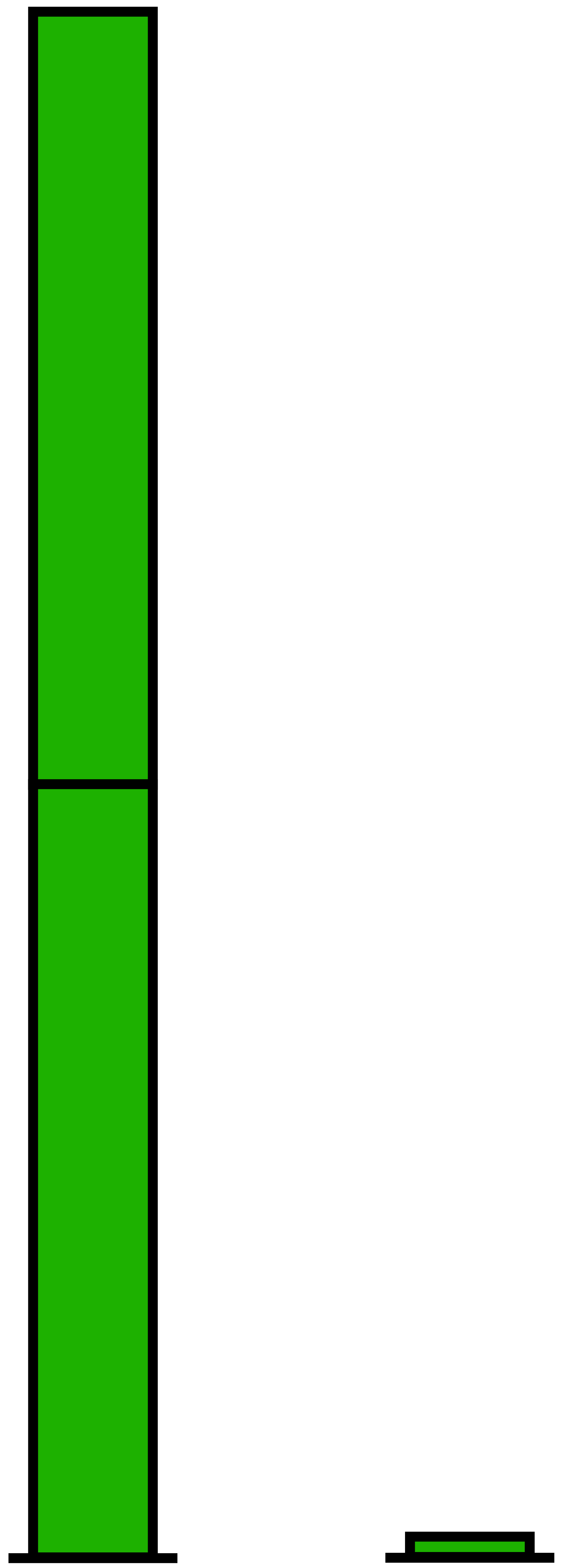
What Maximum Height can be Guaranteed?



What Maximum Height can be Guaranteed?

Lower bound:

- 2 (simple instance)



$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

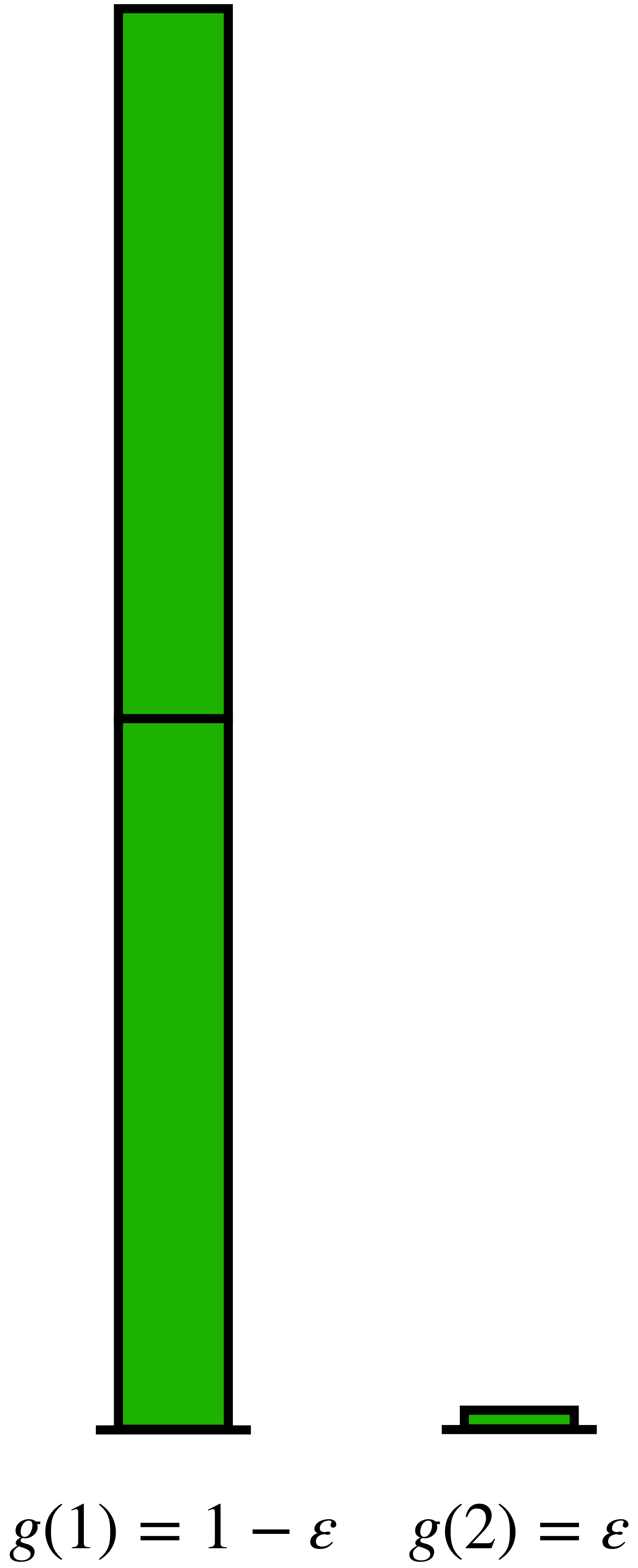
Lower bound:

- 2 (simple instance)

Upper bound:

- 2 (for more general problem)

[Tijdeman, Discr. Math. 1980]



What Maximum Height can be Guaranteed?

Lower bound:

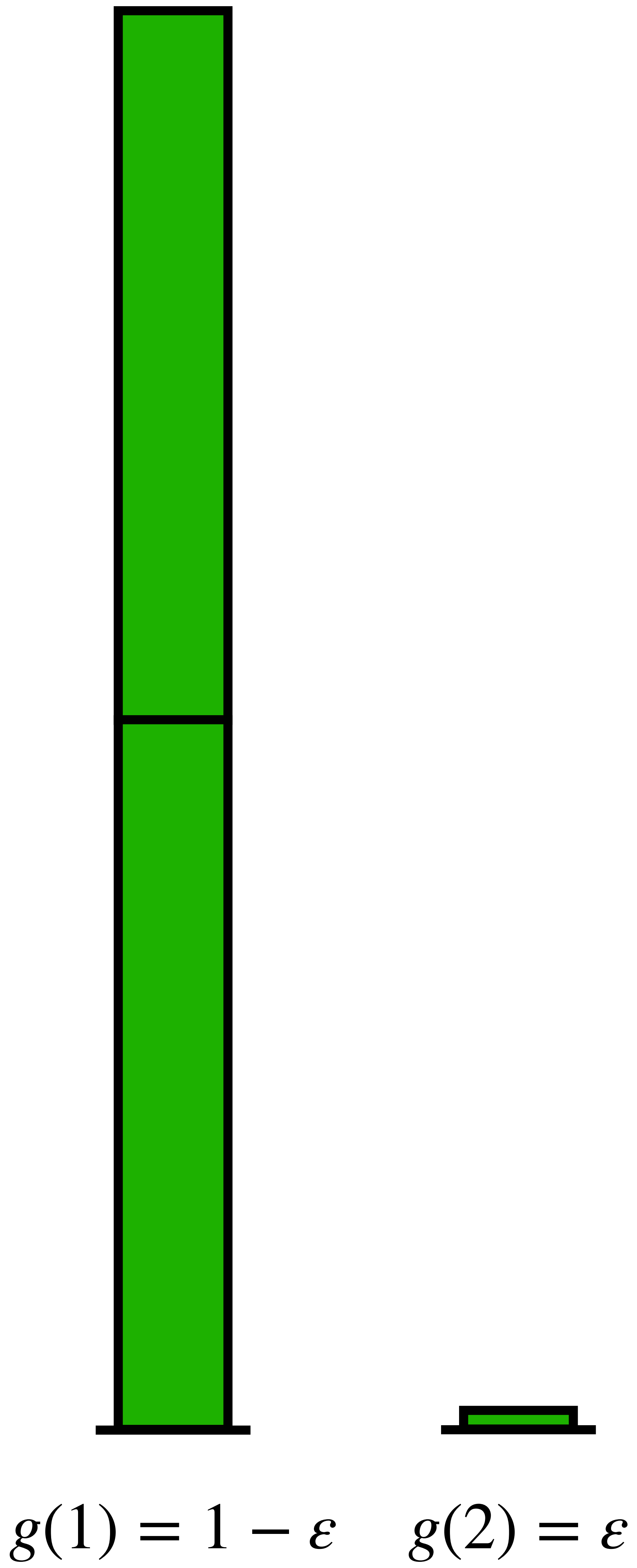
- 2 (simple instance)

Upper bound:

- 2 (for more general problem)

[Tijdeman, Discr. Math. 1980]

Algorithms:



What Maximum Height can be Guaranteed?

Lower bound:

- 2 (simple instance)

Upper bound:

- 2 (for more general problem)

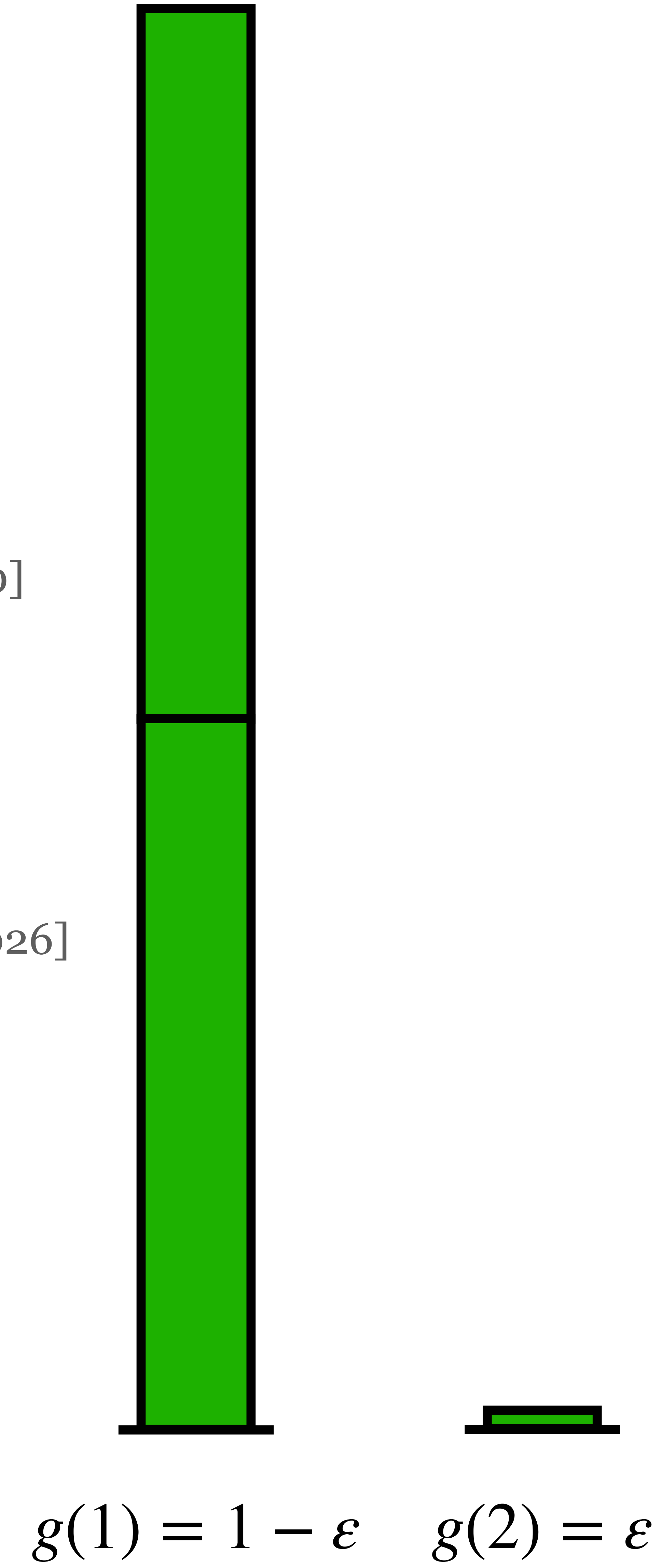
[Tijdeman, Discr. Math. 1980]

Algorithms:

- ReduceMax ("Greedy"): $2.07 \leq \cdot \leq 4$

[Kuszmaul, SPAA'22]

[Jasińska, Kuszmaul, Lee, 2026]



What Maximum Height can be Guaranteed?

Lower bound:

- 2 (simple instance)

Upper bound:

- 2 (for more general problem)

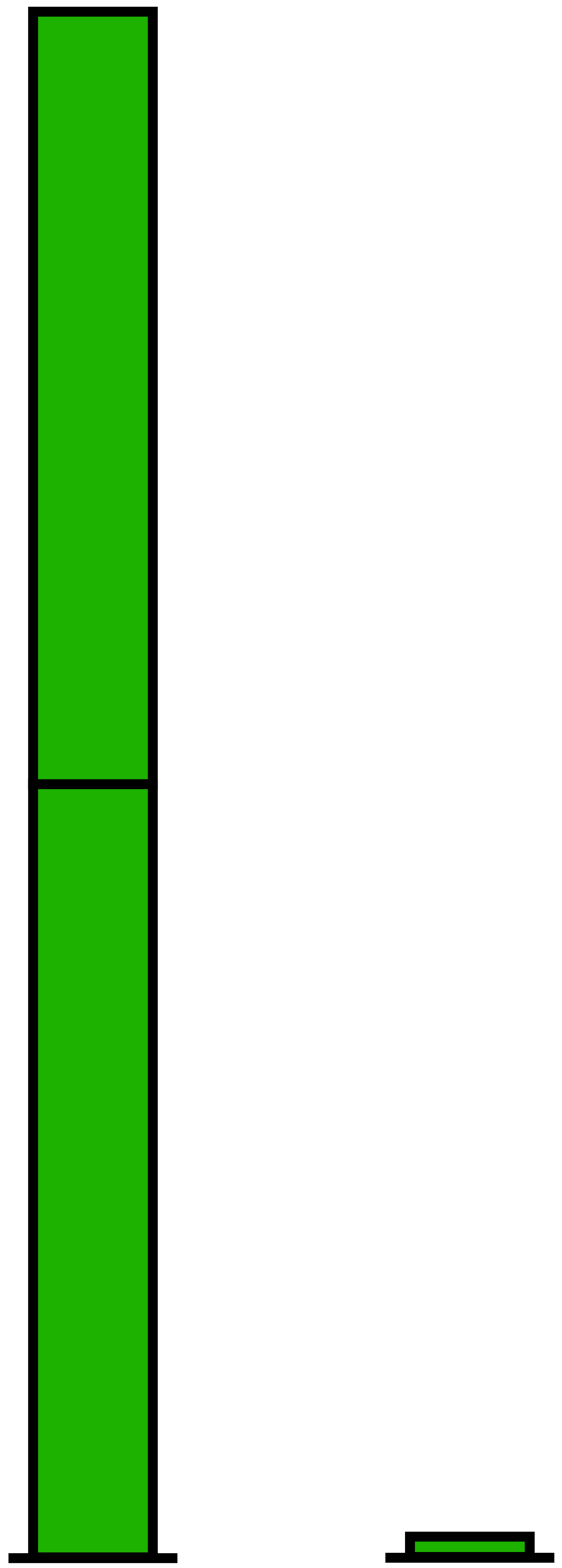
[Tijdeman, Discr. Math. 1980]

Algorithms:

- ReduceMax ("Greedy"): $2.07 \leq \cdot \leq 4$
- ReduceFastest(x) (with a height at least x):

[Kuszmaul, SPAA'22]

[Jasińska, Kuszmaul, Lee, 2026]



$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

Lower bound:

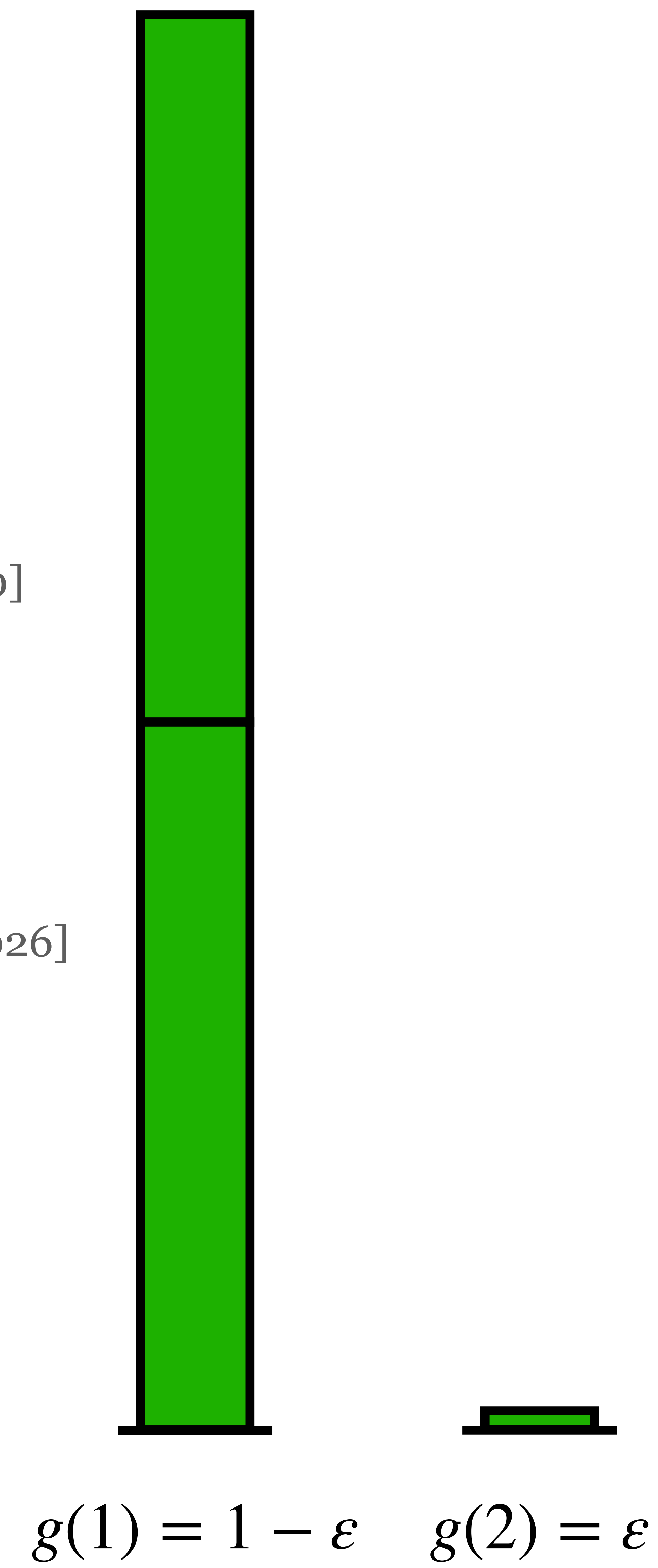
- 2 (simple instance)

Upper bound:

- 2 (for more general problem) [Tijdeman, Discr. Math. 1980]

Algorithms:

- ReduceMax (“Greedy”): $2.07 \leq \cdot \leq 4$ [Kuszmaul, SPAA’22] [Jasińska, Kuszmaul, Lee, 2026]
- ReduceFastest(x) (with a height at least x):
    - For no x better than 2.01. [Kuszmaul, SPAA’22]



What Maximum Height can be Guaranteed?

Lower bound:

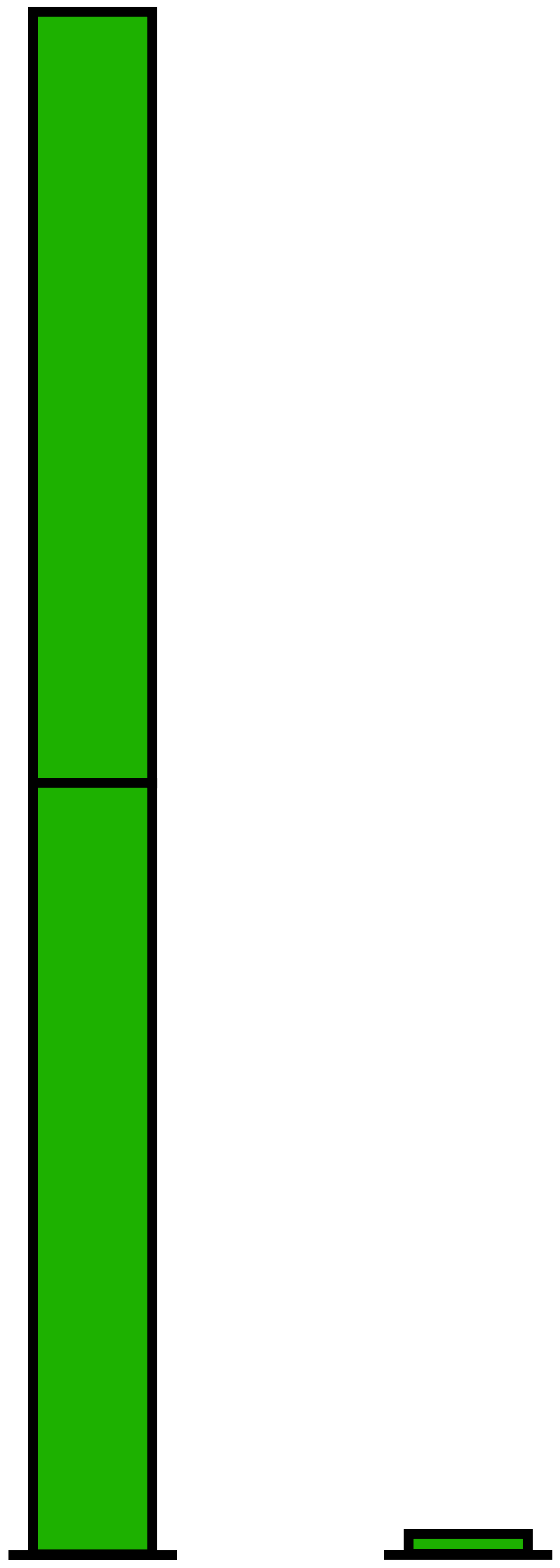
- 2 (simple instance)

Upper bound:

- 2 (for more general problem) [Tijdeman, Discr. Math. 1980]

Algorithms:

- ReduceMax ("Greedy"): $2.07 \leq \cdot \leq 4$ [Kuszmaul, SPAA'22] [Jasińska, Kuszmaul, Lee, 2026]
- ReduceFastest(x) (with a height at least x):
  - For no x better than 2.01. [Kuszmaul, SPAA'22]
  - For $x = 1$, exactly 3. [Kuszmaul, SPAA'22]



What Maximum Height can be Guaranteed?

Lower bound:

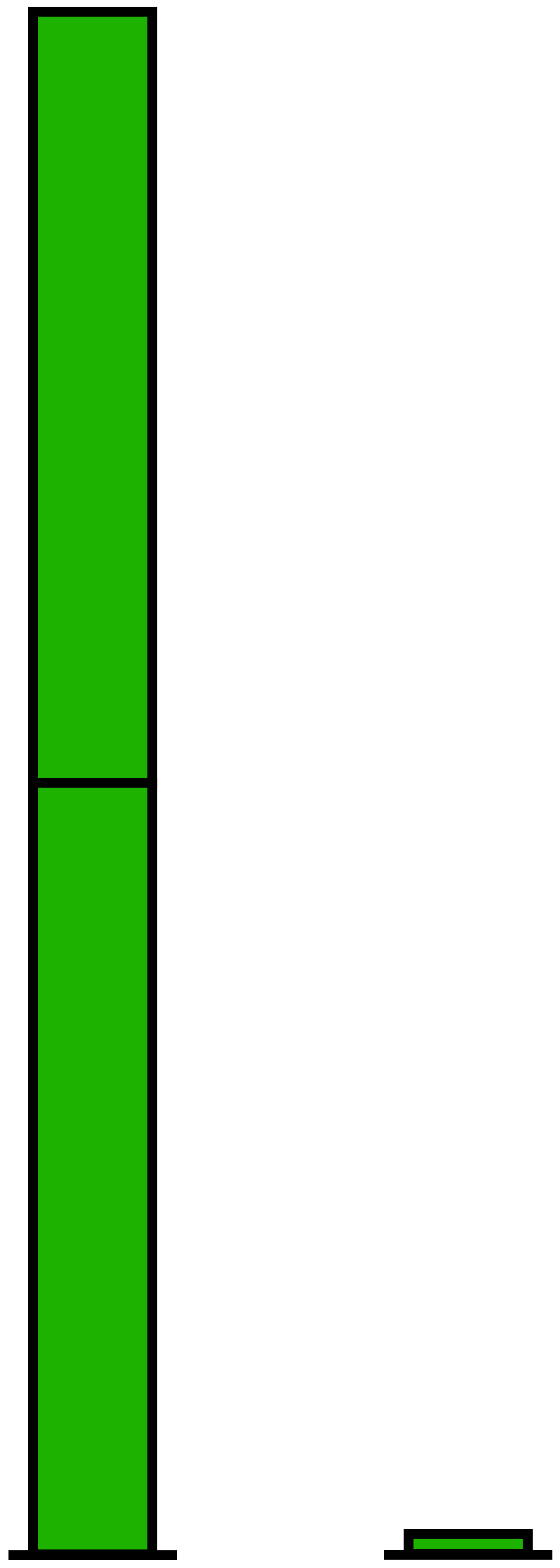
- 2 (simple instance)

Upper bound:

- 2 (for more general problem) [Tijdeman, Discr. Math. 1980]

Algorithms:

- ReduceMax ("Greedy"): $2.07 \leq \cdot \leq 4$ [Kuszmaul, SPAA'22] [Jasińska, Kuszmaul, Lee, 2026]
- ReduceFastest(x) (with a height at least x):
  - For no x better than 2.01. [Kuszmaul, SPAA'22]
  - For $x = 1$, exactly 3. [Kuszmaul, SPAA'22]
- Deadline-Driven Strategy: 2 [Kuszmaul, SPAA'22]



$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

What Maximum Height can be Guaranteed?

