Synthesizing Minimal Tile Sets for Patterned DNA Self-Assembly

Mika Göös & Pekka Orponen
Aalto University *(School of Science and Technology)*

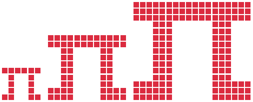
Outline

1. Previous Study
2. Problem Definition
3. Approach of Ma & Lombardi
4. Our Contributions

Previous Study

Shapes modulo Scale

[Soloveichik & Winfree 2004]



Unsolvable

Previous Study

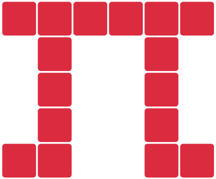
Shapes modulo Scale

[Soloveichik & Winfree 2004]

Unsolvable

Shapes

[Adleman et al. 2002]



NP-hard

Previous Study

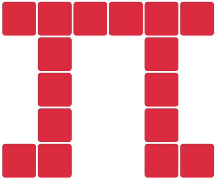
Shapes modulo Scale

[Soloveichik & Winfree 2004]

Unsolvable

Shapes

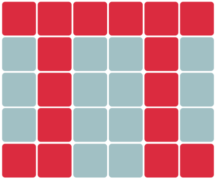
[Adleman et al. 2002]



NP-hard

Patterns

[Ma & Lombardi 2008]

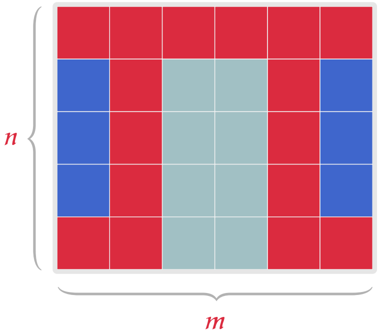


Not known?

Pattern self-Assembly Tile set Synthesis (PATS)

Input

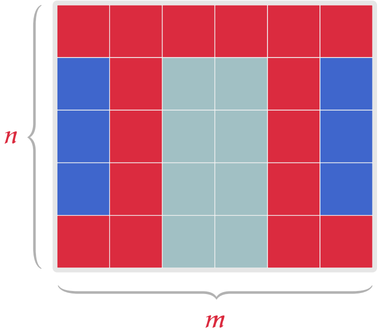
A k-colouring $c : [m] \times [n] \to [k]$



Pattern self-Assembly Tile set Synthesis (PATS)

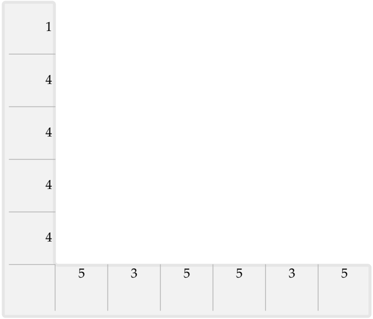
Input

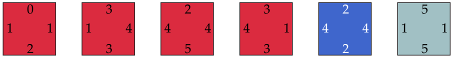
A k-colouring $c : [m] \times [n] \to [k]$



Output

A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$

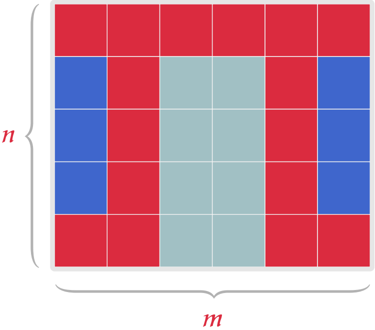




Pattern self-Assembly Tile set Synthesis (PATS)

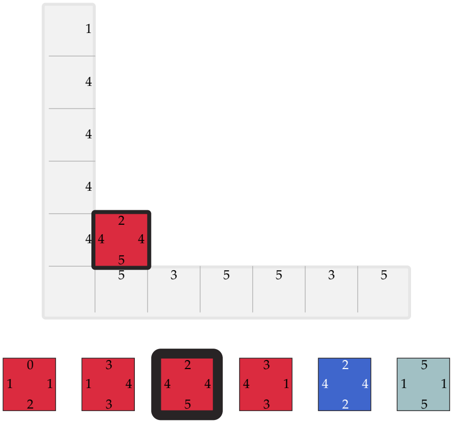
Input

A k-colouring $c : [m] \times [n] \to [k]$



Output

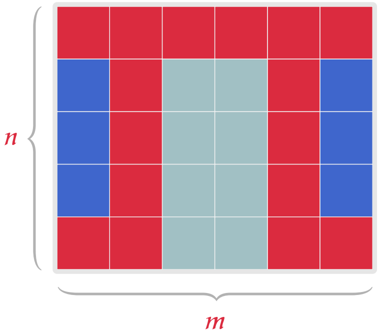
A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

