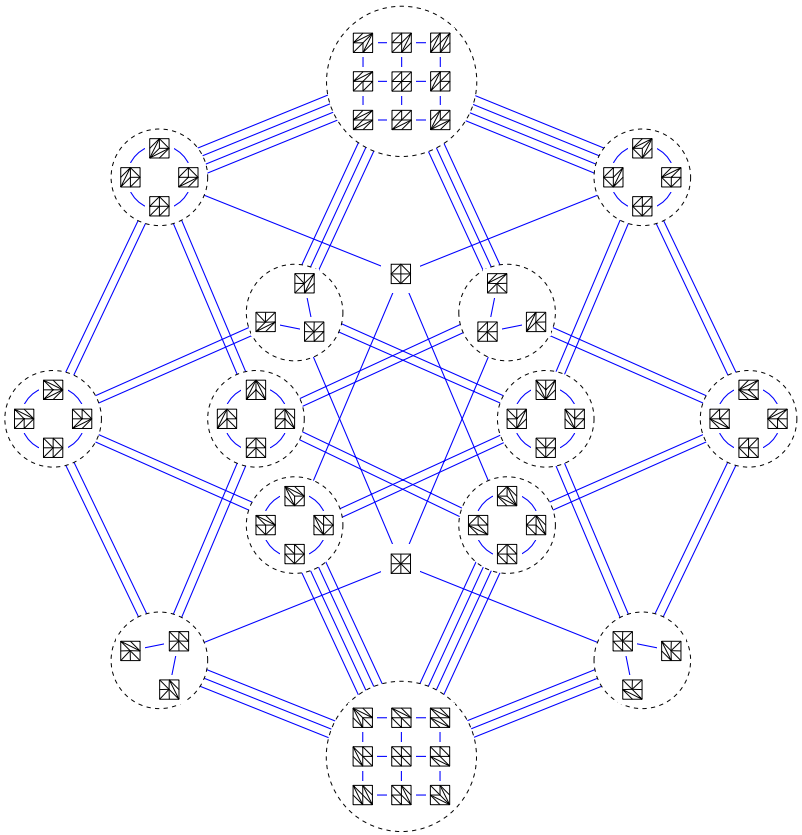
Reconfiguring Undirected Paths

Erik D. Demaine, **David Eppstein**, Adam Hesterberg, Khistij Jain, Anna Lubiw, Ryuhei Uehara, and Yushi Uno

