CS 460/560
Introduction to Computational Robotics
Fall 2019, Rutgers University

Lecture 16
Multi-Robot Path Planning

Instructor: Jingjin Yu

Outline

Multi-robot path planning

- ⇨Applications
- ⇨Formulations

Feasibility of graph-theoretic MPP

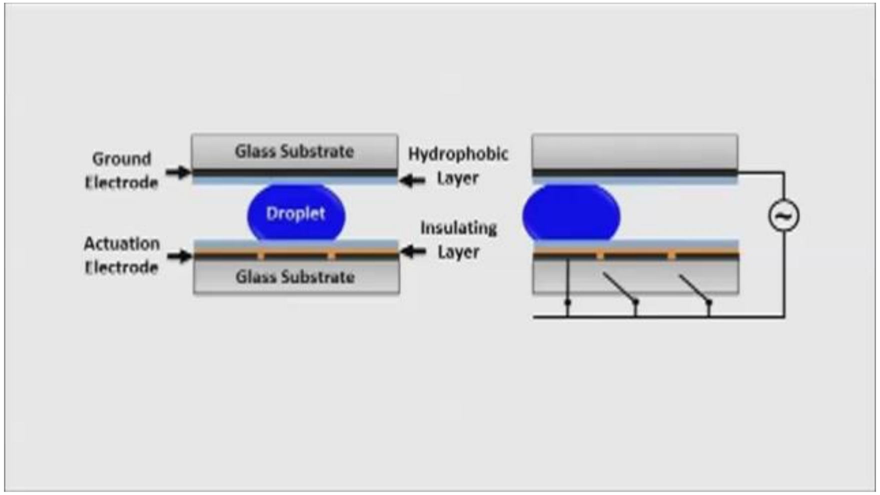
- ⇨The 15-puzzle variant
- ⇨The synchronous rotation variant

Structure and complexity of optimal MPP

Applications – Order Fulfillment

Applications

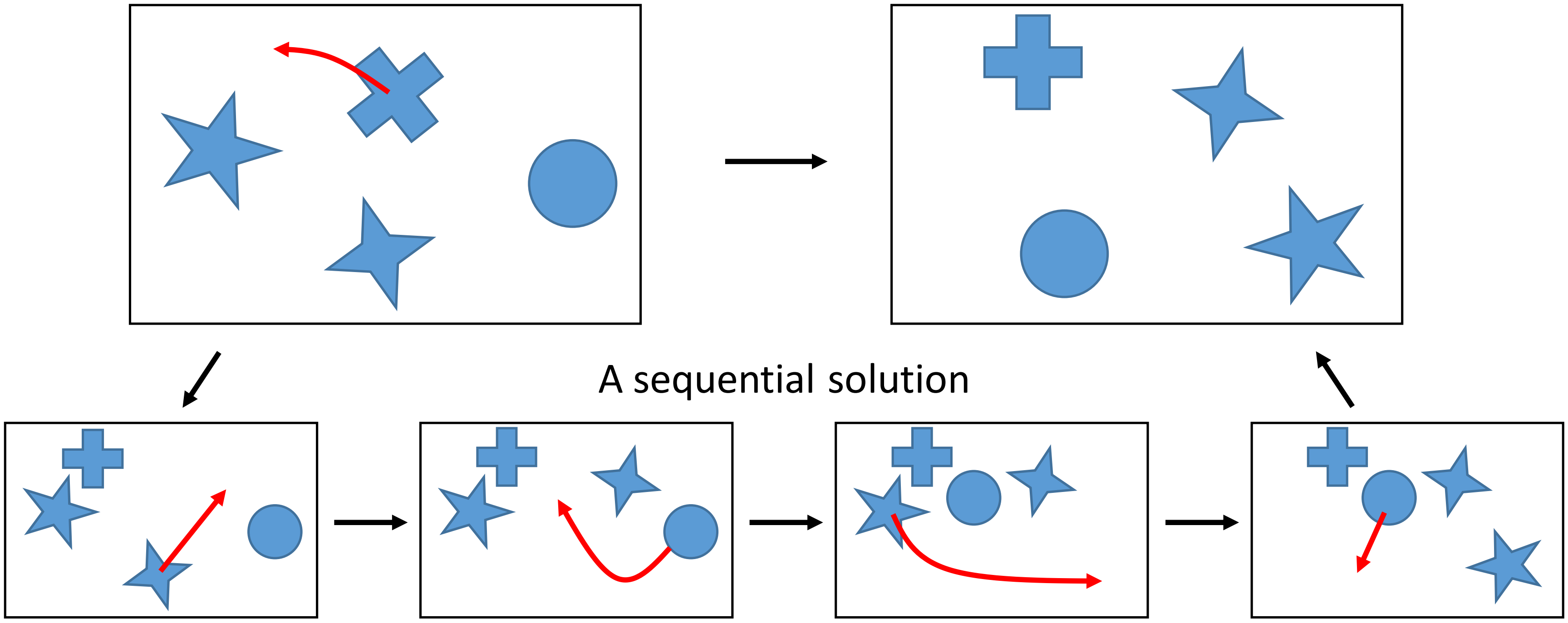
- ⇨Container ports
- ⇨Fulfillment centers
- ⇨Delivery drones
- ⇨Microfluidics (chemical/medical)
- ⇨Future autonomous vehicles
- ⇨Many others