Lower bound:

- 2 (simple instance)

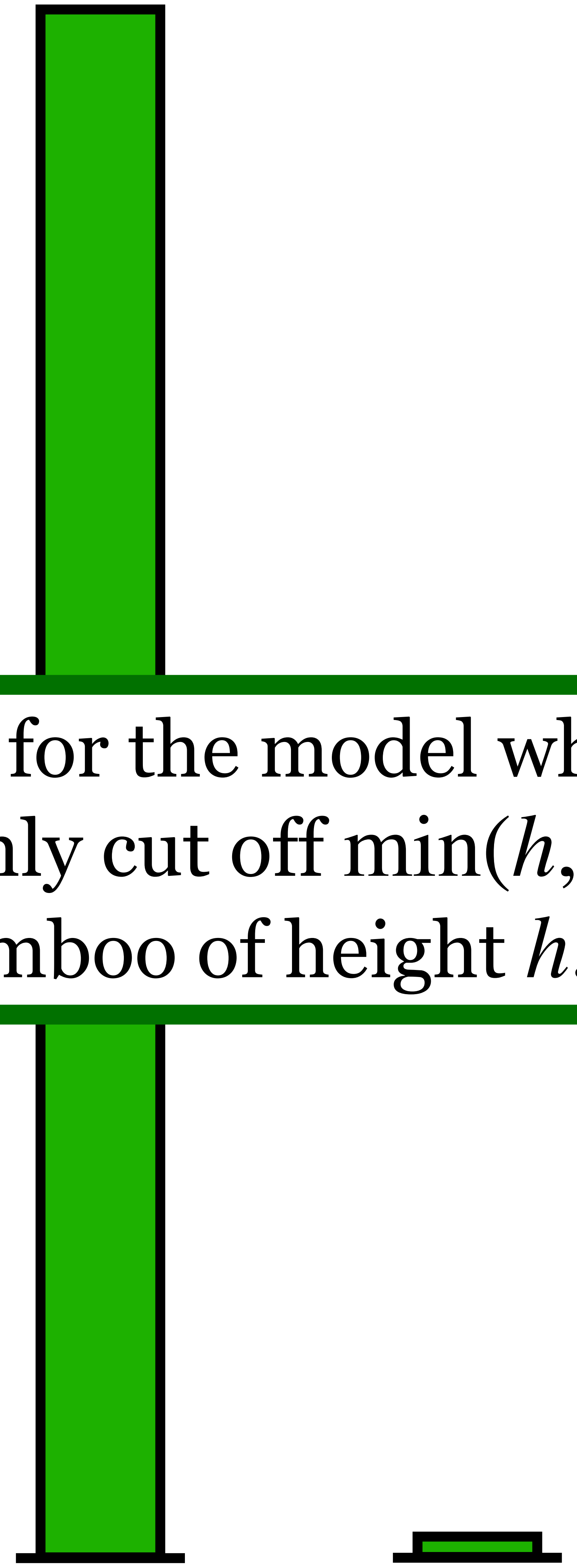
Upper bound:

- 2 (for more general problem) [Tijdeman, Discr. Math. 1980]

Even works for the model where one can only cut off $\min(h,1)$ from bamboo of height h.

Algorithms:

- ReduceMax (“Greedy”): $2.07 \leq \cdot \leq 4$ [Kuszmaul [Jasińska, Kuszmaul, Lee, 2020]
- ReduceFastest(x) (with a height at least x):
    - For no x better than 2.01. [Kuszmaul, SPAA’22]
    - For $x = 1$, exactly 3. [Kuszmaul, SPAA’22]
- Deadline-Driven Strategy: 2 [Kuszmaul, SPAA’22]



$g(1) = 1 - \varepsilon$ $g(2) = \varepsilon$

The Combinatorial Version

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

The Combinatorial Version

Model:

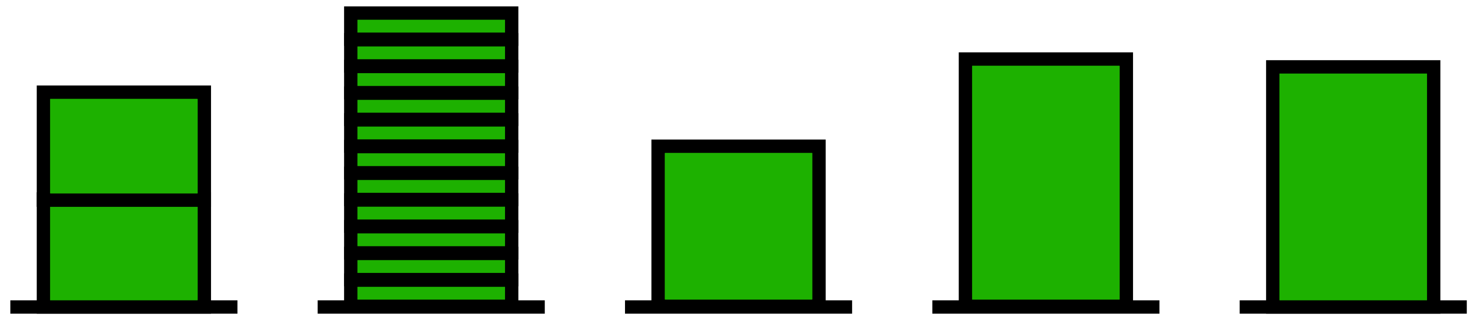
see also [Cicerone et al., CIAC'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]



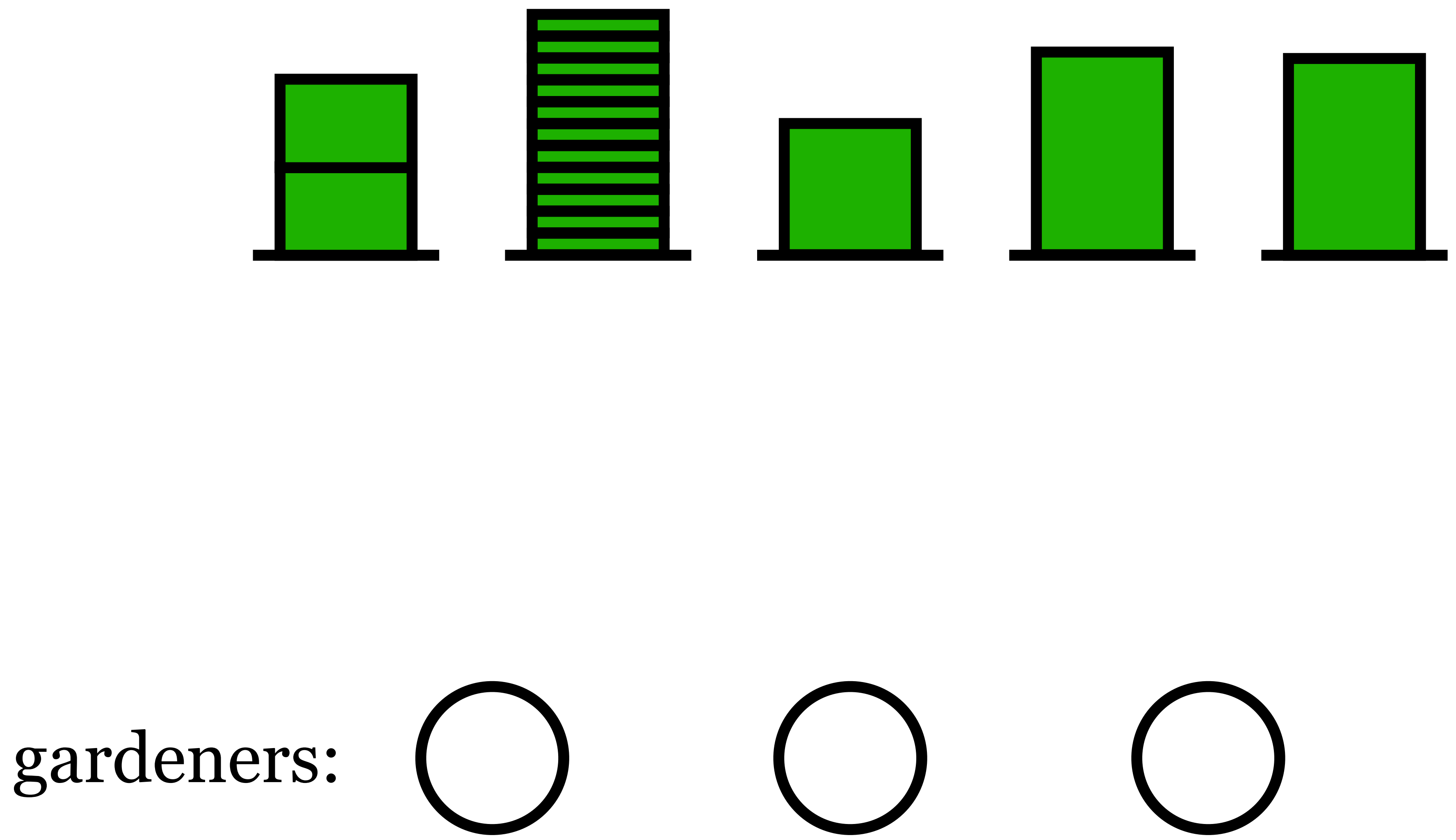
- n *bamboos* (or tasks), initially of *height* (or urgency) 0.

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.

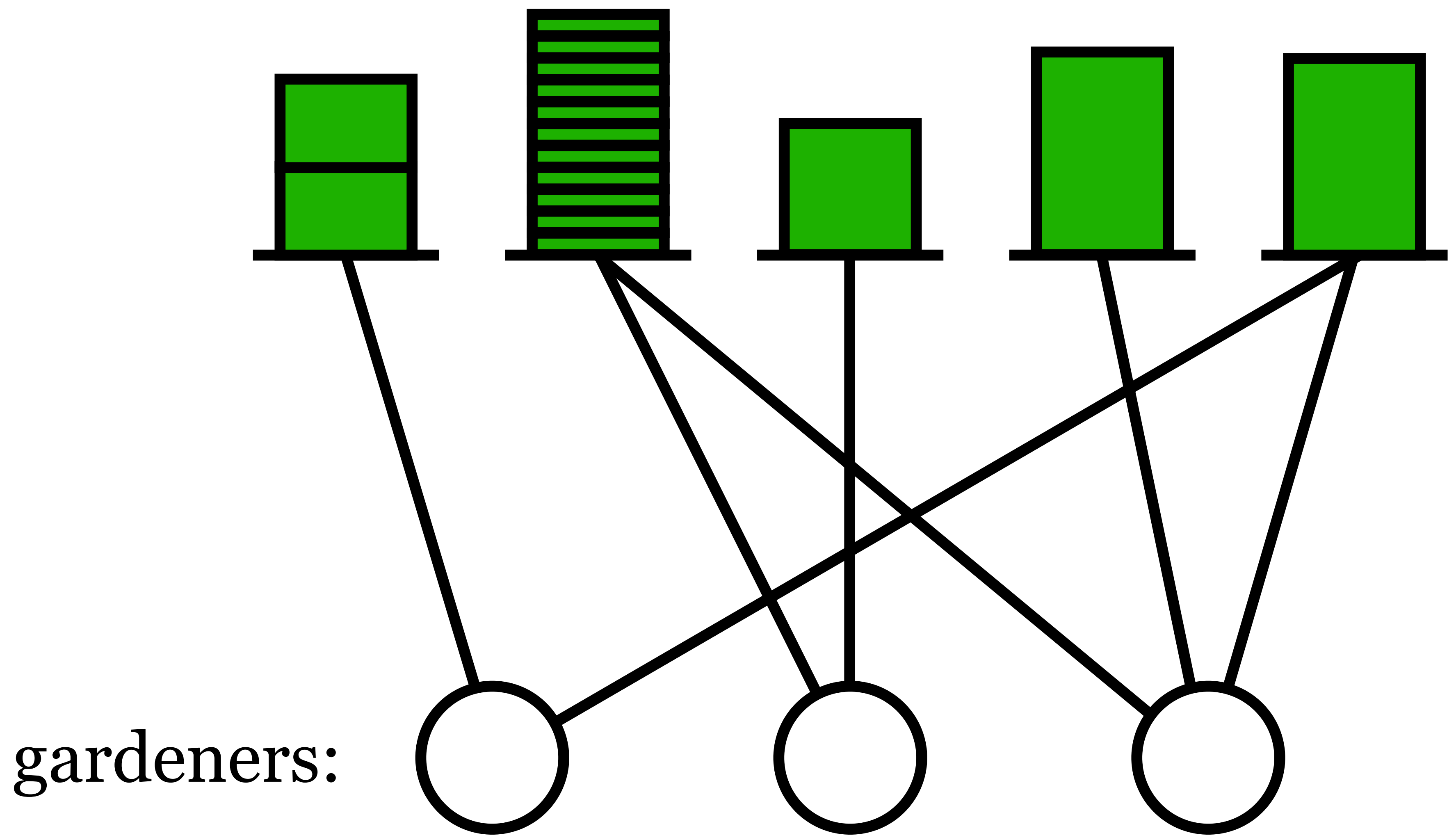


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.

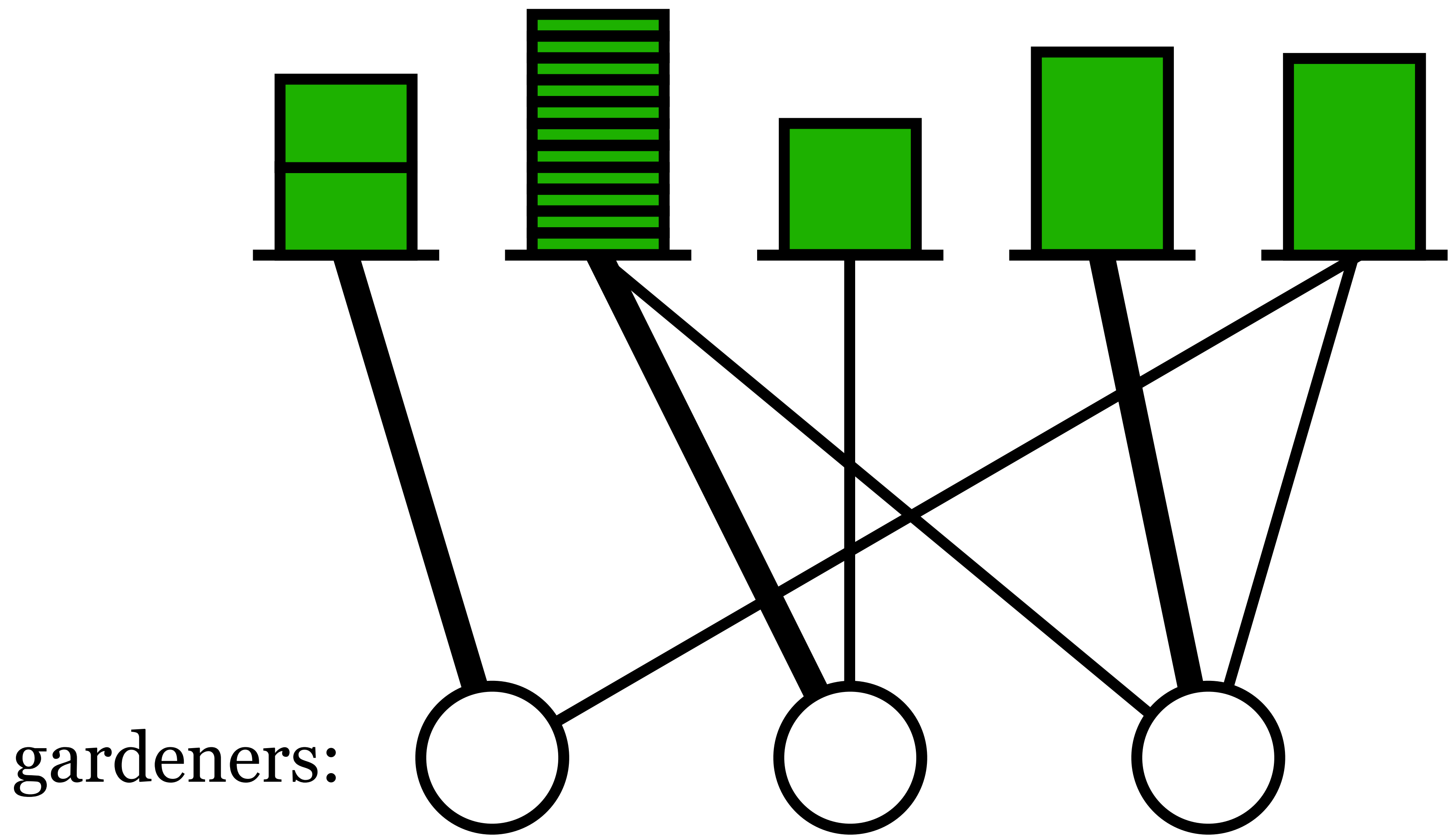


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.

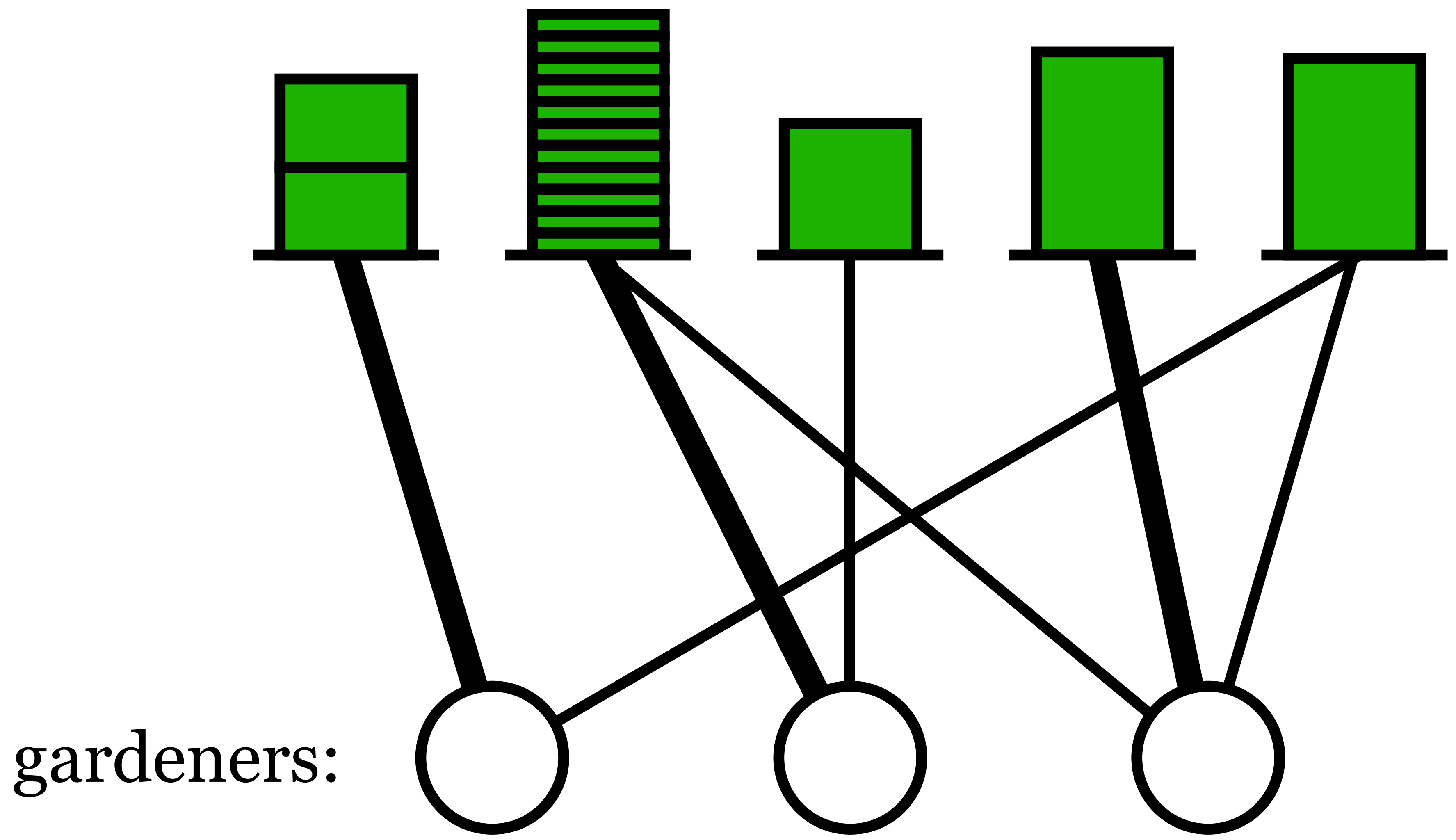


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.

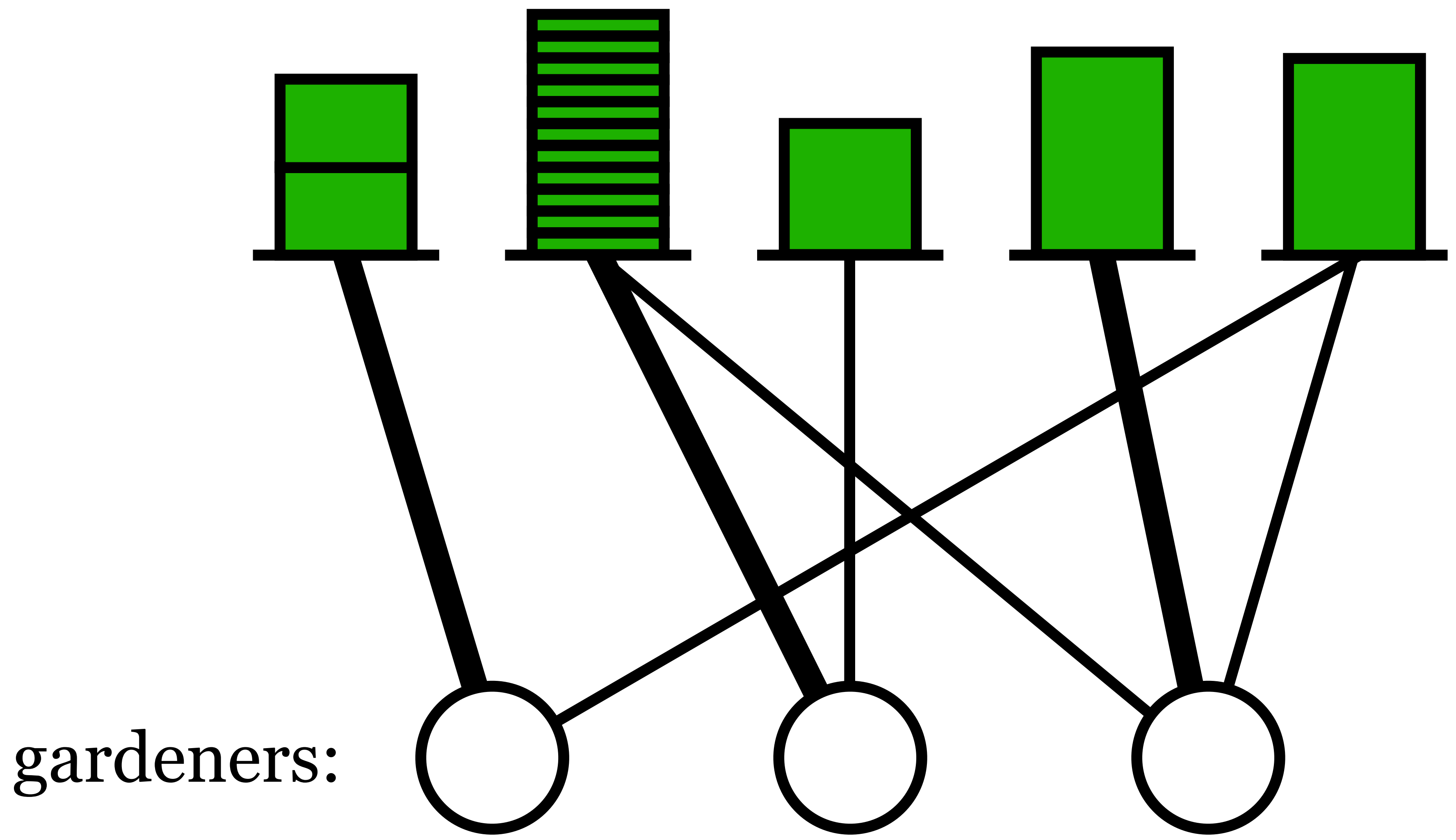


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.
- At each discrete time step (forever):

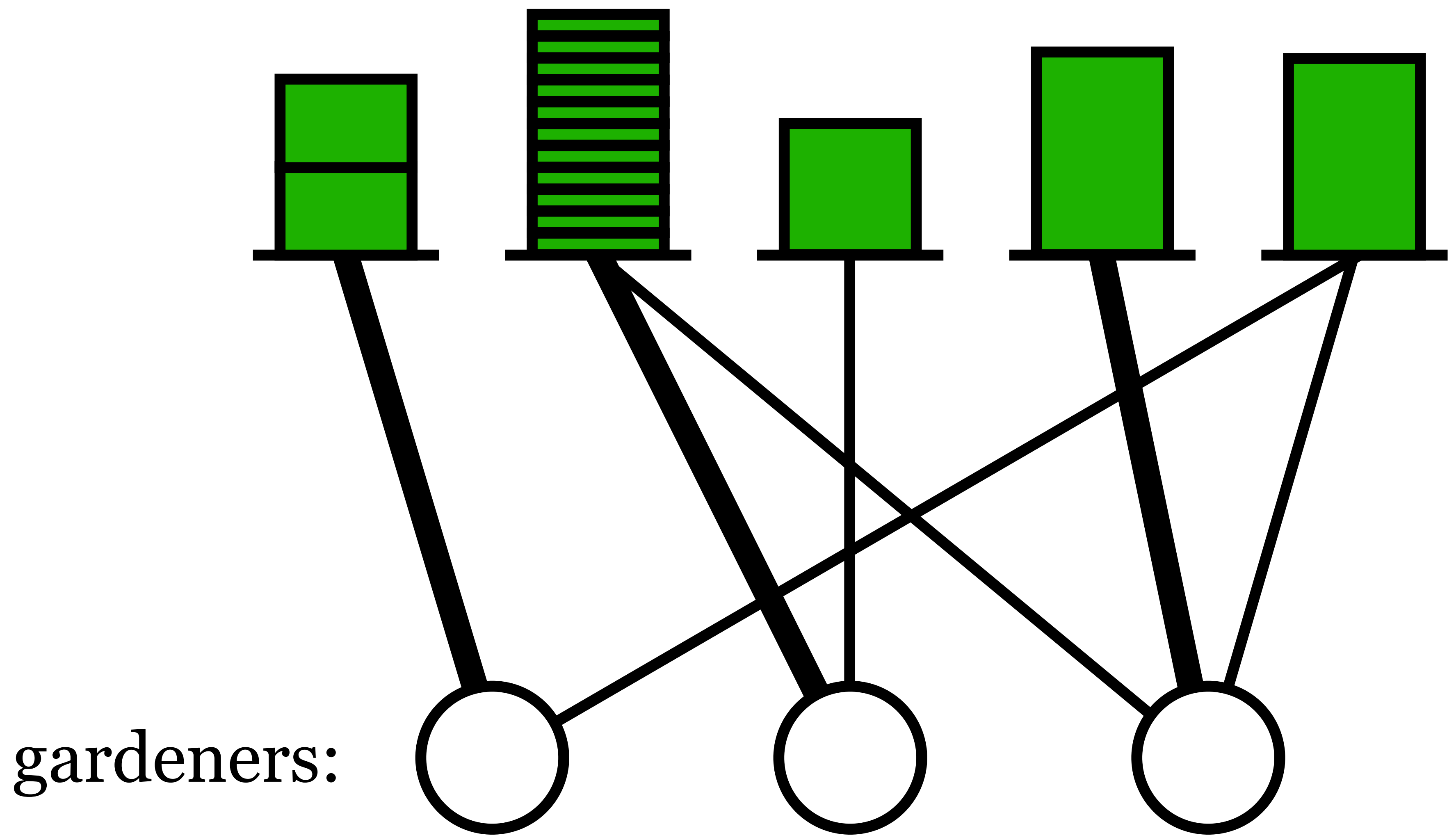


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).

