# Rubik Tables, Stack Rearrangement, and Multi-Robot Routing




**Jingjin Yu**

**Computer Science @ Rutgers University**

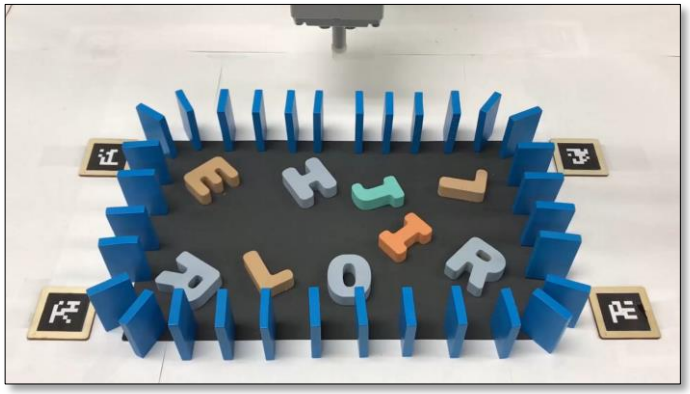


Algorithmic Robotics & Control Lab

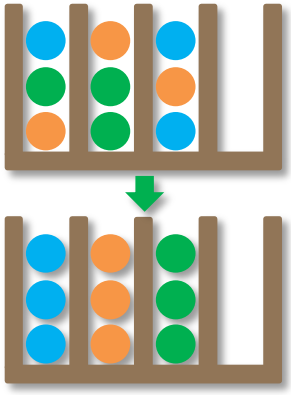
**AAAI'23 MAPF Workshop · Feb 2023**

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

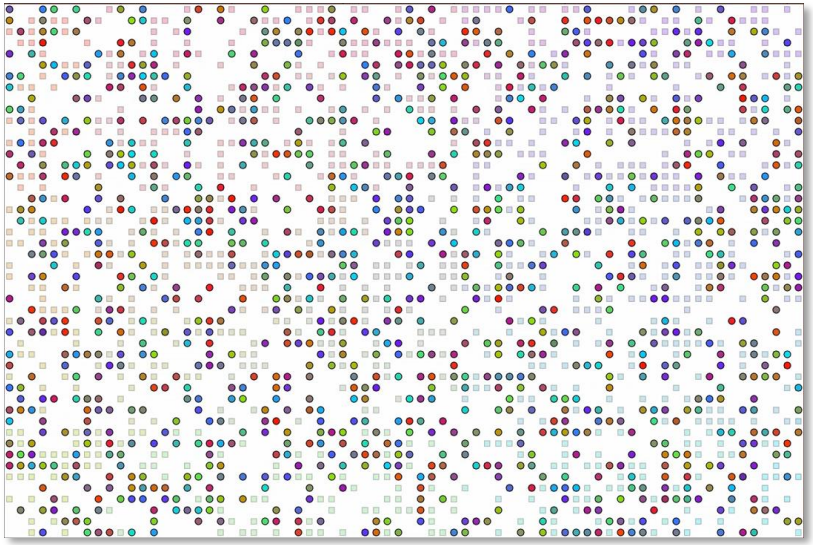
# Background: A “Random Walk” Between Topics



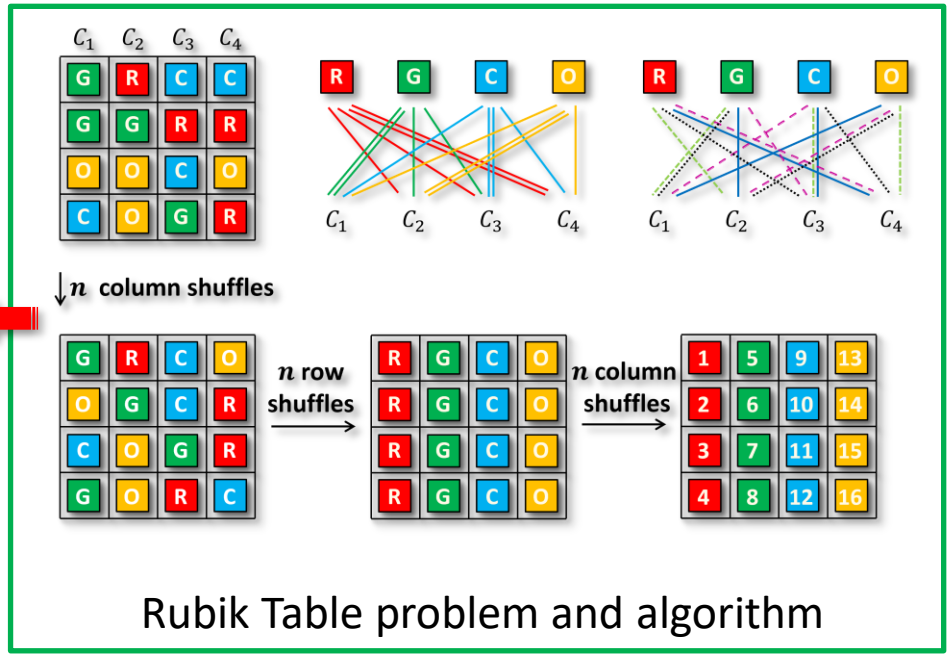
Long-horizon rearrangement




Stack rearrangement



Poly-time 1.$x$-optimal multi-robot routing




Rubik Table problem and algorithm

# Collaborators & Main Publications Discussed

Teng Guo

Mario Szegedy

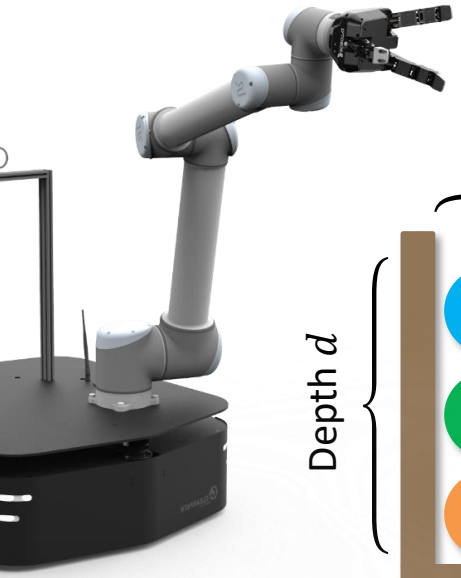
**Main publications discussed:**

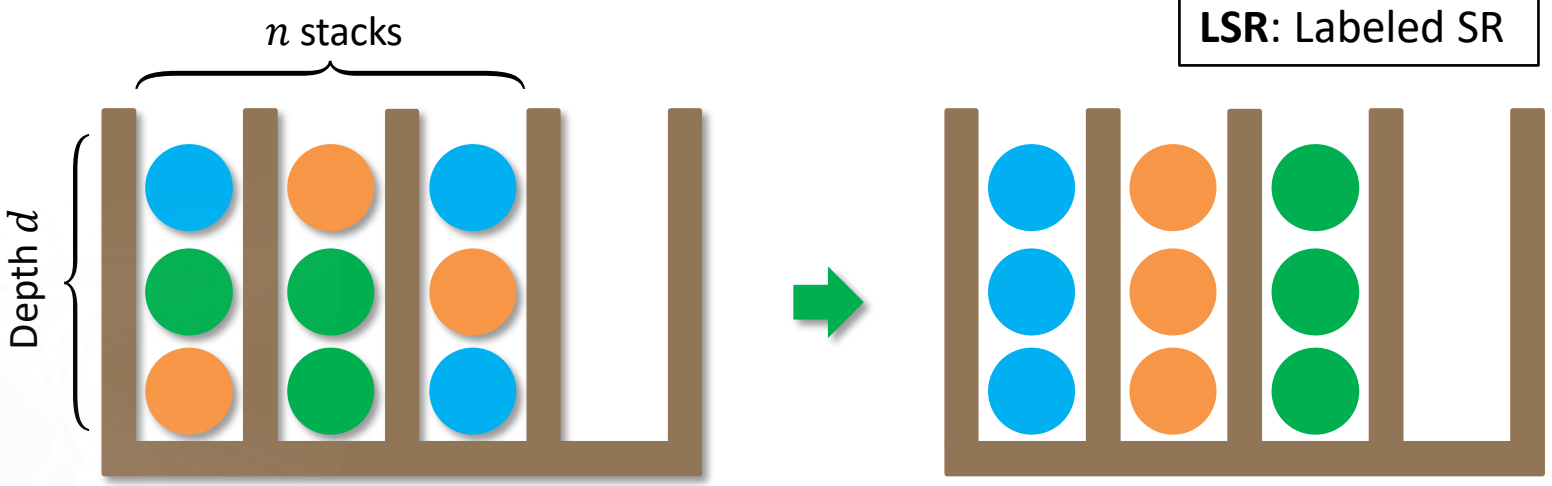
**[IJRR 2022]** M. Szegedy and J. Yu. **Rubik Tables and Object Rearrangement.**

**[RSS 2022]** T. Guo and J. Yu. **Sub-1.5 Time-Optimal Multi-Robot Path Planning on Grids in Polynomial Time.**

NSF amazon

# Stack Rearrangement






**CSR**: Colored SR
**LSR**: Labeled SR

**Objective**: Minimizing the total number of stack pop-push operations

**Proposition (Lower Bound).** LSR on average requires $\Omega(nd\left(1+\frac{\log d}{\log n}\right))$ pop-pushes to solve. CSR on average requires $\Omega(nd)$ pop-pushes.

