Generalizing JPS Symmetry Detection: Canonical Orderings on Graphs

Nathan R. Sturtevant
University of Denver

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Background

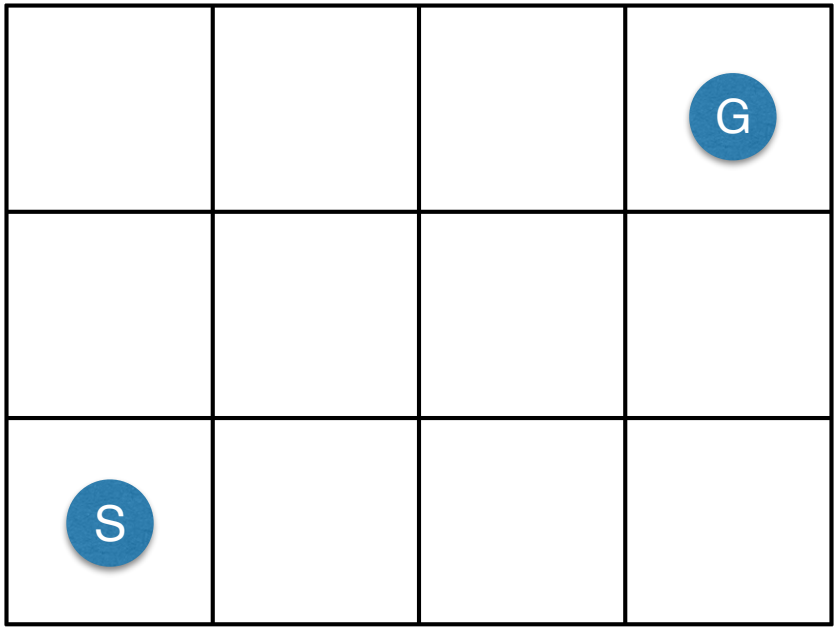
- Jump Point Search has been a popular technique for improving grid-based search
- It is one of the few techniques that requires no pre-processing
  - Pre-processing can improve speed

IJCAI 2016

- Canonical Orderings on Grids
- Reformulations JPS as:
  - Canonical ordering of states
  - Jumping policy
  - Best-first search

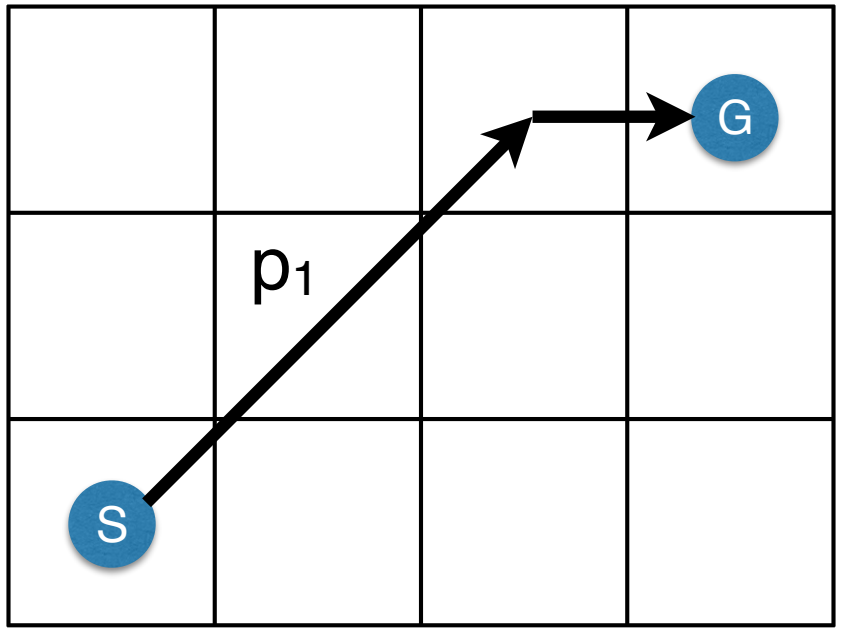
Canonical ordering of paths

- Order all **optimal** paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to p_2