Algorithms and Data Structures Symposium (WADS)
August 2019

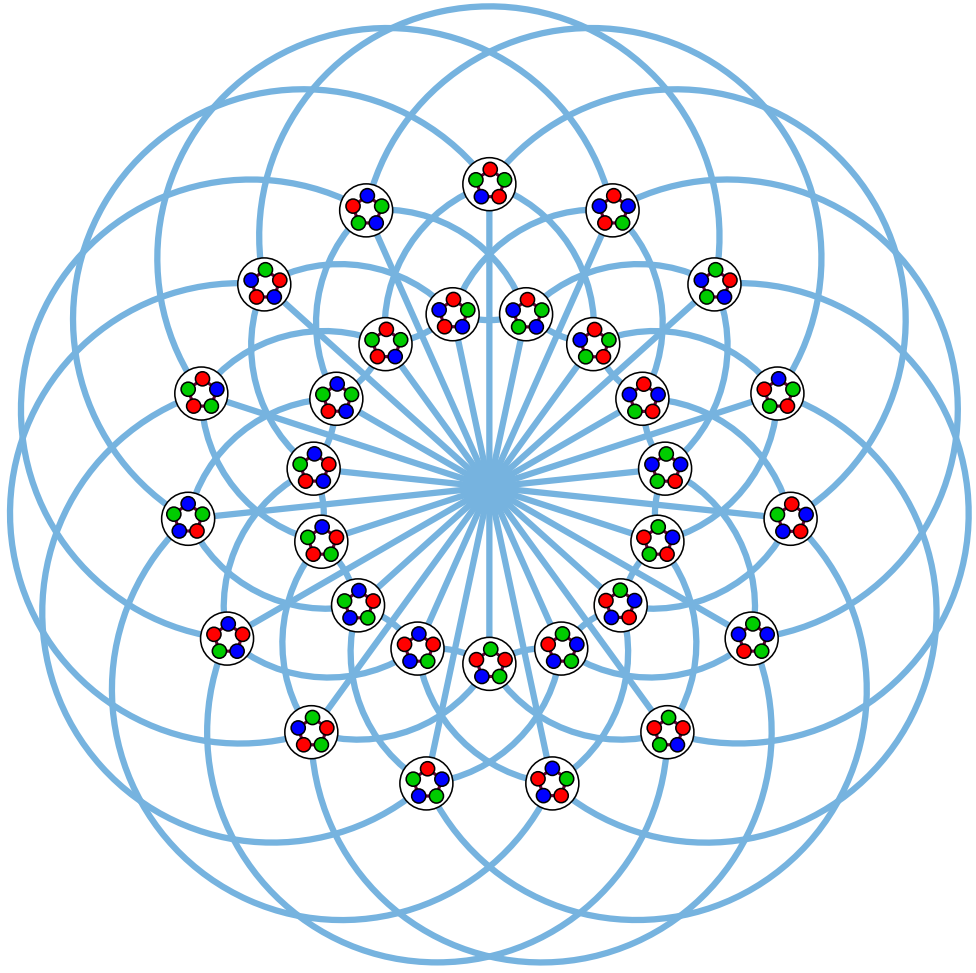
Reconfiguration problems



Construct the state space of solutions of a combinatorial problem, under small changes
(Illustrated: flips in triangulations)

Study the connectivity of the resulting graph
(This paper: existence of a path between given solutions)

Related to MCMC methods for random generation



Wang–Swendsen–Kotecký dynamics on 3-colorings of a 5-cycle

Random walks rapidly mix $\Rightarrow$ random colorings

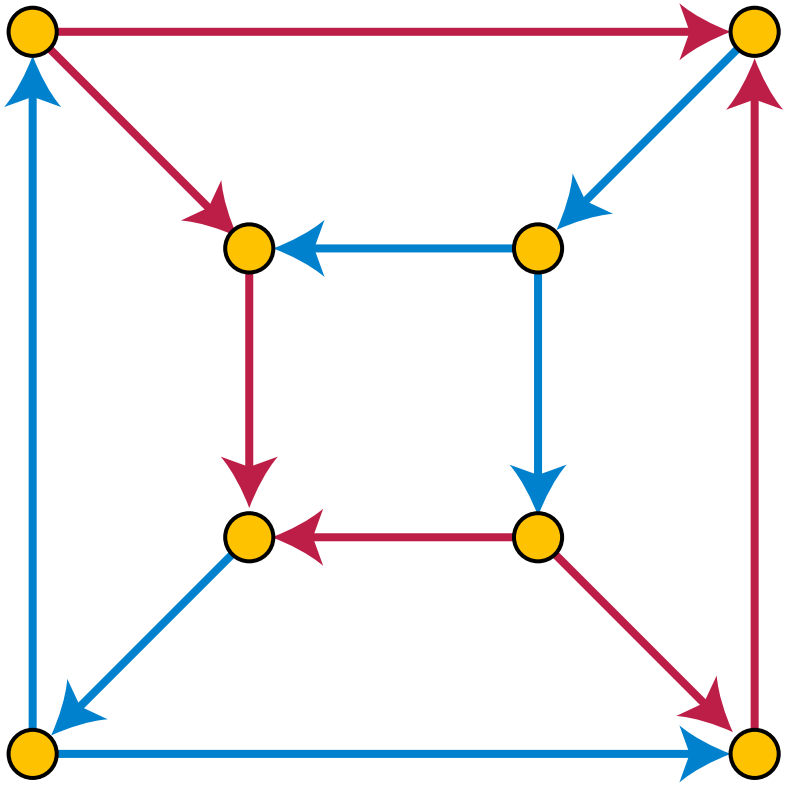
Another application: Puzzle complexity

Many puzzles (here, Rush Hour) involve making sequences of small state changes to reach a goal state