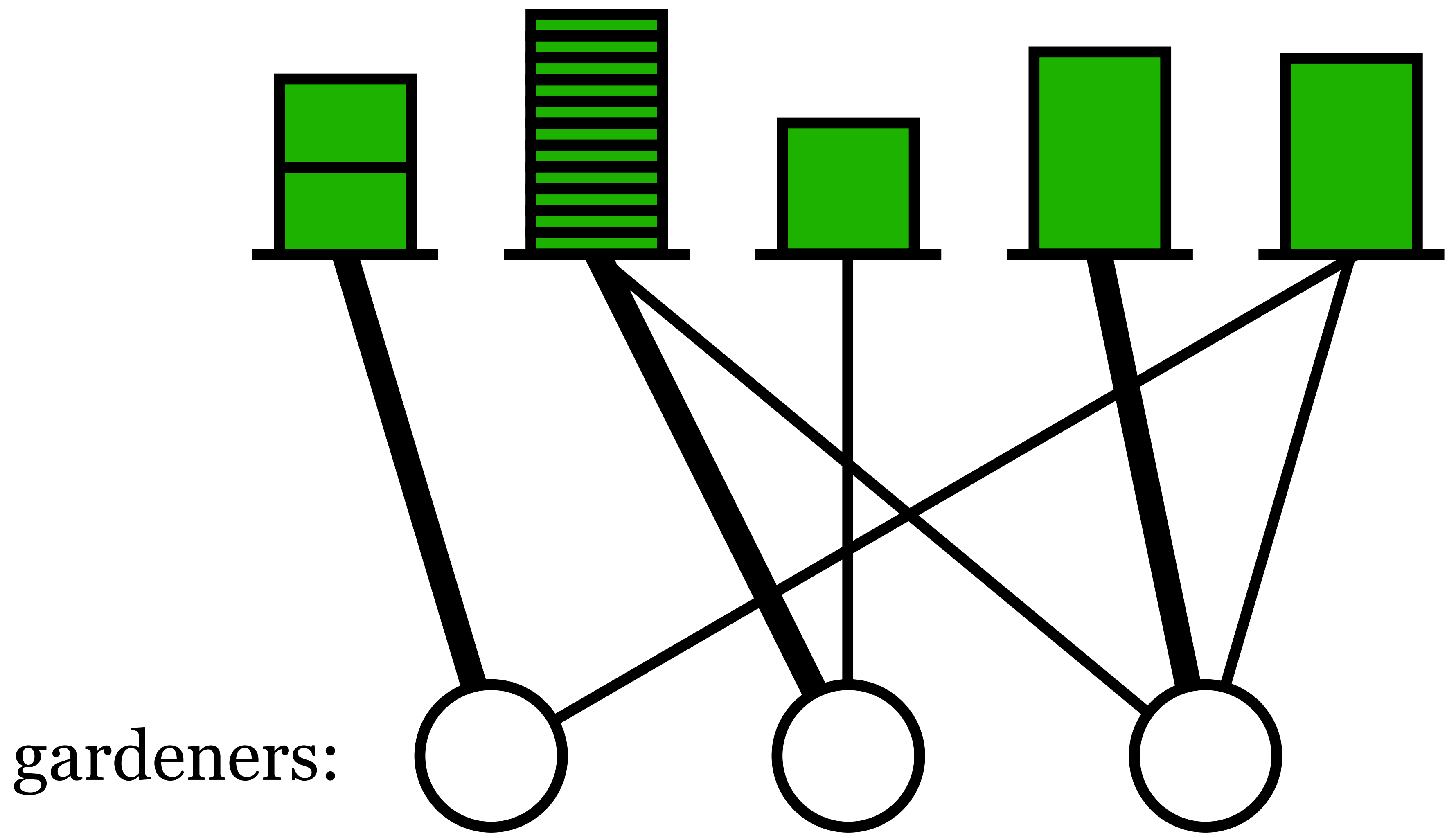


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathscr{I}$ may be *cut* down to 0.

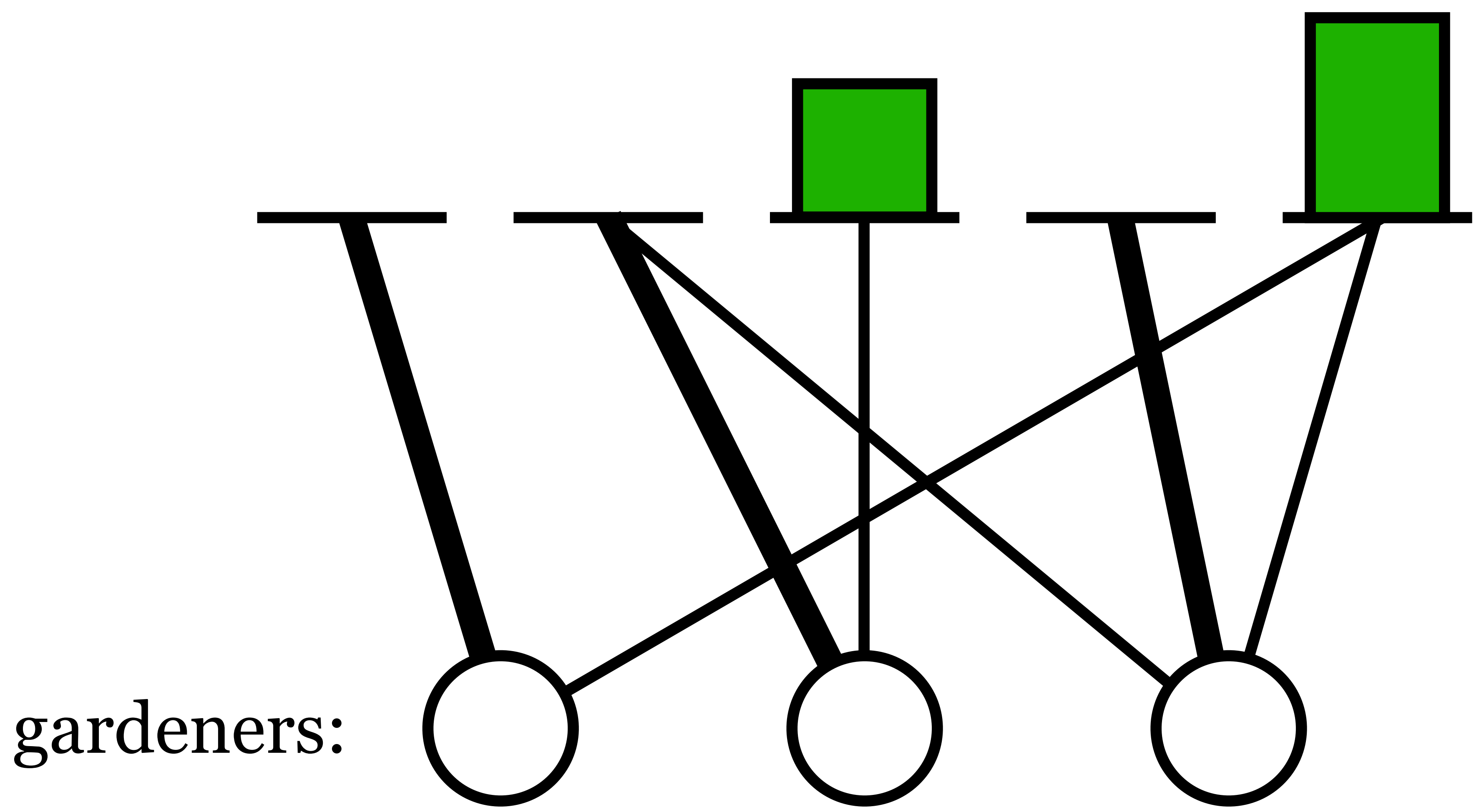


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathscr{I}$ may be *cut* down to 0.

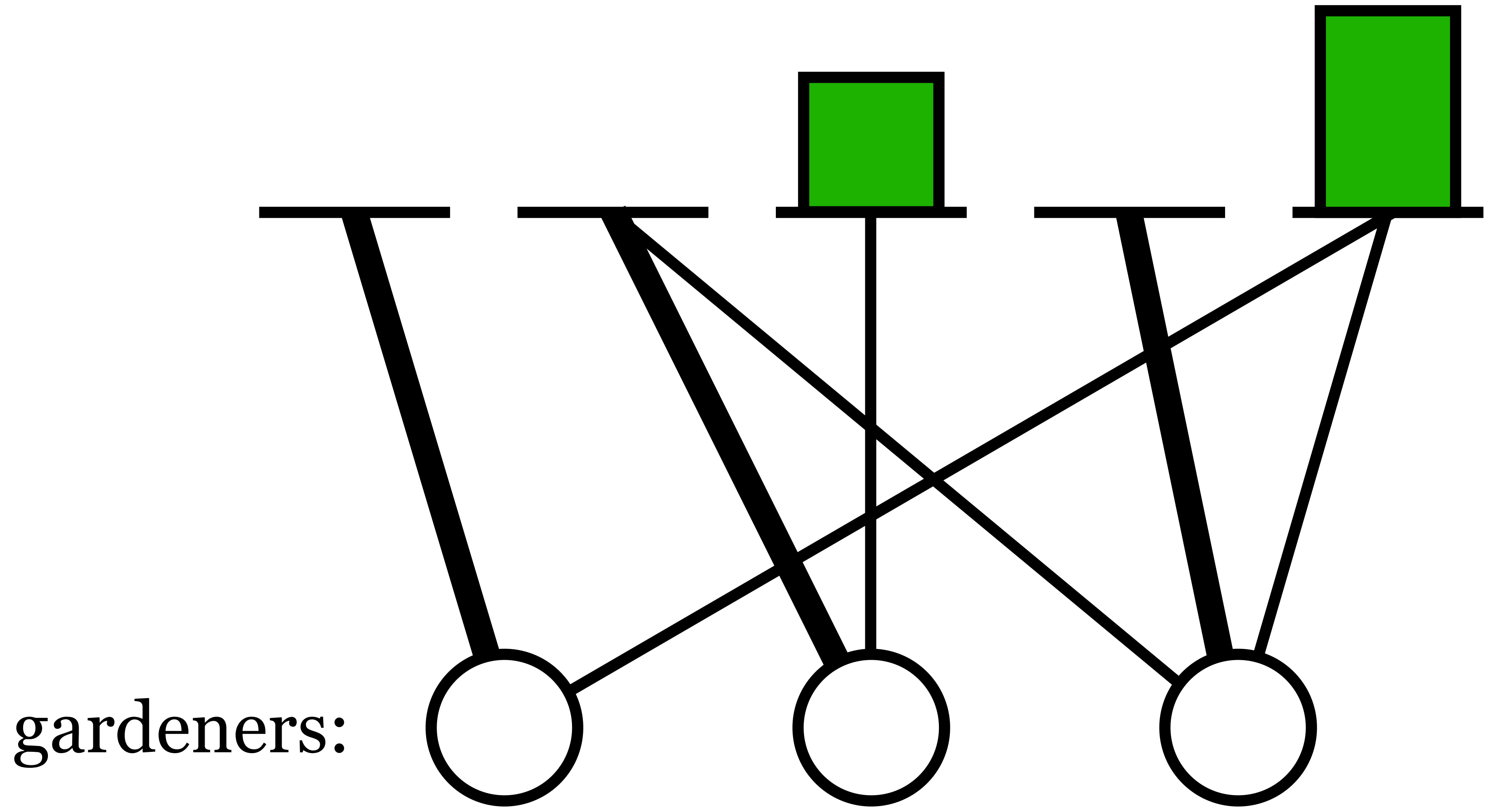


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathcal{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathcal{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.

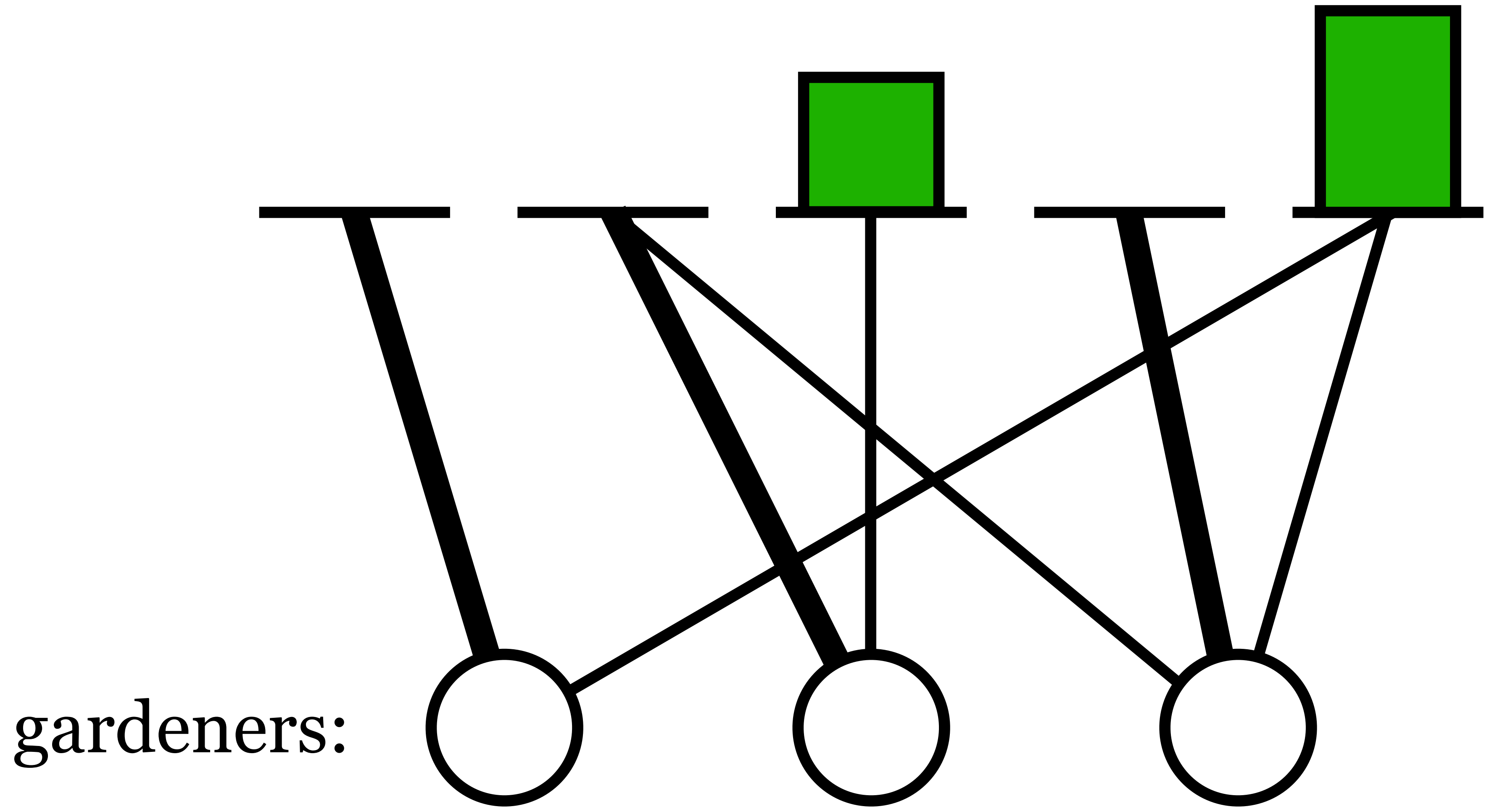


The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathscr{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



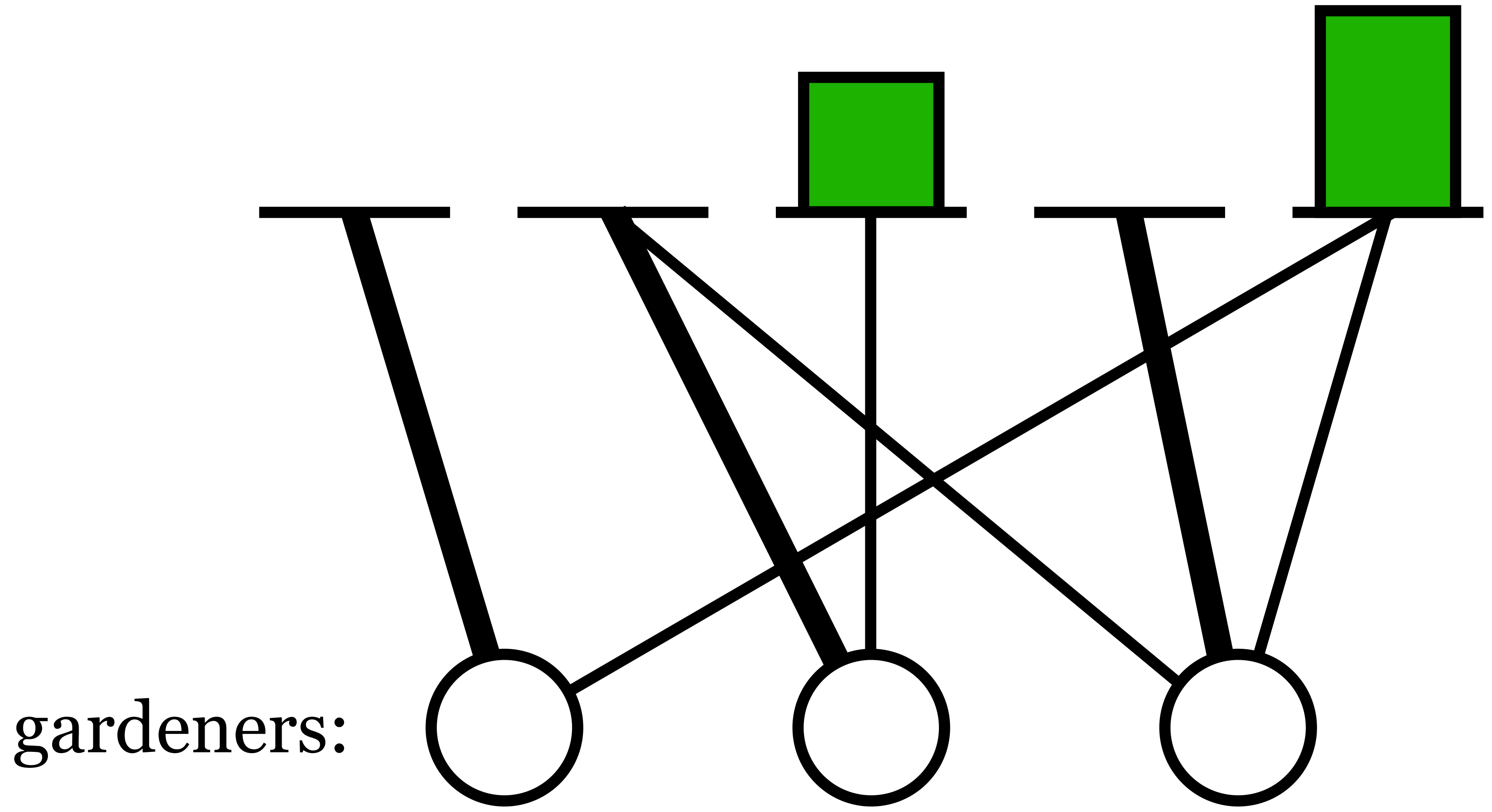
Question:

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathcal{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathcal{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



Question:

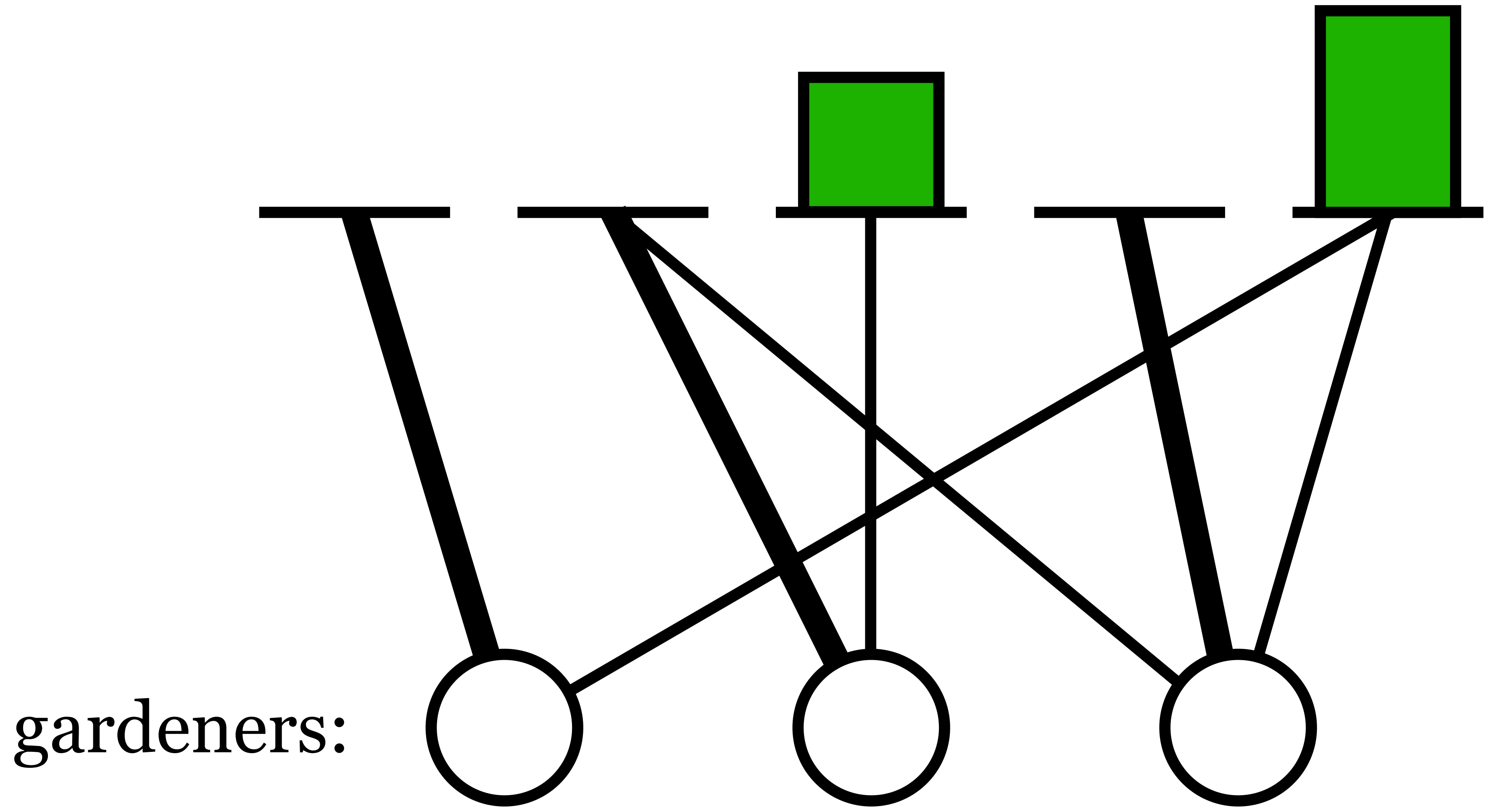
- What height can be guaranteed for all instances?

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathscr{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathscr{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



Question:

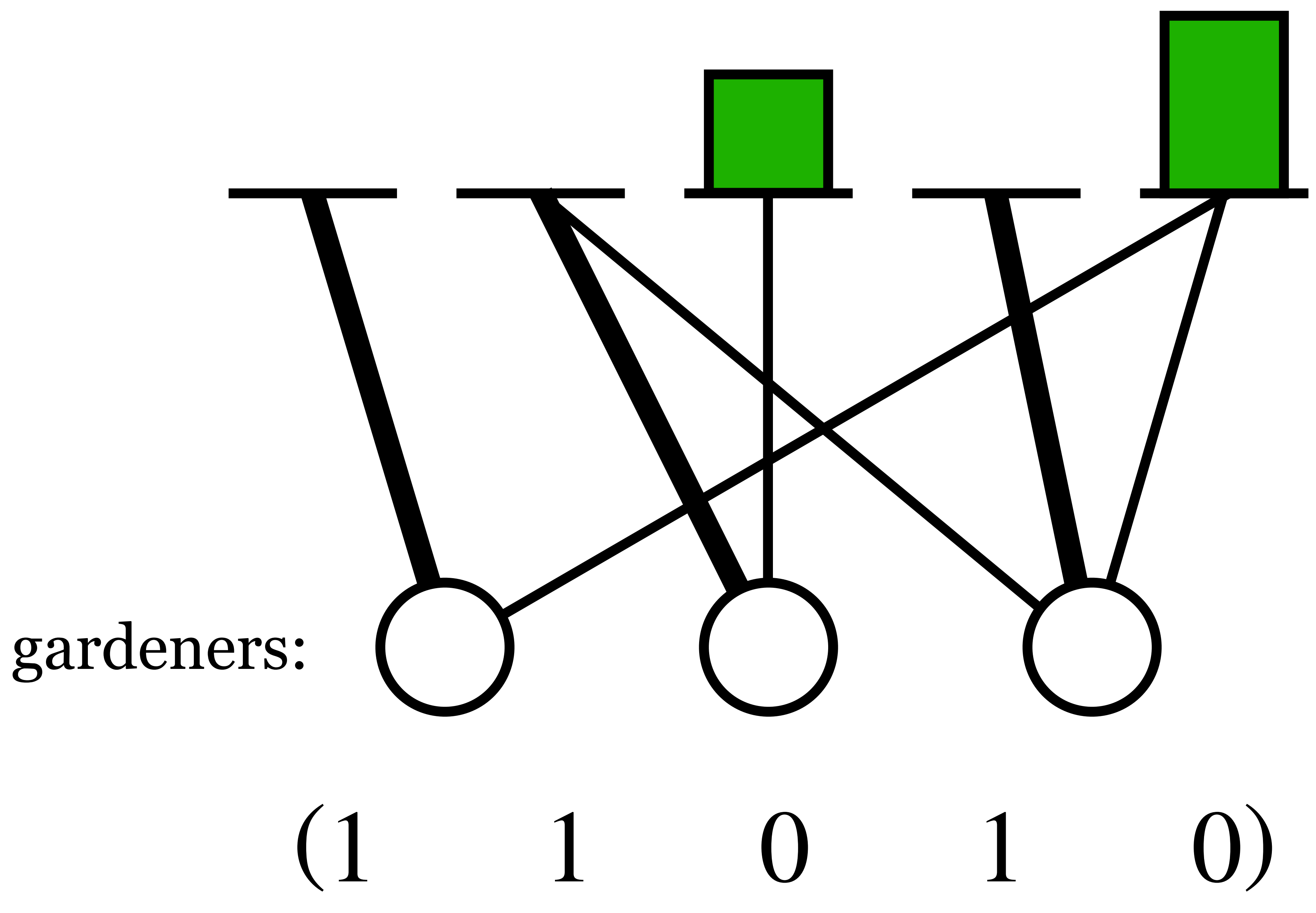
- What height can be guaranteed for all instances?
- Normalization: $\forall e : g(e) = \sum_{I \in \mathscr{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathscr{I}} \lambda(I) = 1$.