$\sim \Omega(nd)$

Proof sketch: there are $(nd)!$ unique LSR instances. In each step, branching factor is no more than $(n+1)^2$. Put these together,

$$\left((n+1)^2\right)^t \geq (nd)! \Rightarrow t \geq nd(1+\frac{\log d}{\log n})$$

# Upper Bounds and the Remaining Gap

**Proposition (Simple Upper Bound).** CSR can be solved using $O(nd \log n)$ pop-pushes. LSR can be solved using $O(nd(\log n + \log d))$ pop-pushes.

Han et al, RA-L 2018

Proof sketch: use divide-n-conquer over $n$ stacks and then sort each stack.

**Proposition (Alternative Upper Bound).** CSR can be solved using $7nd(1 + \log d)$ pop-pushes. LSR can be solved using $7nd + 8nd \log d$ pop-pushes.

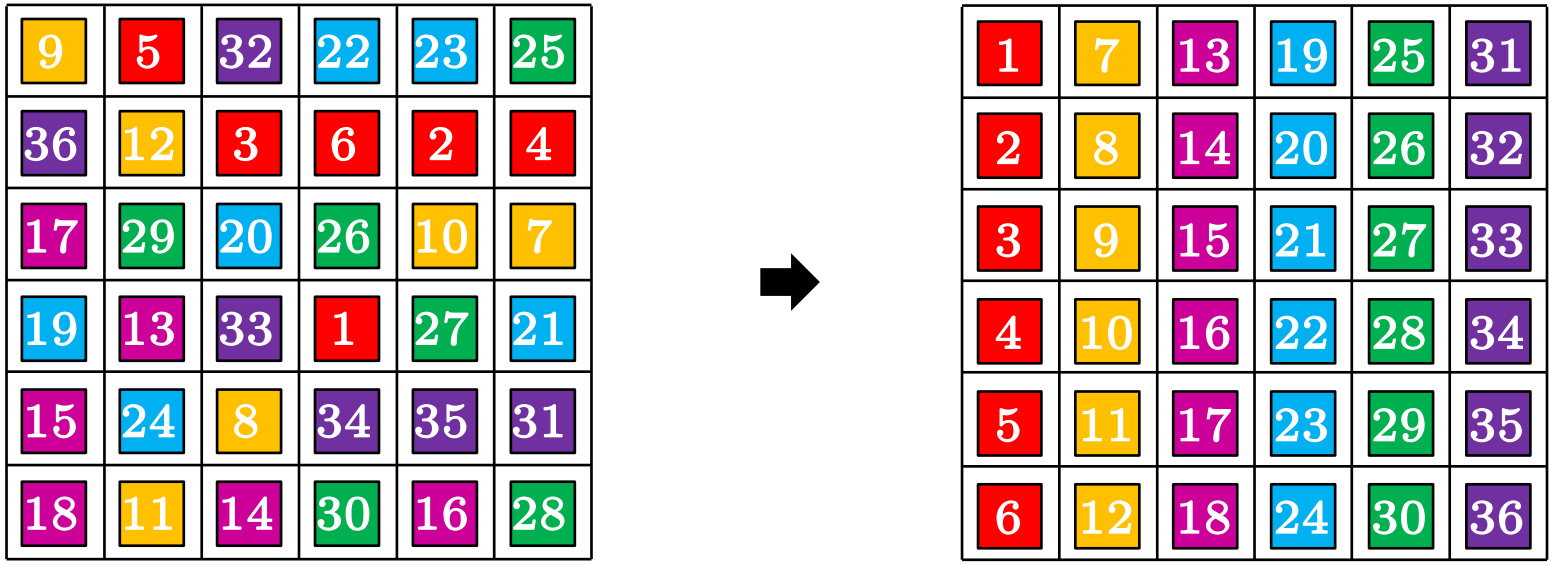
Proof sketch: use divide-n-conquer but with more careful recursion analysis

**Question**: Can we close or narrow the gap, e.g., $\Omega(nd)$ lower bound and $O(nd \log n)$ or $O(nd \log d)$ upper bound for CSR?

**Potential idea**: Multiple levels of recursion over $d$?

# The Rubik Table Problem

**Rubik table problem**: Given an $n \times n$ square filled with $n$ types of items of $n$ each. In a *shuffle*, one may permute a single row or column arbitrarily. How many shuffles are needed for realizing an arbitrary reconfiguration?

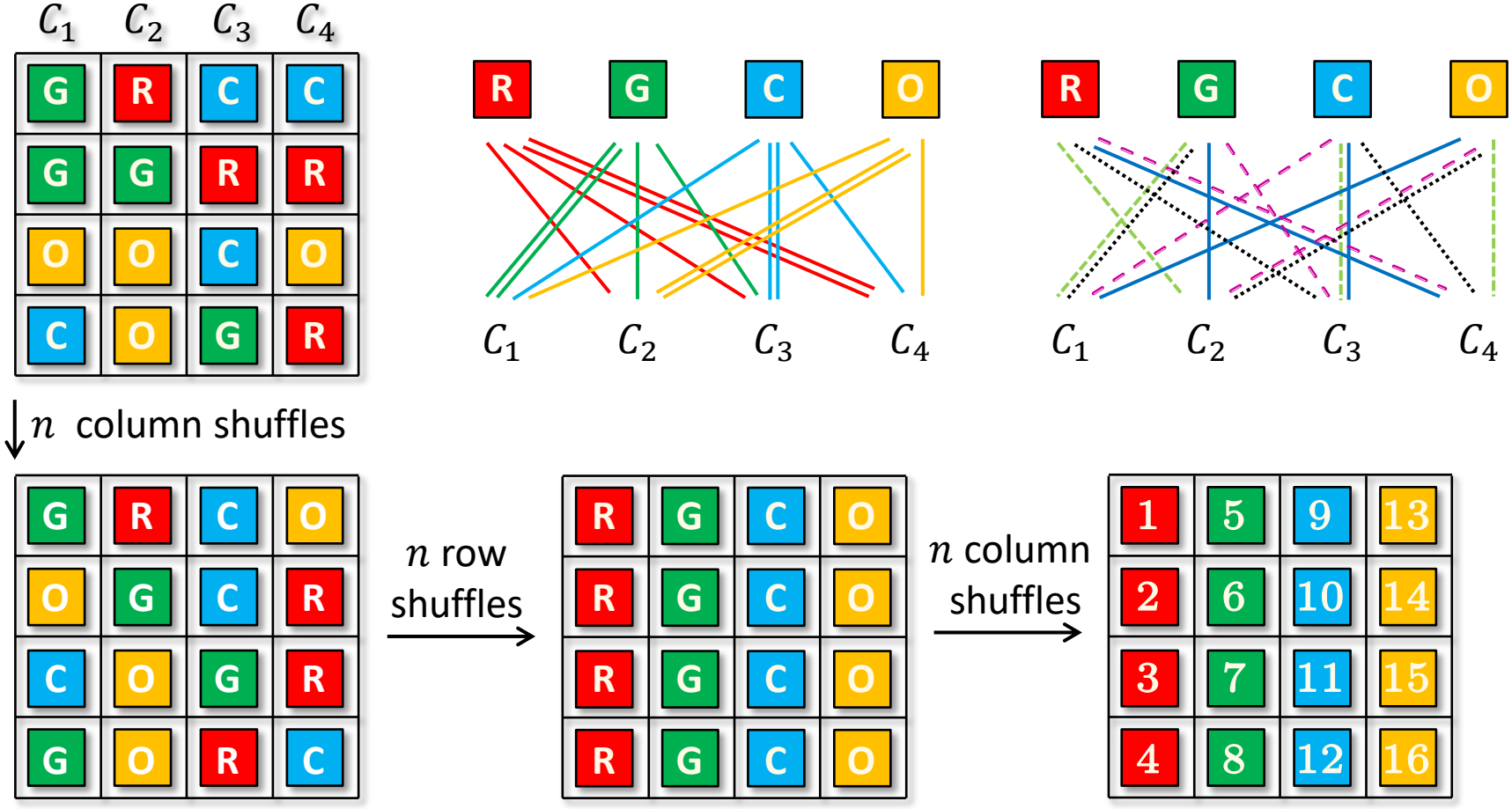



**Theorem.** The Rubik table problem can be solved with $2n$ shuffles.

**Corollary.** The labeled Rubik table problem can be solved with $3n$ shuffles.

# The Rubik Table Algorithm

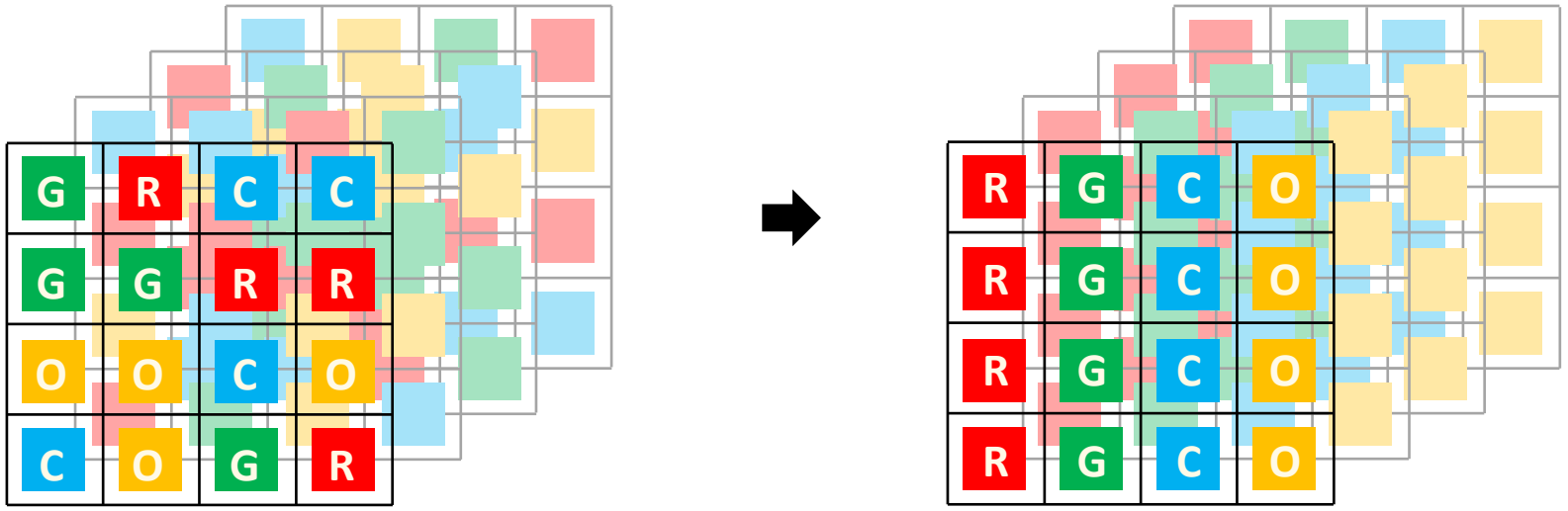
Key step: permute each column so we reach the situation where the $n$ items destined to go to a given column (e.g., a given color) end up in different rows