Reconfiguration = solving the puzzle

Image: Welt-der-Form [2013]

Typical complexity: PSPACE-complete



Often proved by reduction from *nondeterministic constraint logic*

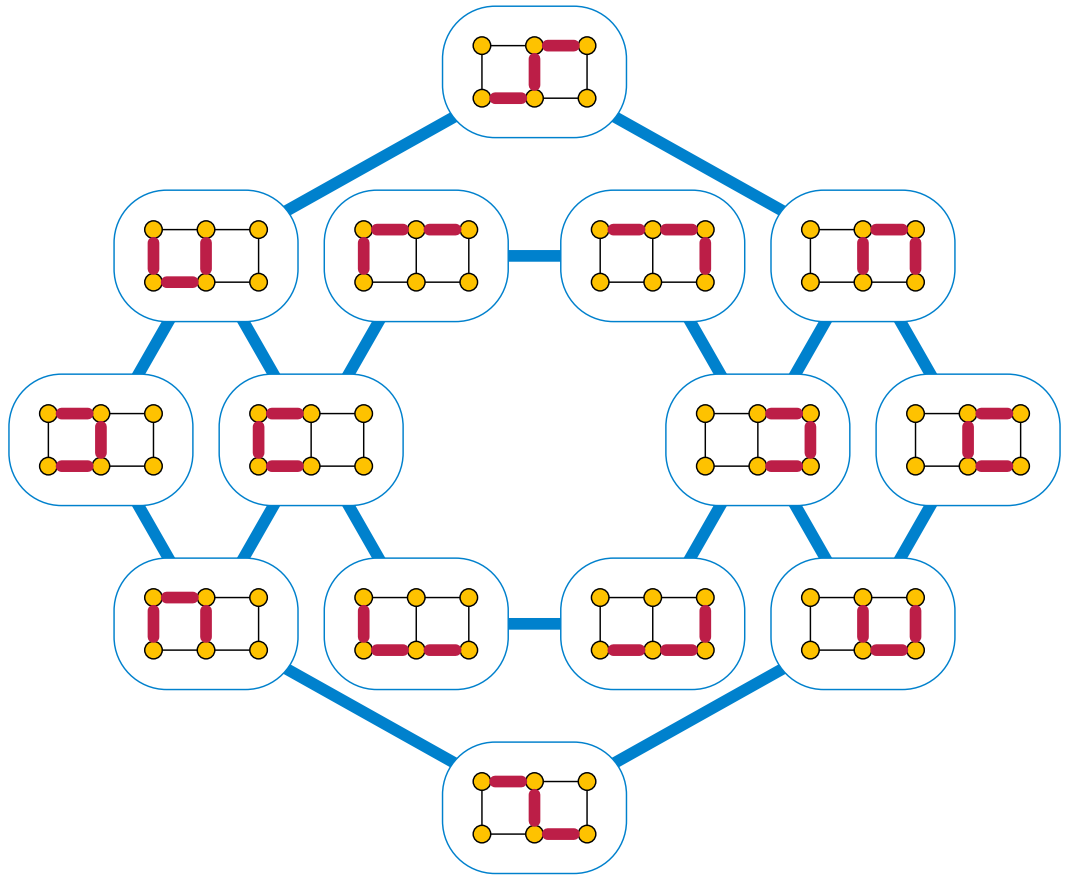
Directed graph with edges of weight 1 (red) or 2 (blue)

Each vertex must have incoming weight ≥ 2 (two red or one blue)

Change by reversing edges

[Hearn and Demaine 2002]