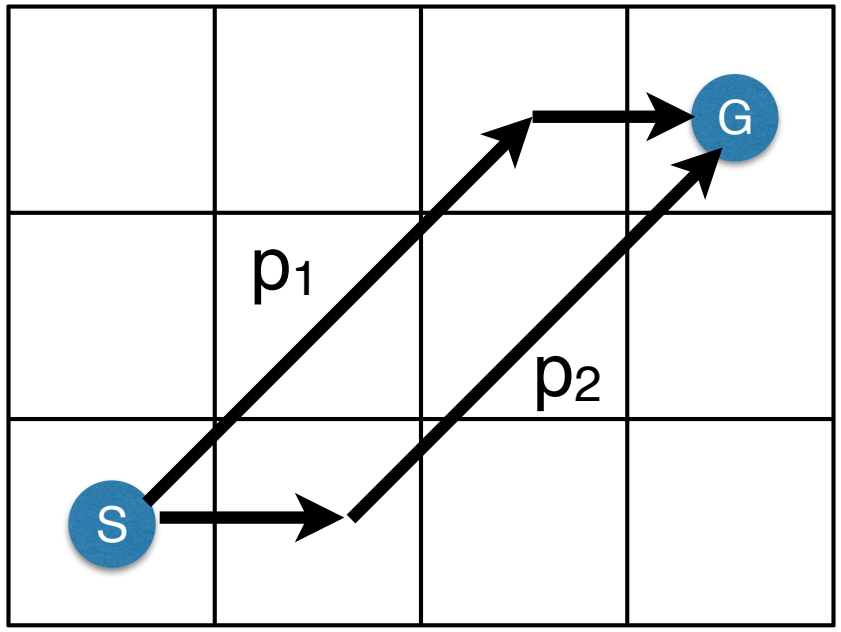
Canonical ordering of paths

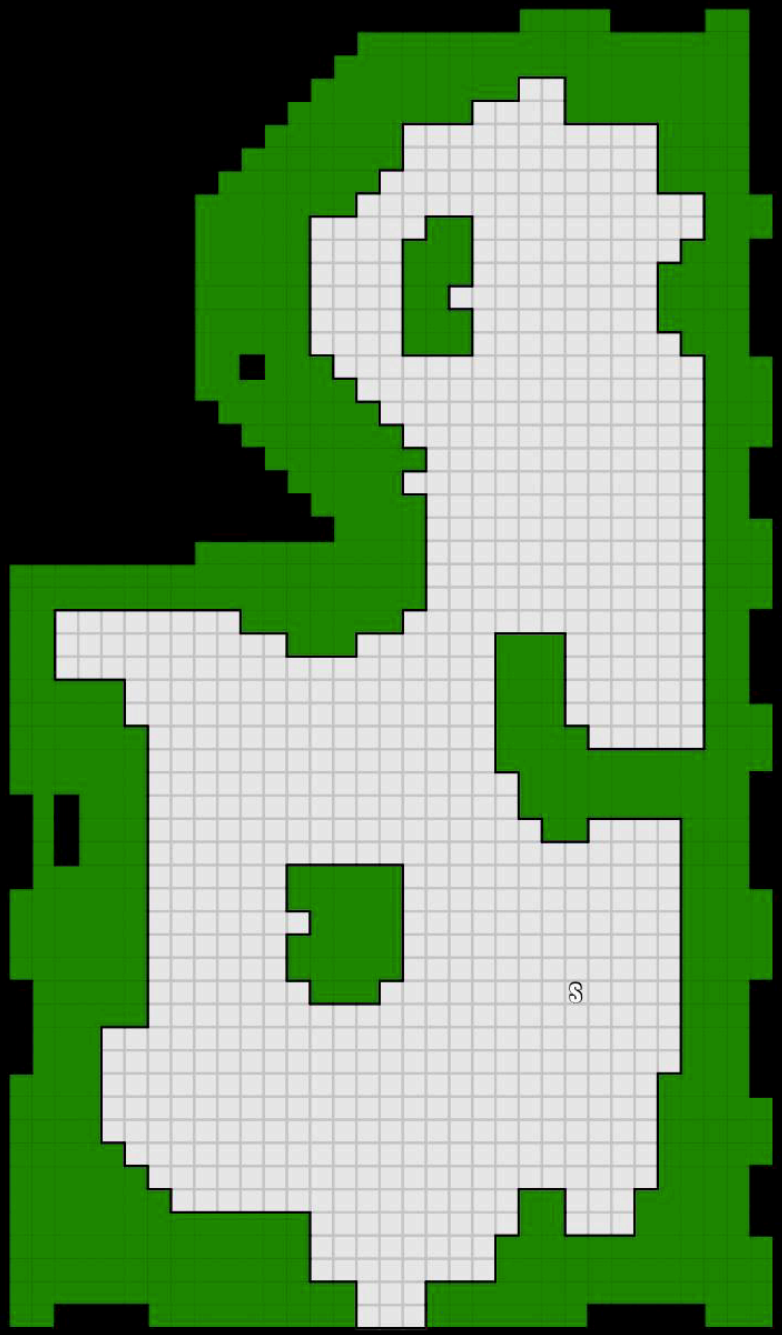
- Order all **optimal** paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to p_2