The algorithm applies to **rectangular tables** without modification

# Fat Rubik Table Problem

**Fat Rubik table problem**: Given an $n \times n \times K$ prism filled with $n$ types of items of $nK$ each. In a *shuffle*, one may permute a single fat row or fat column arbitrarily. How many shuffles are needed for realizing an arbitrary reconfiguration?




**Theorem.** The fat Rubik table problem can be solved with $2n$ shuffles.

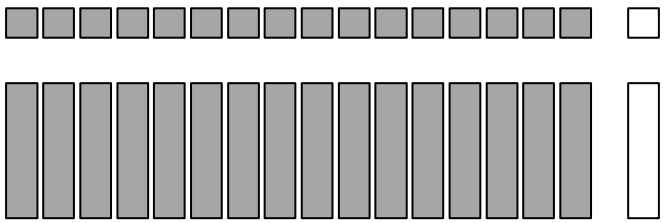
**Corollary.** The labeled fat Rubik table problem can be solved with $3n$ shuffles.

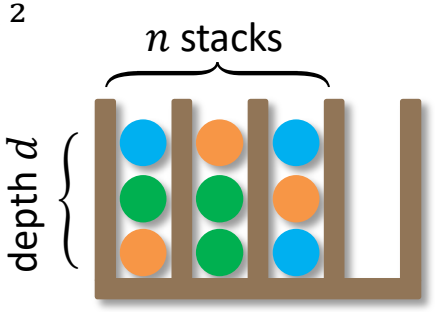
# Sketch of $O(nd)$-Pop-Push Algorithm

For arbitrary fixed $n$, recursively: $d = 1, d = \sqrt{n}, d = n, \dots, d = n^{\frac{m}{2}}$

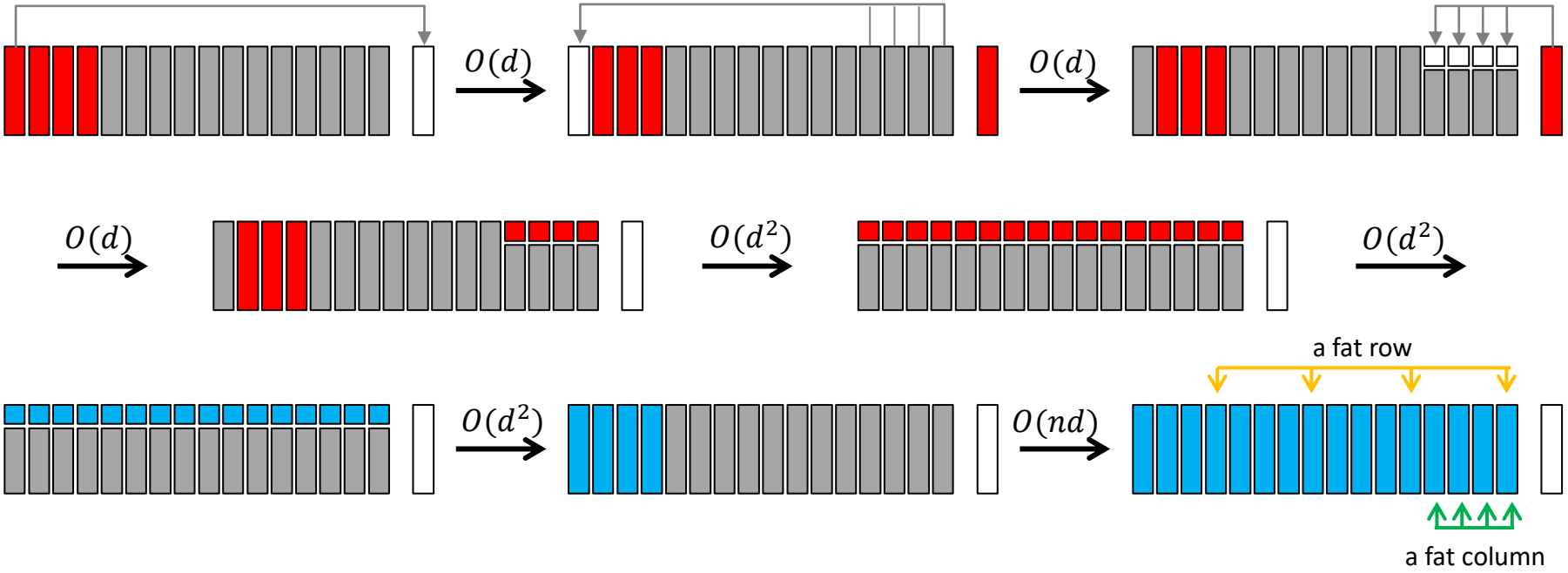
Example: $n = 16, d = 1$

$n = 16, d = \sqrt{n} = 4$



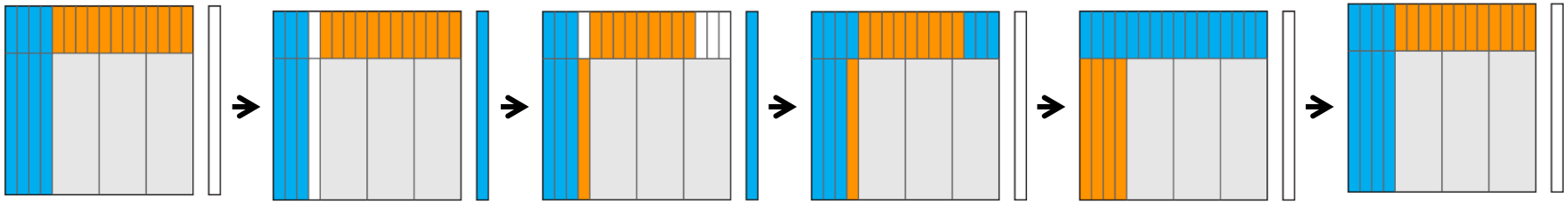



**Simulate** fat column/row shuffles (for applying fat Rubik table algorithm)



# Sketch of $O(nd)$-Pop-Push Algorithm, cont.

$n = 16, d = n = 16$



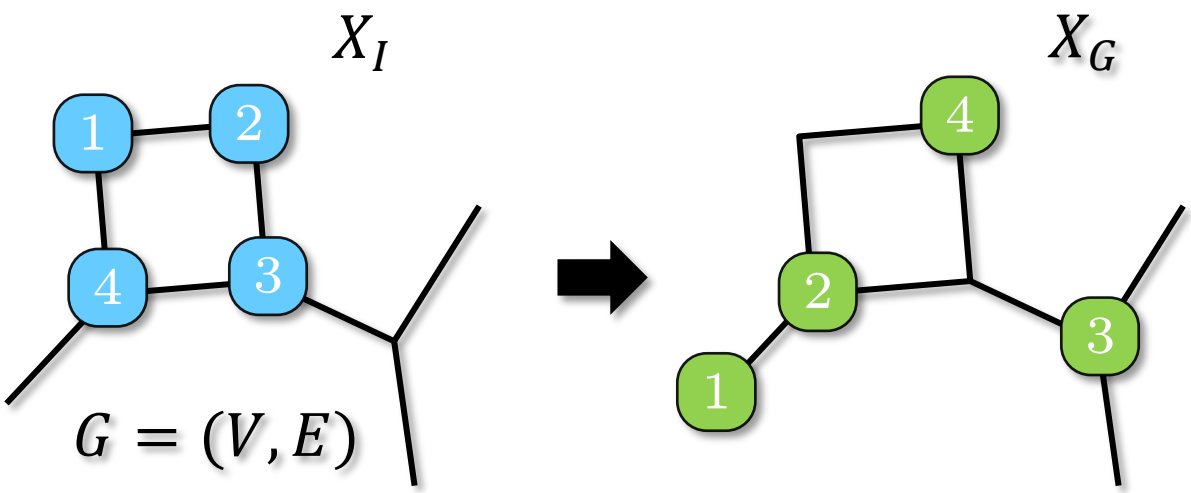
……

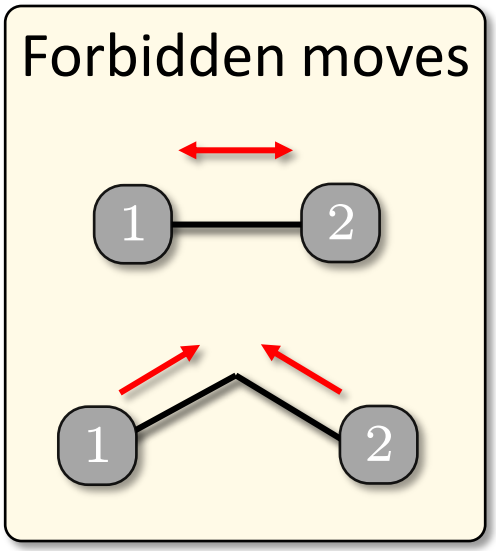
**Theorem (Improved CSR/LSR Upper Bound).** LSR (and therefore, CSR) with $d \leq n^{m/2}$ for $m \geq 0$ can be solved using $O(3^m nd)$ pop-pushes.