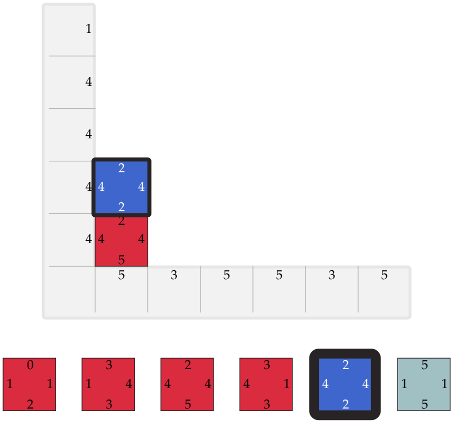
Input

A k-colouring $c : [m] \times [n] \rightarrow [k]$



Output

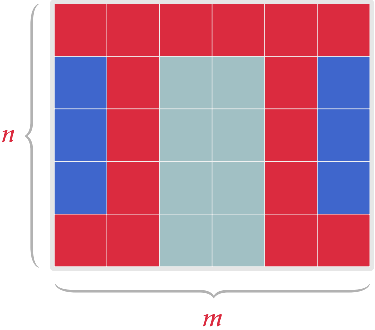
A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

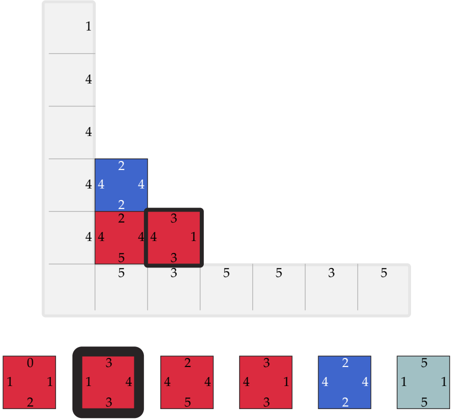
Input

A k-colouring $c : [m] \times [n] \to [k]$



Output

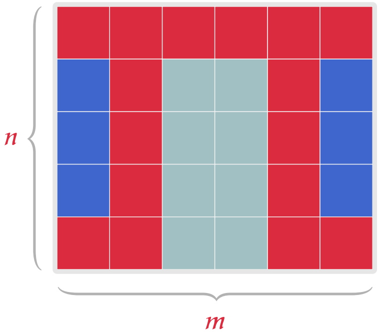
A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

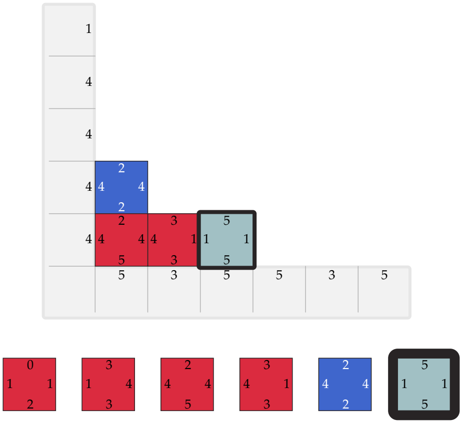
Input

A k-colouring $c : [m] \times [n] \to [k]$



Output

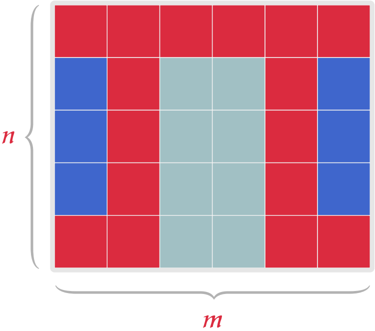
A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

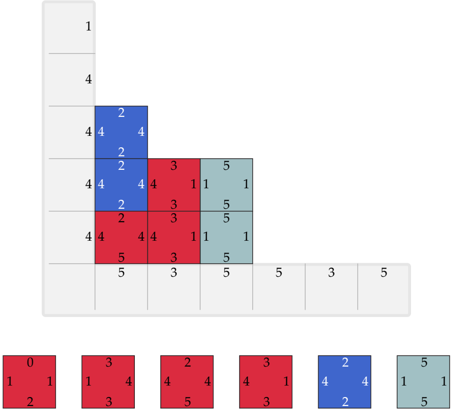
Input

A k-colouring $c : [m] \times [n] \rightarrow [k]$



Output

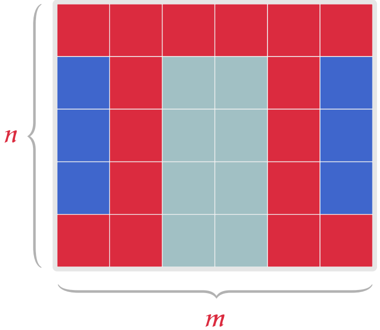
A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

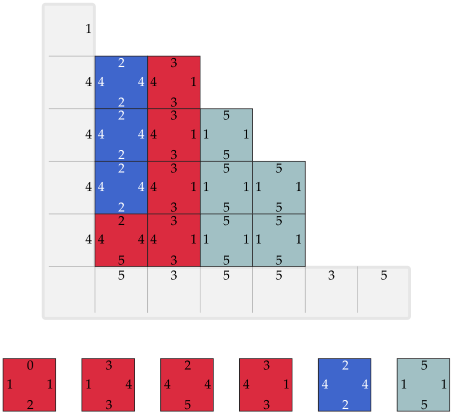
Input

A k-colouring $c : [m] \times [n] \to [k]$



Output

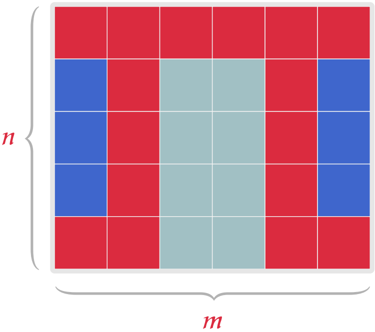
A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