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathcal{I})$.
- At each discrete time step (forever):
    - Each bamboo *grows* by $g(e)$ (constant over time).
    - Some bamboos $I \in \mathcal{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



Question:

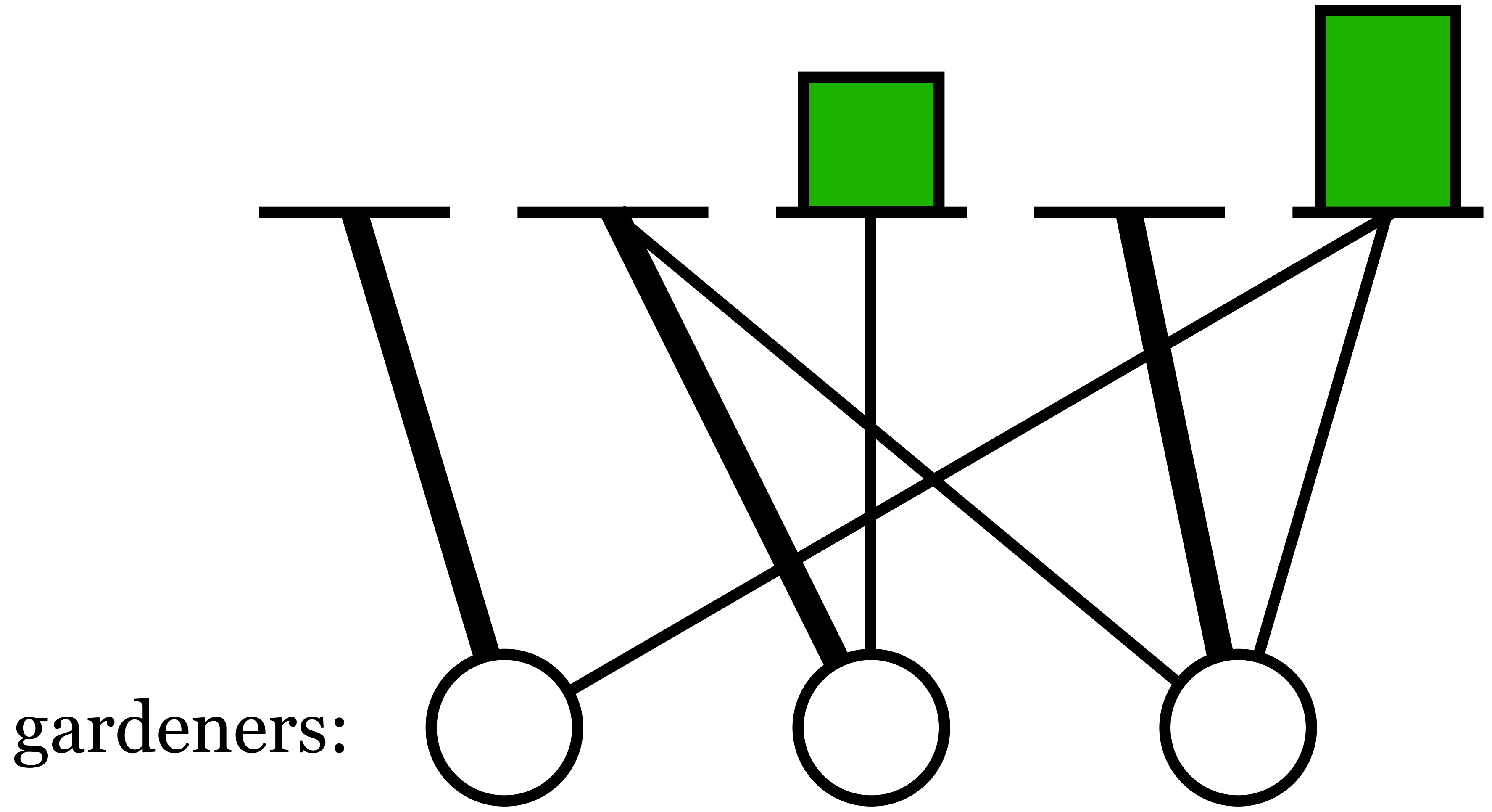
- What height can be guaranteed for all instances?
- Normalization: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- n *bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathcal{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathcal{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



$$\begin{matrix}(1 & 1 & 0 & 1 & 0)\\(0 & 0 & 1 & 1 & 1)\\(1 & 0 & 0 & 1 & 1)\end{matrix}$$

Question:

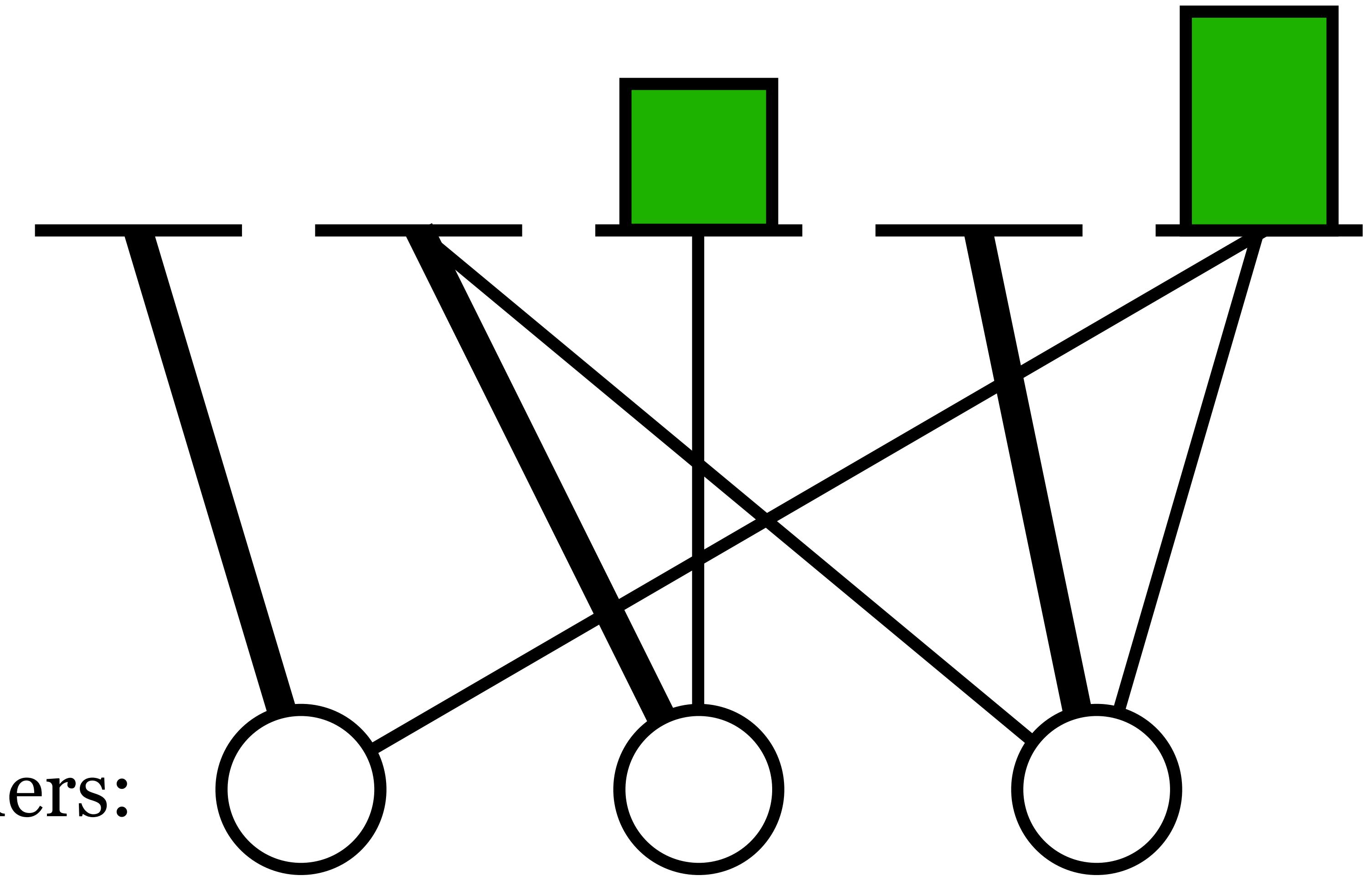
- What height can be guaranteed for all instances?
- Normalization: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathcal{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathcal{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



gardeners:

$$\begin{array}{l} 0.1 \times (1 \quad 1 \quad 0 \quad 1 \quad 0) \\ 0.6 \times (0 \quad 0 \quad 1 \quad 1 \quad 1) \\ 0.3 \times (1 \quad 0 \quad 0 \quad 1 \quad 1) \end{array}$$

Question:

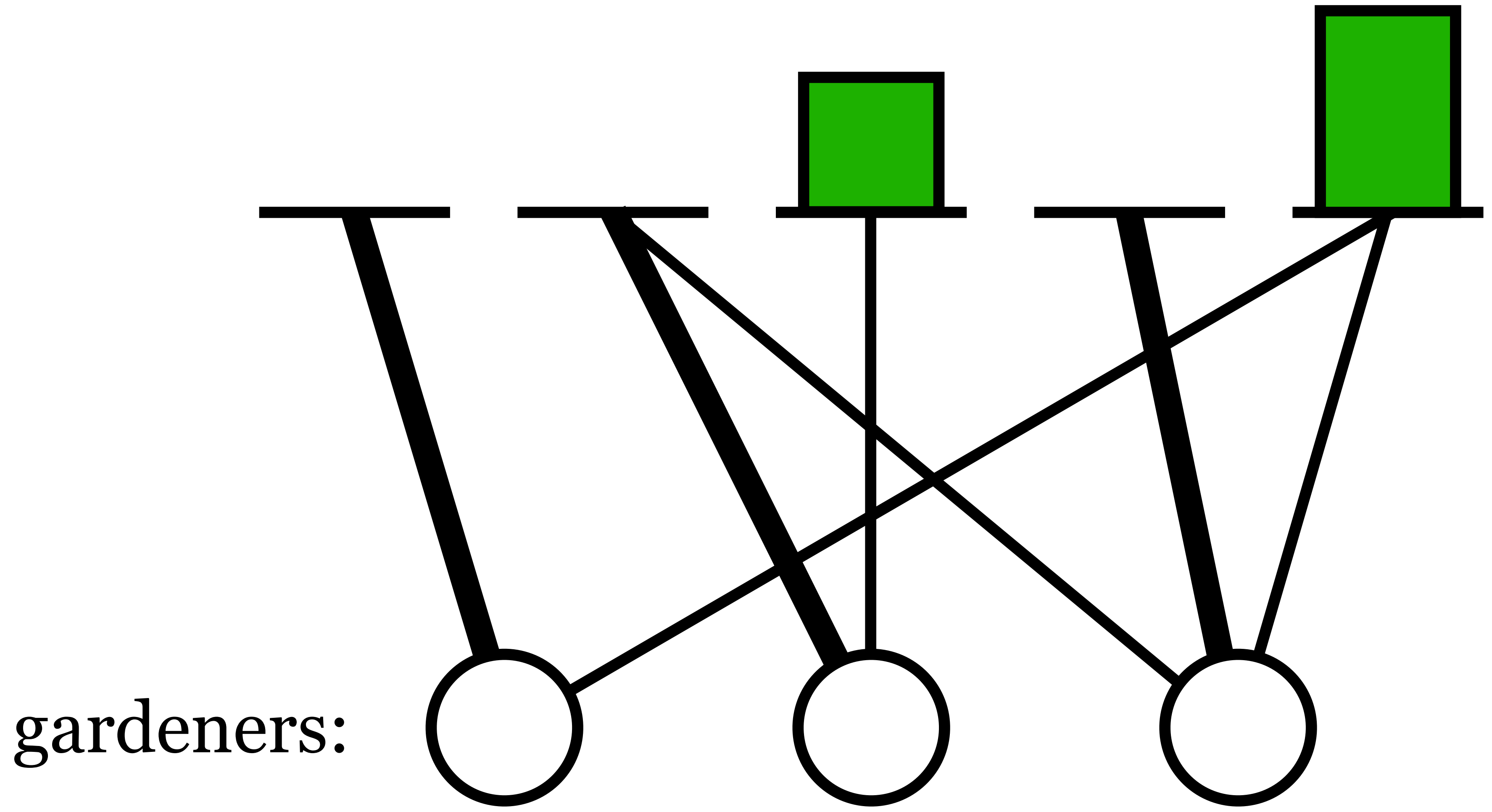
- What height can be guaranteed for all instances?
- Normalization: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.

The Combinatorial Version

Model:

see also [Cicerone et al., CIAC'17]

- *n bamboos* (or tasks), initially of *height* (or urgency) 0.
- A (downward-closed) set system $([n], \mathcal{I})$.
- At each discrete time step (forever):
  - Each bamboo *grows* by $g(e)$ (constant over time).
  - Some bamboos $I \in \mathcal{I}$ may be *cut* down to 0.
- Goal: minimize maximum height that ever occurs.



$$\begin{array}{rccccc}
0.1 \times & (1 & 1 & 0 & 1 & 0) \\
0.6 \times & (0 & 0 & 1 & 1 & 1) \\
0.3 \times & (1 & 0 & 0 & 1 & 1) \\
\hline
g: & (0.4 & 0.1 & 0.6 & 1 & 0.9)
\end{array}$$

Question:

- What height can be guaranteed for all instances?
- Normalization: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.

The Combinatorial Version: Related Work

The Combinatorial Version: Related Work

- Originally introduced in the context of approximation algorithms. [Cicerone et al., CIAC'17]

The Combinatorial Version: Related Work

- Originally introduced in the context of approximation algorithms. [Cicerone et al., CIAC'17]
- When E is ground set, and $\mathscr{I}$ is set of matchings in given graph (V, E), then problem has been considered under the name *polyamorous scheduling*. [Gasieniec et al., FUN'24]

The Combinatorial Version: Related Work

- Originally introduced in the context of approximation algorithms. [Cicerone et al., CIAC'17]
- When E is ground set, and $\mathscr{I}$ is set of matchings in given graph (V, E), then problem has been considered under the name *polyamorous scheduling*. [Gasieniec et al., FUN'24]
    - Normalization is the same on bipartite graphs.

The Combinatorial Version: Related Work

- Originally introduced in the context of approximation algorithms. [Cicerone et al., CIAC'17]
- When E is ground set, and $\mathscr{I}$ is set of matchings in given graph (V, E), then problem has been considered under the name *polyamorous scheduling*. [Gasieniec et al., FUN'24]
    - Normalization is the same on bipartite graphs.
    - Normalization is the same up to factor $3/2$ on general graphs.

The Combinatorial Version: Related Work

- Originally introduced in the context of approximation algorithms. [Cicerone et al., CIAC'17]
- When E is ground set, and $\mathscr{I}$ is set of matchings in given graph (V, E), then problem has been considered under the name *polyamorous scheduling*. [Gasieniec et al., FUN'24]
  - Normalization is the same on bipartite graphs.
  - Normalization is the same up to factor $3/2$ on general graphs.
  - Achievable height is known to lie between 2 (trivial) and 4. [Biktairov et al., SOSA'25]

The Combinatorial Version: Related Work

- Originally introduced in the context of approximation algorithms. [Cicerone et al., CIAC'17]
- When E is ground set, and $\mathscr{I}$ is set of matchings in given graph (V, E), then problem has been considered under the name *polyamorous scheduling*. [Gasieniec et al., FUN'24]
  - Normalization is the same on bipartite graphs.
  - Normalization is the same up to factor $3/2$ on general graphs.
  - Achievable height is known to lie between 2 (trivial) and 4. [Biktairov et al., SOSA'25]
- Similar extensions have, e.g., also been considered for variants such as the adversarial variant (cup games). [Im et al., Oper. Res. Lett. 2021]

A Lower Bound

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

$$\begin{pmatrix} 0 & 1 & & 1 & 1 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & & 1 & 0 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & & 0 & 1 \end{pmatrix} \quad 2k \text{ allowed sets}$$

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

all length-$2k$ bitstrings
with precisely k ones

$$\begin{pmatrix} 0 & 1 & \cdots & 1 & 1 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & & 1 & 0 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & & 0 & 1 \end{pmatrix}$$

$2k$ allowed sets

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

$$\begin{pmatrix} 0 & 1 & & 1 & 1 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & & 1 & 0 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & & 0 & 1 \end{pmatrix}$$

all length-$2k$ bitstrings with precisely k ones

$2k$ allowed sets

(take downward closure)

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

all length-$2k$ bitstrings with precisely k ones

$$\begin{array}{r} 1/(2k) \times \\ 1/(2k) \times \\ 1/(2k) \times \\ \vdots \\ 1/(2k) \times \\ 1/(2k) \times \\ 1/(2k) \times \end{array} \begin{pmatrix} 0 & 1 & & 1 & 1 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & & 1 & 0 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & & 0 & 1 \end{pmatrix}$$

$2k$ allowed sets

(take downward closure)

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

all length-$2k$ bitstrings with precisely k ones

$$\begin{array}{r}
1/(2k) \times \\
1/(2k) \times \\
1/(2k) \times \\
\vdots \\
1/(2k) \times \\
1/(2k) \times \\
1/(2k) \times
\end{array}
\begin{pmatrix}
0 & 1 & & 1 & 1 \\
1 & 0 & \cdots & 1 & 0 \\
0 & 0 & & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
1 & 0 & & 1 & 0 \\
1 & 0 & \cdots & 1 & 0 \\
0 & 1 & & 0 & 1
\end{pmatrix}$$

$2k$ allowed sets

(take downward closure)

$$g: \quad (1/2 \quad 1/2 \quad \cdots \quad 1/2 \quad 1/2)$$

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

all length-$2k$ bitstrings with precisely k ones

$$\begin{array}{r c}
1/(2k)\times & \\
1/(2k)\times & \\
1/(2k)\times & \\
\vdots & \\
1/(2k)\times & \\
1/(2k)\times & \\
1/(2k)\times & \\
\hline
g: & (1/2 \quad 1/2 \quad \cdots \quad 1/2 \quad 1/2)
\end{array}
\begin{pmatrix}
0 & 1 & & 1 & 1 \\
1 & 0 & \cdots & 1 & 0 \\
0 & 0 & & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
1 & 0 & & 1 & 0 \\
1 & 0 & \cdots & 1 & 0 \\
0 & 1 & & 0 & 1
\end{pmatrix}$$

$2k$ allowed sets

(take downward closure)

For any schedule of length k exists element that is not cut.

A Lower Bound

$$n = \binom{2k}{k} \text{ bamboos}$$

all length-$2k$ bitstrings with precisely k ones

$$\begin{array}{r} 1/(2k)\times \\ 1/(2k)\times \\ 1/(2k)\times \\ \vdots \\ 1/(2k)\times \\ 1/(2k)\times \\ 1/(2k)\times \end{array} \begin{pmatrix} 0 & 1 & & 1 & 1 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & & 1 & 0 \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & & 0 & 1 \end{pmatrix}$$

$$g: \quad (1/2 \quad 1/2 \quad \cdots \quad 1/2 \quad 1/2)$$

$2k$ allowed sets

(take downward closure)

For any schedule of length k exists element that is not cut. $\Rightarrow$ Height $k/2 \in \Theta(\log n)$ reached.

Can We Match the $\Omega(\log n)$ Lower Bound?

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathscr{I}$ with $e \in I$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathcal{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathscr{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.
  - $E_{\text{fast}} := \{e \in E \mid g(e) > \tau\}$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathcal{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.
  - $E_{\text{fast}} := \{e \in E \mid g(e) > \tau\}$.
    - Interpret λ as probability distribution, and draw every set independently from it.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \le \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathscr{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.
  - $E_{\text{fast}} := \{e \in E \mid g(e) > \tau\}$.
    - Interpret λ as probability distribution, and draw every set independently from it.
    - $\Pr[e \in E_{\text{fast}}$ exceeds height $3 \ln n$ at $t] \le (1 - g(e))^{3 \ln n / g(e)} \le e^{-3 \ln n} \le n^{-3}$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathcal{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.
  - $E_{\text{fast}} := \{e \in E \mid g(e) > \tau\}$.
    - Interpret λ as probability distribution, and draw every set independently from it.
    - $\Pr[e \in E_{\text{fast}}$ exceeds height $3 \ln n$ at $t] \leq (1 - g(e))^{3 \ln n / g(e)} \leq e^{-3 \ln n} \leq n^{-3}$.
    - $\Pr[$any element exceeds height $3 \ln n$ in n time steps$] \leq n \cdot n \cdot n^{-3} = n^{-1} < 1$.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathcal{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.
  - $E_{\text{fast}} := \{e \in E \mid g(e) > \tau\}$.
    - Interpret λ as probability distribution, and draw every set independently from it.
    - Pr[$e \in E_{\text{fast}}$ exceeds height $3 \ln n$ at t] $\leq (1 - g(e))^{3 \ln n / g(e)} \leq e^{-3 \ln n} \leq n^{-3}$.
    - Pr[any element exceeds height $3 \ln n$ in n time steps] $\leq n \cdot n \cdot n^{-3} = n^{-1} < 1$.
    - So length-n height-$(3 \ln n)$ schedule exists; repeat forever.

Can We Match the $\Omega(\log n)$ Lower Bound?

Approach:

see also [Cicerone et al., CIAC'17]

- Define $\tau := 3 \cdot (\ln n)/n$.
- Partition E into the following two sets, and handle them at alternating time steps:
  - $E_{\text{slow}} := \{e \in E \mid g(e) \leq \tau\}$.
    - Consider elements $e \in E_{\text{slow}}$ one by one and choose arbitrary set $I \in \mathcal{I}$ with $e \in I$.
    - Bamboos will only grow to $O(\log n)$.
  - $E_{\text{fast}} := \{e \in E \mid g(e) > \tau\}$.
    - Interpret λ as probability distribution, and draw every set independently from it.
    - $\Pr[e \in E_{\text{fast}}$ exceeds height $3 \ln n$ at $t] \leq (1 - g(e))^{3 \ln n / g(e)} \leq e^{-3 \ln n} \leq n^{-3}$.
    - $\Pr[$any element exceeds height $3 \ln n$ in n time steps$] \leq n \cdot n \cdot n^{-3} = n^{-1} < 1$.
    - So length-n height-$(3 \ln n)$ schedule exists; repeat forever.

Conclusion: Can guarantee precisely $\Theta(\log n)$.

Special Set Systems

Special Set Systems

Matroids: A set system $(E, \mathscr{I})$ is called a *matroid* if the following hold:

Special Set Systems

Matroids: A set system $(E, \mathscr{I})$ is called a *matroid* if the following hold:

- $\mathscr{I} \neq \emptyset$

Special Set Systems

Matroids: A set system $(E, \mathscr{I})$ is called a *matroid* if the following hold:

- $\mathscr{I} \neq \emptyset$
- For all $I \in \mathscr{I}$ and $I' \subseteq I$, it holds that $I' \in \mathscr{I}$ (downward-closed).

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \backslash I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \backslash I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

- uniform matroid (given integer $k \geq 1$):

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

- uniform matroid (given integer $k \geq 1$):
  - $I \in \mathcal{I}$ if $|I| \leq k$.

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \backslash I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

- uniform matroid (given integer $k \geq 1$):
  - $I \in \mathcal{I}$ if $|I| \leq k$.
- graphic matroid (given undirected graph (V, E)):

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \backslash I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

- uniform matroid (given integer $k \geq 1$):
    - $I \in \mathcal{I}$ if $|I| \leq k$.
- graphic matroid (given undirected graph (V, E)):
    - $I \in \mathcal{I}$ if (V, I) is forest.

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \backslash I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

- uniform matroid (given integer $k \geq 1$):
    - $I \in \mathcal{I}$ if $|I| \leq k$.
- graphic matroid (given undirected graph (V, E)):
    - $I \in \mathcal{I}$ if (V, I) is forest.
- transversal matroid (for given bipartite graph $(E \cup V_2, E')$ with bipartition E, V_2):

Special Set Systems

Matroids: A set system $(E, \mathcal{I})$ is called a *matroid* if the following hold:

- $\mathcal{I} \neq \emptyset$
- For all $I \in \mathcal{I}$ and $I' \subseteq I$, it holds that $I' \in \mathcal{I}$ (downward-closed).
- For all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, there exists $e \in I_2 \backslash I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Some examples:

- uniform matroid (given integer $k \geq 1$):
  - $I \in \mathcal{I}$ if $|I| \leq k$.
- graphic matroid (given undirected graph (V, E)):
  - $I \in \mathcal{I}$ if (V, I) is forest.
- transversal matroid (for given bipartite graph $(E \cup V_2, E')$ with bipartition E, V_2):
  - $I \in \mathcal{I}$ if I can be covered by a matching.