**Corollary (Linear CSR/LSR Upper Bound).** For arbitrary fixed real number $c > 0$, LSR (and therefore, CSR) with $d \leq \lceil cn \rceil$ can be solved using $O(nd)$ pop-pushes.

**Conclusion:** We "**almost**" closed the gap for CSR (lower bound: $\Omega(nd)$) and LSR (lower bound: $\Omega(nd\ (1 + \log d\ /\ \log n))$

# [Discrete] Multi-Robot Path Planning (MRPP)

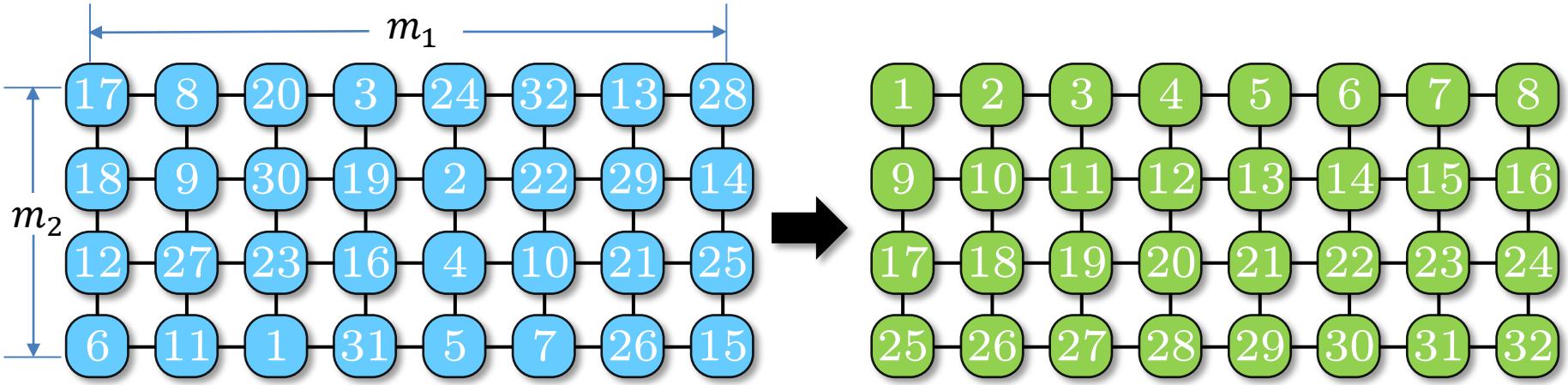



Given $(G, X_I, X_G)$, seek collision free $P = \{p_1, \dots, p_n\}$

- ▷ Min makespan: $\min_{P \in \mathcal{P}} \max_{p_i \in P} time(p_i)$
- ▷ NP-hard even for grids (Demaine et al. 2018)
- ▷ Best known poly-time algorithm computes $O(1)$-opt solution with large constants (>10)
- ▷ Note: we focus on **expected** $O(1)$-opt here


# MRPP on Grids: Main Results



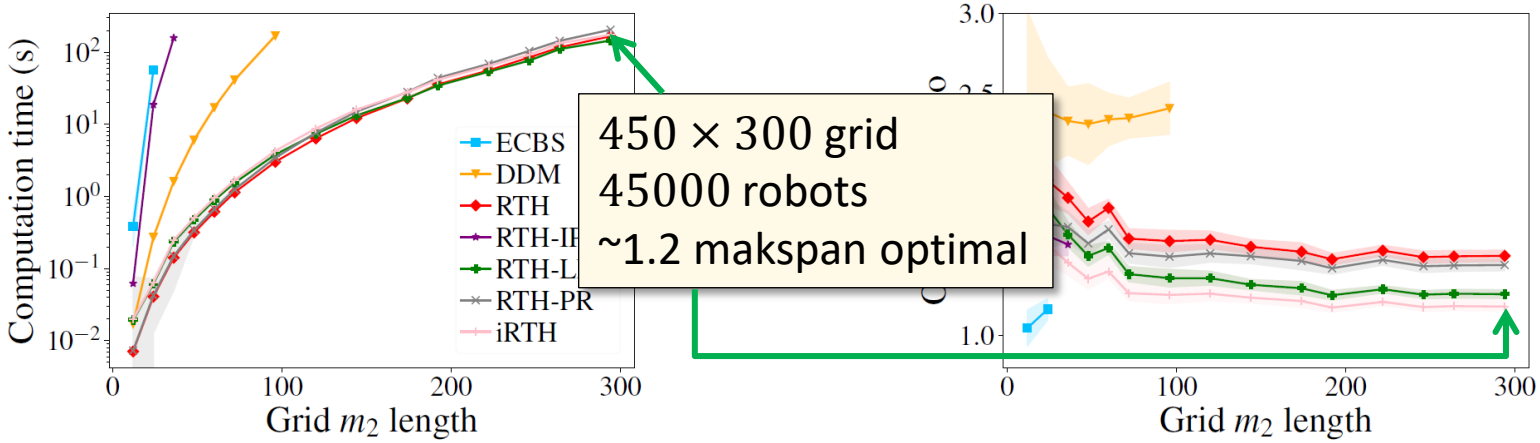

**1/2**

**Theorem ($\leq 1/3$ Density).** On a 2D $m_1 \times m_2$ grid, $m_1 \geq m_2$, for $n \leq \frac{m_1 m_2}{3}$ robots with randomly distributed start and goal configurations, a $(1 + \frac{m_2}{m_1+m_2})$ makespan-optimal solution can be computed in low polynomial time, with high probability.

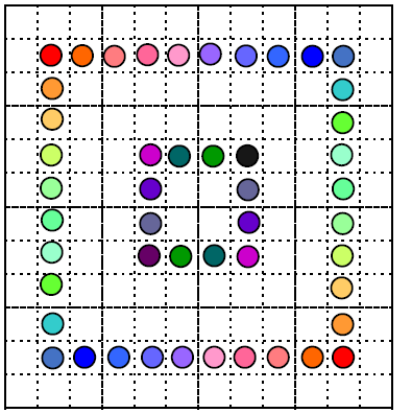
**Theorem (100% Density).** On a 2D $m_1 \times m_2$ grid, $m_1 \geq m_2$, for $n = m_1 m_2$ robots with randomly distributed start and goal configurations, a **4**$(1 + \frac{m_2}{m_1+m_2})$ makespan-optimal solution can be computed in low polynomial time, with high probability.

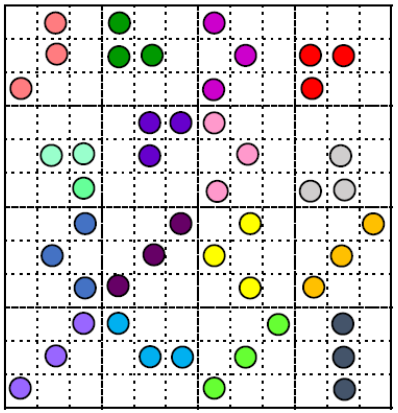
# Simulation Based Evaluation (1/3 Density)

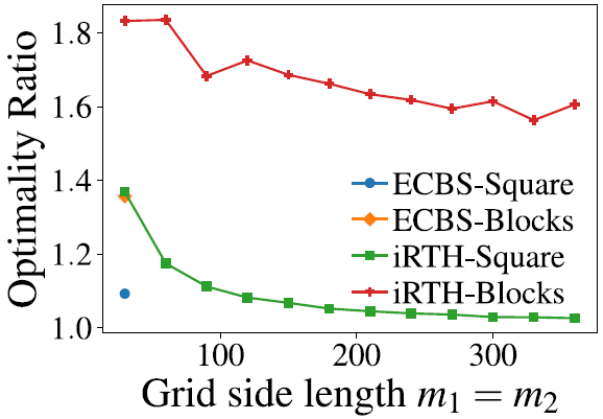
Scalability and comparison on $m_1 \times m_2$ grids with $m_1 : m_2 = 3 : 2$




Performance on specific patterns, $m_1 : m_2 = 1 : 1$







# Achieving 1.$x$-Opt for 1/3 Robot Density

**Theorem (Upper Bound).** For an MRPP instance on an $m_1 \times m_2$ grid with random $X_I, X_G$ for $n = \frac{m_1 m_2}{3}$ robots, in low polynomial time and with high probability, an $[\boldsymbol{m_1 + 2m_2 + o(m_1)}]$-makespan solution can be computed.

**Theorem (Lower Bound).** On an $m_1 \times m_2$ grid with random $X_I, X_G$ for $n = \frac{m_1 m_2}{3}$ robots, with high probability, an MRPP instance has an makespan of $\boldsymbol{m_1 + m_2 - o(m_1)}$.

**Theorem (Sub 1.5-optimality in Poly Time).** On an $m_1 \times m_2$ grid with random $X_I, X_G$ for $n = \frac{m_1 m_2}{3}$ robots, a $(\mathbf{1} + \frac{\boldsymbol{m_2}}{\boldsymbol{m_1 + m_2}})$ makespan-optimal solution can be computed in low polynomial time, with high probability.