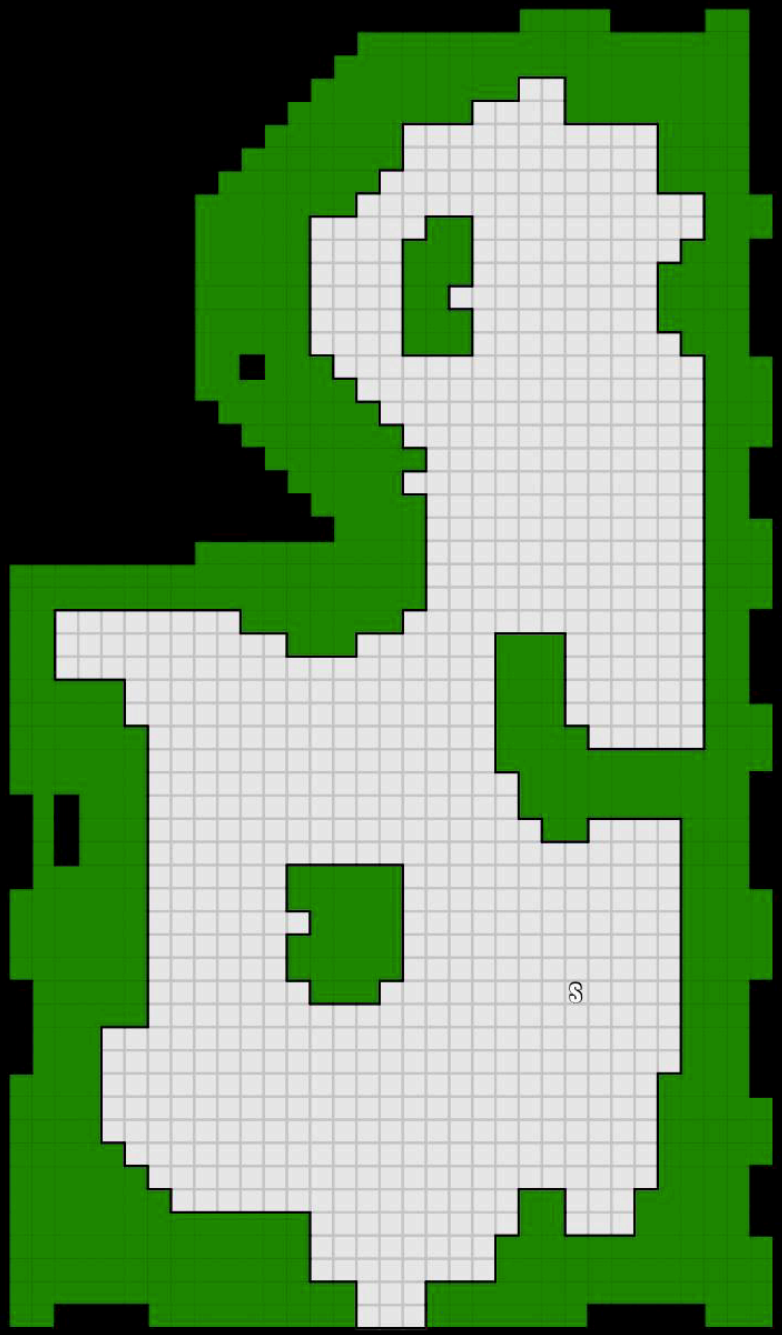


