# CSCI-564
# Constraint Processing and Heuristic Search

Lecture 9 – Pattern Database

StFX

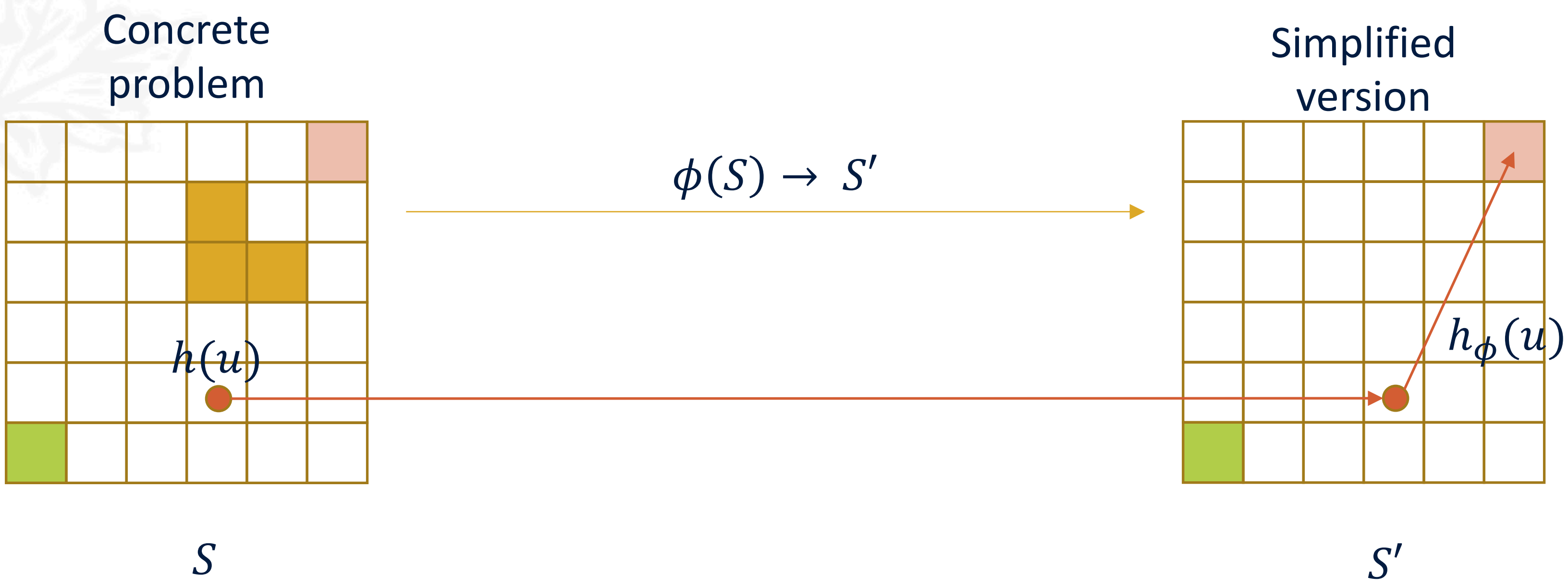
# Recap

- Previously we saw that heuristics can be calculated on demand.
  - Calculated online.
- We need to ensure that calculating that the computing effort is less than a blind search.
- If you save the heuristics, the number will grow over time.

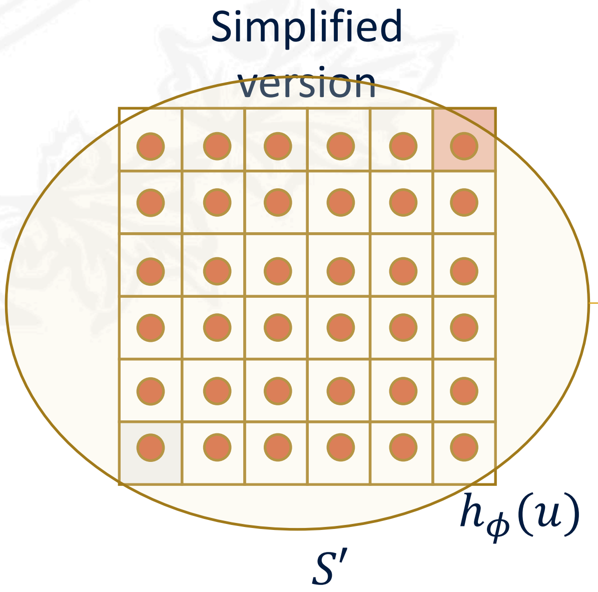
Do you have another approach in mind?