**UB:** $\boldsymbol{m_1 + 2m_2 + o(m_1)}$

---

**LB:** $\boldsymbol{m_1 + m_2 - o(m_1)}$

||

$(\mathbf{1} + \frac{\boldsymbol{m_2}}{\boldsymbol{m_1 + m_2}})$

$$m_1 \approx m_2,\ \left(1 + \frac{m_2}{m_1 + m_2}\right) \to 1.5$$

$$m_1 \gg m_2,\ \left(1 + \frac{m_2}{m_1 + m_2}\right) \to 1.0$$

$\Big\} \to$ (1,1.5] optimality asymptotically, high probability, low-poly time

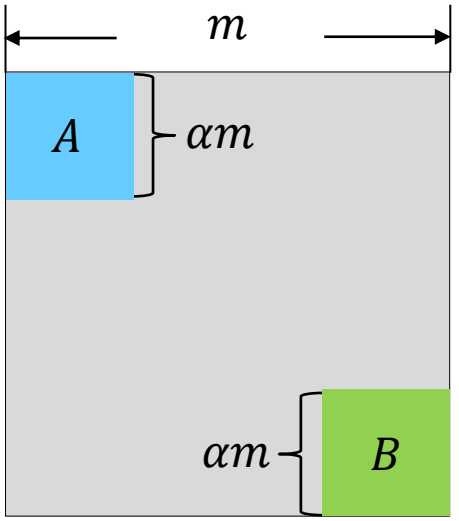
# Establishing the Lower Bound

**Observation.** On an $m \times m$ grid with random $X_I, X_G$, a robot starts from $A$ and goes to $B$ with probability $\alpha^4$, yielding a minimum makespan of $2(1-2\alpha)m$.

**Observation.** For $cm^2$ robots, the probability that at least one robot has a minimum makespan of $2(1-2\alpha)m$ is no less than $p = 1-(1-\alpha^4)^{cm^2} > 1 - e^{-c\alpha^4 m^2}$

Using $(1-x)^y < e^{-xy}$ for $0 < x < 1$ and $y > 0$

For arbitrary fixed $\alpha > 0$, let $m \to \infty$, then $p \to 1$, yielding

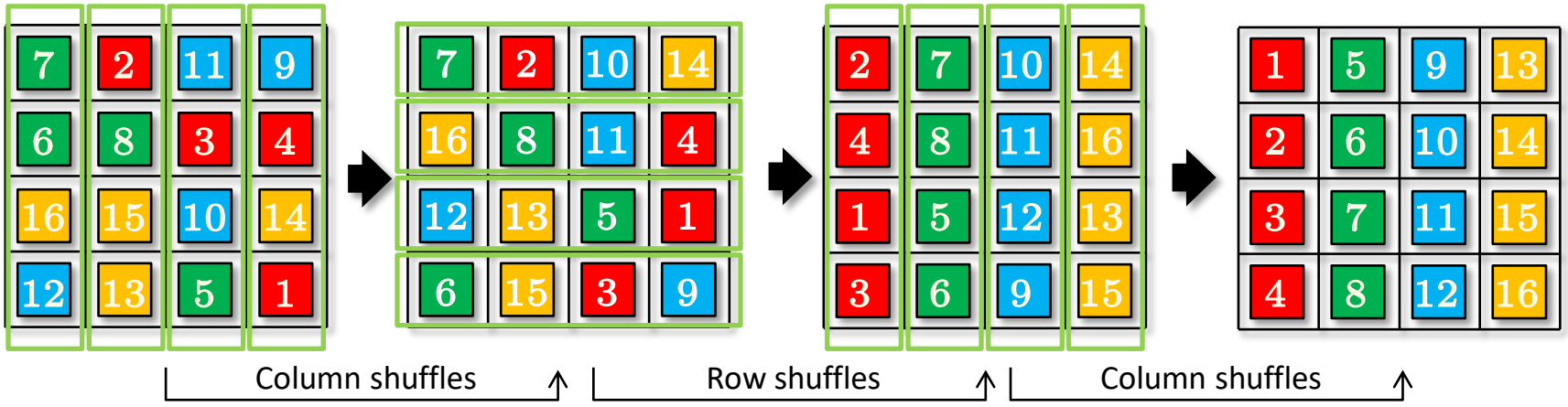



**Proposition.** The minimum makespan of a random MRPP instance on an $m_1 \times m_2$ grid with $\Omega(m_1 m_2)$ robots is $m_1 + m_2 - o(m_1)$ with arbitrarily high probability as $m_1 \to \infty$

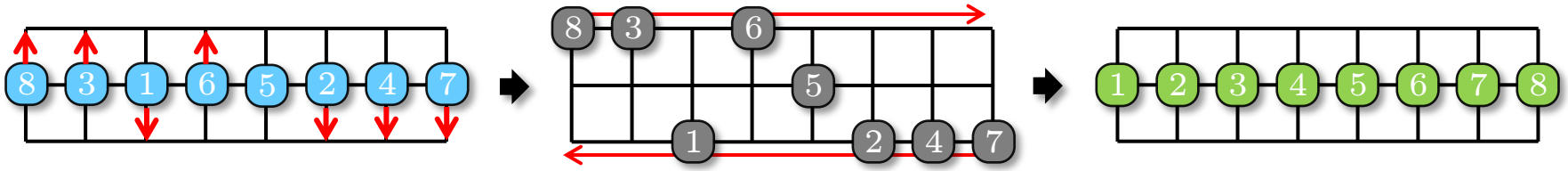
# Establishing the Upper Bound – Outline

Strategy: breaking into subproblems (for 1/3 and 100%)

- Subproblem 1: row/column shuffle based high level planning

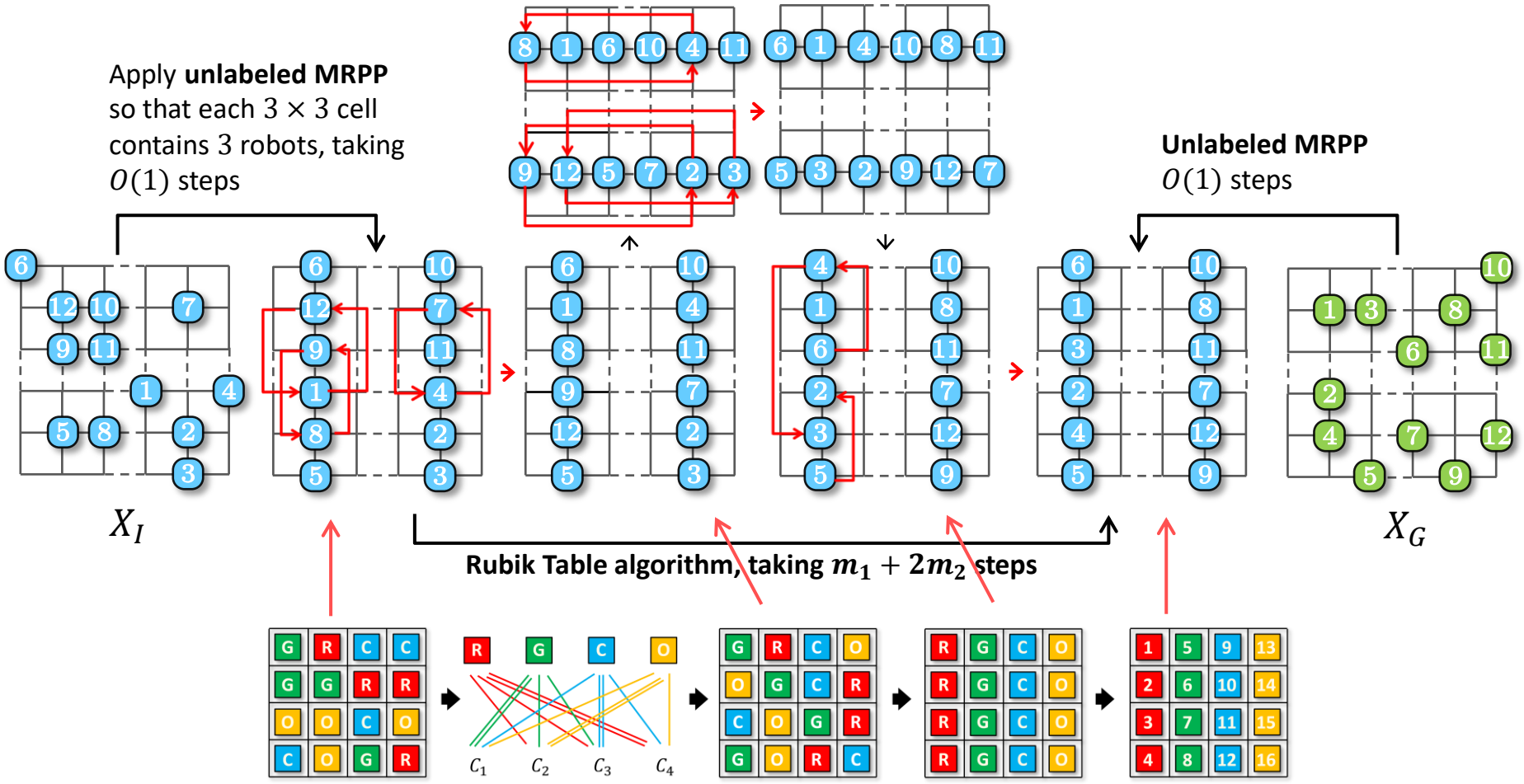



- Subproblem 2: realizing shuffles for robots
  - For 1/3 density, relatively easy