Continuous Domain

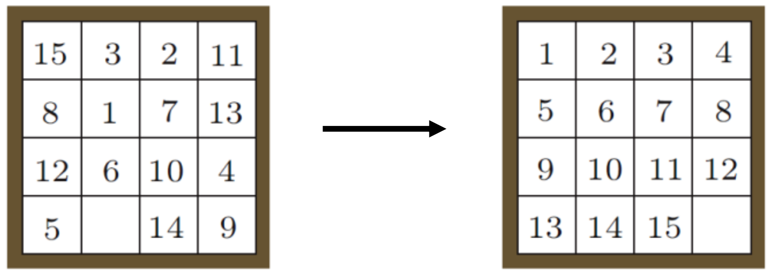
Continuous formulation

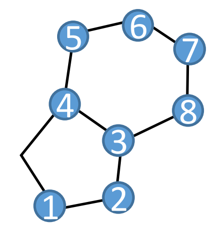
- ⇨Some n objects, usually close to each other (crowded)
- ⇨Usually in bounded region (why?)
- ⇨The objects must be moved from one configuration to another
- ⇨No collision is allowed
- ⇨This problem (finding any solution) is PSPACE-hard



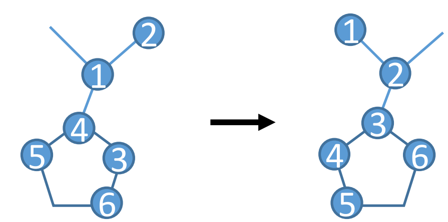
Discrete Domain: Single Move per Step

Origin: the 15-puzzle



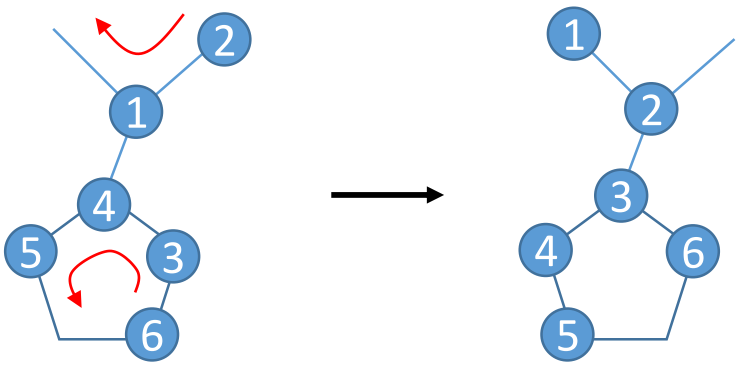


- ⇨ 4×4 game board, 15 square game pieces
- ⇨ Only pieces adjacent to the empty space can move
- ⇨ This is generalized first to $(N^2 - 1)$-puzzle
- ⇨ Then generalized to **2-connected graph** with n vertices and $n - 1$ robots
- ⇨ Then generalized to general graph with n vertices and up to $n - 1$ robots
- ⇨ In these problems only one robot may move at a time
- ⇨ Let's call these problems $\mathrm{MPP_s}$
- ⇨ Also known as the pebble motion problem (PMG)



Discrete Domain: Parallel Moves

For $\mathrm{MPP_s}$ on a n-vertex graph, moves may be parallelized



⇨Essentially, there are multiple "move sequences"

⇨Each sequence requires one empty vertex in the head

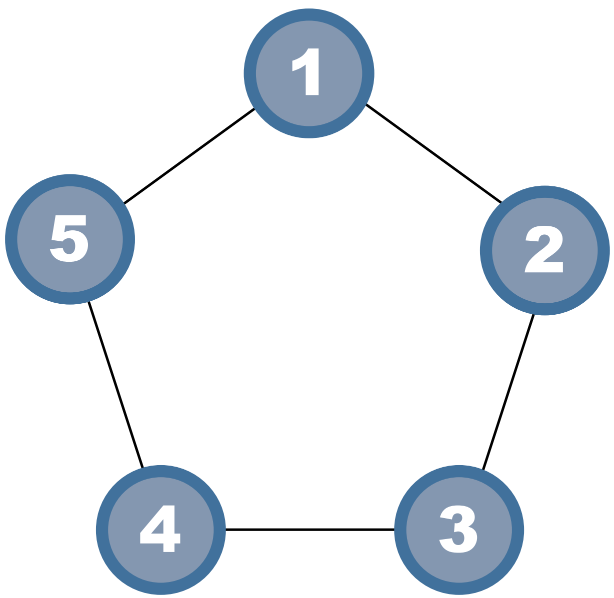
⇨It is possible that all robots move in the same step