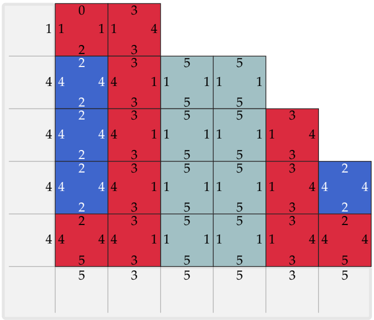
Input

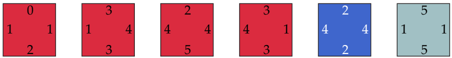
A k-colouring $c : [m] \times [n] \to [k]$



Output

A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$

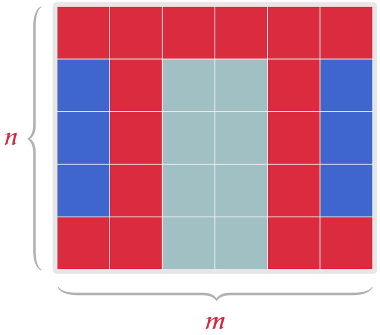




Pattern self-Assembly Tile set Synthesis (PATS)

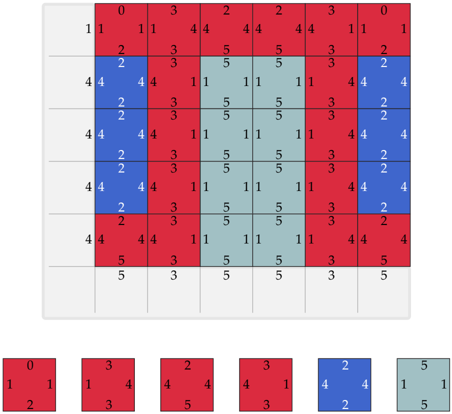
Input

A k-colouring $c : [m] \times [n] \rightarrow [k]$



Output

A Tile Assembly System $\mathscr{T} = (T, \mathcal{S}, s, 2)$



Pattern self-Assembly Tile set Synthesis (PATS)

Given: A k-colouring $c : [m] \times [n] \to [k]$.

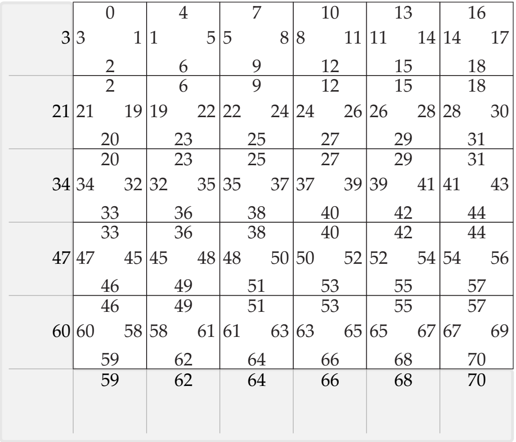
Find: A tile assembly system $\mathscr{T} = (T, \mathcal{S}, s, 2)$ s.t.

P1. The tiles in T have bonding strength 1.

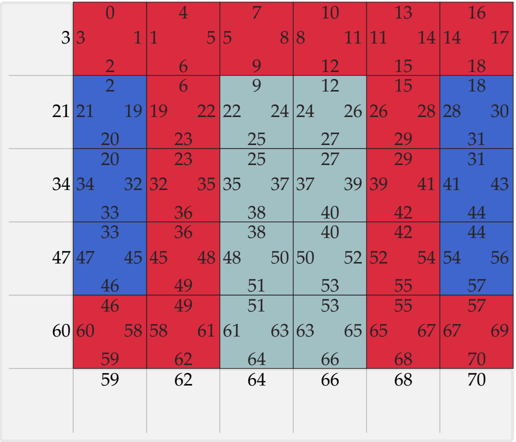
P2. The domain of $\mathcal{S}$ is $[0, m] \times \{0\} \cup \{0\} \times [0, n]$ and all the terminal assemblies have the domain $[0, m] \times [0, n]$.

P3. There exists a colouring $d : T \to [k]$ such that for each terminal assembly $\mathcal{A} \in \mathrm{Term}\,\mathscr{T}$ we have $d(\mathcal{A}(x, y)) = c(x, y)$ for all $(x, y) \in [m] \times [n]$.