# Pattern Database

- We could evaluate the entire abstract search space before search in the concrete problem.



# Pattern Database




Pattern Database

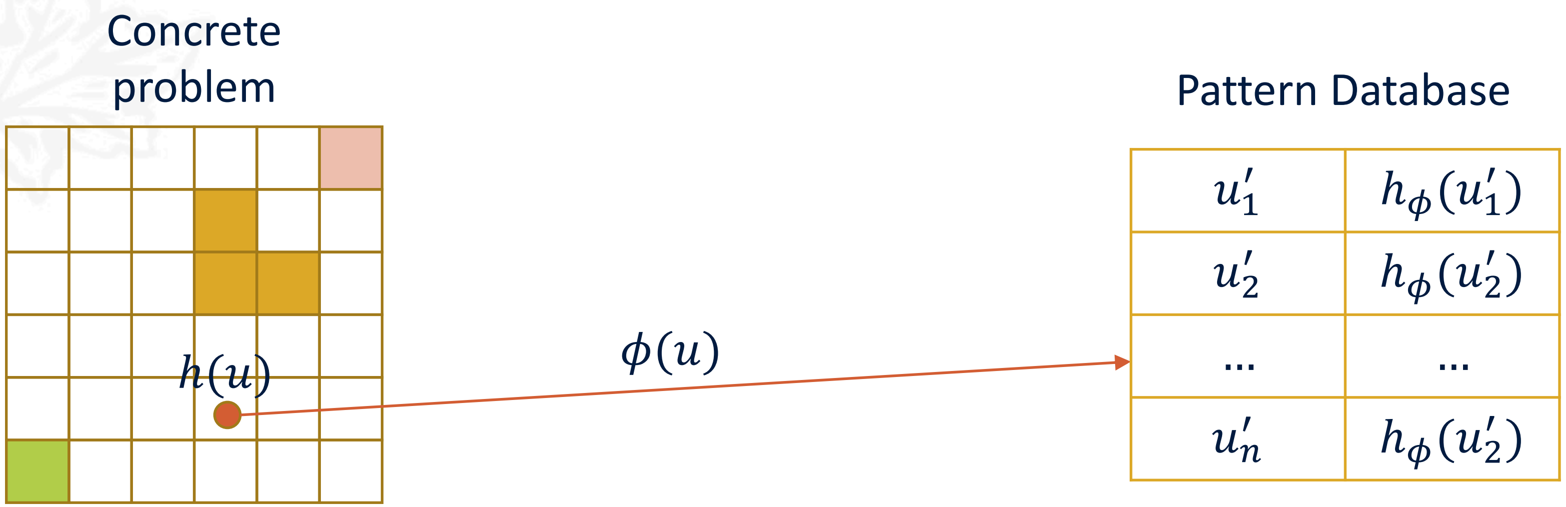
| | |
|---|---|
| $u_1'$ | $h_\phi(u_1')$ |
| $u_2'$ | $h_\phi(u_2')$ |
| … | … |
| $u_n'$ | $h_\phi(u_2')$ |

Shortest path from $u_1{}'$ to $\phi(t)$

Number of state in $S'$

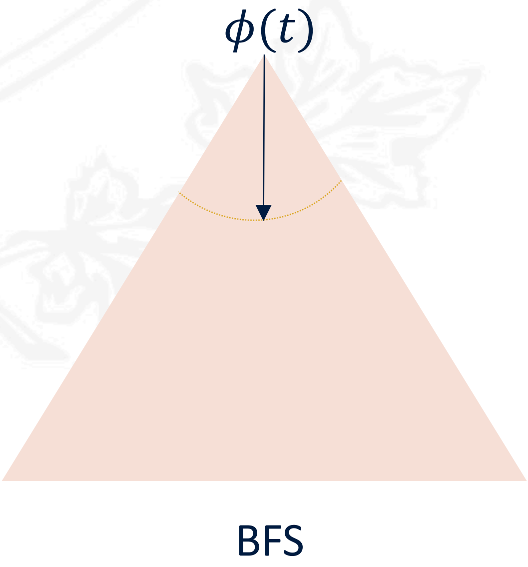
# Pattern Database

- During the concrete search we are looking directly inside the pattern database.