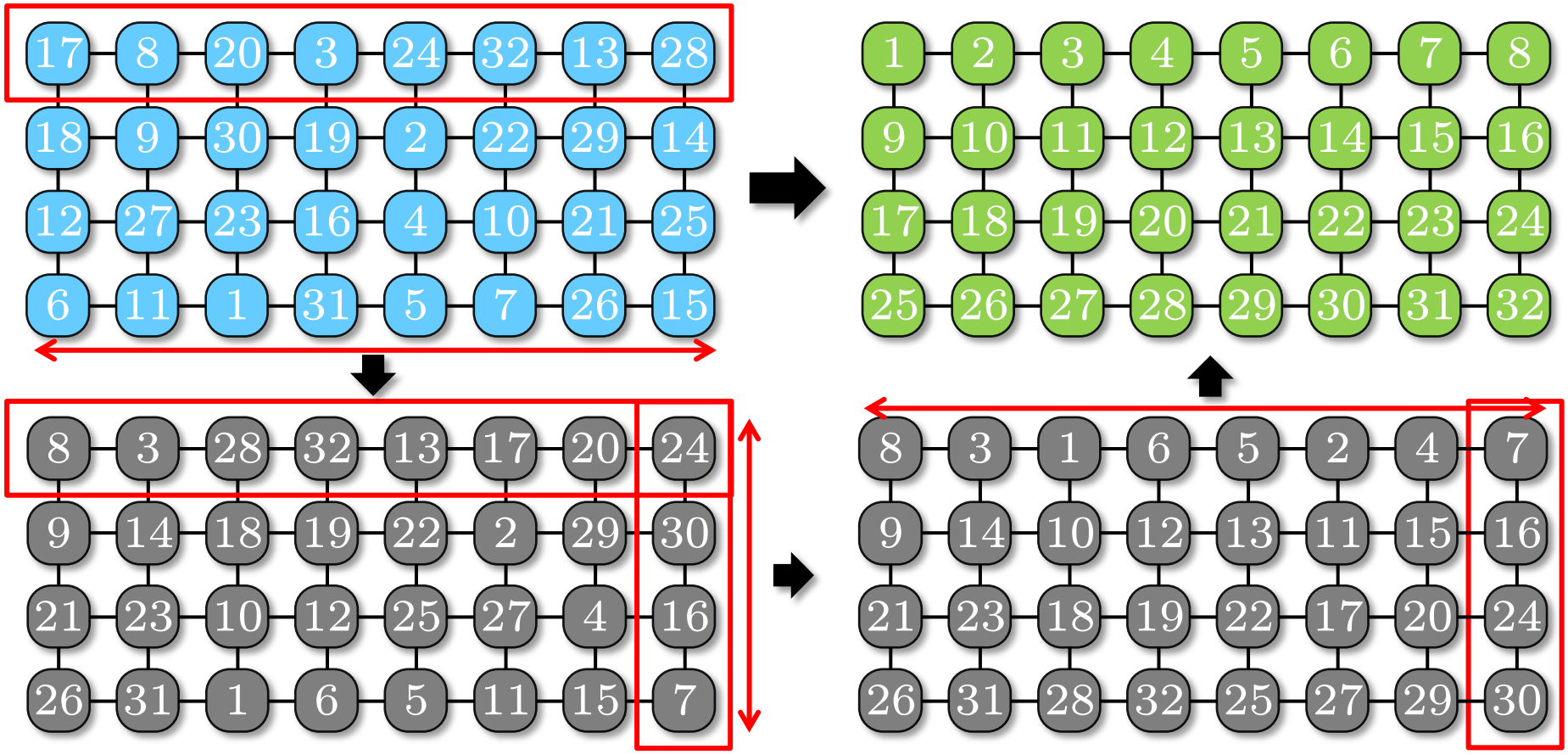

  - More difficult for 100% density

# Upper Bound for 1/3 Robot Density

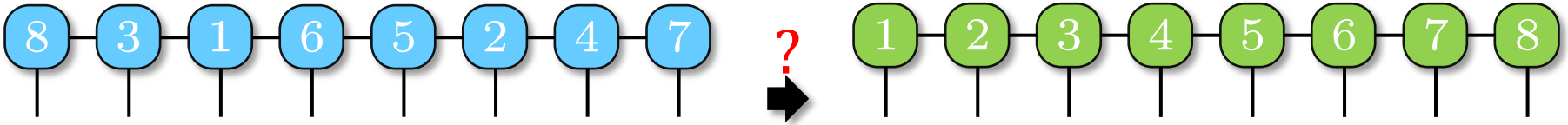



Makespan of the overall algorithm is $m_1 + 2m_2 + o(m_1)$

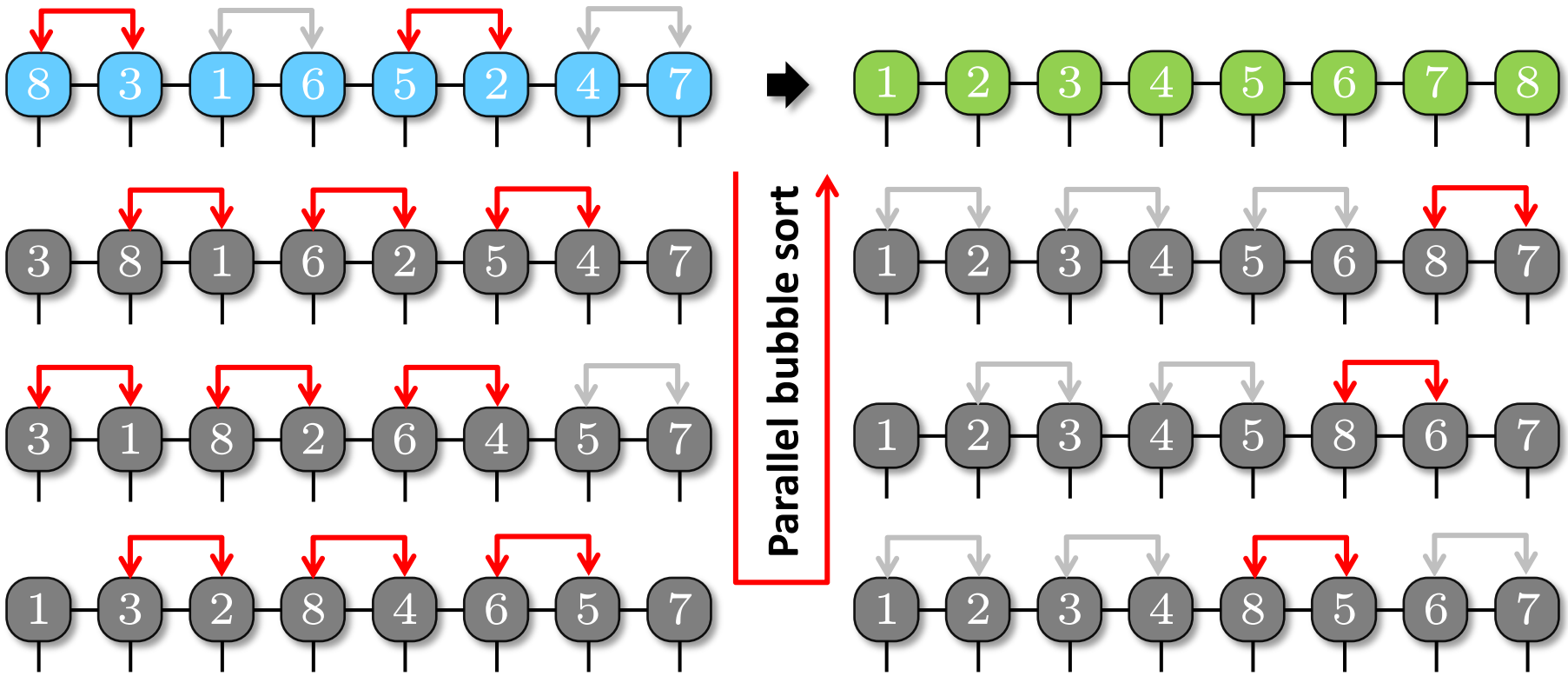
# MRPP with 100% Density




Rubik Table reduces MRPP at 100% density to realizing **row shuffles on grids.**



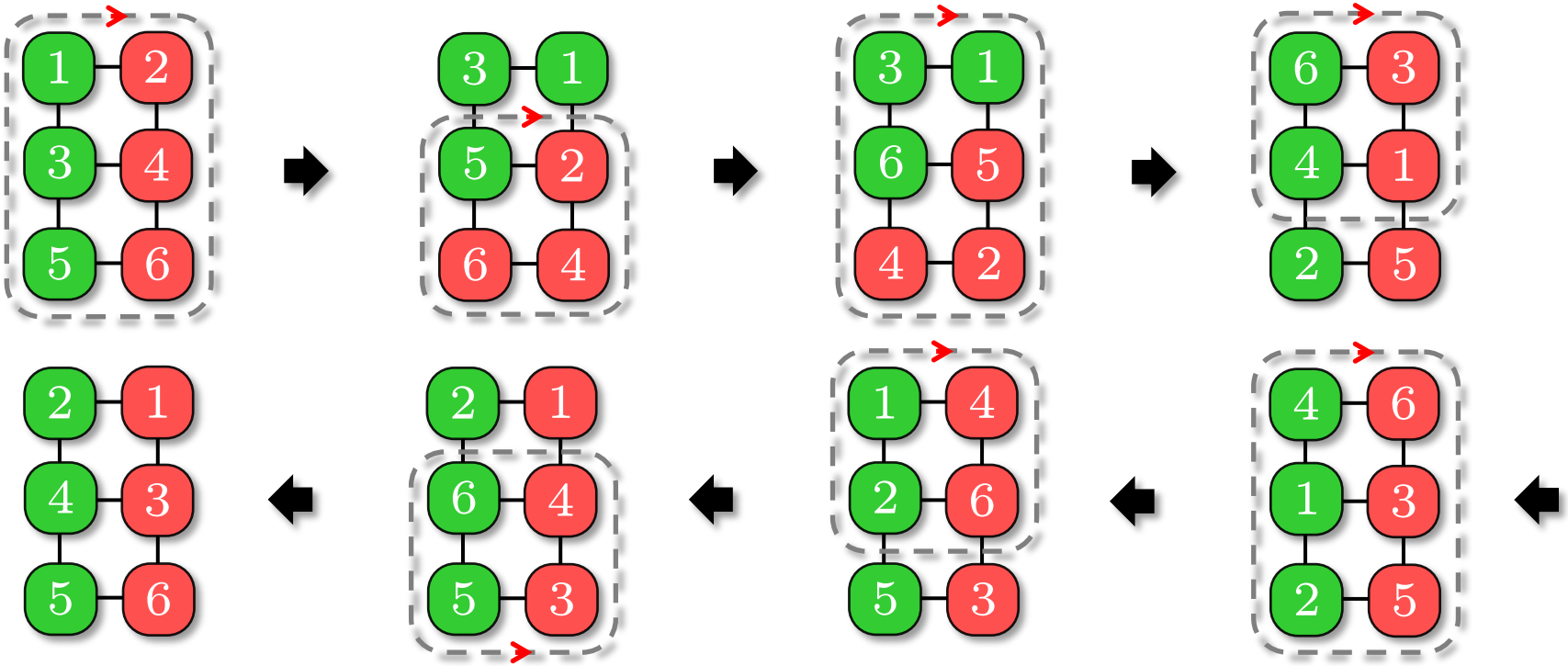
# Reducing Line Shuffle to Pair Swaps




Parallel bubble sort reduces row shuffle to **pairwise shuffles**

# Realizing Pair Swaps on Grids

**Theorem.** Arbitrary reconfiguration on a $2 \times 3$ grid can be done in 7 steps.

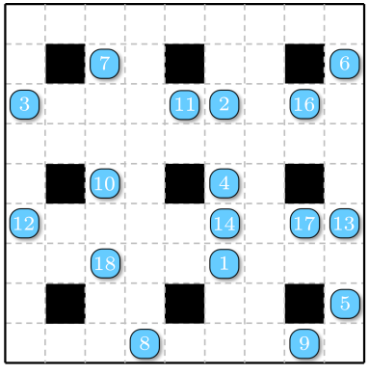


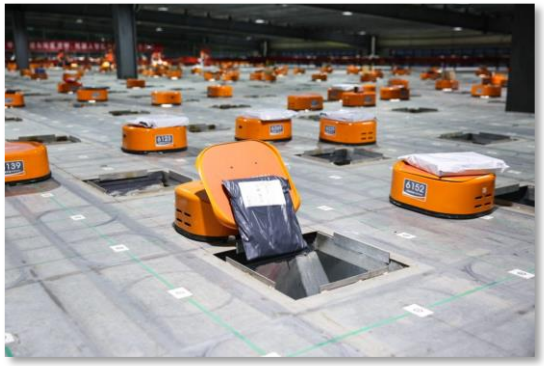