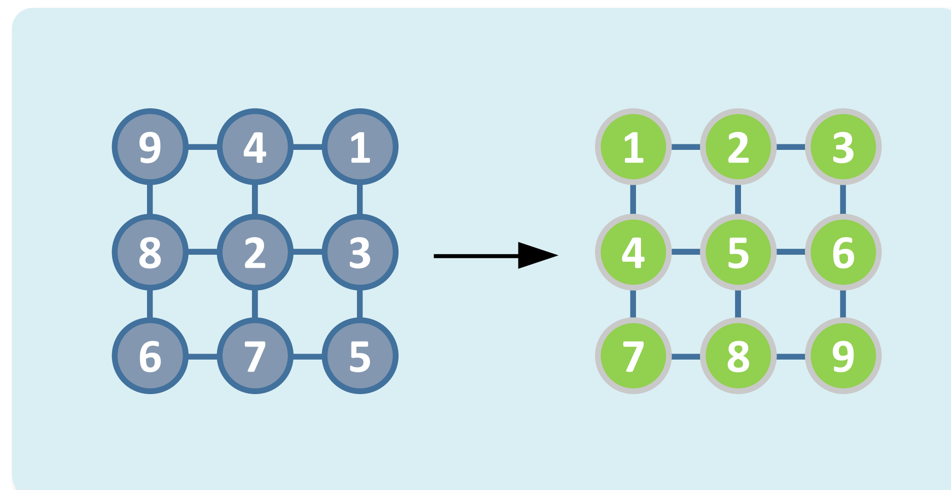
⇨Let's call this $\mathrm{MPP_p}$

Discrete Domain: Parallel Moves w/ Rotation

$\mathrm{MPP_p}$ still does not reflect full capabilities of modern robots

⇨Robots should not always require empty vertex to move, e.g.,



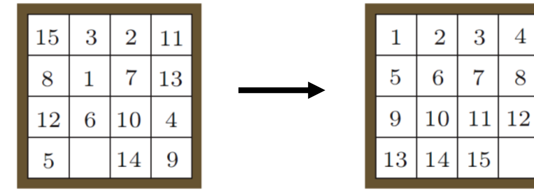


⇨We can further allow **rotations** in addition to parallel moves

⇨Let's call this $\mathrm{MPP_r}$

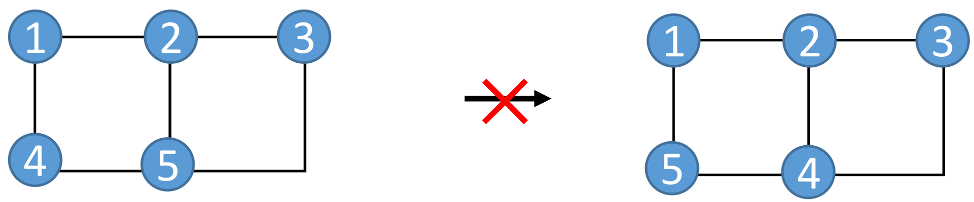
⇨Problems can be feasible even when there are no empty vertices!