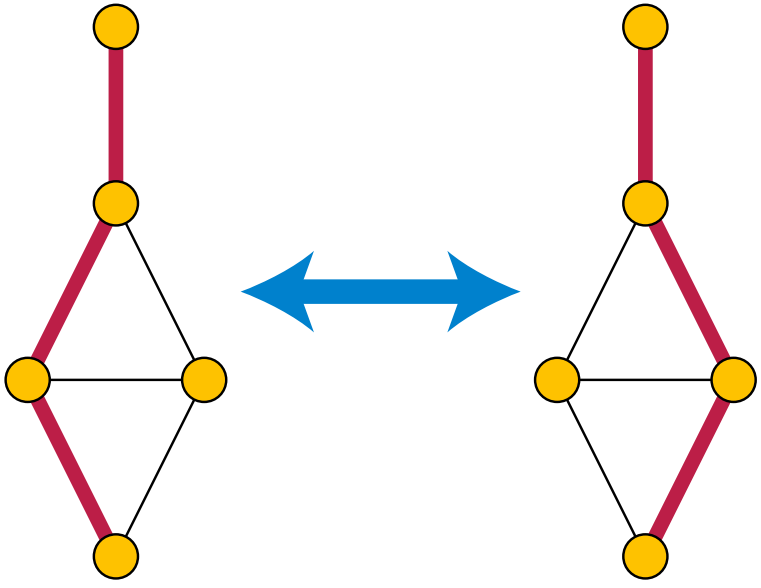
Our problem: Path reconfiguration



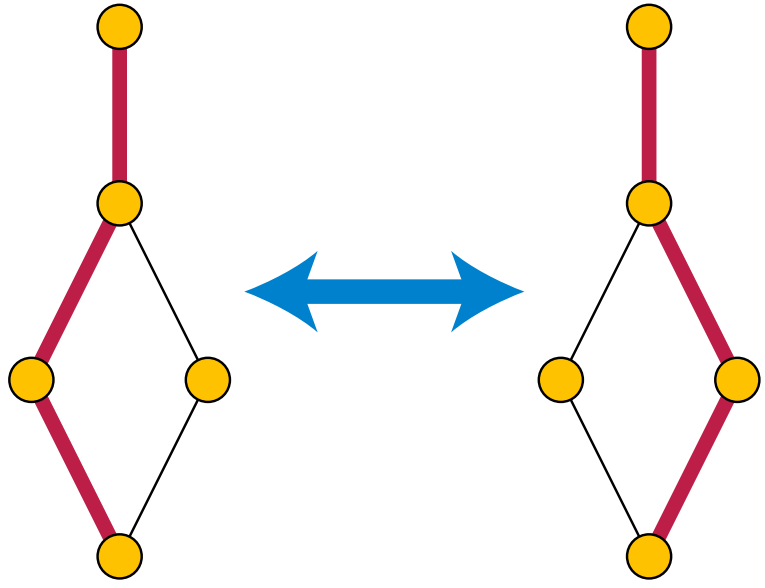
States = paths of same length in an undirected graph

Moves = slide path one step in either direction
(Equivalently, add edge at one end and remove at other end)

Earlier path reconfiguration uses less natural moves



Token-sliding: replace path vertex by neighbor



Token-jumping: replace path vertex by arbitrary vertex

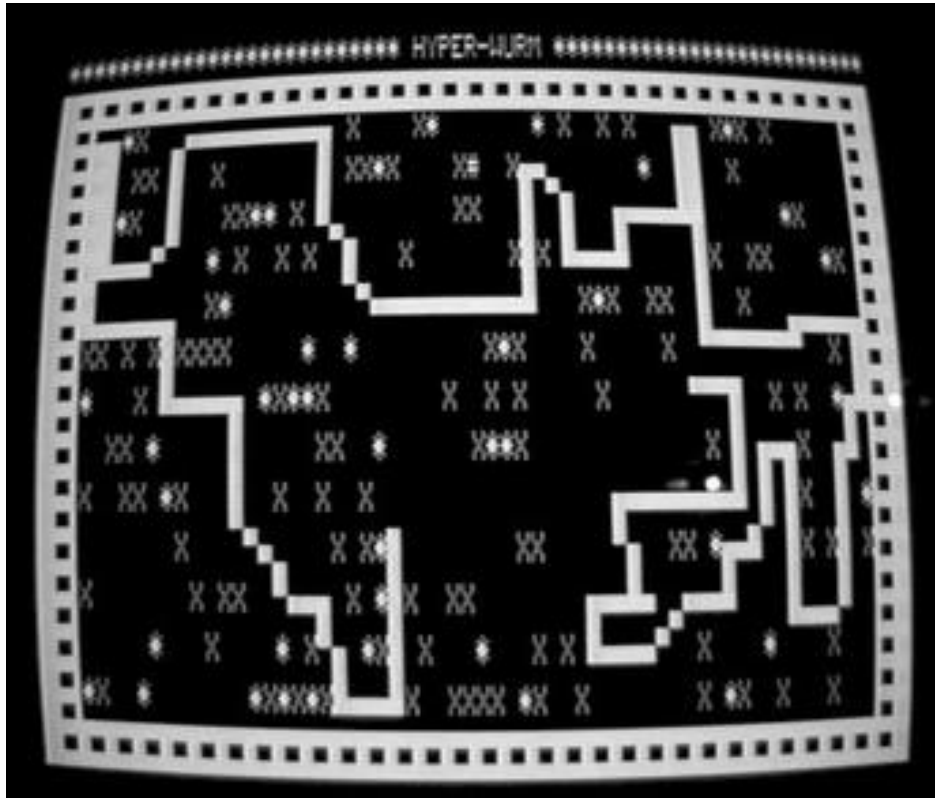
[Kamiński et al. 2011; Bonsma 2013; Hanaka et al. 2018]