Towards a Better Bound for Matroids

Towards a Better Bound for Matroids

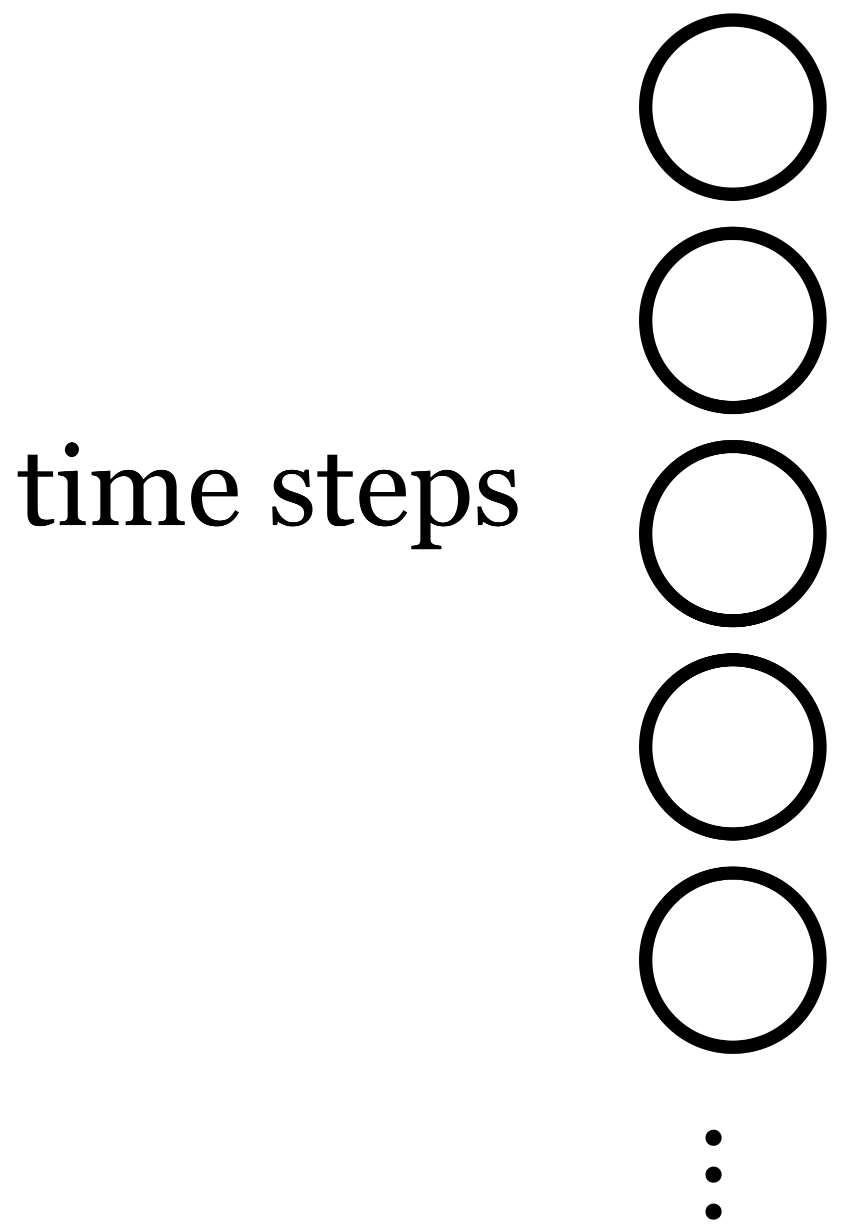
- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.

Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:

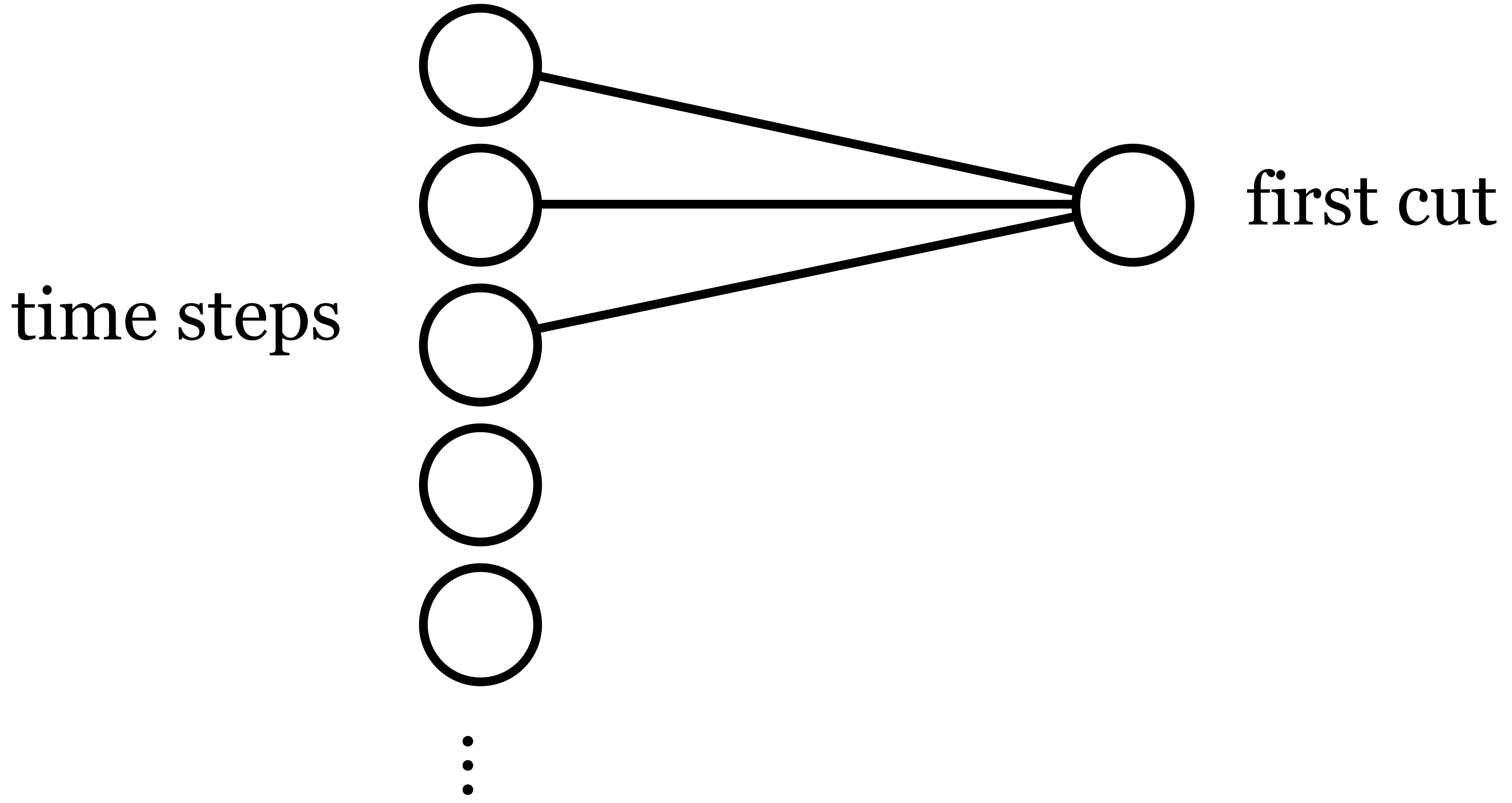
Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:



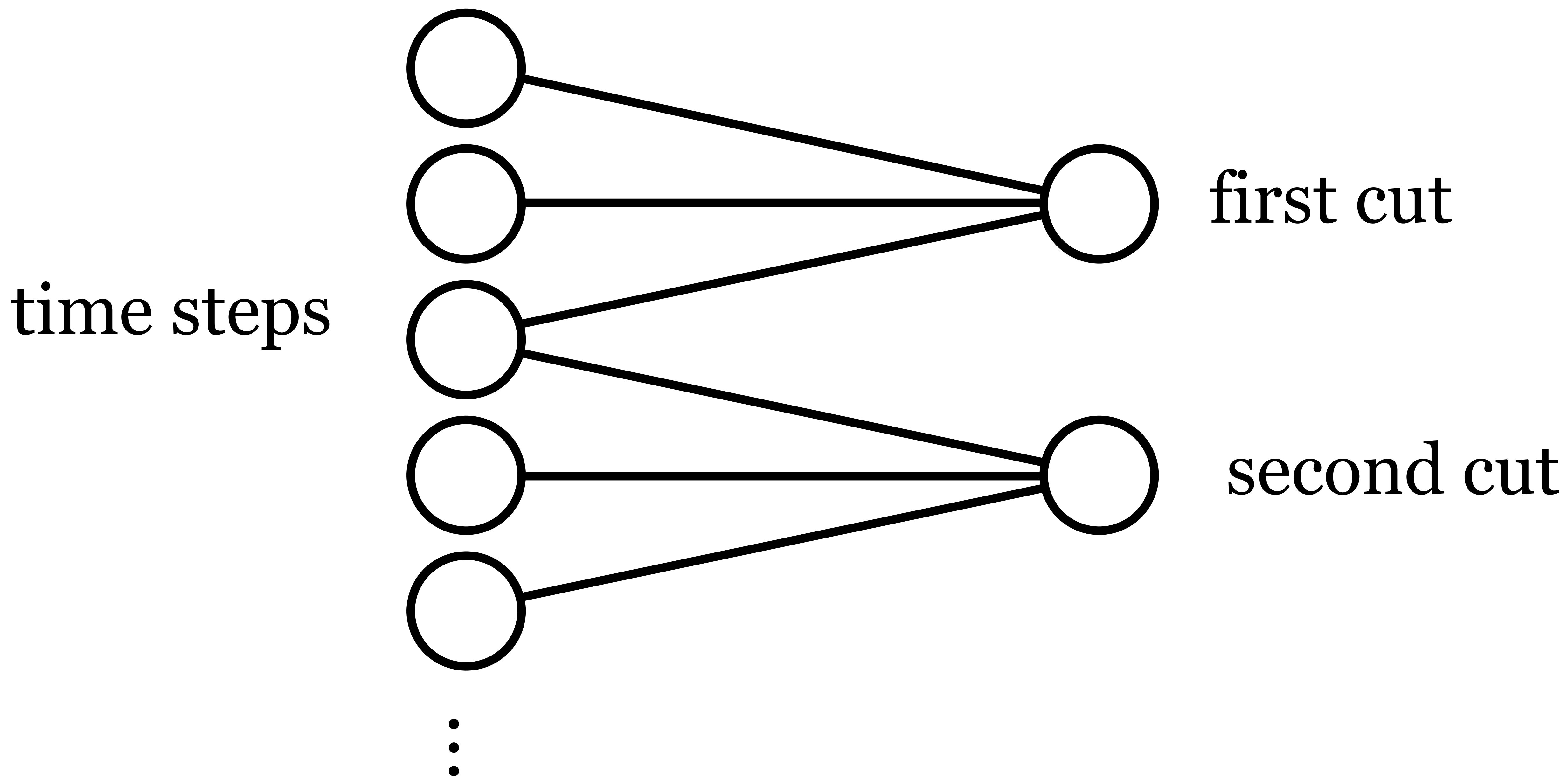
Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:



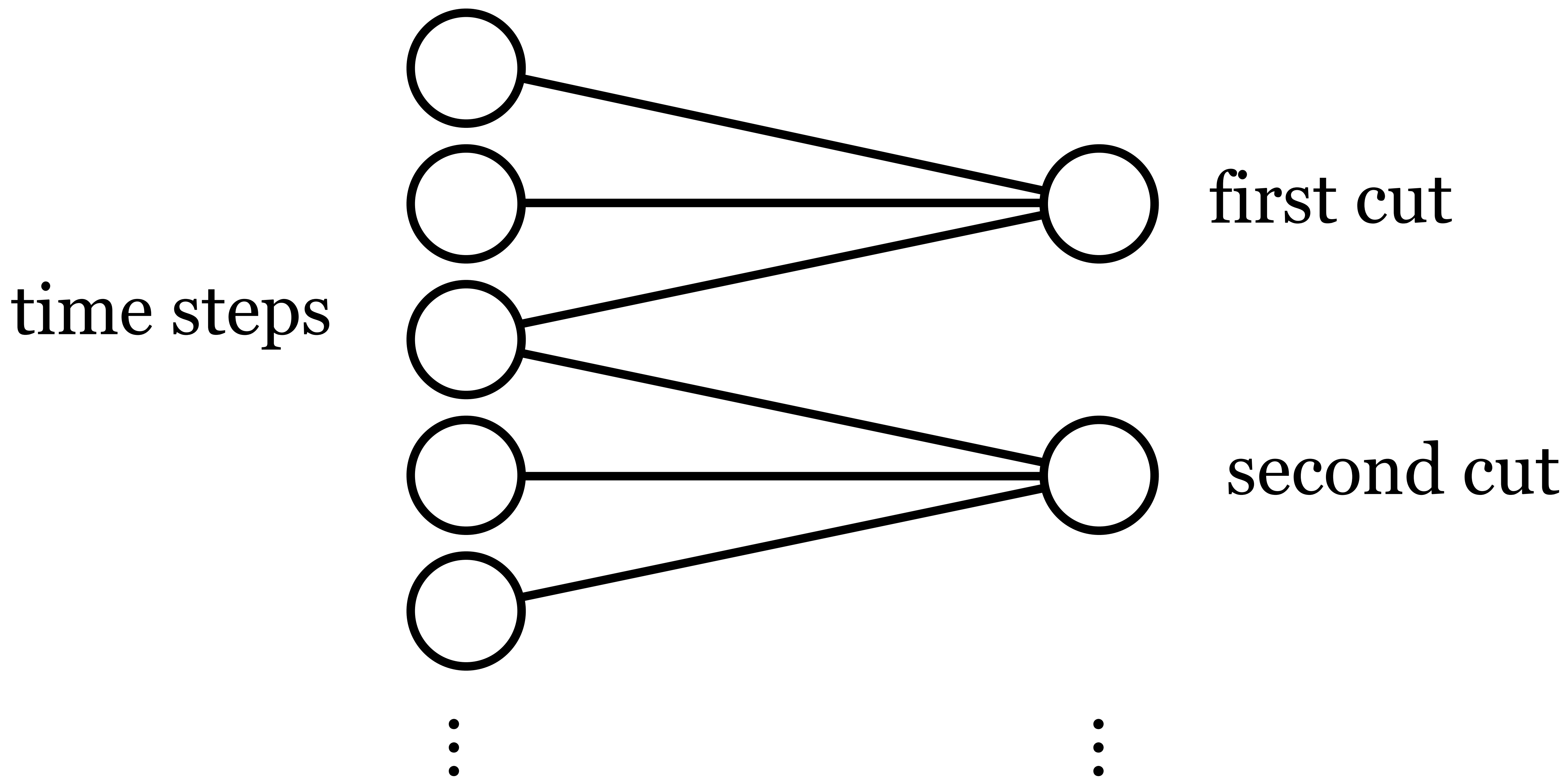
Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:



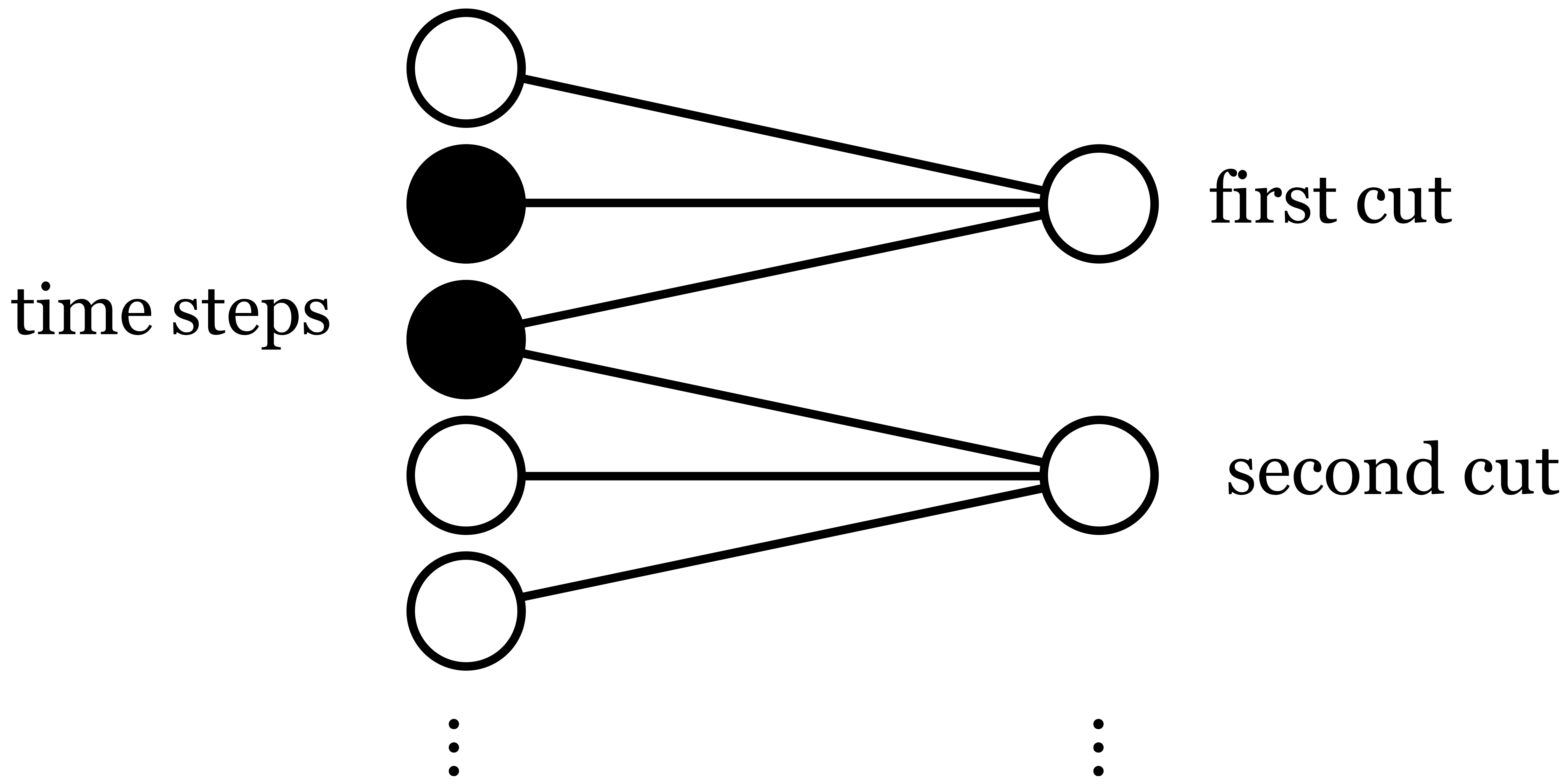
Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:



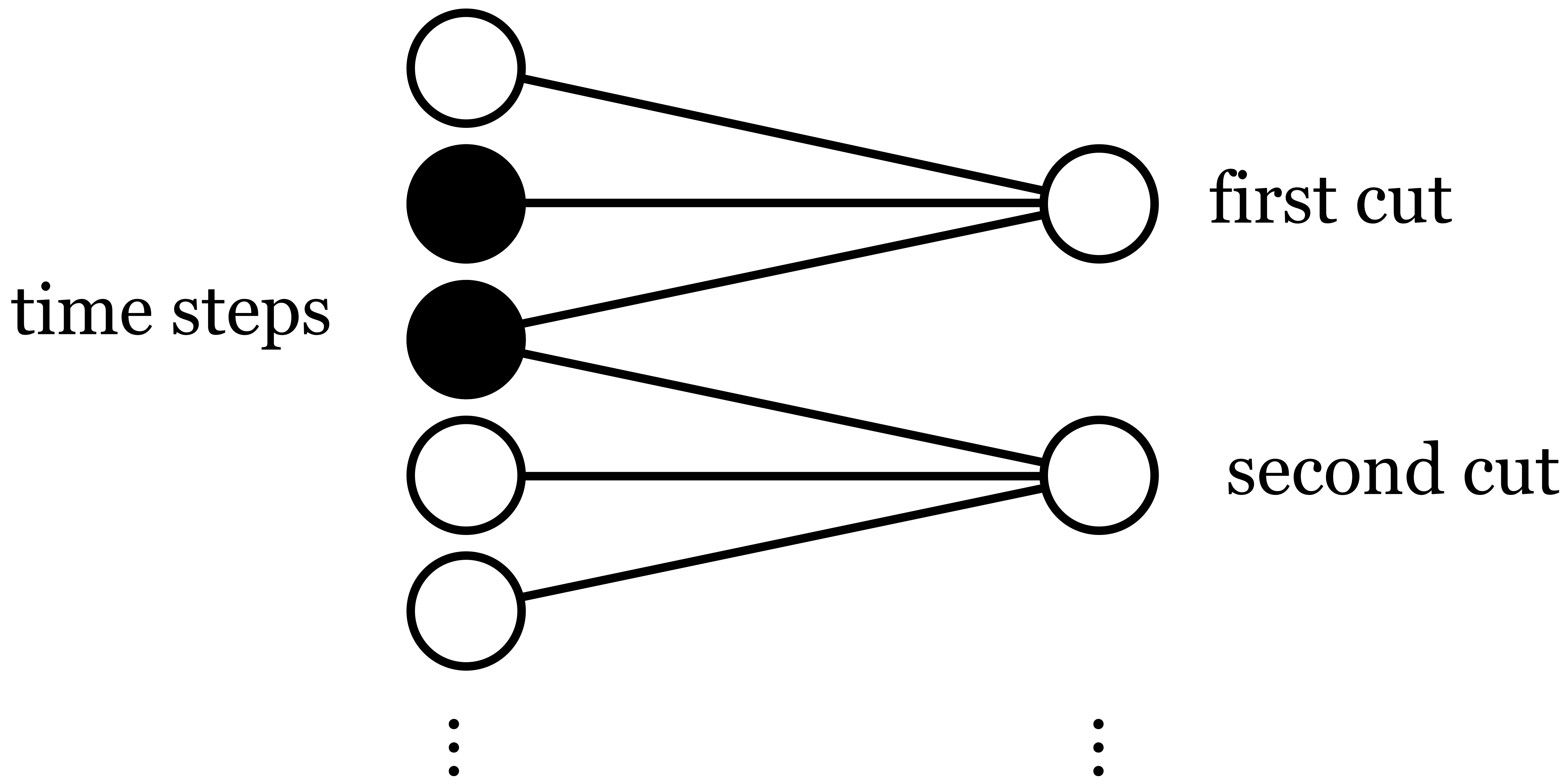
Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:



Towards a Better Bound for Matroids

- **Idea:** Find schedule, such that, for any $e \in E$ and time t, e has been cut ℓ times where $|tg(e) - \ell| < 1$.
- Consider such e:



Thus: Feasible (in above sense) iff we cut according to basis of above transversal matroid M_e.

Towards a Better Bound for Matroids: Proof Sketch

Towards a Better Bound for Matroids: Proof Sketch

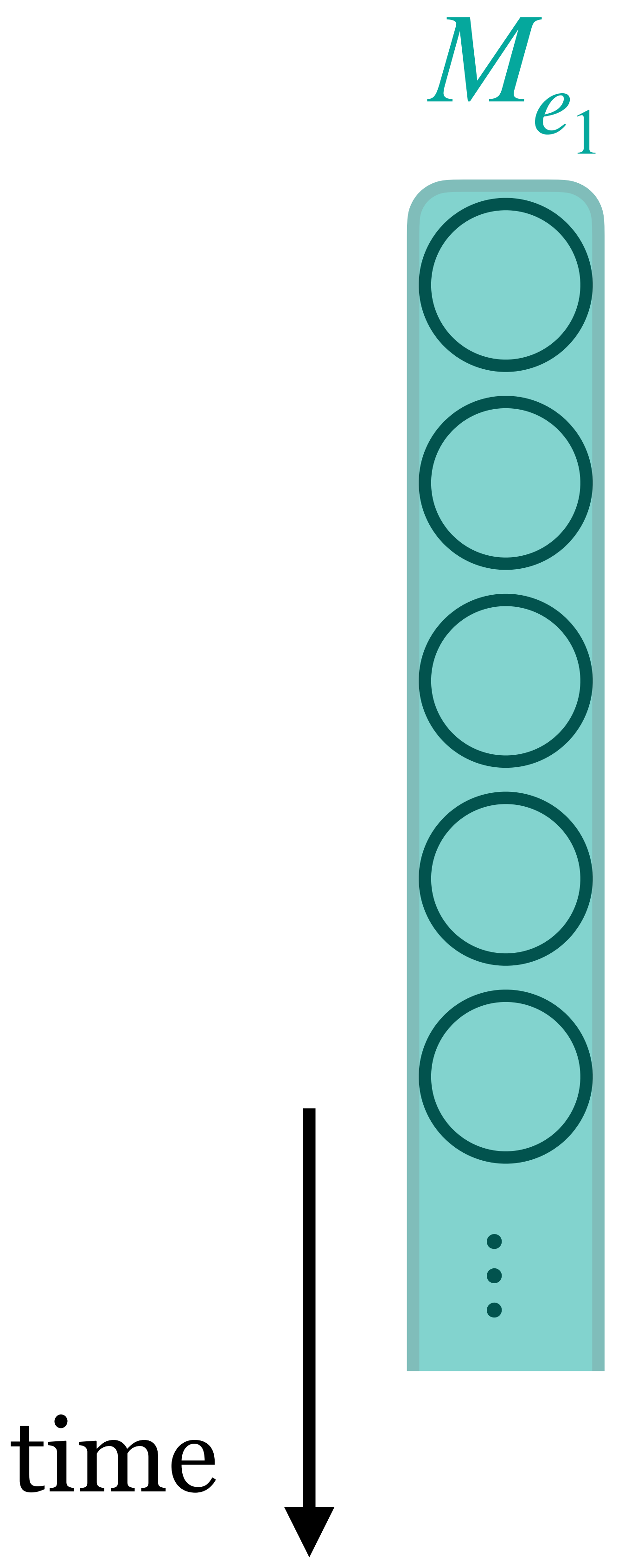
- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.

Towards a Better Bound for Matroids: Proof Sketch

- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.