# Pattern Database

- How can we create this database?
  - Run BFS backward!

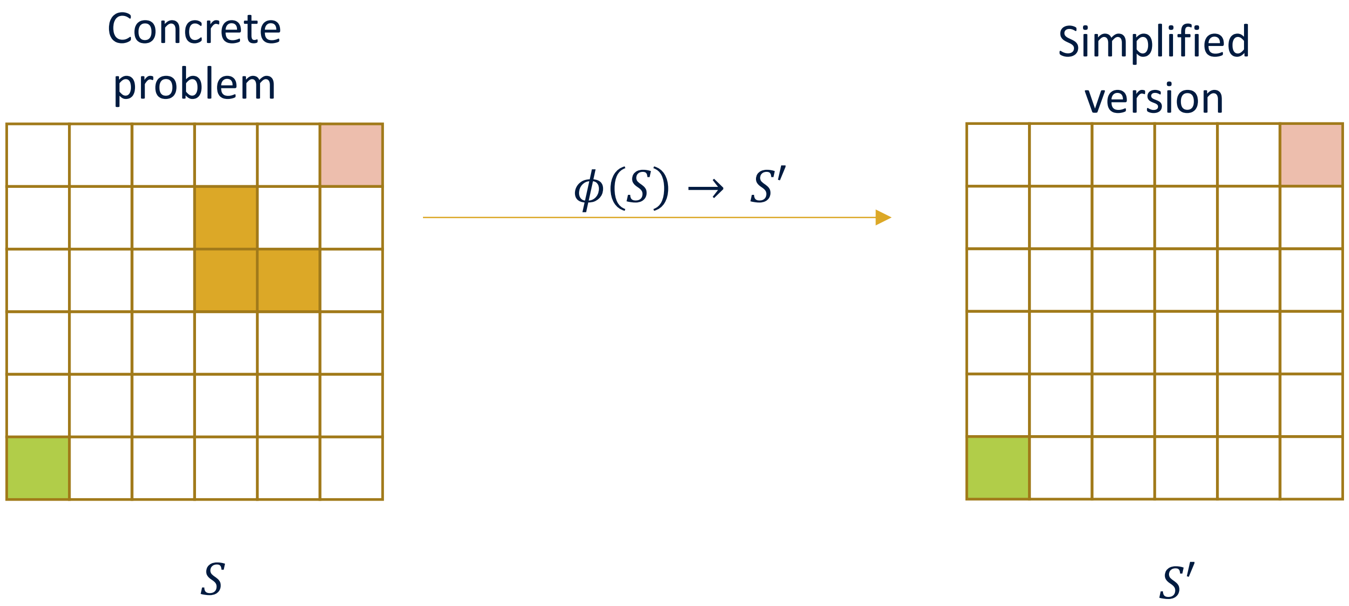



It assumes that there is a set of actions $A^{-1} = \{a^{-1} | a \in A\}$.

- For each action $a$, it exists an inverse action $a^{-1}$.
- Such that $v = a(u)$ iff $i = a^{-1}(v)$.

# Pattern Database

- Exercise:
  - For the grid problem propose a set of action $A^{-1}$.
  - What do you conclude?

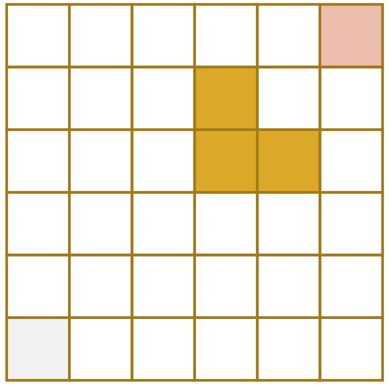


# Pattern Database

- Exercise:
  - For the grid problem propose a set of action $A^{-1}$.
  - What do you conclude?

  - The set of actions $A^{-1}$ is equal to $A$ .
  - This types of problem is called reversible (undirected graph problem).