Intuition: Model trains

Can each of the three trains move to the positions of the others?

(Yes, but it may have to move backwards)

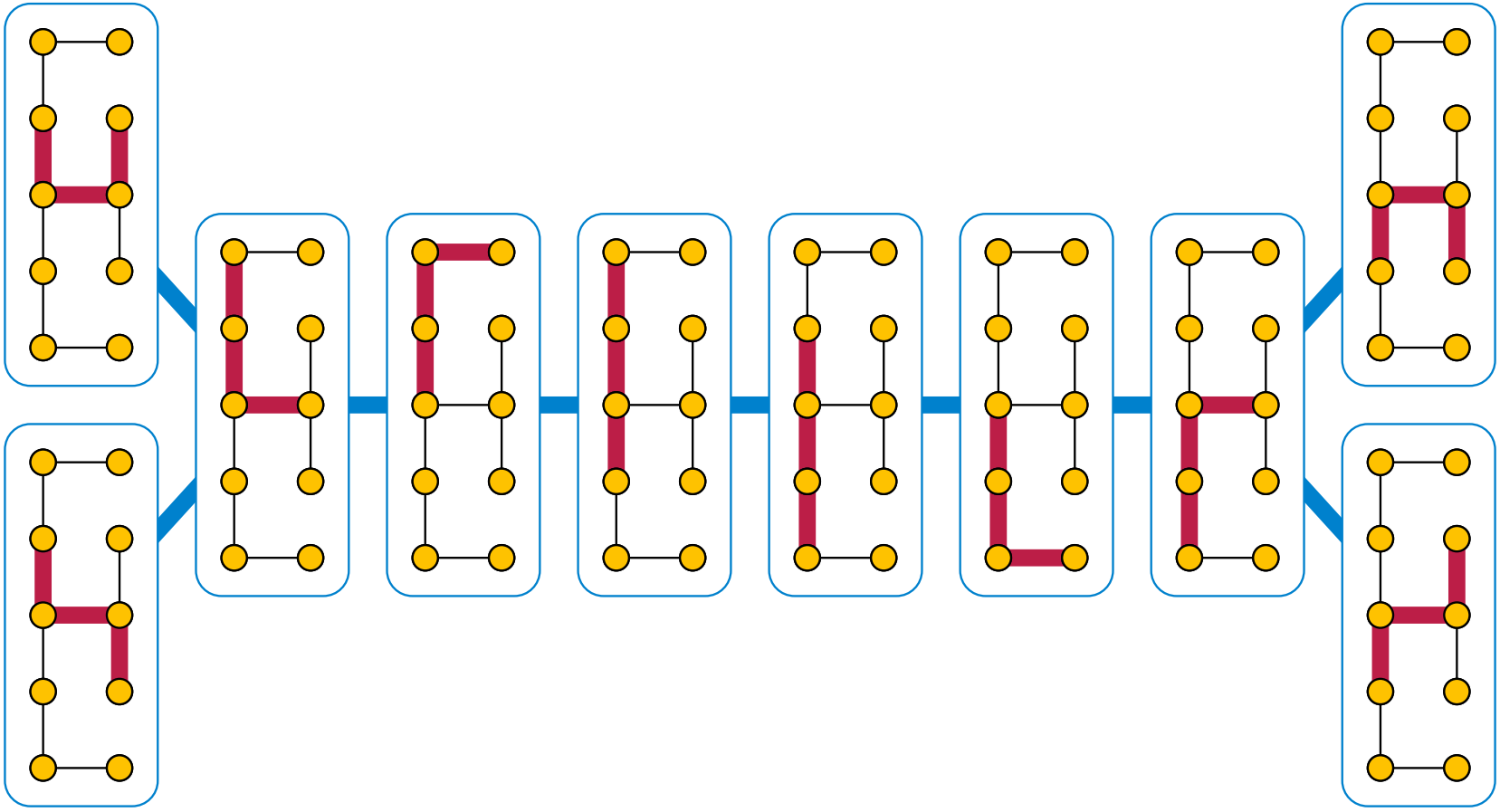
Also related: Snake video game