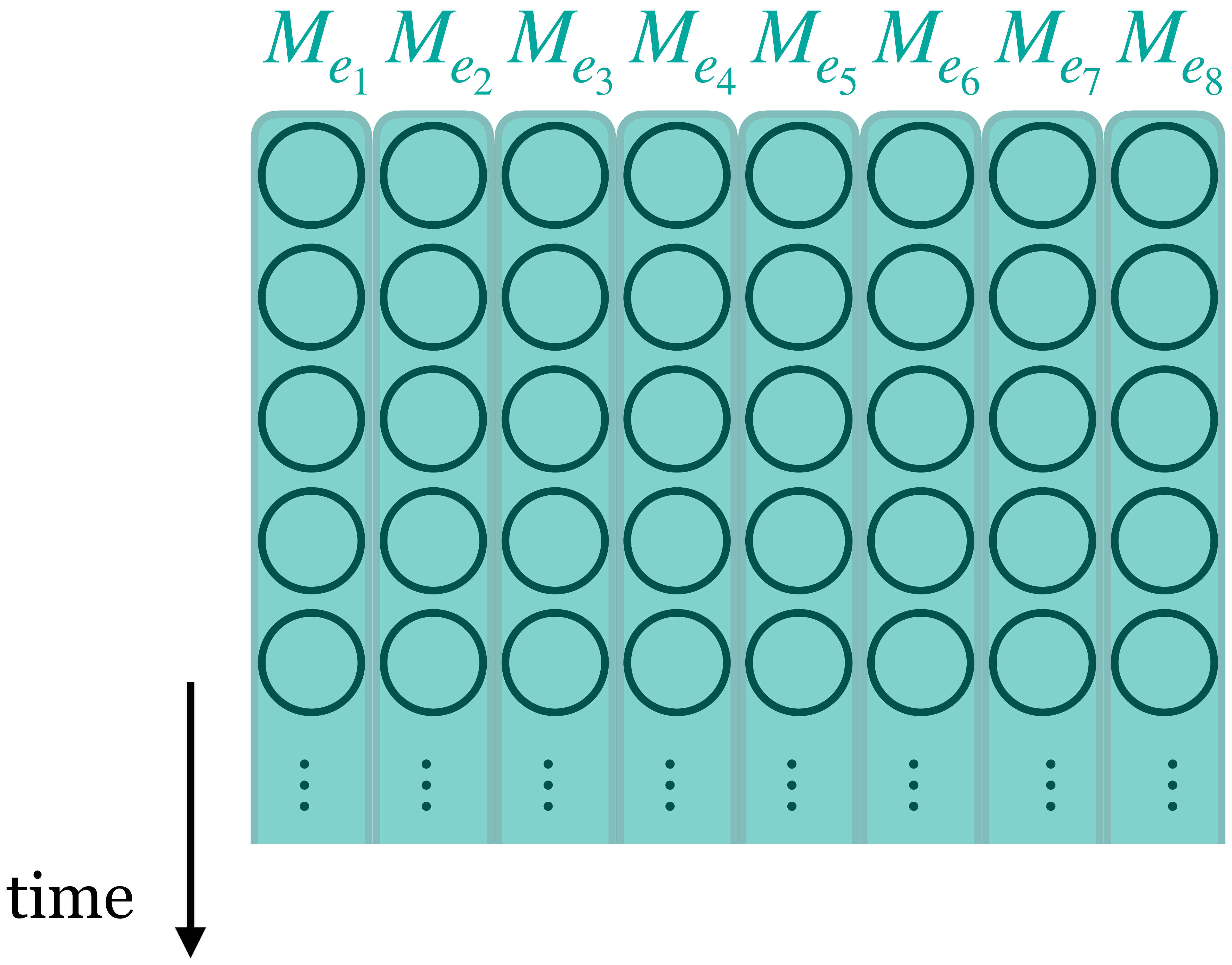
Towards a Better Bound for Matroids: Proof Sketch

- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



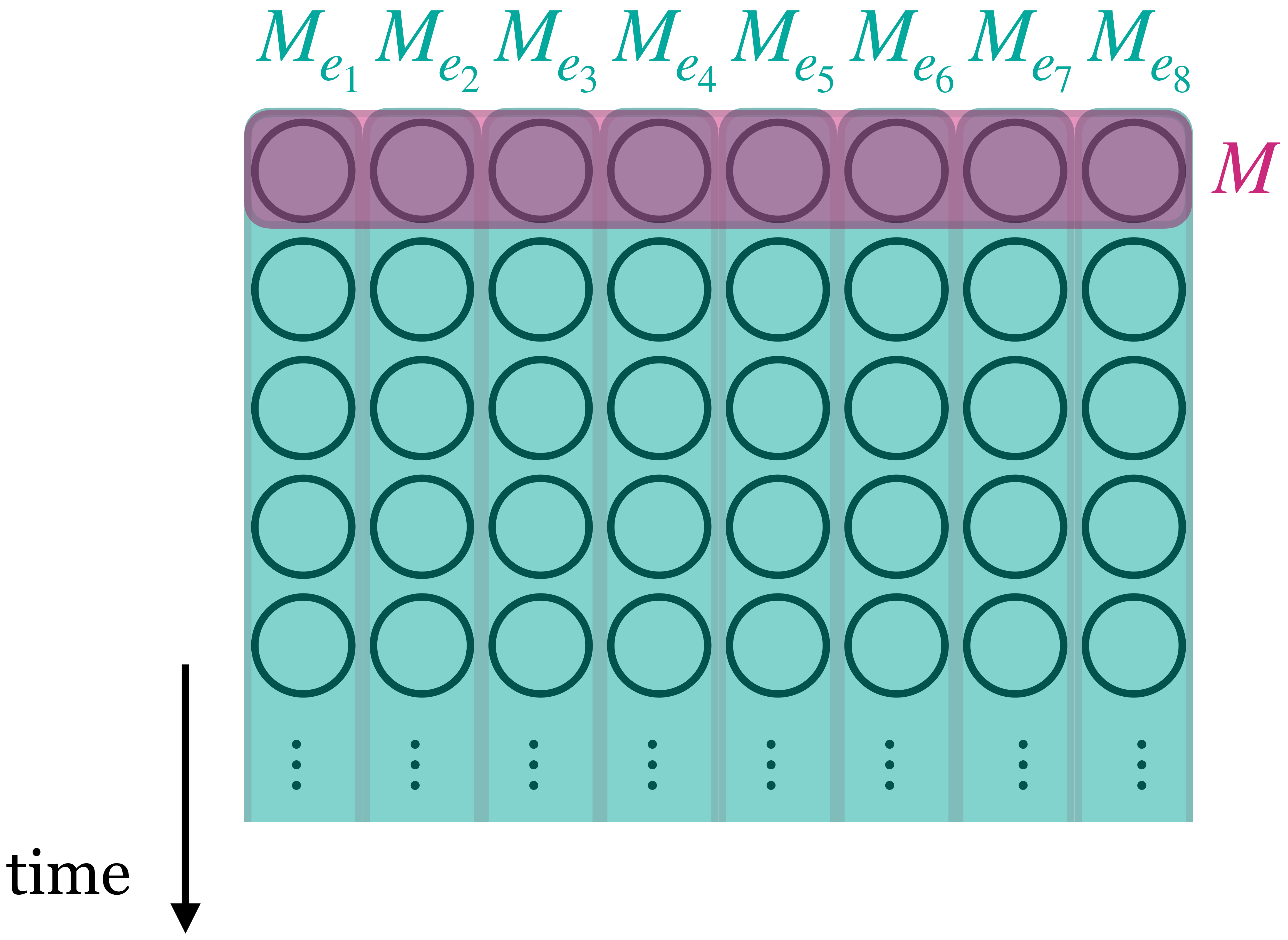
Towards a Better Bound for Matroids: Proof Sketch

- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



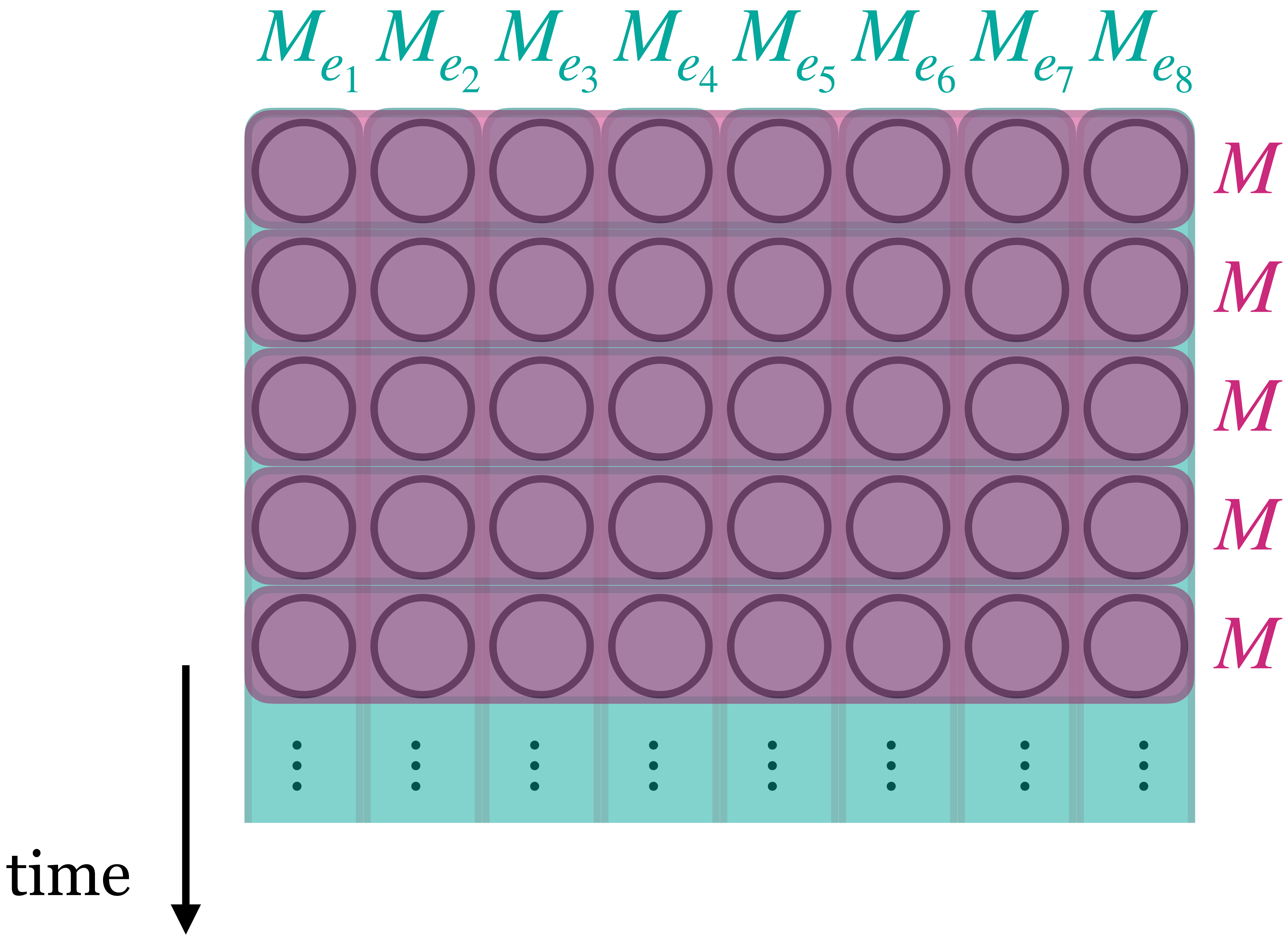
Towards a Better Bound for Matroids: Proof Sketch

- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



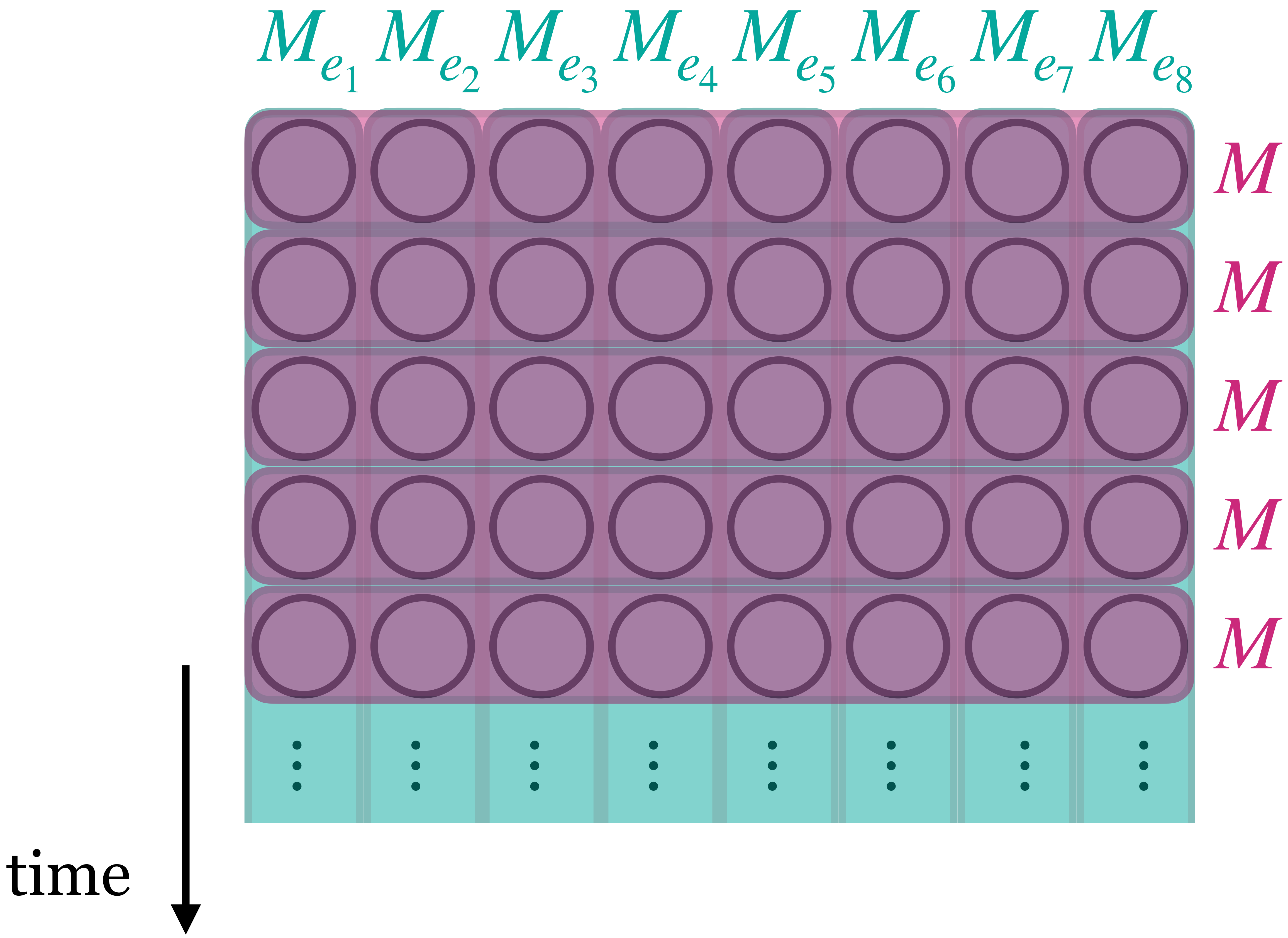
Towards a Better Bound for Matroids: Proof Sketch

- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



Towards a Better Bound for Matroids: Proof Sketch

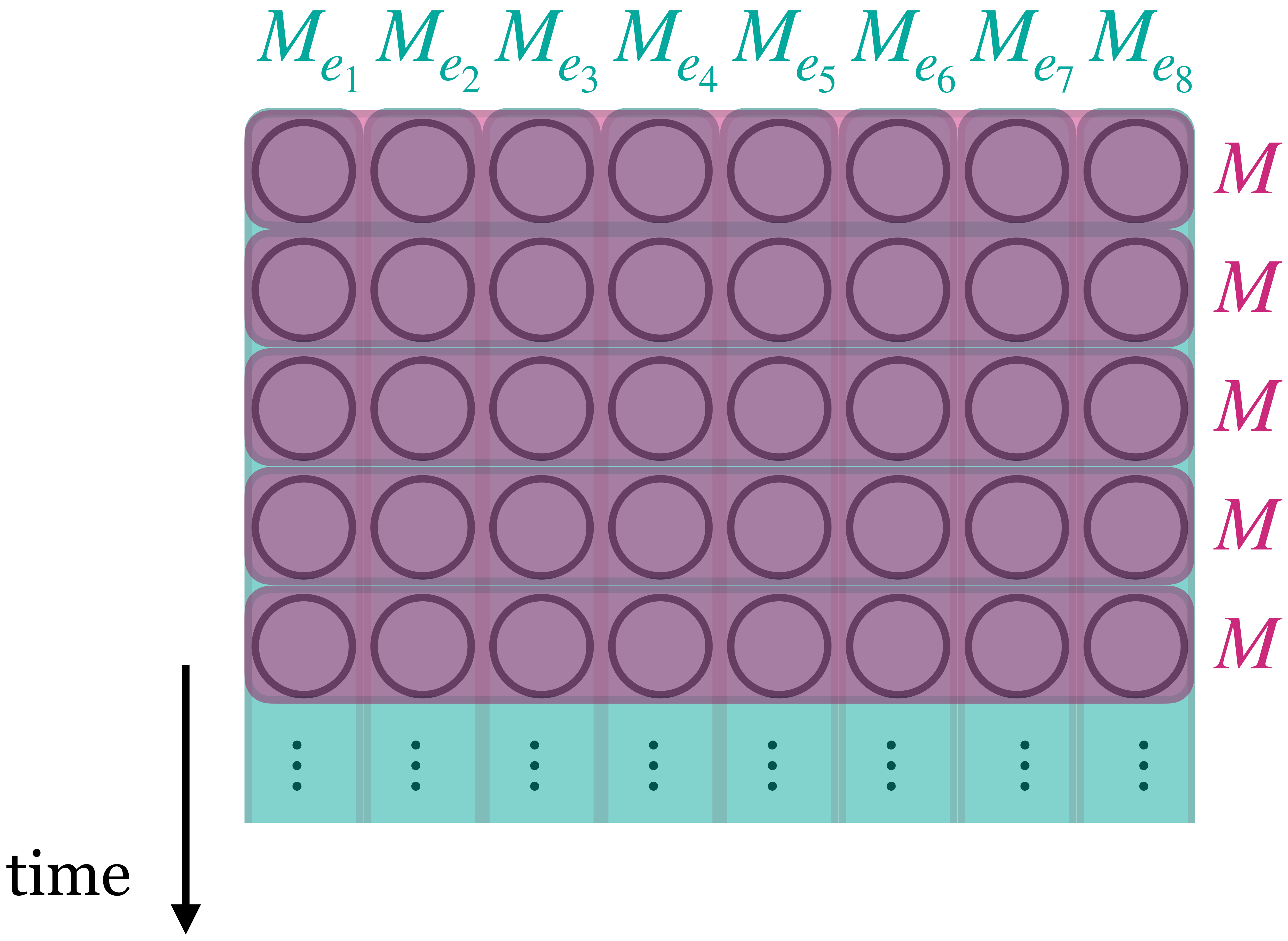
- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



- Because of the above: want common basis.

Towards a Better Bound for Matroids: Proof Sketch

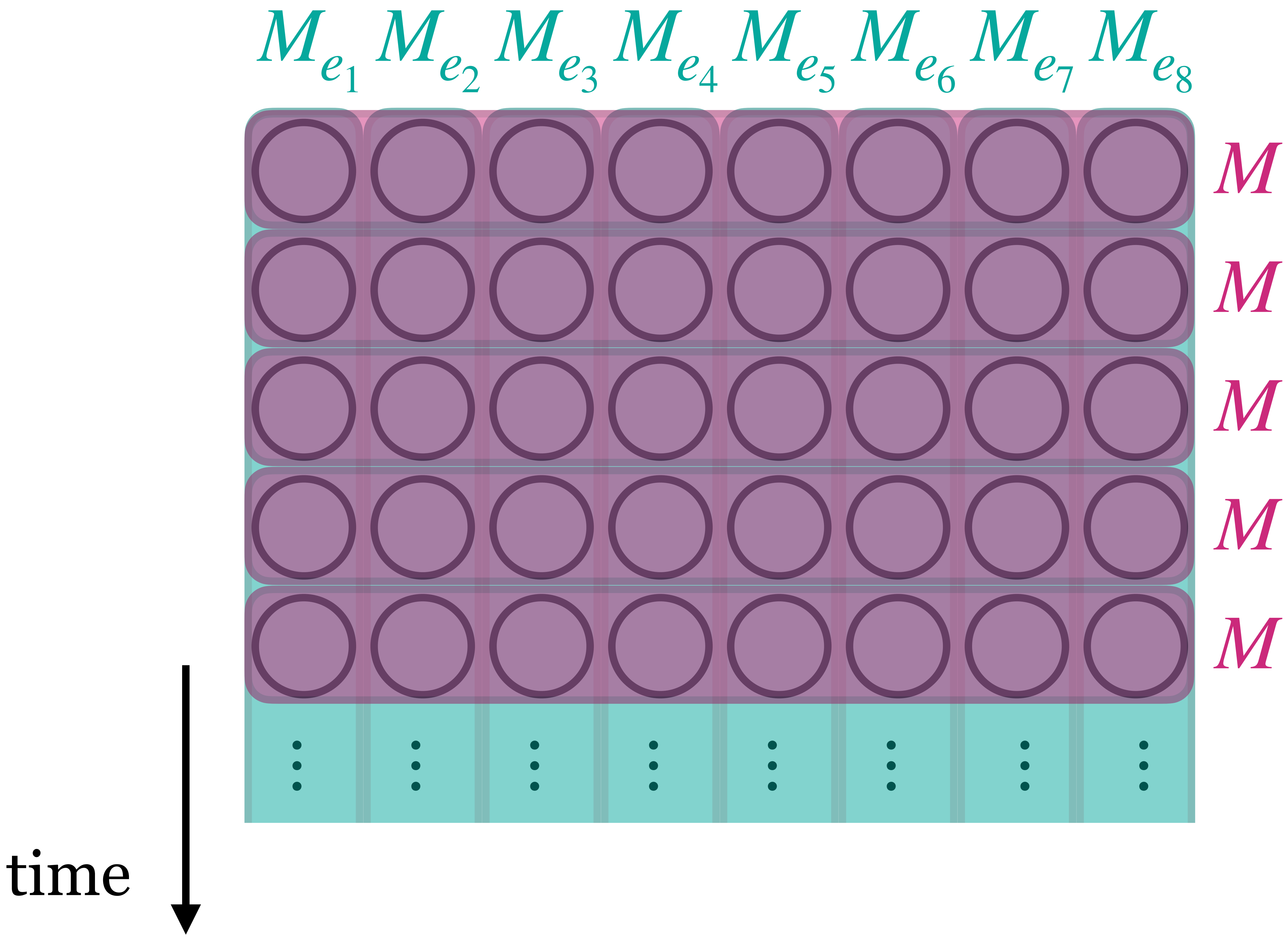
- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



- Because of the above: want common basis.
- Lemma: assigning $g(e)$ to element (t, e) corresponds to point in matroid-intersection polytope.

Towards a Better Bound for Matroids: Proof Sketch

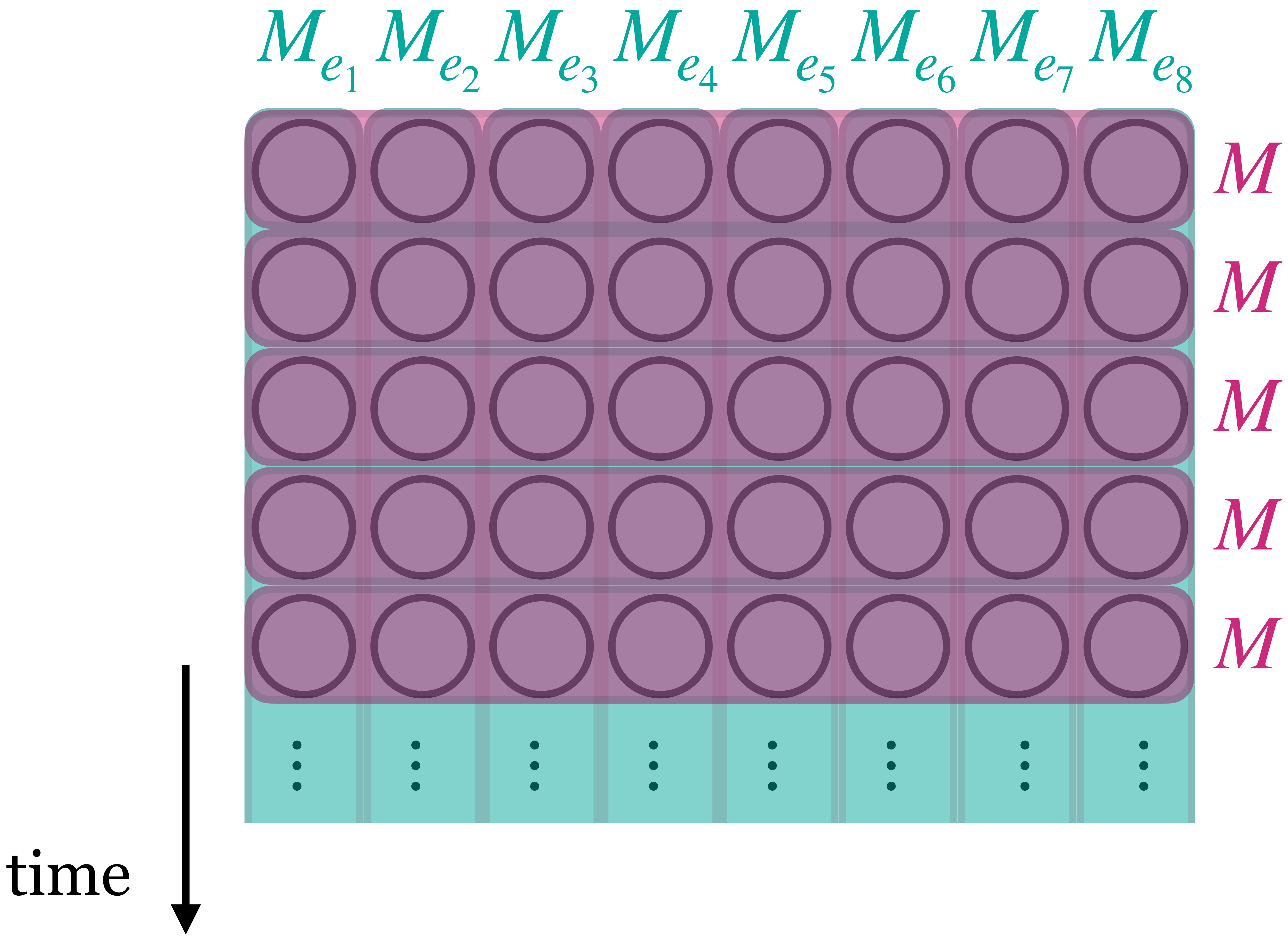
- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



- Because of the above: want common basis.
- Lemma: assigning $g(e)$ to element (t, e) corresponds to point in matroid-intersection polytope.
  - Essentially, use integrality of fractional matching polytope in bipartite graphs.