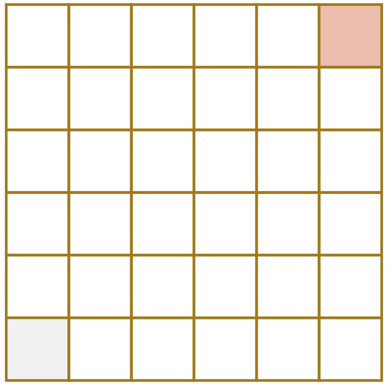
Concrete problem



$S$

$\phi(S) \rightarrow S'$

Simplified version



$S'$

# Pattern Database

- Works with weighted graph
- Dijkstra can be used to create a pattern database.

# $(n^2 - 1)$-Puzzle Problem

- $(n^2 - 1)$-Puzzle Problem:
  - States in $(n^2$-1)-Puzzle problem: $\frac{(n^2)!}{2}$
  - 181,440 possible states for 8-Puzzle
  - $1.05 \times 10^{13}$ possible states for 15-Puzzle

- What heuristic can you use?
  - The number of misplaced tiles
  - The maximum Manhattan distance
  - …

| | | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Goal

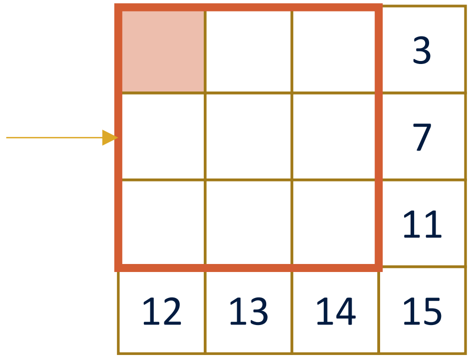
| | | | |
|---|---|---|---|
| 1 | 4 | 5 | 6 |
| 10 | | 2 | 11 |
| 14 | 3 | 7 | 15 |
| 8 | 9 | 13 | 12 |

Start

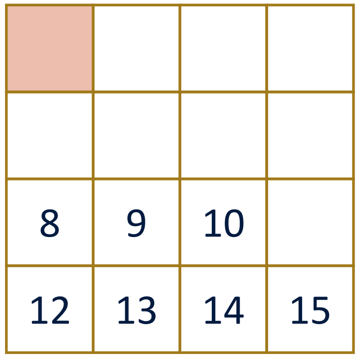
# $(n^2 - 1)$-Puzzle Problem

- Why are we calling it pattern database and not simply database?
  - One abstraction consist to ignore some tiles.
    - Their labels are replaced with a symbol (or nothing).
  - The remaining set of tiles is called a pattern.
    - The fringe
    - The corner

You don't care what is inside.



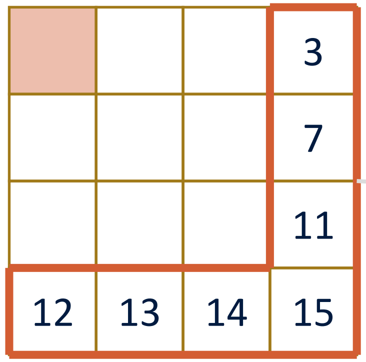

Fringe




Corner

# $(n^2 - 1)$-Puzzle Problem

- You store each combination of a pattern in the database
- Then, you calculate the heuristic value with your heuristic function(Manhattan distance, etc.)




Pattern Database

| | |
|---|---|
| $u'_1$ | $h_\phi(u'_1)$ |
| $u'_2$ | $h_\phi(u'_2)$ |
| … | … |
| $u'_n$ | $h_\phi(u'_2)$ |

# $(n^2 - 1)$-Puzzle Problem

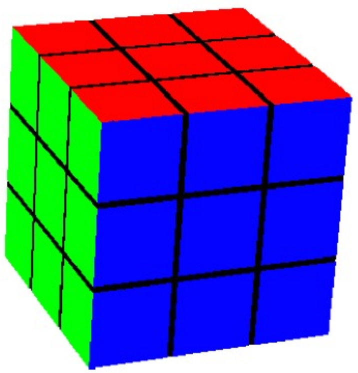
- Combining Manhattan distance and one pattern reduces the number of expanded by two orders of magnitude.
- Combined with both patterns, it reduces by three orders of magnitude.