But in Snake, the motion is only one way
and paths grow rather than staying the same length

Some graphs have small state spaces

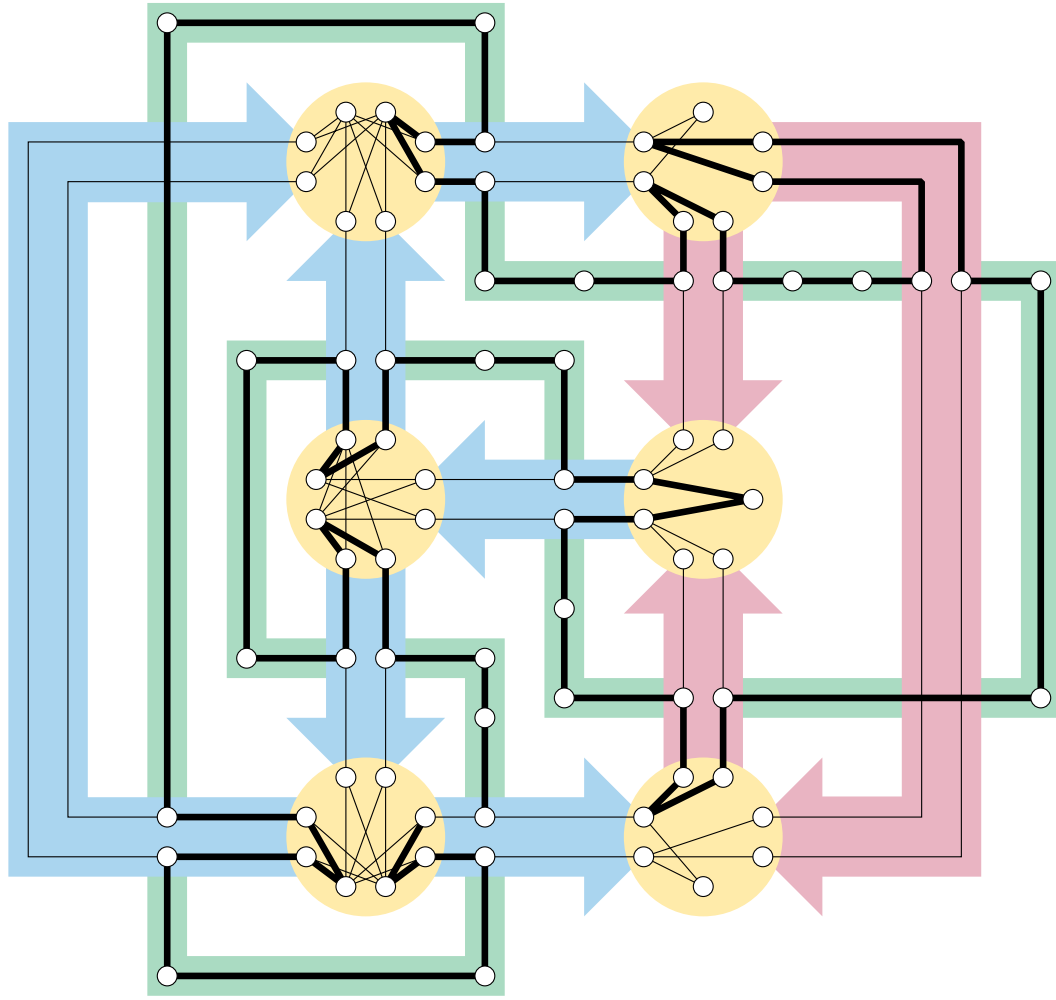
In trees, paths are determined by endpoints $\Rightarrow O(n^2)$ states