**Theorem.** On a 2D $m_1 \times m_2$ grid, $m_1 \geq m_2$, for $n = m_1 m_2$ robots with randomly distributed start and goal configurations, a $7(1 + \frac{m_2}{m_1 + m_2})$ makespan-optimal solution can be computed in low polynomial time, with high probability.

# Extensions: Obstacles and Higher Dimensions

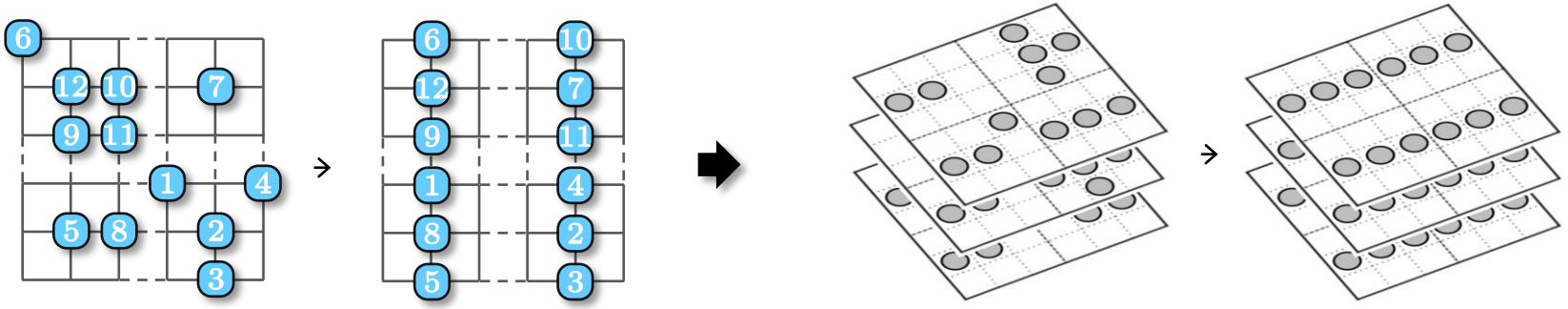
Rubik Table algorithms naturally applies to

- Grids with regularly distributed obstacles
  - Applicable to parcel sorting



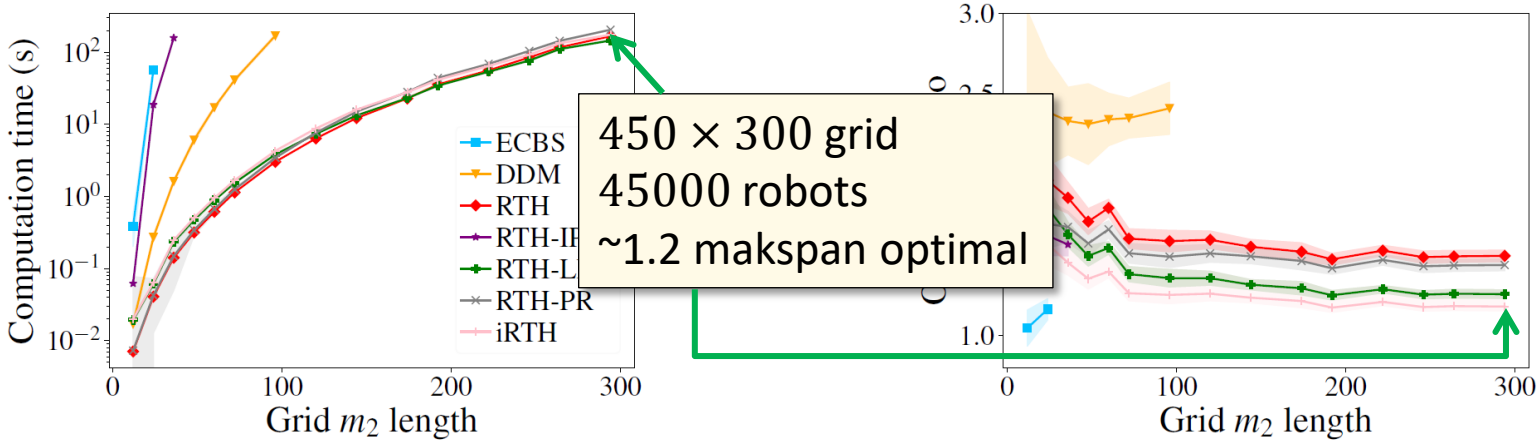



- And arbitrary dimensions

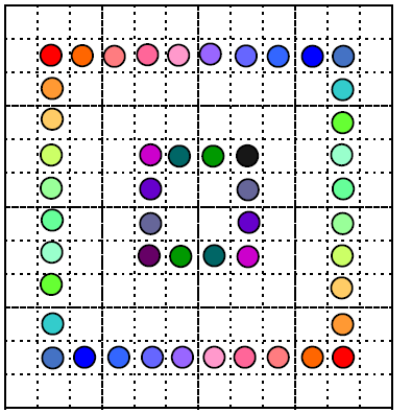


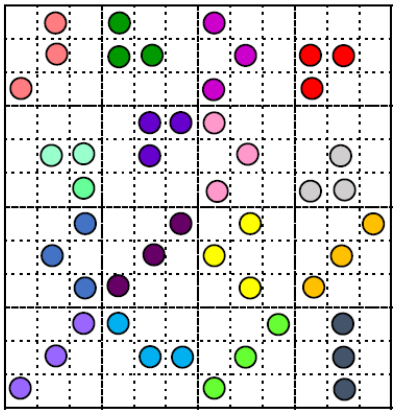
# Simulation Based Evaluation (1/3 Density)

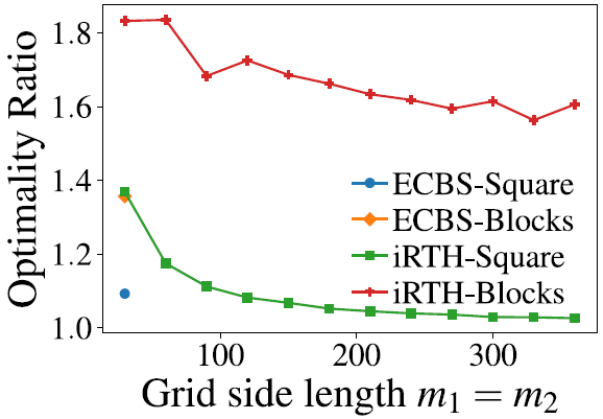
Scalability and comparison on $m_1 \times m_2$ grids with $m_1 : m_2 = 3 : 2$