| Experiment | Total nodes | Tree size (%) | Improvement |
|---|---|---|---|
| MD | 36,302,808,031 | 100.00 | 1 |
| MD+FR | 105,067,478 | 0.29 | 346 |
| MD+CO | 83,125,633 | 0.23 | 437 |
| MD+FC | 34,987,894 | 0.10 | 1038 |

Number of nodes decreases

# Rubik's Cube

- A Rubik's Cube is composed of:
  - 27 cubies: 26 visible
- State:
  - 8 corners: 3 colors
  - 12 edges: 2 colors
  - 6 middles: 1 color



- The number of states: $8!\times 3^8 \times 12!\times 2^{12}/12 \approx 43\times 10^{18}$
- The actions:
  - Rotating 90° clockwise
  - Rotating 180°
  - Rotating 270° (90 counterclockwise)

# Rubik's Cube

- What happens if we expand the search tree?

Branching Factor
$b = 3\times 6$ Faces = 18

After first move, the second step cannot be on the same face

b = 15

Forbid move that twist two faces in a row in opposite order

b = 13.35

number of nodes $2.47\times 10^{20}$

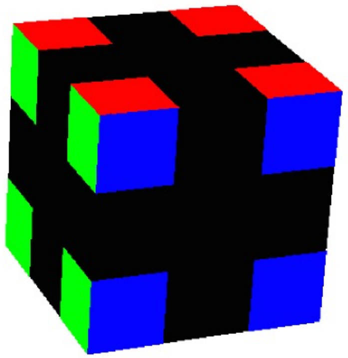
$> 4.3\times 10^{19}$

| Depth | Nodes |
|---|---|
| 1 | 18 |
| 2 | 243 |
| 3 | 3,240 |
| 4 | 43,254 |
| 5 | 577,368 |
| 6 | 7,706,988 |
| 7 | 102,876,480 |
| 8 | 1,373,243,544 |
| 9 | 18,330,699,168 |
| 10 | 244,686,773,808 |
| 11 | 3,266,193,870,720 |
| 12 | 43,598,688,377,184 |
| 13 | 581,975,750,199,168 |
| 14 | 7,768,485,393,179,328 |
| 15 | 103,697,388,221,736,960 |
| 16 | 1,384,201,395,738,071,424 |
| 17 | 18,476,969,736,848,122,368 |
| 18 | 246,639,261,965,462,754,048 |

**Nodes in search tree as a function of depth**

# Rubik's Cube

- Pattern Database:
  - 8 corner cubies: $8!\times 3^7 = 88{,}179{,}840$ possible combinations, require 44,089,920 bytes of memory (42 megabytes) – improved heuristic to 8.764
  - 6 of 12 edge cubies: $\frac{12!}{6!}\times 2^6 = 42{,}577{,}920$ states, require 21,288,960 bytes (20 megabytes) – improved heuristic to 7.668
  - Combine 8 corner and two groups of 6 edge cubies require a memory of 82 megabytes – improved heuristic to 8.878



# Rubik’s Cube

- Ten solvable instances of Rubik's Cube, by making 100 random moves each, starting from the goal state.

| Problem | Depth | Nodes Generated |
|---|---|---|
| 1 | 16 | 3,720,885,493 |
| 2 | 17 | 11,485,155,726 |
| 3 | 17 | 64,937,508,623 |
| 4 | 18 | 126,005,368,381 |
| 5 | 18 | 262,228,269,081 |
| 6 | 18 | 344,770,394,346 |
| 7 | 18 | 502,417,601,953 |
| 8 | 18 | 562,494,969,937 |
| 9 | 18 | 626,785,460,346 |
| 10 | 18 | 1,021,814,815,051 |

# Multiple Pattern Databases

- We can improve the pattern database method, by using multiple pattern databases.

Manhattan distance → (Mapping different states into smaller patterns) → Pattern Databases → (Break the large pattern database into different smaller ones) → Multiple Pattern Databases

# Multiple Pattern Databases

- Granularity: A vector indication how many constants in the original domain are mapped to each constant in the abstract domain.