Feasibility of the 15-Puzzle

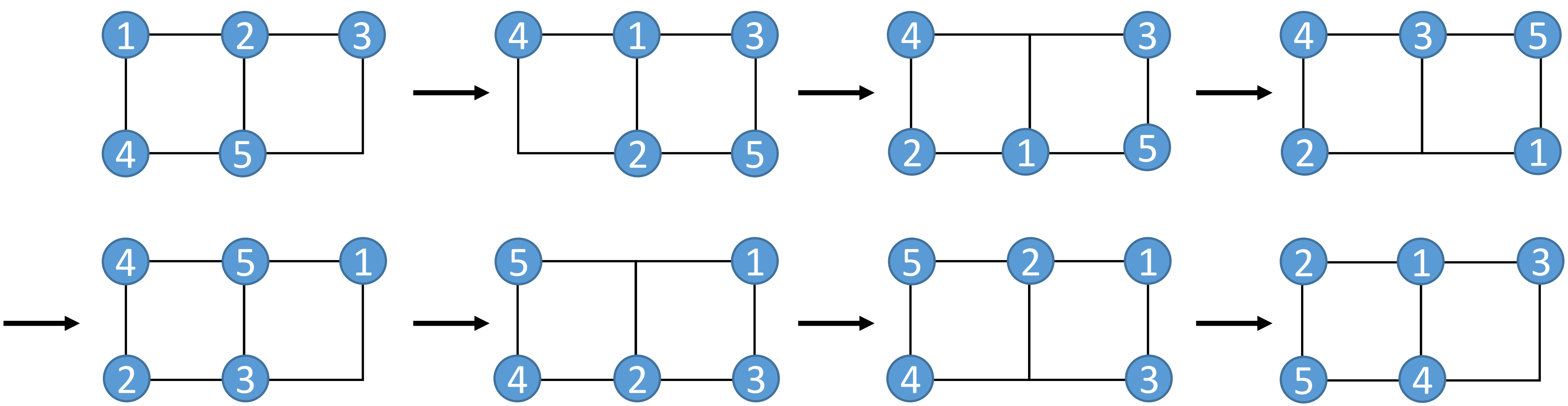


The 15-puzzle (and $(N^2 - 1)$-puzzle in general) is not always feasible

⇨In particular, if two robots are "swapped", it defines an infeasible problem

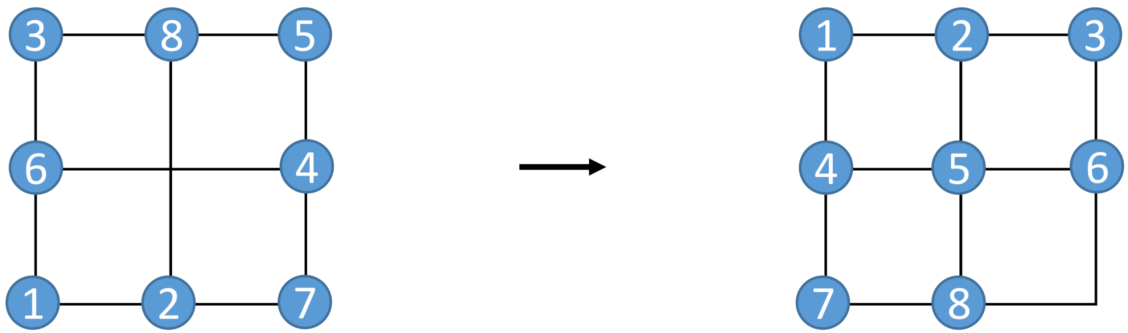


⇨What happens if we legally swap 4 and 5?



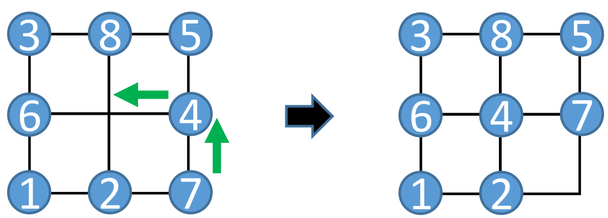
⇨It also (always) flip at least one other pair (in this case, 1 and 2)

A Simple Method for $N^2 - 1$ Puzzle Feasibility

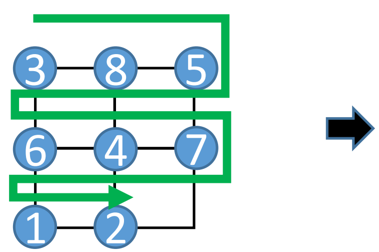


Steps

1. Move the empty spot to the lower right (doesn't matter how you do it)



2. Flatten the square row by row



38564712

3. Bubbling each number from 1 and count number of moves

38564712	Moved 1 over 385647, 6 numbers →	13856472
13856472	Moved 2 over 385647, 6 numbers →	12385647
12385647	Nothing to do for 3, 0 number →	12385647
12385647	Moved 4 over 856, 3 numbers →	12348567
12348567	Moved 5 over 8, 1 number →	12345867
12345867	Moved 6 over 8, 1 number →	12345687
12345687	Moved 7 over 8, 1 number →	12345678

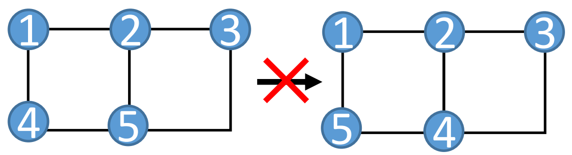
4. Sum up $X = 6 + 6 + 0 + 3 + 1 + 1 + 1 = 18$

5. Odd – infeasible. Even – feasible. $X = 18$, instance is feasible

Solving of the 15-Puzzle (I)

For $(N^2 - 1)$-puzzles

⇨The only source of infeasibility comes from this

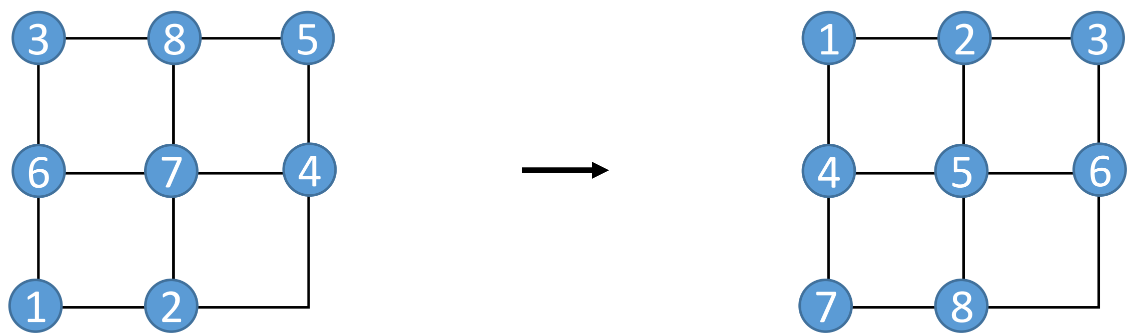


⇨How can we detect this?