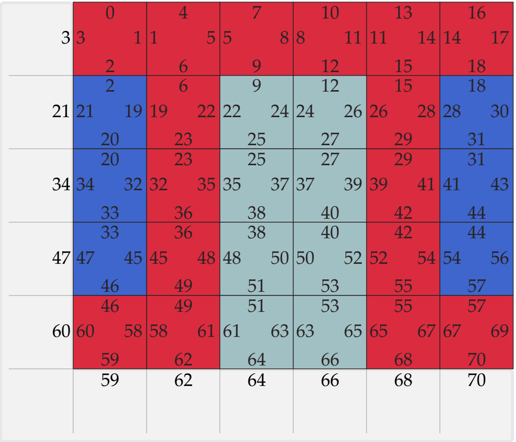
Approach of Ma & Lombardi



Approach of Ma & Lombardi



Approach of Ma & Lombardi



To minimize Tile set size:

- Merge glues
- Merge tiles

If conflicts arise:

- Continue merging!

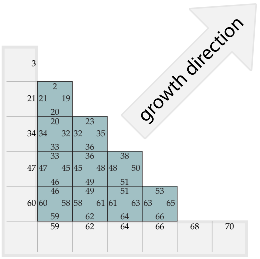
Our Contributions

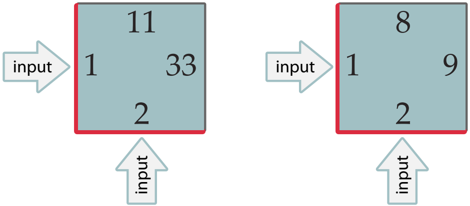
We present

1. Extension of the work of Ma & Lombardi
2. Branch & Bound algorithm
3. Pruning heuristics

Determinism

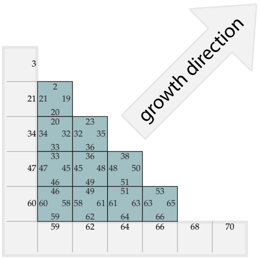
Lemma: **Minimal solutions to the PATS problem are** ***deterministic***