Towards a Better Bound for Matroids: Proof Sketch

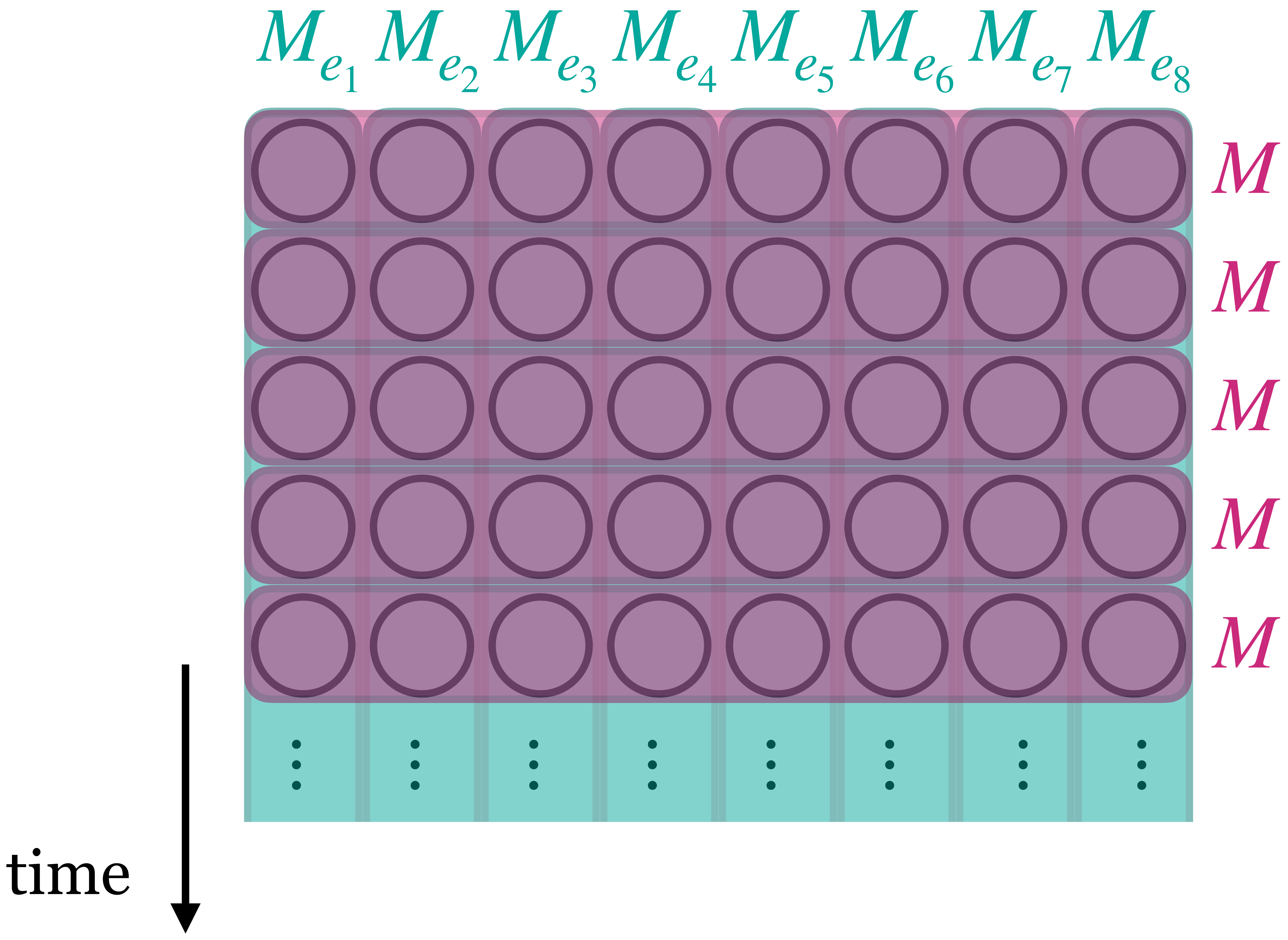
- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



- Because of the above: want common basis.
- Lemma: assigning $g(e)$ to element (t, e) corresponds to point in matroid-intersection polytope.
  - Essentially, use integrality of fractional matching polytope in bipartite graphs.
- Invoke integrality theorem for said polytope.

Towards a Better Bound for Matroids: Proof Sketch

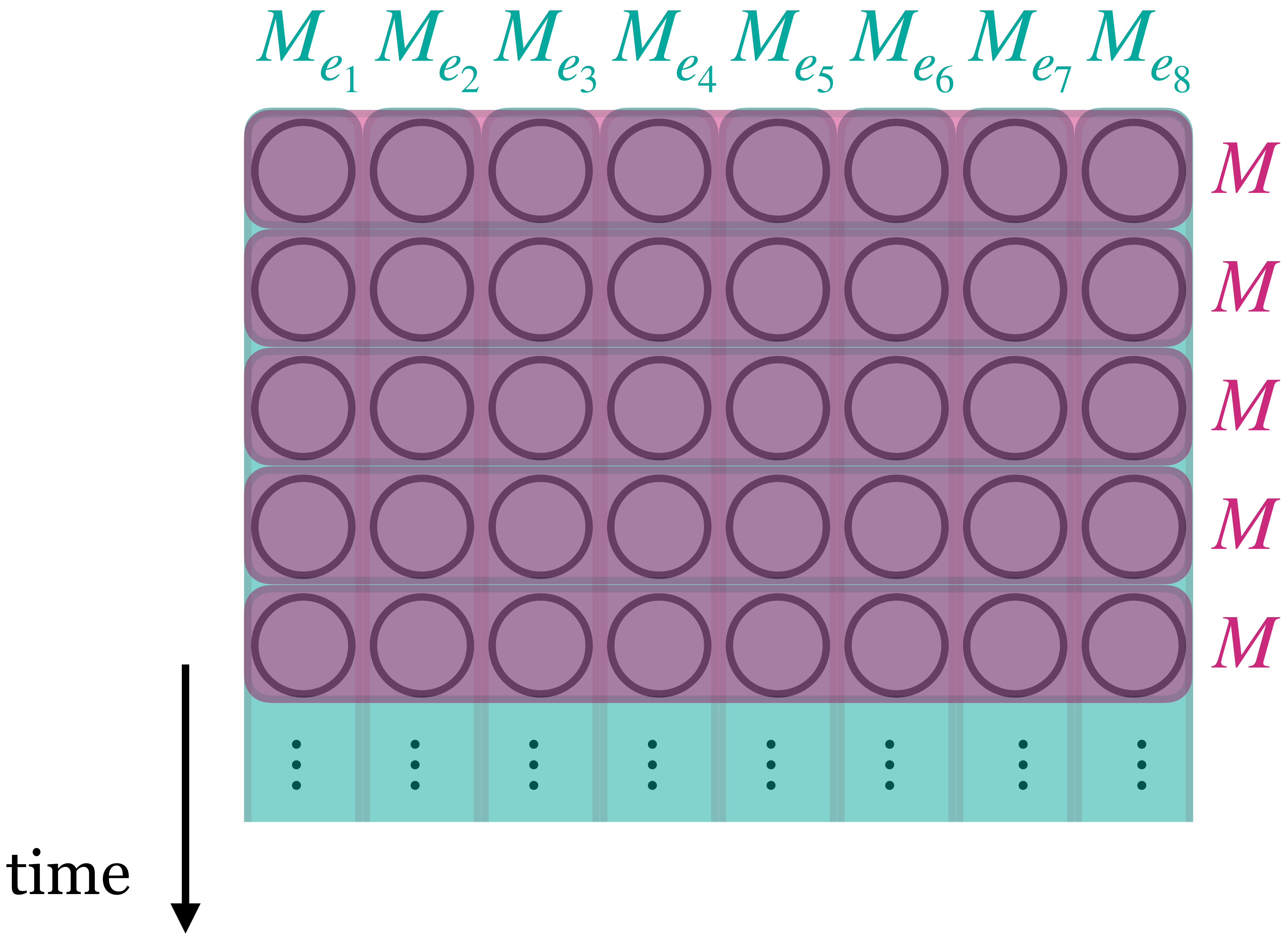
- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



- Because of the above: want common basis.
- Lemma: assigning $g(e)$ to element (t, e) corresponds to point in matroid-intersection polytope.
  - Essentially, use integrality of fractional matching polytope in bipartite graphs.
- Invoke integrality theorem for said polytope.
- Get integral point with same sum of entries.

Towards a Better Bound for Matroids: Proof Sketch

- Recall: $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$ where $\sum_{I \in \mathcal{I}} \lambda(I) = 1$.
- Assume w.l.o.g. there does not exist λ with $\sum_{I \in \mathcal{I}} \lambda(I) < 1$ and $\forall e : g(e) = \sum_{I \in \mathcal{I} : e \in I} \lambda(I)$.



- Because of the above: want common basis.
- Lemma: assigning $g(e)$ to element (t, e) corresponds to point in matroid-intersection polytope.
  - Essentially, use integrality of fractional matching polytope in bipartite graphs.
- Invoke integrality theorem for said polytope.
- Get integral point with same sum of entries.
- Corresponds to said common basis.

Overview of Height Guarantees

Overview of Height Guarantees

	upper bound: existence	lower bound
general set system	$O(\log n)$	$\Omega(\log n)$
general matroids	2	2

Overview of Height Guarantees

	upper bound		lower bound
	poly-time	existence	
general set system		$O(\log n)$	$\Omega(\log n)$
general matroids		2	2

Overview of Height Guarantees

	upper bound		lower bound
	poly-time	existence	
general set system	$O(\log n)$	$O(\log n)$	$\Omega(\log n)$
general matroids	$-$	2	2

Overview of Height Guarantees

	upper bound		lower bound
	poly-time	existence	
general set system	$O(\log n)$	$O(\log n)$	$\Omega(\log n)$
general matroids	$-$	2	2
graphic matroids			
laminar matroids			
uniform matroids			
partition matroids			

Overview of Height Guarantees

	upper bound		lower bound
	poly-time	existence	
general set system	$O(\log n)$	$O(\log n)$	$\Omega(\log n)$
general matroids	–	2	2
graphic matroids		2	2
laminar matroids		2	2
uniform matroids		2	2
partition matroids		2	2

Overview of Height Guarantees

	upper bound		lower bound
	poly-time	existence	
general set system	$O(\log n)$	$O(\log n)$	$\Omega(\log n)$
general matroids	–	2	2
graphic matroids	4	2	2
laminar matroids	4	2	2
uniform matroids	2	2	2
partition matroids	2	2	2

The Pinwheel Scheduling Problem

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.