⇨Through counting the number of moves for each robot

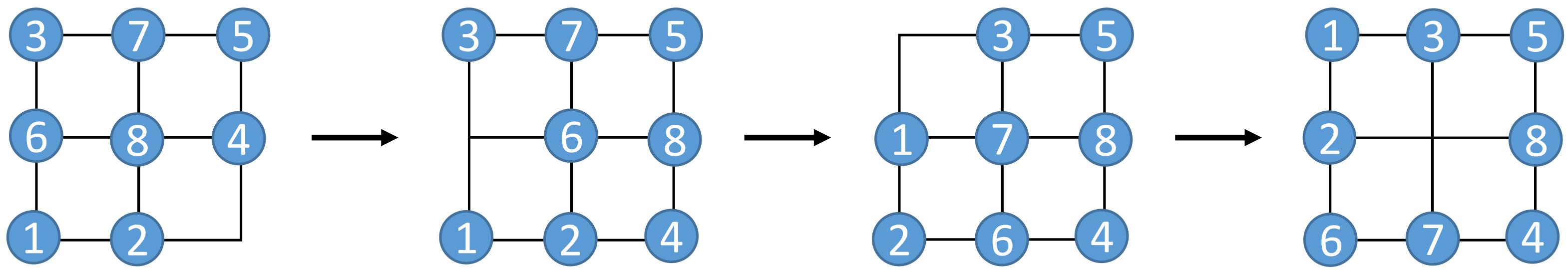
Now suppose we have a feasible problem, how do we solve it?

⇨We will use the 8-puzzle as an example

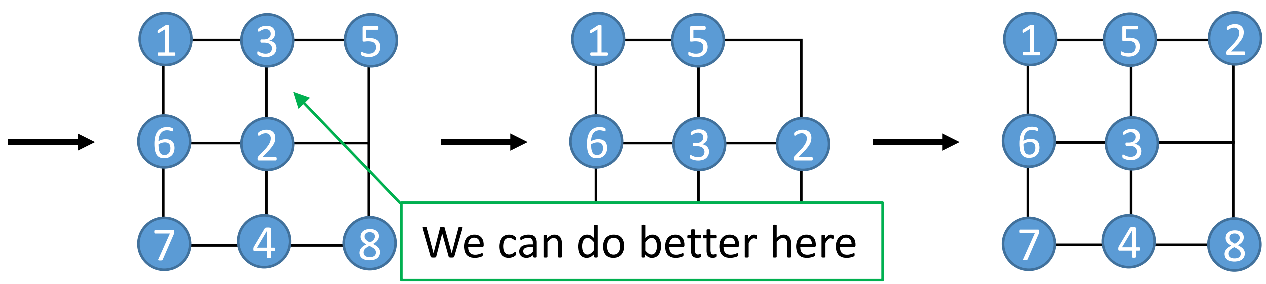


Solving the 15-Puzzle (II)

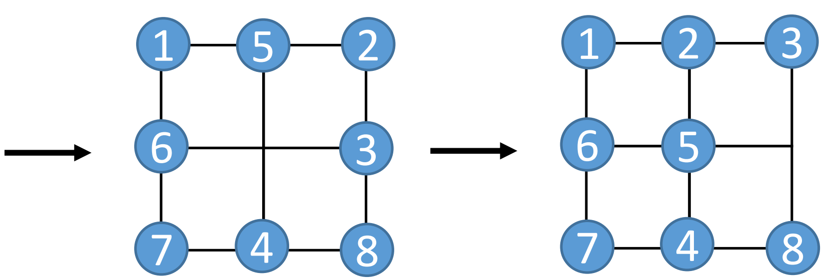
First, move 1 to its goal



Then, move 2 to 3's goal



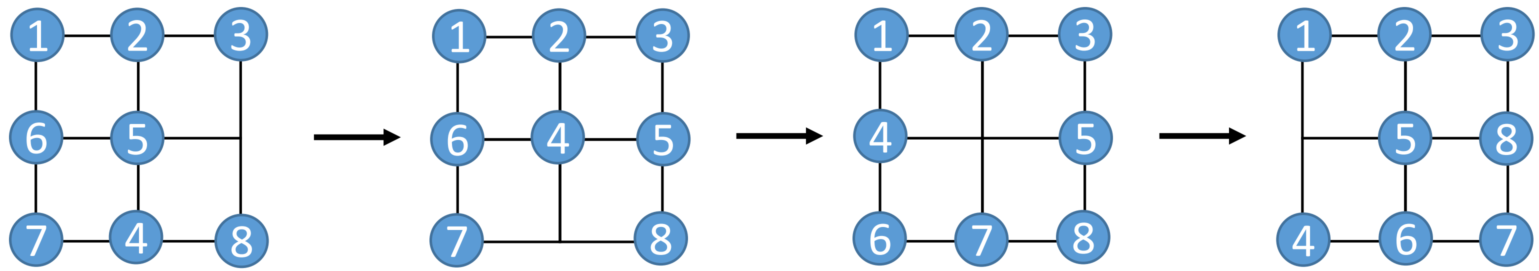
Then, move 3 to 6's goal, followed by moving 2,3 to their goals