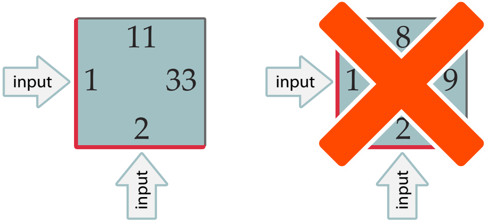




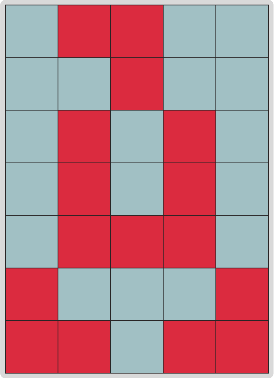
Determinism

Lemma: **Minimal solutions to the PATS problem are *deterministic***

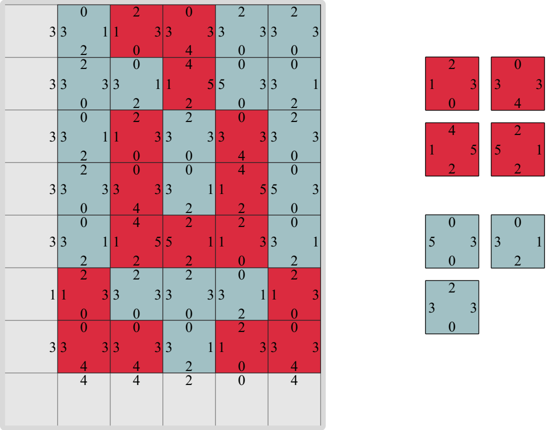




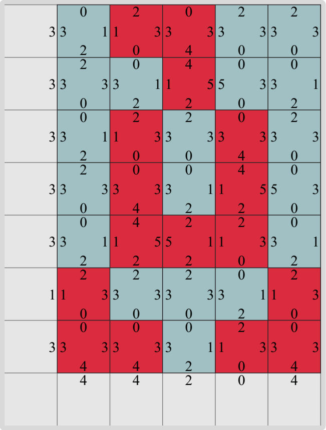
Partition Centric View

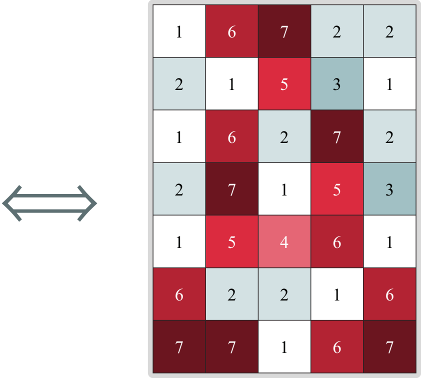


Partition Centric View



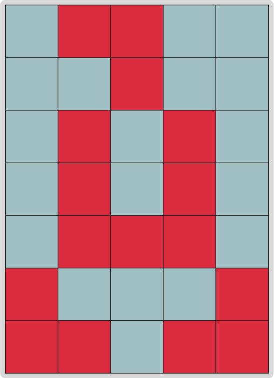
Partition Centric View



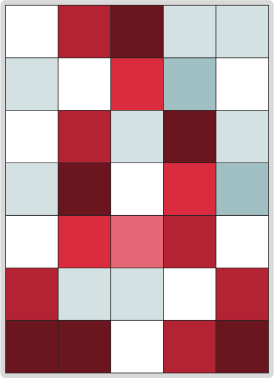


Contructible partition of $[m] \times [n]$

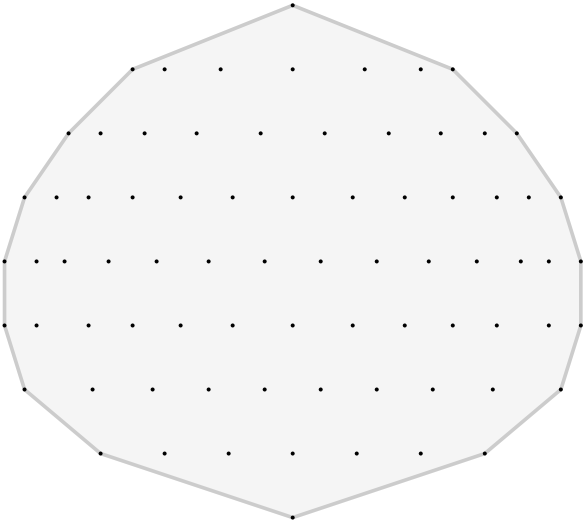
Partition Centric View



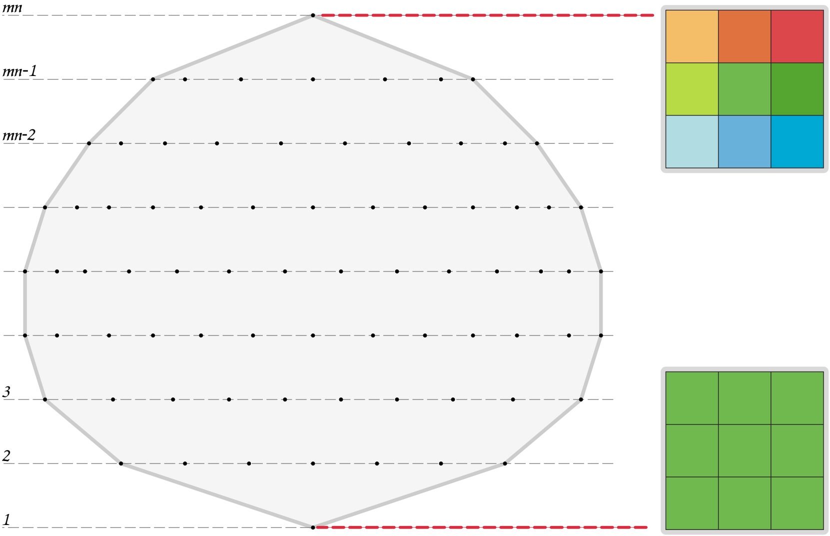
is **coarser** than