Concrete domain

| | | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

$\phi_1$ →

Abstract domain

| | | | |
|---|---|---|---|
| | $x$ | $x$ | 3 |
| $x$ | $x$ | $x$ | 7 |
| $x$ | $x$ | $x$ | 11 |
| 12 | 13 | 14 | 15 |

$\phi_2$ →

Abstract domain

| | | | |
|---|---|---|---|
| | $x$ | $x$ | 3 |
| $x$ | $x$ | $x$ | $z$ |
| $x$ | $x$ | $x$ | 11 |
| $y$ | 13 | $y$ | $z$ |

The granularity of $\phi_1$ is <8,1,1,1,1,1,1,1,1>

The granularity of $\phi_2$ is <8,2,2,1,1,1,1>

| Concrete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | blank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\phi_1$ | $x$ | $x$ | 3 | $x$ | $x$ | $x$ | 7 | $x$ | $x$ | $x$ | 11 | 12 | 13 | 14 | 15 | blank |
| $\phi_2$ | $x$ | $x$ | 3 | $x$ | $x$ | $x$ | $z$ | $x$ | $x$ | $x$ | 11 | $y$ | 13 | $y$ | $z$ | blank |

# Multiple Pattern Databases

- Using $n$ pattern databases of size $m/n$ instead of one pattern database of size $m$ improves search performance.
- Experiments with the 8 puzzle:
  - Taking tow or smaller pattern databases results in very significant reductions in nodes over using a single large pattern database.
  - Using $n = 10$ reduces the number of nodes generated almost an order of magnitude over a single pattern database.

| Granularity | PDB Size | $n$ | Nodes Generated | CPU (secs) |
|---|---|---|---|---|
| $\langle 6,2,1 \rangle$ | 252 | 20 | 585 | 0.04 |
| $\langle 6,1,1,1 \rangle$ | 504 | 10 | 460 | 0.02 |
| $\langle 5,3,1 \rangle$ | 504 | 10 | 725 | 0.03 |
| $\langle 4,3,1,1 \rangle$ | 2,520 | 2 | 1,212 | 0.02 |
| $\langle 3,3,2,1 \rangle$ | 5,040 | 1 | 3,842 | 0.07 |

# Multiple Pattern Databases

- Rubik's Cube

| Granularity ⟨ corners ⟩⟨ edges ⟩ | PDB Size | $n$ | Nodes Generated |
|---|---|---|---|
| $\langle 8\rangle\langle 4,4,1,1,1,1\rangle$ | 13,305,600 | 8 | 2,654,689 |
| $\langle 8\rangle\langle 3,3,3,1,1,1\rangle$ | 17,740,800 | 6 | 2,639,969 |
| $\langle 8\rangle\langle 4,3,1o,1o,1,1,1\rangle$ | 26,611,200 | 4 | 3,096,919 |
| $\langle 8\rangle\langle 4,3,1o,1,1,1,1\rangle$ | 53,222,400 | 2 | 5,329,829 |
| $\langle 8\rangle\langle 4,3,1,1,1,1,1\rangle$ | 106,444,800 | 1 | 61,465,541 |

# Multiple Pattern Databases

- Proposition 1: Maxing smaller pattern databases could replace small $h$-values by larger ones, and substantially reduce the number of patterns with very small $h$-values

- Proposition 2: Eliminating low $h$-values is more important for improving search performance than retaining large $h$-values.

*Multiple Pattern Databases*, R. C. Holte, etc. ICAPS 2004

# Multiple Pattern Databases

- Making the pattern databases too small has a negative impact on performance.

| Granularity | PDB Size | $n$ | Nodes Generated | CPU (secs) |
|---|---|---|---|---|
| $\langle 6, 2, 1 \rangle$ | 252 | 20 | 3,132 | 0.112 |
| $\langle 6, 1, 1, 1 \rangle$ | 504 | 10 | 2,807 | 0.056 |
| $\langle 5, 3, 1 \rangle$ | 504 | 10 | 2,173 | 0.044 |
| $\langle 4, 3, 1, 1 \rangle$ | 2,520 | 2 | 3,902 | 0.027 |
| $\langle 3, 3, 2, 1 \rangle$ | 5,040 | 1 | 18,665 | 0.113 |

9 pancake puzzle