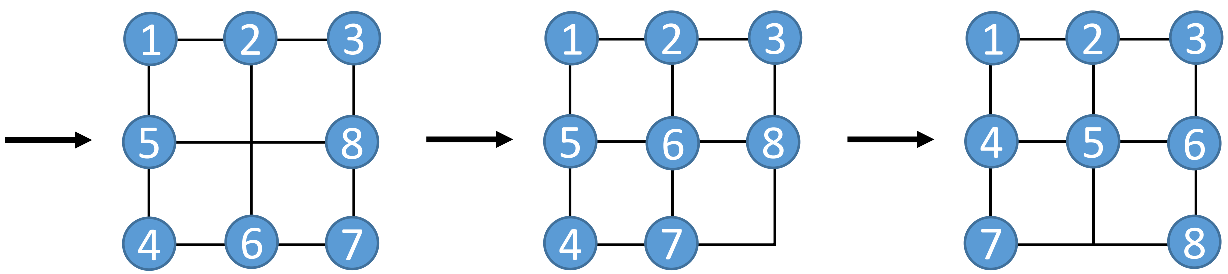
First row is solved!

Solving the 15-Puzzle (III)

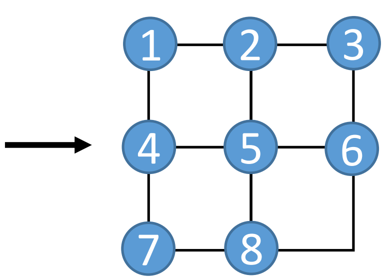
Then, solve the first column similarly. In this case, move 4 to 7’s goal



Then, move 7 to 8’s goal, and solve 4 and 7



Then, the last piece should be readily solvable



Solving the 15-Puzzle (IV)

Procedure for solving the $(N^2 - 1)$-puzzle

- ⇨Check feasibility
- ⇨Solve the first (top) row
- ⇨Solve the first (left most) column, now we have a $((N-1)^2 - 1)$-puzzle
- ⇨Repeat the above steps until we get a 3-puzzle
- ⇨The 3-puzzle should be readily solvable

The same procedure applies to $N \times M$-puzzles where $N \neq M$

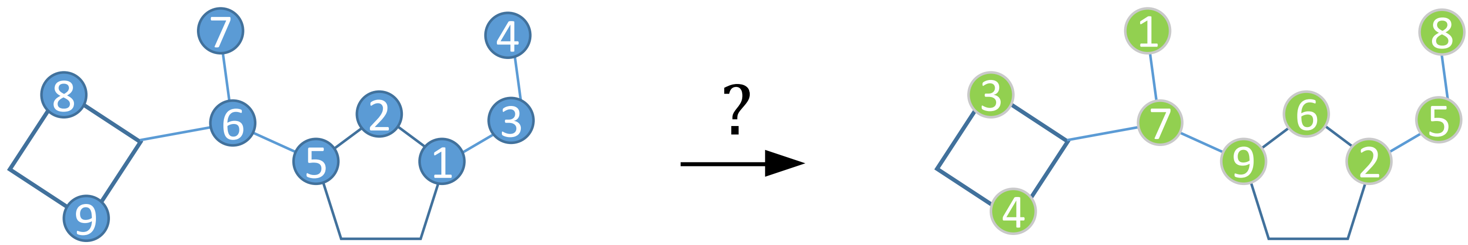
How many moves are needed?

- ⇨For each robot, may need to move $\mathrm{O}(N)$ robots
- ⇨Each robot needs to be moved $\mathrm{O}(1)$ times
- ⇨So $O(N)$ total moves
- ⇨For all $N^2 - 1$ robots, $O(N^3)$ total moves

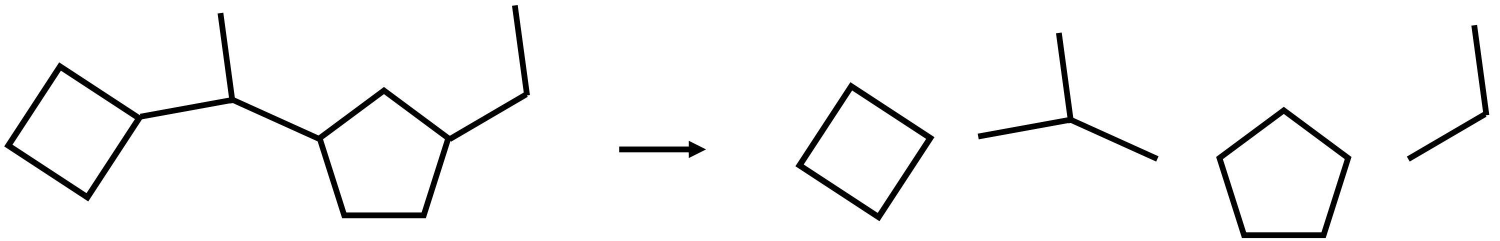
The method generalizes to 2-connected graphs

General Graphs

How about general graphs?