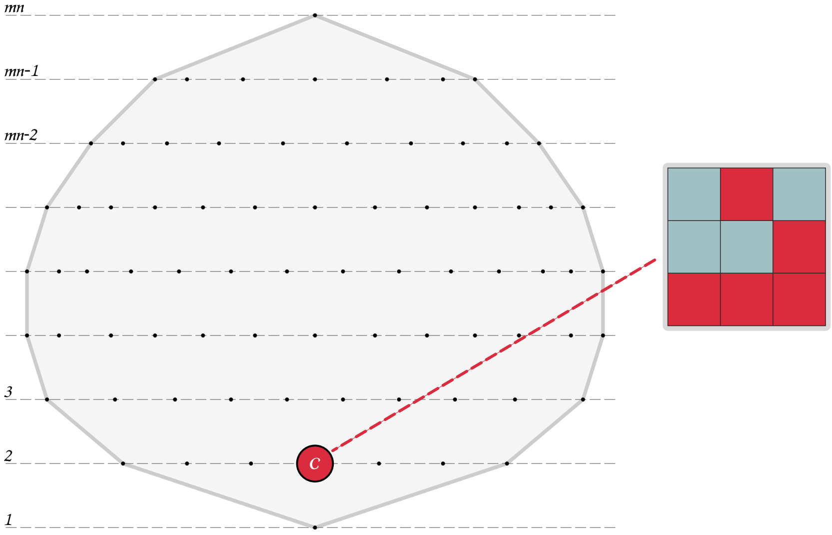
Searching the Lattice of Partitions



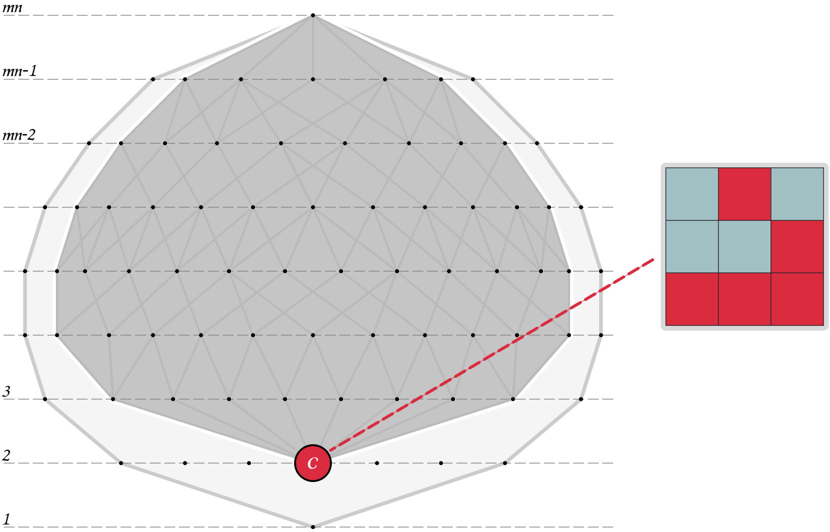
Searching the Lattice of Partitions



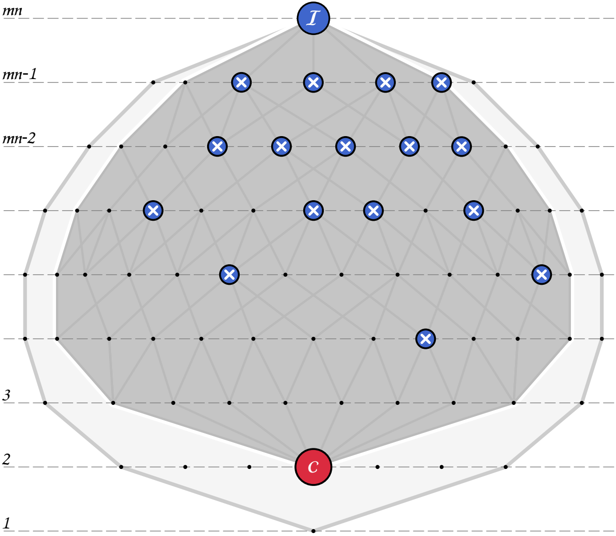
Searching the Lattice of Partitions



Searching the Lattice of Partitions

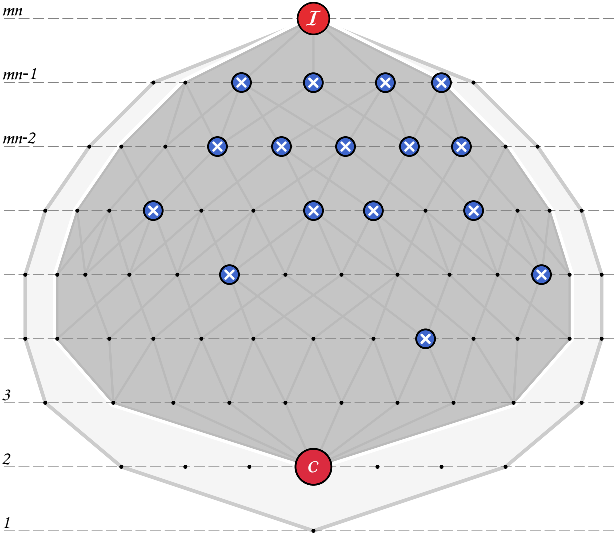


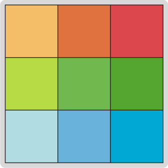
Searching the Lattice of Partitions



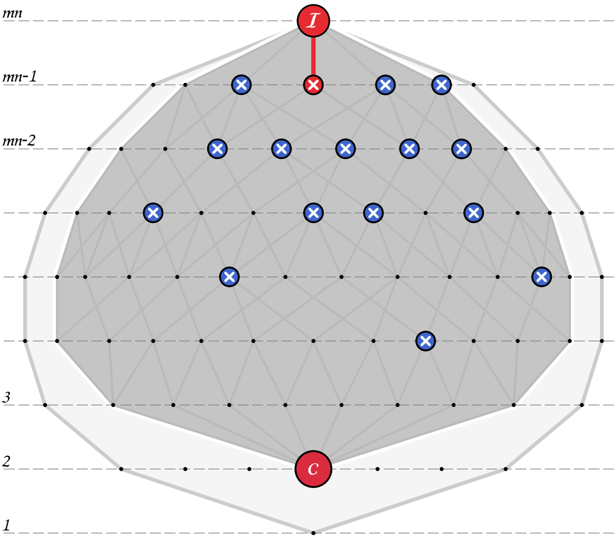
Contructible
partition

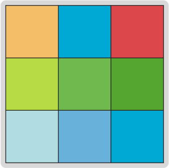
Searching the Lattice of Partitions



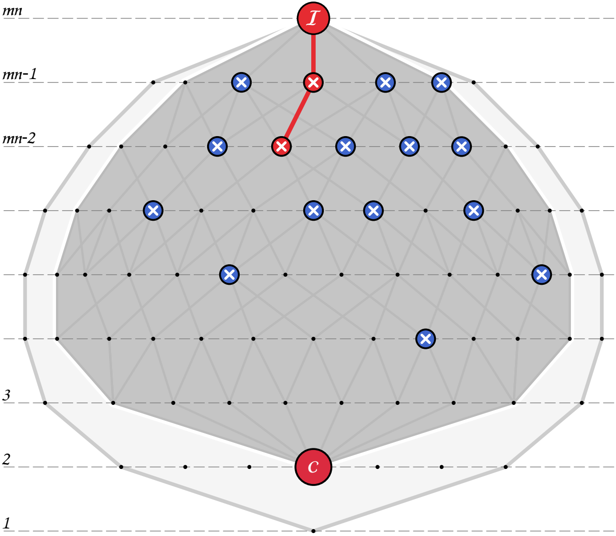


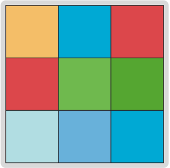
Searching the Lattice of Partitions