With some care, can reconfigure paths in trees in $O(n)$ time

(or find shortest reconfiguration sequences)

The general problem has the usual complexity



PSPACE-complete (even for bounded bandwidth) from NCL

The reduction involves long paths, so what about short ones?

Our main result

Path reconfiguration is fixed-parameter tractable in path length

That is, in n-vertex graphs with path length k, we get time

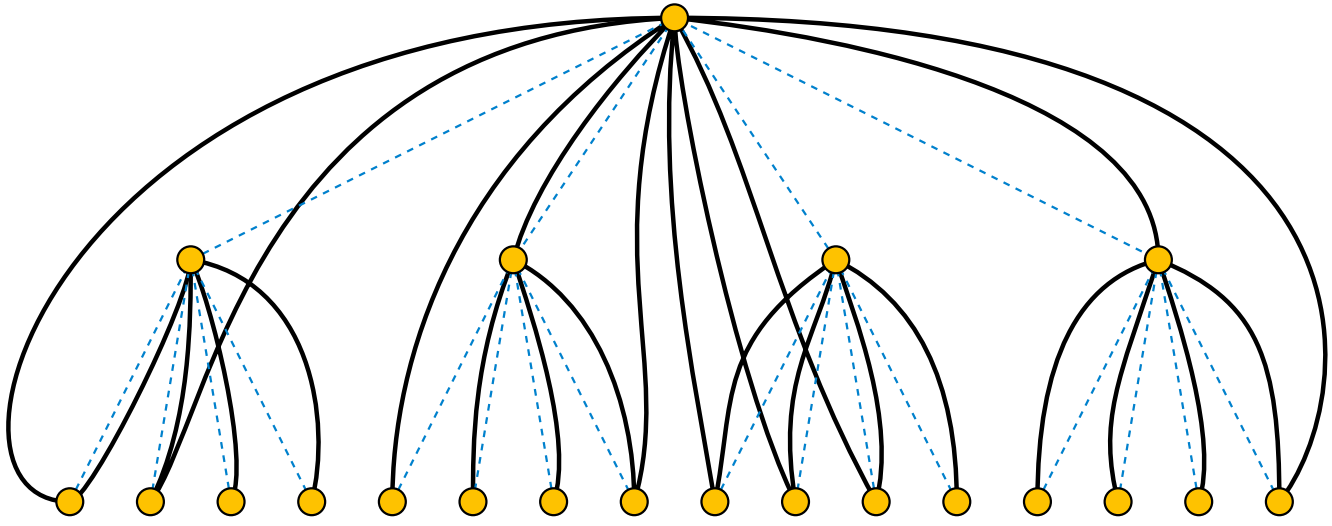
$$O(f(k) \cdot n^c)$$

where c is a constant
and f is a computable (but large) function

(Does not depend on the structure of the graph!)

First idea: Treedepth

Depth(G) = min height of a tree for which all edges in G connect ancestor-descendant pairs (tree edges might not belong to G)