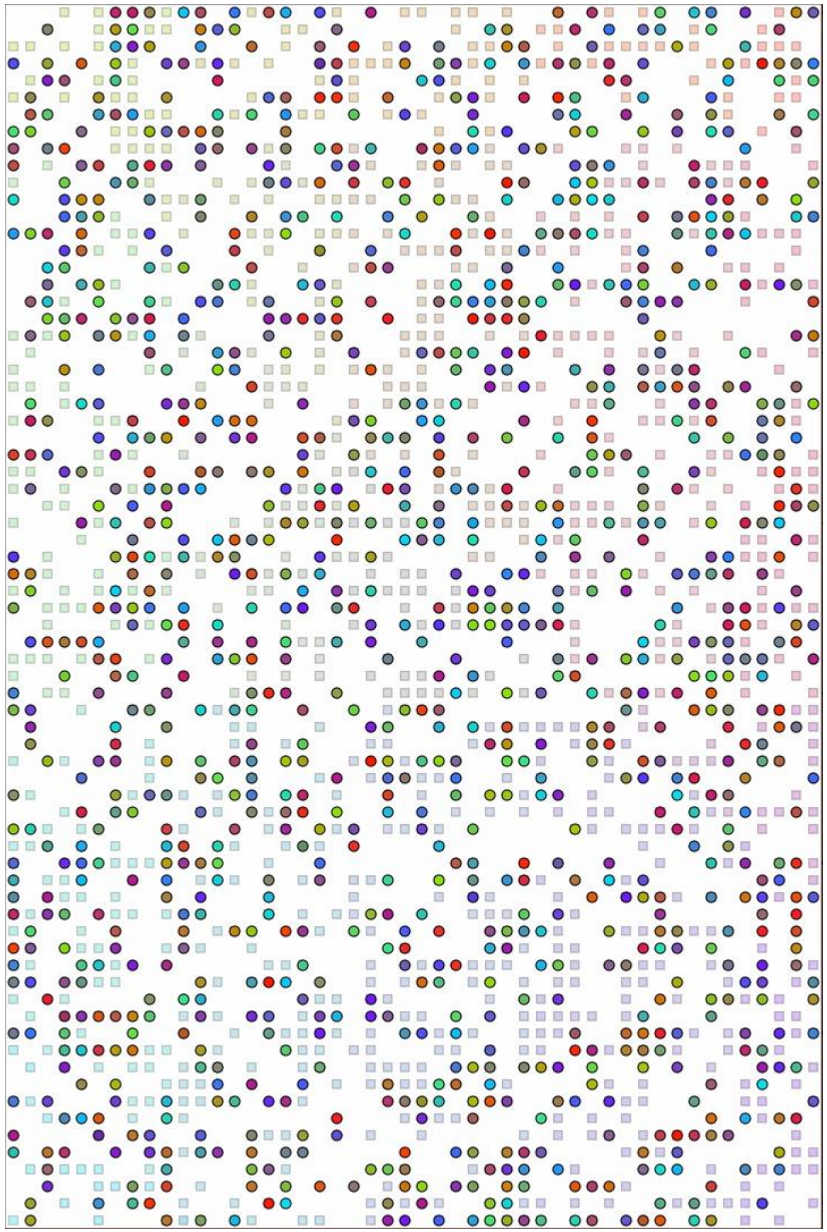

Performance on specific patterns, $m_1 : m_2 = 1 : 1$

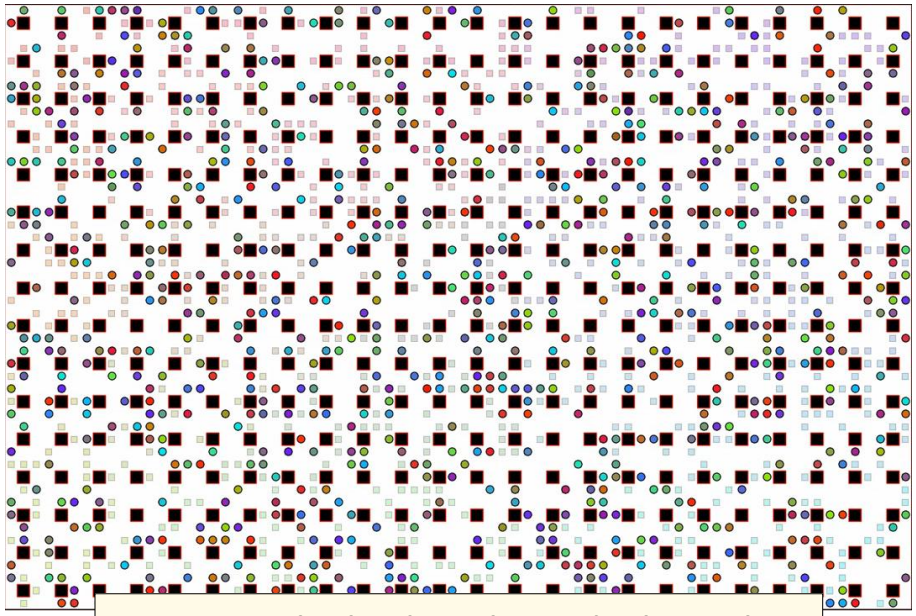




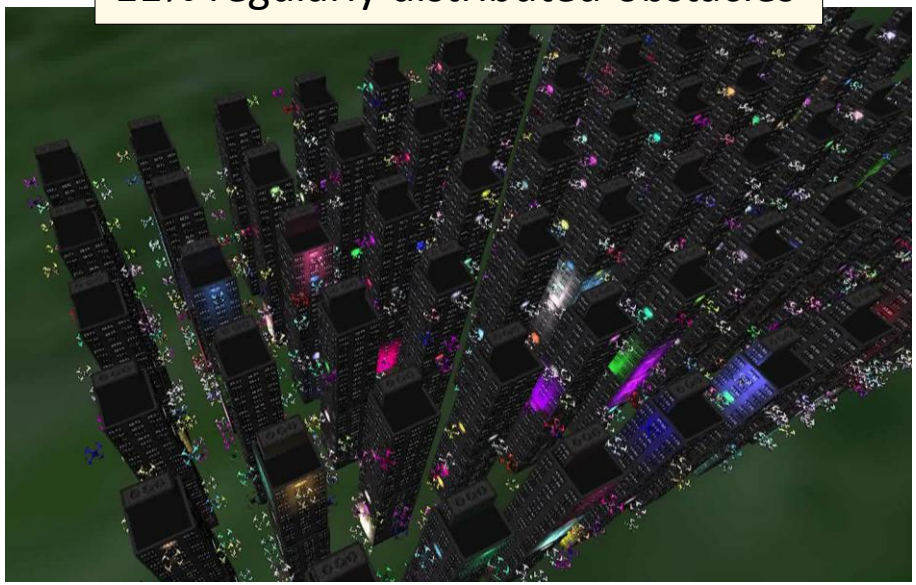


# Selected Simulations

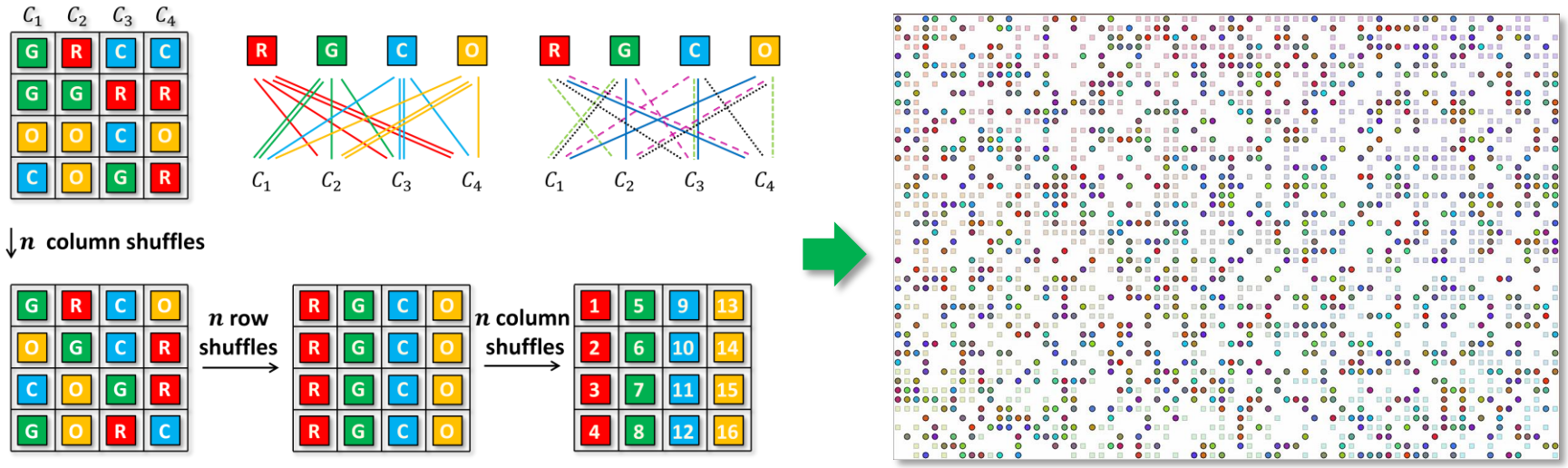




11% regularly distributed obstacles



# Rubik Tables, Stack Rearrangement, and MRPP




**Some natural open questions**

- Even better optimality?
- Other optimality objectives?
  - Can directly generalize to provide 9/2 of maximum possible throughput
- More general obstacle support?