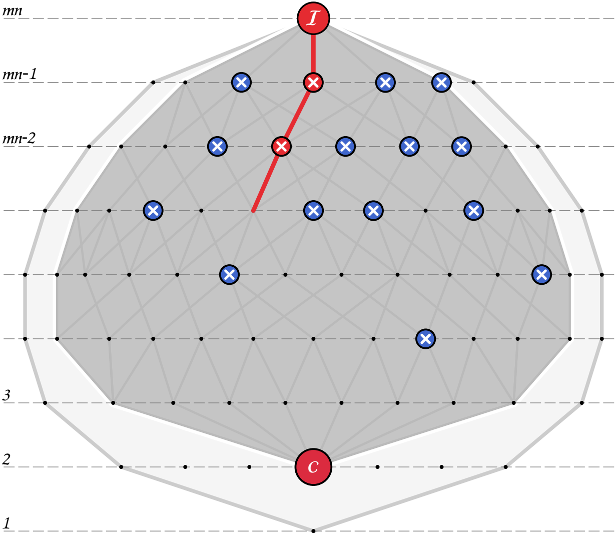


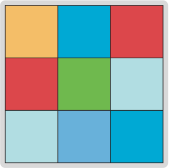
Searching the Lattice of Partitions



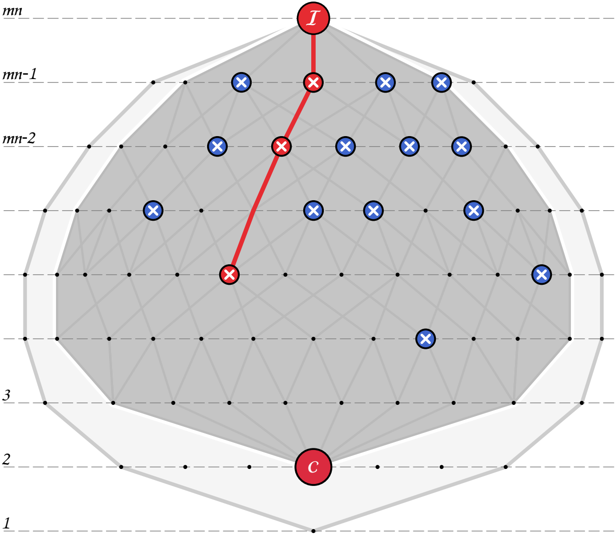


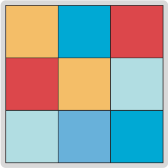
Searching the Lattice of Partitions



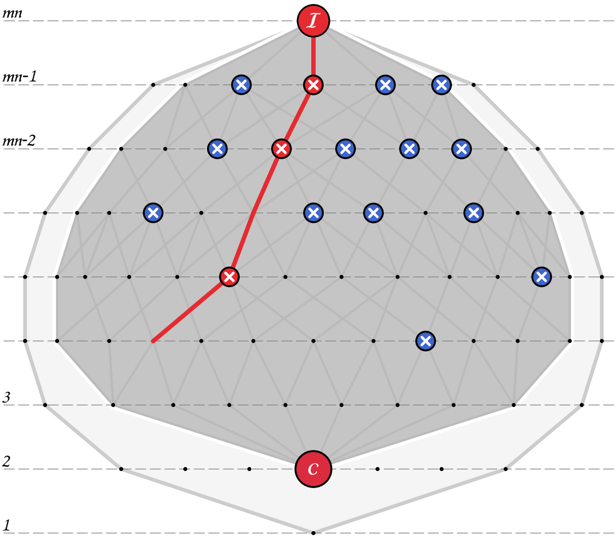


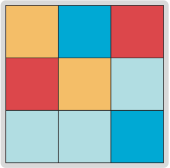
Searching the Lattice of Partitions



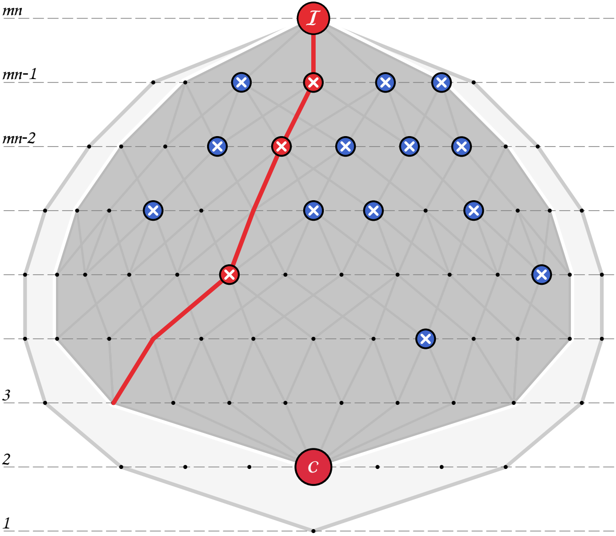


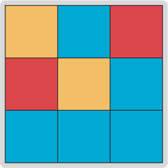
Searching the Lattice of Partitions



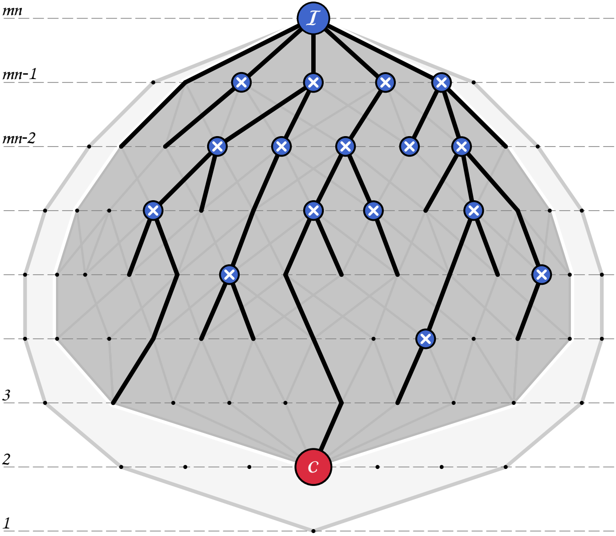


Searching the Lattice of Partitions





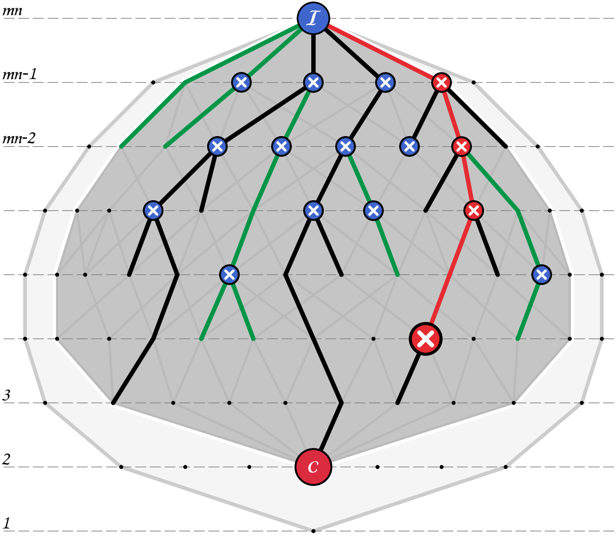
Searching the Lattice of Partitions