⇨We can break it into trees and 2-connected graphs



⇨We already can solve 2-connected graphs

⇨How about trees?

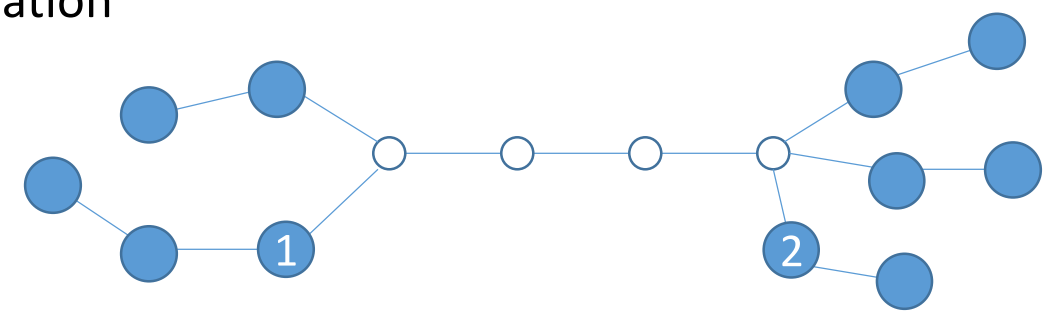
Partitioning Robots on Trees

We can solve the problem by partitioning robots into **equivalence classes**

⇨ Two robots belong to the same equivalence classes if they can exchange locations

⇨ It turns out this is easy to determine (well, intuitively)

⇨ Two robots 1 and 2 that are "adjacent" (i.e., no other robots between them on the tree) cannot exchange (i.e., not equivalent) if they have the following configuration



⇨ If there is one more empty vertex, then 1 and 2 are equivalent

⇨ This can be checked (in linear time)

⇨ Combined with 2-connected graph solver, solves $\mathrm{MPP_S}$ on general graph

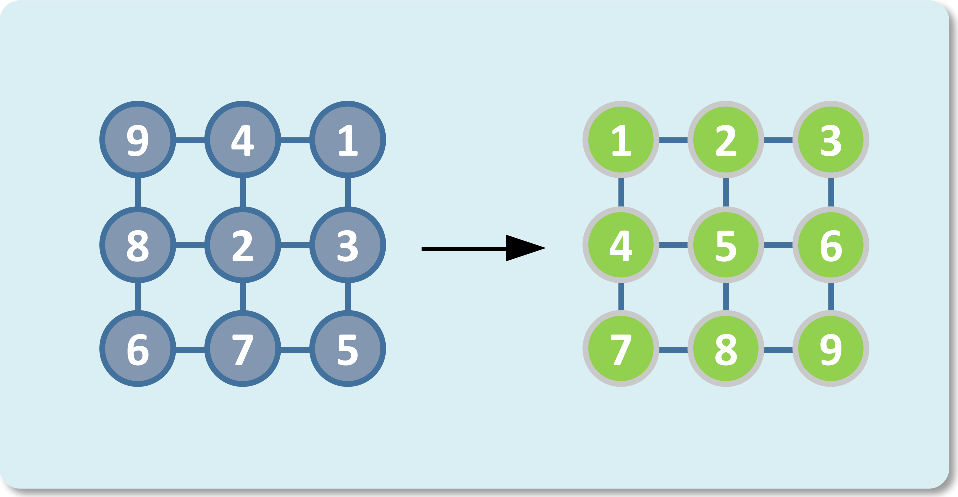
Feasibility with Parallel Moves

$\mathrm{MPP_p}$ feasibility is the same as $\mathrm{MPP_s}$

- ⇨Why?
- ⇨Every parallel move can be carried out as step-by-step single robot moves

$\mathrm{MPP_r}$, it's slightly different

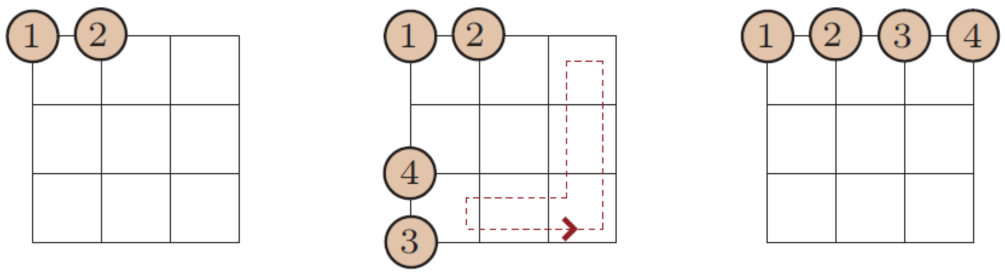
- ⇨We can decompose the problem into trees and 2-edge-connected components
- ⇨But, we need some methods for solving problems like this (N^2-puzzle)