Low depth $\Rightarrow$ many repeated subgraphs

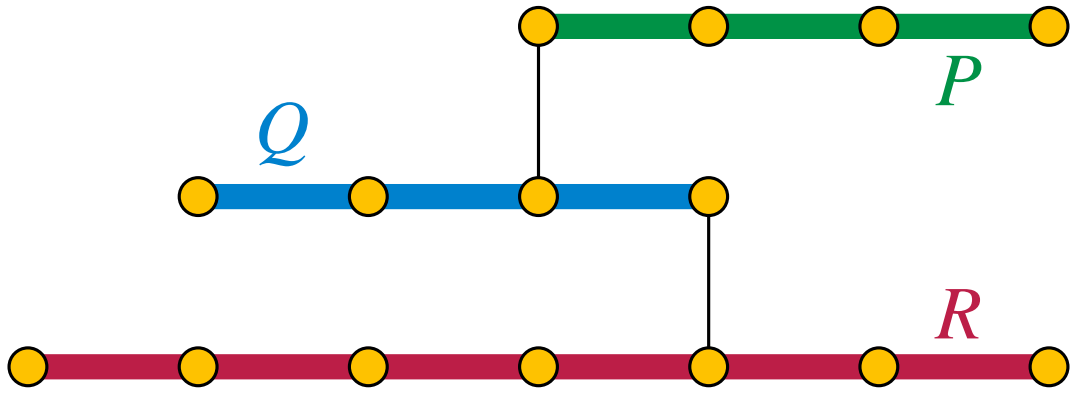
Never need more than $\approx k/2$ copies of each distinct subgraph

Traverse tree bottom-up keeping only $O_k(1)$ children at each level

Result: an equivalent instance with total size $O_k(1)$

Second idea: Long paths

Loose path = length $2k$, disjoint from both start and goal paths



Any two states contained in loose paths can reach each other:

1. Slide along first loose path to free up exit vertex
2. Slide along a path between the two loose paths
3. Slide into correct position on second loose path

So if we can get from start and goal to loose paths, we win!

Win-win

Either the treewidth or the loose path method works!

From start state, add edges one at a time to reachable subgraph

- Until finding a loose path, reachable treedepth must be low
- Use low treedepth to check loose path existence efficiently
- Use low treedepth to test ability to move onto nearby edges
- Stop adding once we find a loose path

Do same from goal state

Either both searches reach loose paths, or completed reachable set has low treedepth

Conclusions

Path reconfiguration is:

Hard for long paths in arbitrary graphs

Linear-time for trees, and easy for tree-like graphs
(FPT for graphs of low circuit rank,
and XP for graphs with small feedback vertex sets)

Easy (FPT) for arbitrary graphs and bounded-length paths

Related: Also FPT for bounded-length paths that can only move unidirectionally [Gupta et al. 2019]

Open: How hard is it to find the shortest reconfiguration sequence for instances with bounded-length paths?

References and image credits, I

Paul Bonsma. The complexity of rerouting shortest paths. *Theoretical Computer Science*, 510:1–12, 2013. doi: 10.1016/j.tcs.2013.09.012.

Siddharth Gupta, Guy Sa'ar, and Meirav Zehavi. The parameterized complexity of motion planning for snake-like robots. Electronic preprint arxiv:1903.02445, March 2019.

Tesshu Hanaka, Takehiro Ito, Haruka Mizuta, Benjamin Moore, Naomi Nishimura, Vijay Subramanya, Akira Suzuki, and Krishna Vaidyanathan. Reconfiguring spanning and induced subgraphs. In Lusheng Wang and Daming Zhu, editors, *Computing and Combinatorics: 24th International Conference, COCOON 2018, Qing Dao, China, July 2–4, 2018, Proceedings*, volume 10976 of *Lecture Notes in Computer Science*, pages 428–440. Springer, 2018. doi: 10.1007/978-3-319-94776-1_36.

References and image credits, II

Robert A. Hearn and Erik D. Demaine. The nondeterministic constraint logic model of computation: reductions and applications. In *Automata, Languages and Programming: 29th International Colloquium, ICALP 2002, Málaga, Spain, July 8–13, 2002 Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2002. doi: 10.1007/3-540-45465-9_35.

Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths. *Theoretical Computer Science*, 412(39): 5205–5210, 2011. doi: 10.1016/j.tcs.2011.05.021.

Welt-der-Form. Rush Hour, a sliding block puzzle, manufactured by ThinkFun. CC-BY-SA licensed image, 2013. URL `https://commons.wikimedia.org/wiki/File:Rush_Hour_sliding_block_puzzle.jpg`.