Our **B&B algorithm**

1. Node-disjoint search tree
2. Uses memory $\mathbf{poly}(mn)$
3. Branching *only* on constructible partitions

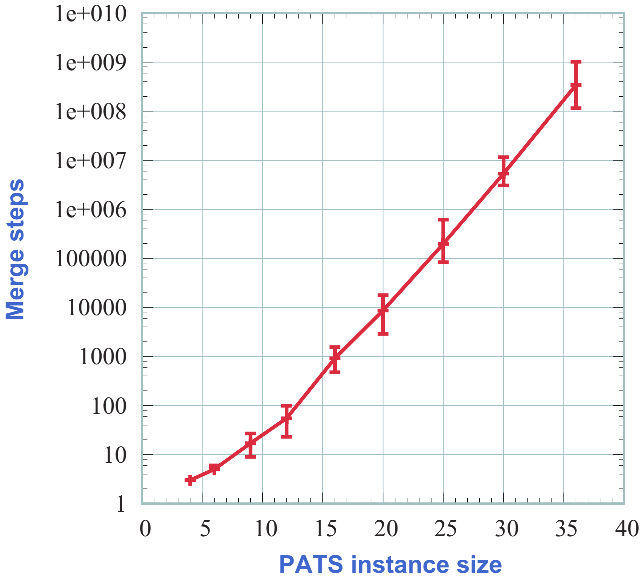
Searching the Lattice of Partitions



Our **B&B algorithm**

1. Node-disjoint search tree
2. Uses memory $\mathbf{poly}(mn)$
3. Branching *only* on constructible partitions
4. Cheap bounding function

Running time on random 2-coloured instances



$\sim 2^{mn}$

PATS problem remains challenging

- Open Problems:
  1. Is it **NP**-hard?
  2. Faster algorithms?
  3. Generalize to infinite finite-period patterns
- PATS is of practical importance

Thank you!