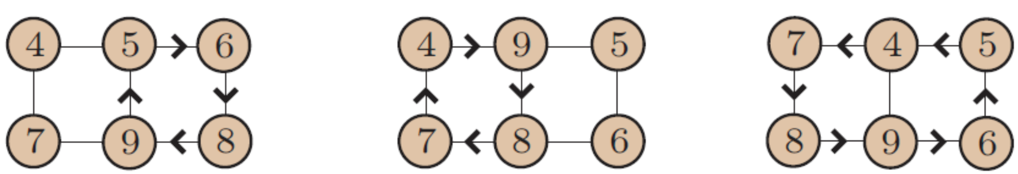
Solving N^2 puzzles

N^2-puzzle ($\mathrm{MPP_r}$) is similar to $(N^2 - 1)$-puzzle ($\mathrm{MPP_s}$)

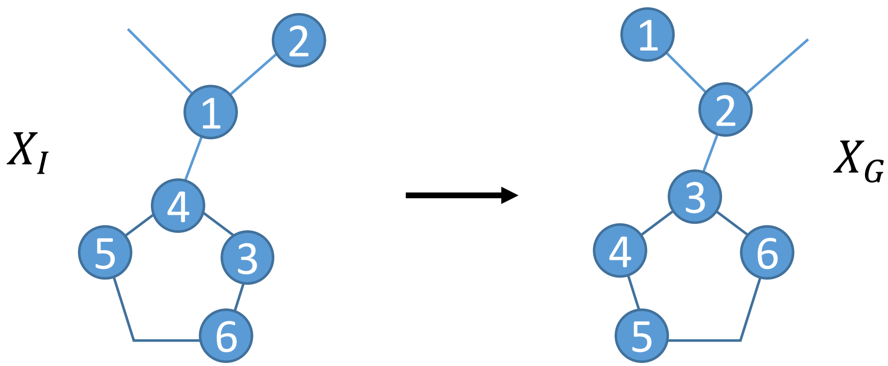
- ⇨ We can first solve the top row (and left most column)



- ⇨ This yields an $(N-1)^2$-puzzle
- ⇨ We do this until we get to 3×3, and solve the top row
- ⇨ This leaves us with a 2×3 puzzle, which can be solved
  - ⇨ E.g., exchange 8 and 9



Optimal Formulations



MPP Problem: (G, X_I, X_G), solution: collision free $P = \{p_1, \ldots, p_n\}$

Optimality objectives (minimization):

⇨ Max time (makespan): $\min\limits_{P \in \mathcal{P}} \max\limits_{p_i \in P} time(p_i)$

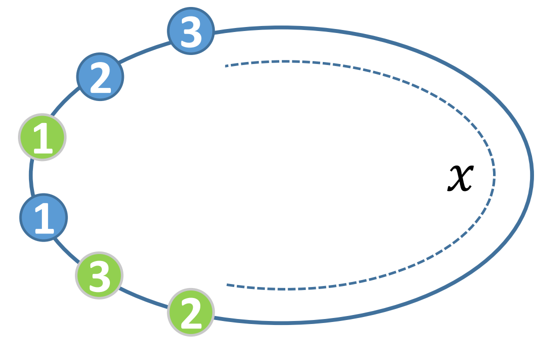
⇨ Total time: $\min\limits_{P \in \mathcal{P}} \sum_{p_i \in P} time(p_i)$

⇨ Max distance: $\min\limits_{P \in \mathcal{P}} \max\limits_{p_i \in P} length(p_i)$

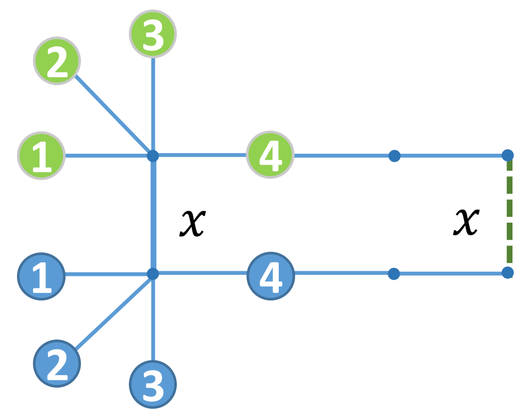
⇨ Total distance: $\min\limits_{P \in \mathcal{P}} \sum_{p_i \in P} length(p_i)$

Incompatibility of the Formulations

	Makespan	**Total time**
Clockwise	$x + 1$	$2x + 3$
Counterclockwise	$x + 4$	$x + 12$



	Total distance	**Total time**
Left path only	$4x + 8$	$4x + 14$
Using right path	$4x + 10$	$4x + 13$



A pair of the four MPP objectives on makespan, total time, max distance, and total distance demonstrates a Pareto-optimal structure.