$n = 3$

a_e : 2 3 99

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

$n = 3$

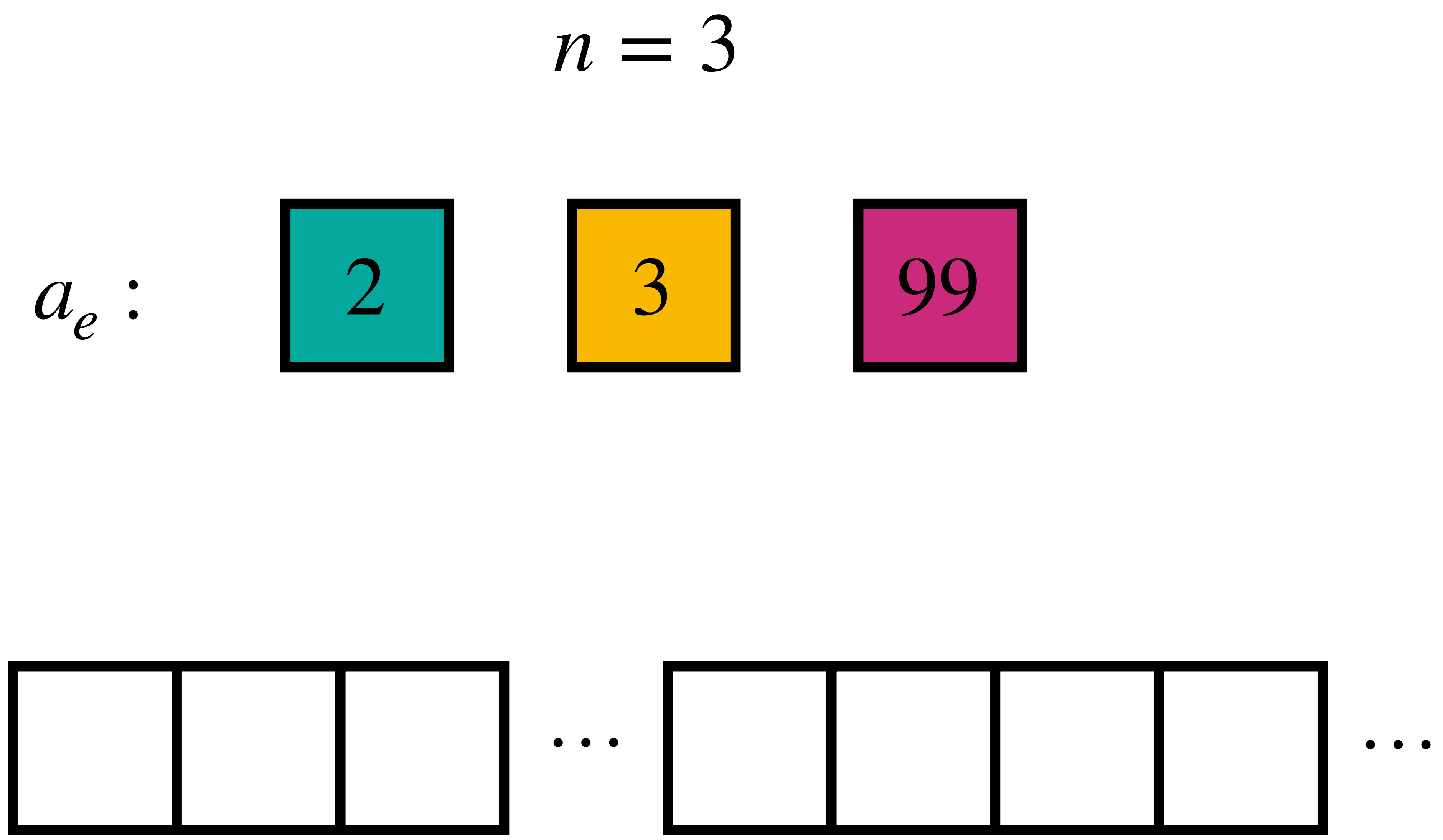
a_e : 2 3 99

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

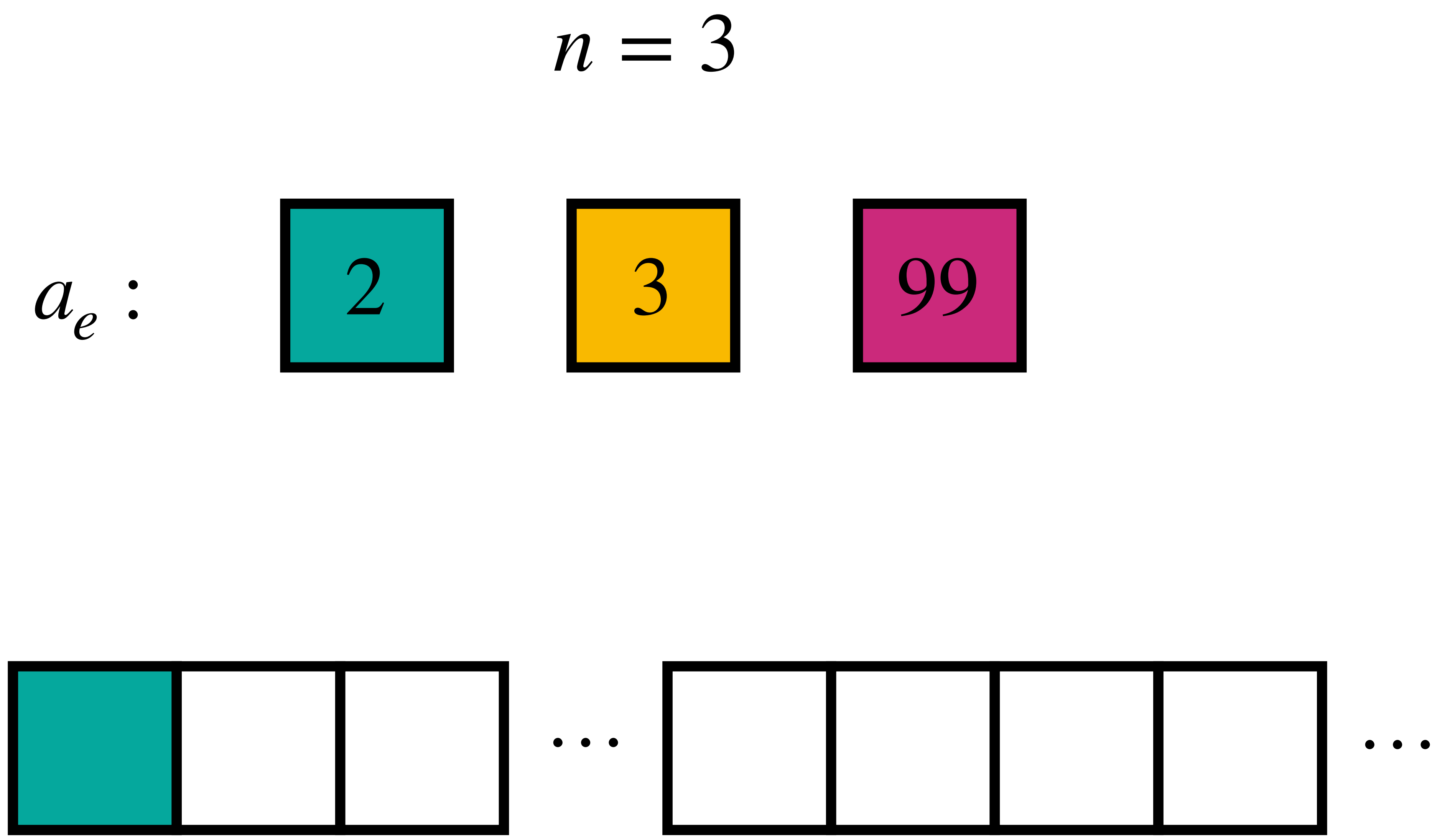


The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

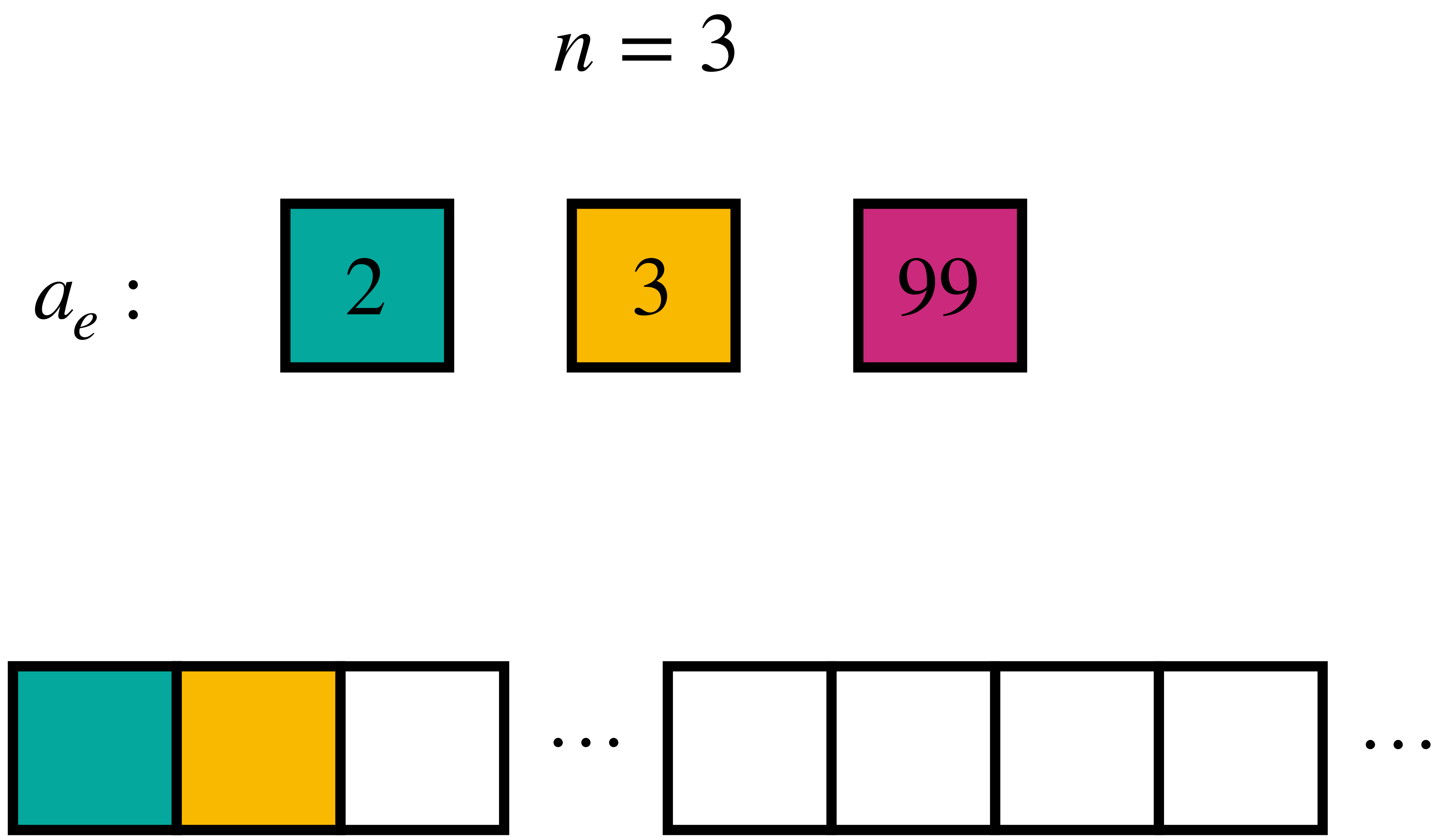


The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

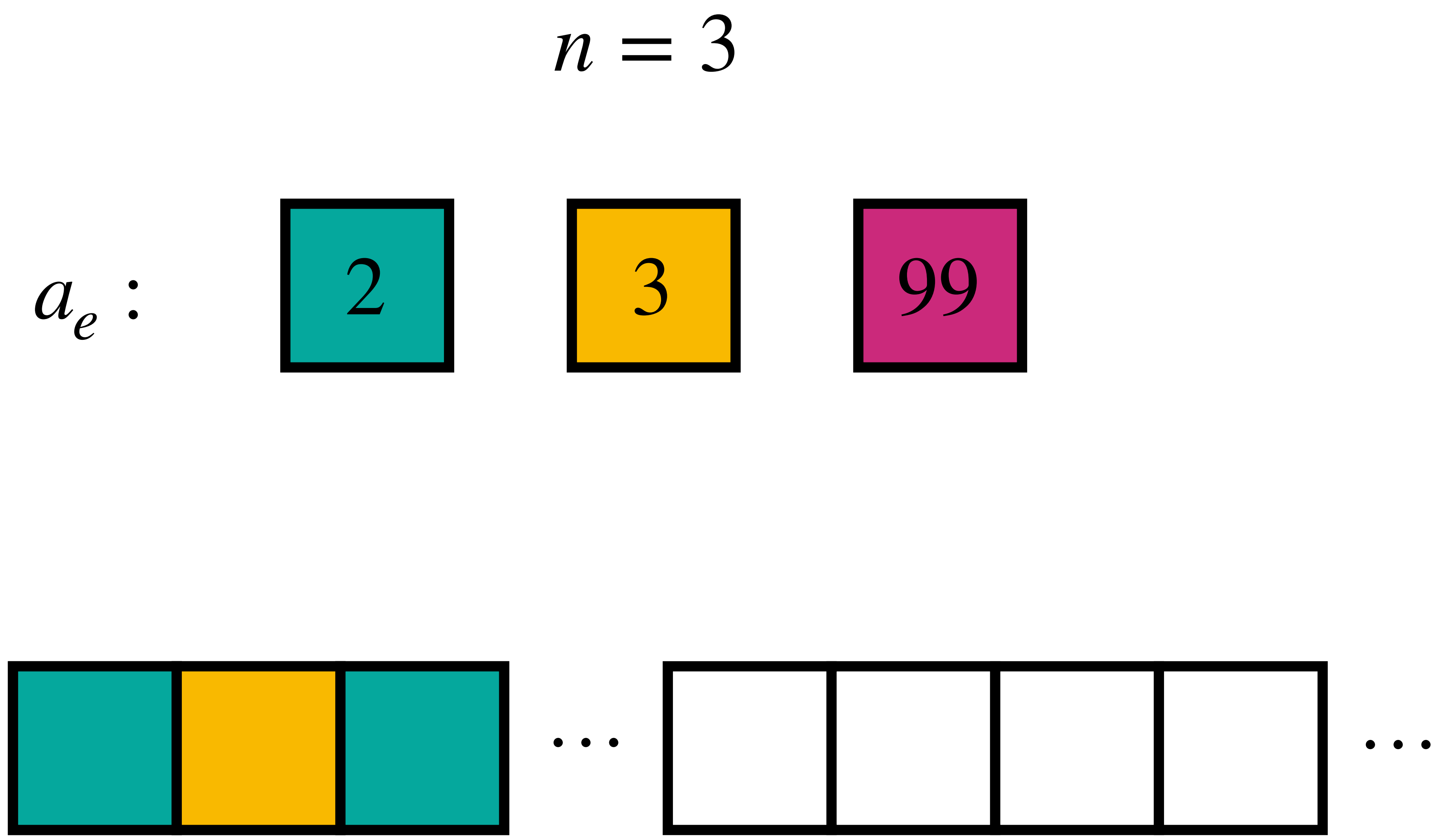


The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

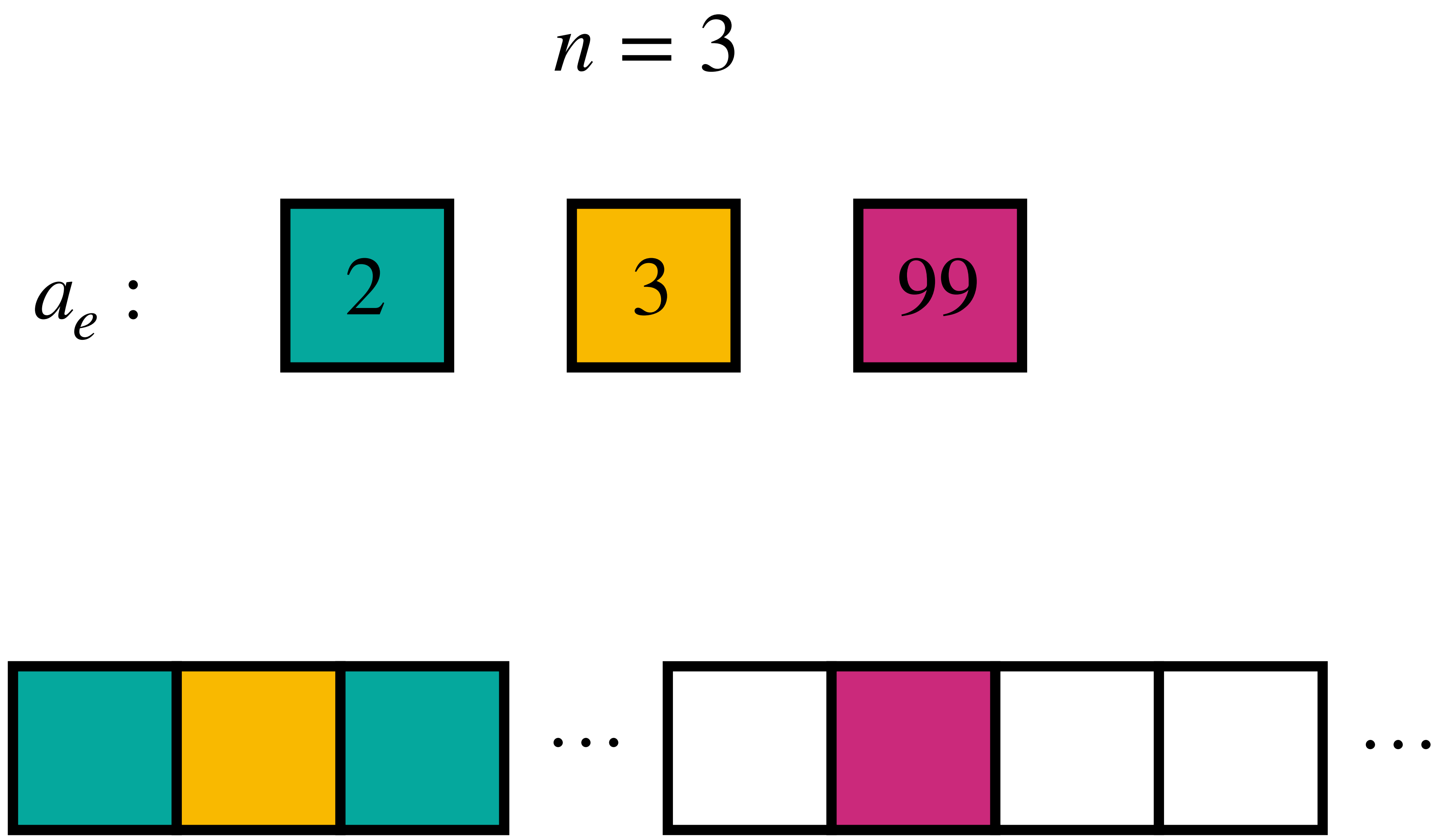


The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

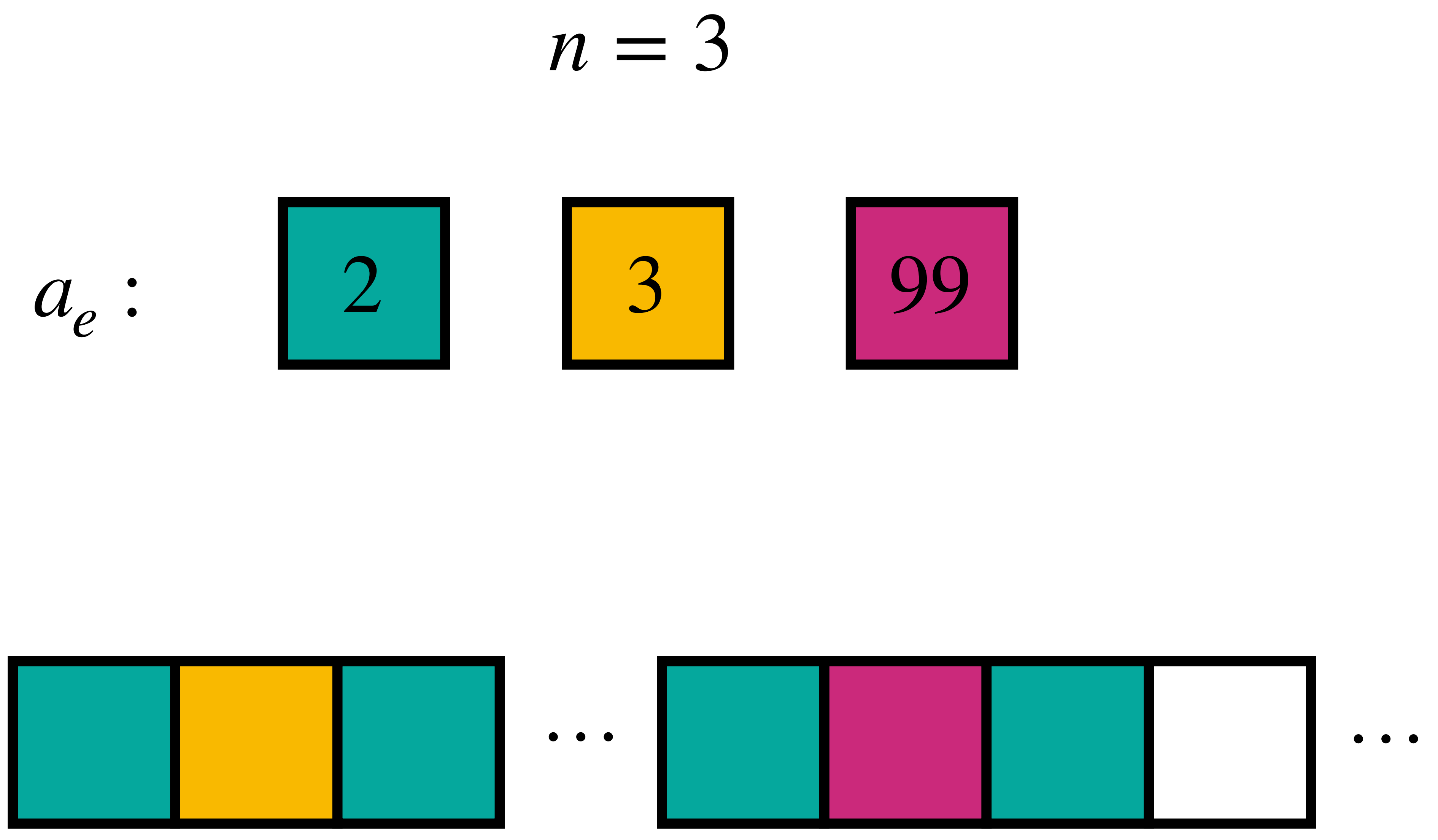


The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

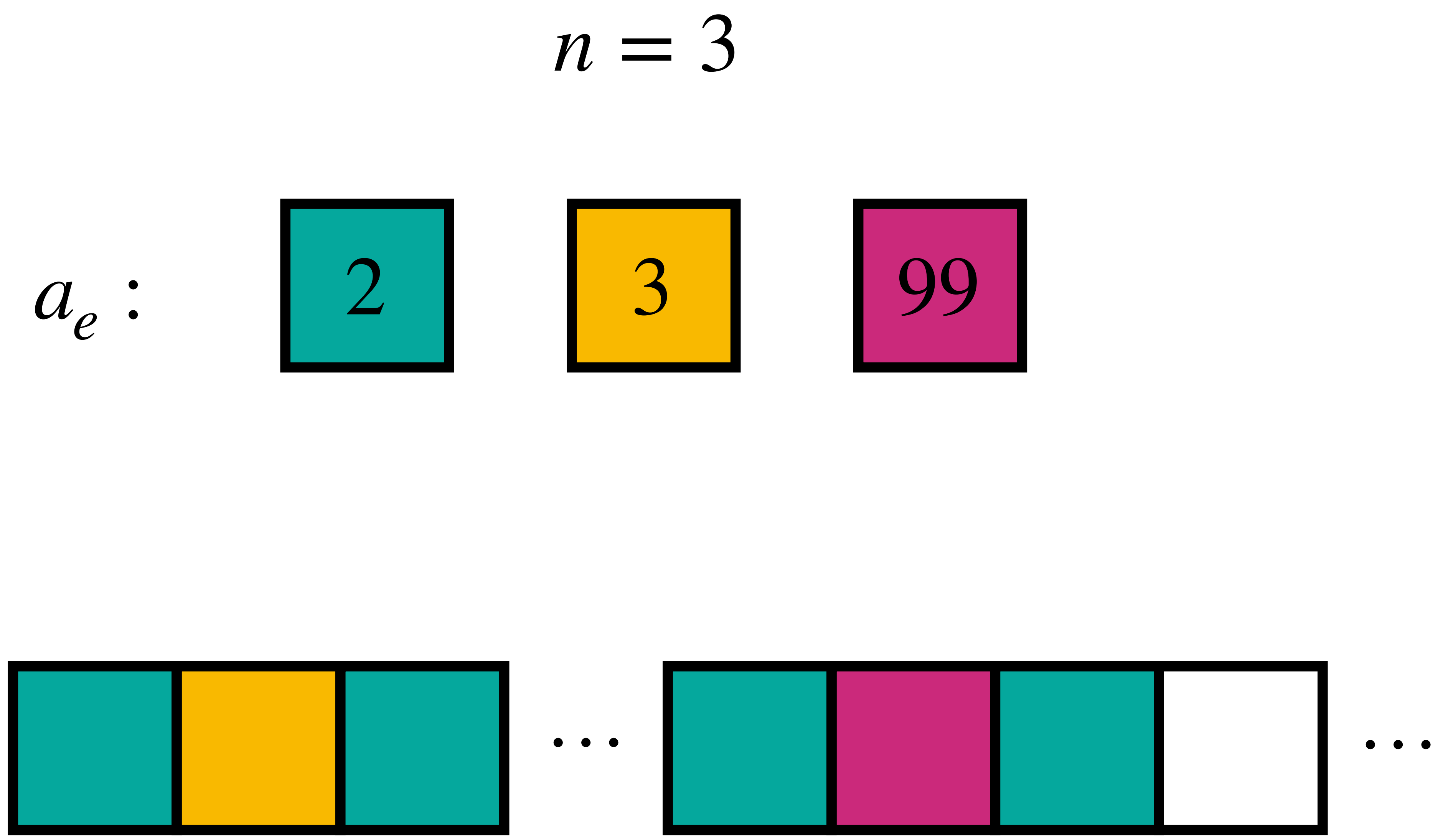


The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.



no schedule exists!

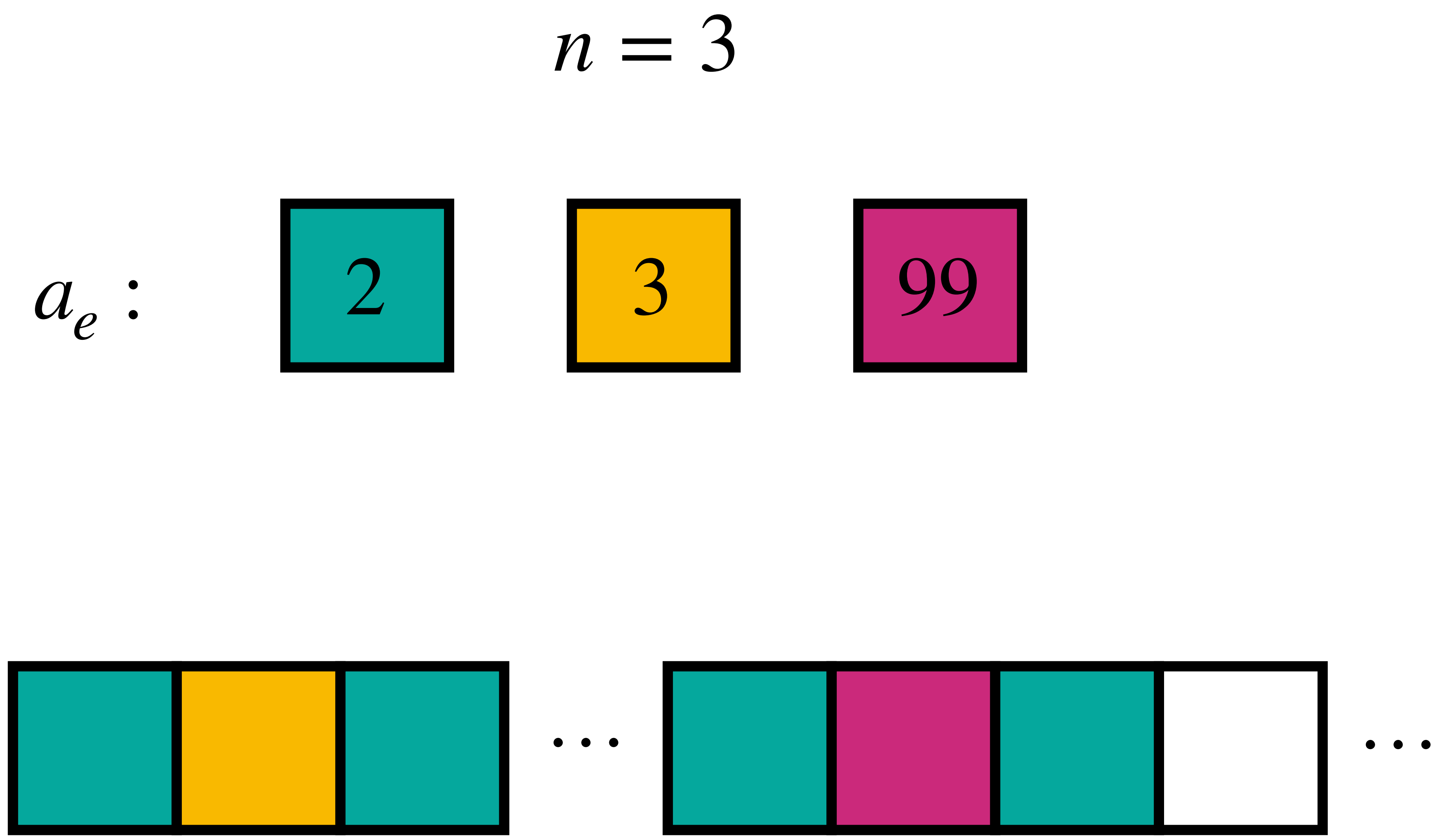
The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

Question:



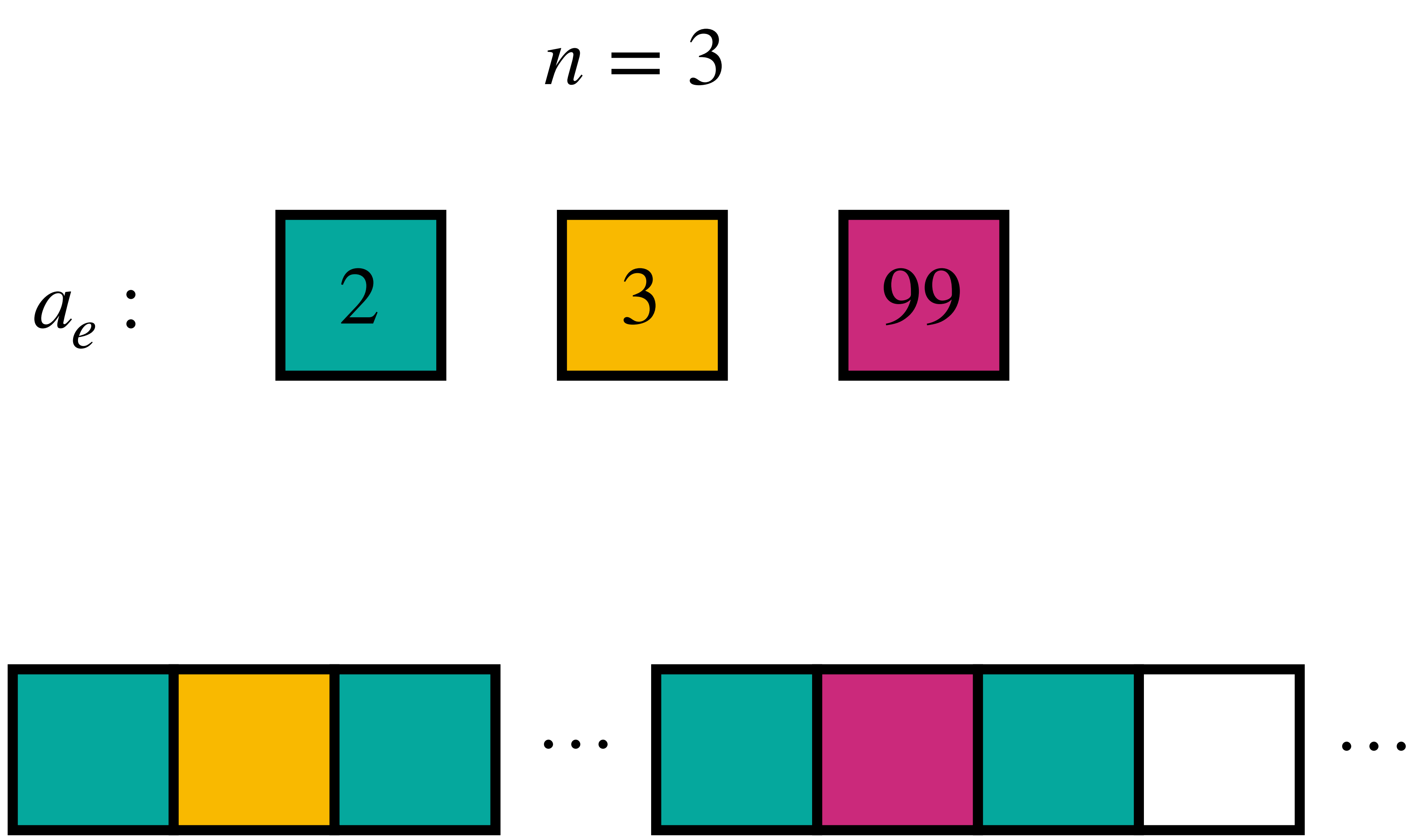
no schedule exists!

The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.



no schedule exists!

Question:

- What is the maximum *density* $\sum_{e=1}^{n} 1/a_e$ that guarantees that there exists a schedule?

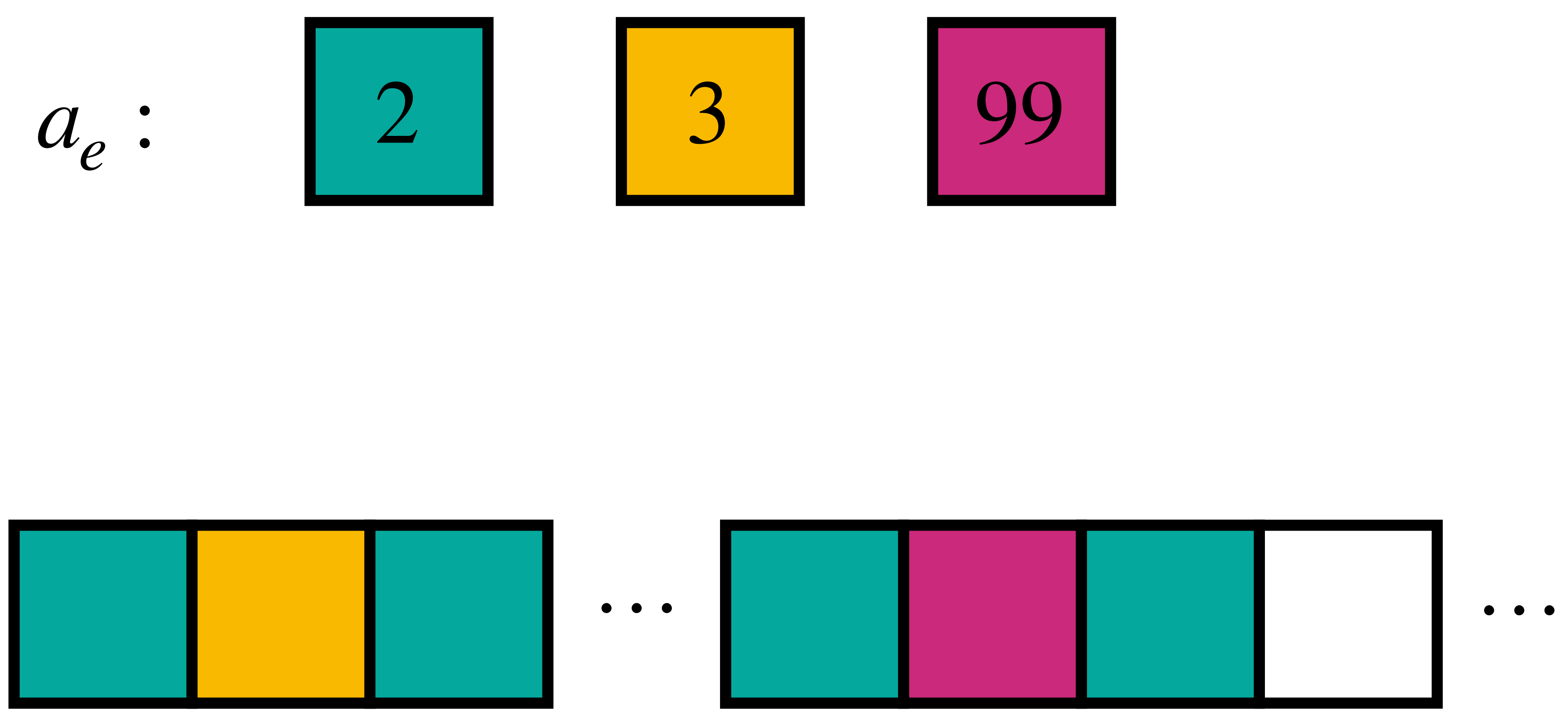
The Pinwheel Scheduling Problem

Model:

[Holte et al., HICCS'89]

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- Goal: Compute perpetual schedule, scheduling one task at a time, fulfilling all frequency requirements.

$n = 3$



no schedule exists!

Question:

- What is the maximum *density* $\sum_{e=1}^{n} 1/a_e$ that guarantees that there exists a schedule?
- Was shown to be precisely 5/6 recently.

[Kawamura, STOC'24]

The Combinatorial Version

The Combinatorial Version

Model:

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

- Density: Minimum $\sum_{I \in \mathcal{I}} \lambda(I)$ such that $\sum_{I \in \mathcal{I}: e \in I} \lambda(I) \geq 1/a_e$ for all e.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

- Density: Minimum $\sum_{I \in \mathcal{I}} \lambda(I)$ such that $\sum_{I \in \mathcal{I} : e \in I} \lambda(I) \geq 1/a_e$ for all e.
- What is the maximum density that guarantees that there exists a schedule?

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

- Density: Minimum $\sum_{I \in \mathcal{I}} \lambda(I)$ such that $\sum_{I \in \mathcal{I} : e \in I} \lambda(I) \geq 1/a_e$ for all e.
- What is the maximum density that guarantees that there exists a schedule?

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

- Density: Minimum $\sum_{I \in \mathcal{I}} \lambda(I)$ such that $\sum_{I \in \mathcal{I} : e \in I} \lambda(I) \geq 1/a_e$ for all e.
- What is the maximum density that guarantees that there exists a schedule?

Using our Results on BGT:

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
    - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

- Density: Minimum $\sum_{I \in \mathcal{I}} \lambda(I)$ such that $\sum_{I \in \mathcal{I} : e \in I} \lambda(I) \geq 1/a_e$ for all e.
- What is the maximum density that guarantees that there exists a schedule?

Using our Results on BGT:

- Threshold is at most $1/2$ for matroids.

The Combinatorial Version

Model:

- n tasks, each task e with frequency requirement $a_e \in \mathbb{N}$.
  - has to be scheduled at least every a_e time steps.
- A (downward-closed) set system $([n], \mathcal{I})$.
- Goal: Compute perpetual schedule, scheduling some $I \in \mathcal{I}$ at every time, fulfilling all frequency requirements.

Question:

- Density: Minimum $\sum_{I \in \mathcal{I}} \lambda(I)$ such that $\sum_{I \in \mathcal{I} : e \in I} \lambda(I) \geq 1/a_e$ for all e.
- What is the maximum density that guarantees that there exists a schedule?

Using our Results on BGT:

- Threshold is at most $1/2$ for matroids.
- Threshold is $\Theta(1/\log n)$ for general set systems.

Other Directions (not comprehensive)

Other Directions (not comprehensive)

- Complexity considerations:

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P.
  - NP-Hardness known for combinatorial variants.
- Approximation Algorithms:

survey by [Kawamura, 2025]

[Cicerone et al., CIAC'17]

[Biktairov et al., SOSA'25]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Approximation Algorithms:
  - Current best for vanilla BGT: 9/7. [Mishra, SODA'26]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Approximation Algorithms:
  - Current best for vanilla BGT: 9/7. [Mishra, SODA'26]
  - Also considered for matchings and general combinatorial constraints. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Approximation Algorithms:
  - Current best for vanilla BGT: 9/7. [Mishra, SODA'26]
  - Also considered for matchings and general combinatorial constraints. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Cup Games (adversarial growth rates). e.g., [Adler et al., SPAA'03]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Approximation Algorithms:
  - Current best for vanilla BGT: 9/7. [Mishra, SODA'26]
  - Also considered for matchings and general combinatorial constraints. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Cup Games (adversarial growth rates). e.g., [Adler et al., SPAA'03]
- Covering version of Pinwheel Scheduling

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Approximation Algorithms:
  - Current best for vanilla BGT: 9/7. [Mishra, SODA'26]
  - Also considered for matchings and general combinatorial constraints. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Cup Games (adversarial growth rates). e.g., [Adler et al., SPAA'03]
- Covering version of Pinwheel Scheduling
  - Exact density threshold known. [Mishra, SODA'26] [Kawamura and Kobayashi, 2025]

Other Directions (not comprehensive)

- Complexity considerations:
  - Unknown whether deciding schedulability of Pinwheel instance is in P. survey by [Kawamura, 2025]
  - NP-Hardness known for combinatorial variants. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Approximation Algorithms:
  - Current best for vanilla BGT: 9/7. [Mishra, SODA'26]
  - Also considered for matchings and general combinatorial constraints. [Cicerone et al., CIAC'17] [Biktairov et al., SOSA'25]
- Cup Games (adversarial growth rates). e.g., [Adler et al., SPAA'03]
- Covering version of Pinwheel Scheduling
  - Exact density threshold known. [Mishra, SODA'26] [Kawamura and Kobayashi, 2025]
- k-visits problem (finite version of Pinwheel Scheduling). [Kanellopoulos et al., SODA'26]

Future Work

Future Work

Some of the many open problems:

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
    - Performance of simple algorithms (e.g., ReduceMax) for BGT.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
    - Performance of simple algorithms (e.g., ReduceMax) for BGT.
    - Complexity of BGT and Pinwheel Scheduling.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):
  - Polynomial-time algorithm guaranteeing height 2 for combinatorial BGT.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):
  - Polynomial-time algorithm guaranteeing height 2 for combinatorial BGT.
  - Exact density guarantees for Pinwheel Scheduling.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):
  - Polynomial-time algorithm guaranteeing height 2 for combinatorial BGT.
  - Exact density guarantees for Pinwheel Scheduling.
  - Complexity and approximation algorithms.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):
  - Polynomial-time algorithm guaranteeing height 2 for combinatorial BGT.
  - Exact density guarantees for Pinwheel Scheduling.
  - Complexity and approximation algorithms.
- Beyond matroids:

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):
  - Polynomial-time algorithm guaranteeing height 2 for combinatorial BGT.
  - Exact density guarantees for Pinwheel Scheduling.
  - Complexity and approximation algorithms.
- Beyond matroids:
  - Close gaps for Polyamorous Scheduling.

Future Work

Some of the many open problems:

- Vanilla version (one task per time step):
  - Performance of simple algorithms (e.g., ReduceMax) for BGT.
  - Complexity of BGT and Pinwheel Scheduling.
  - Improved approximation algorithms for BGT.
- Matroid version (independent set per time step):
  - Polynomial-time algorithm guaranteeing height 2 for combinatorial BGT.
  - Exact density guarantees for Pinwheel Scheduling.
  - Complexity and approximation algorithms.
- Beyond matroids:
  - Close gaps for Polyamorous Scheduling.
  - Consider general 2-systems or even k-Systems.

Thank You!