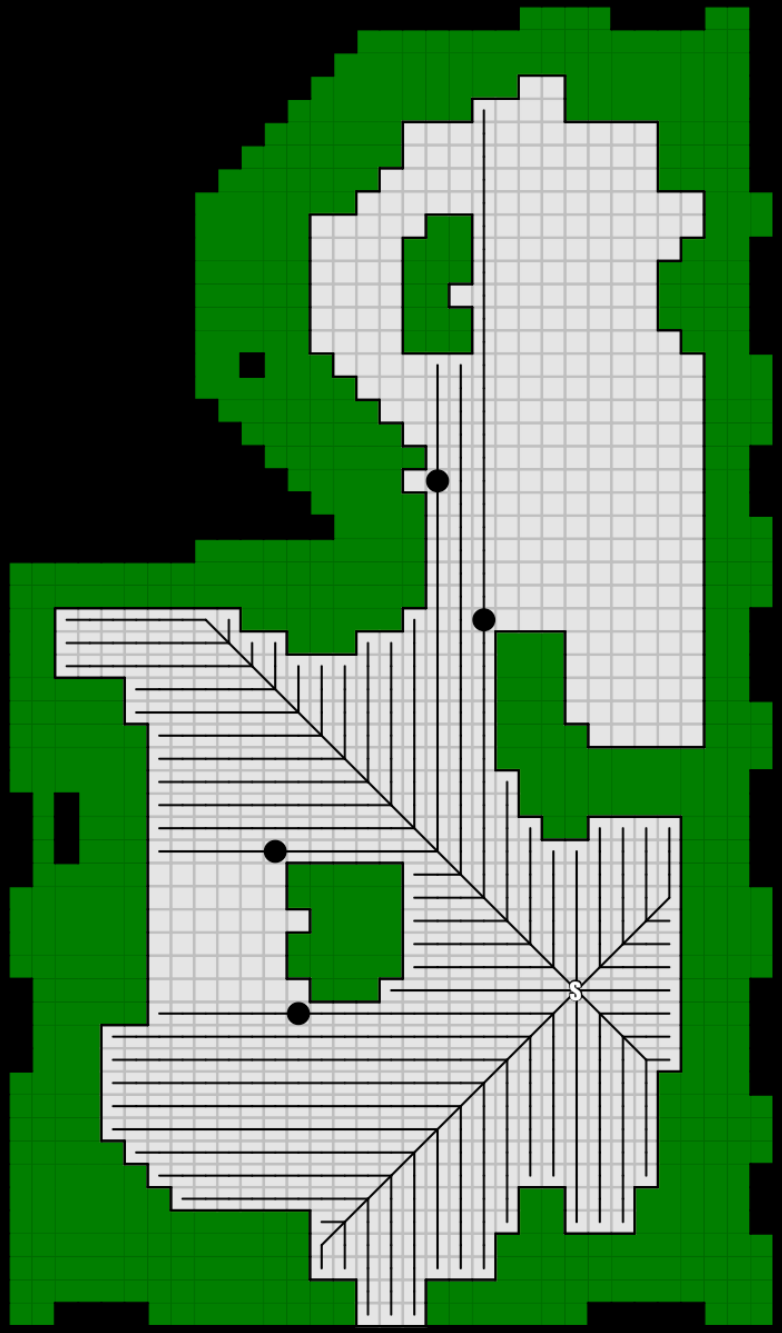
Canonical ordering of paths

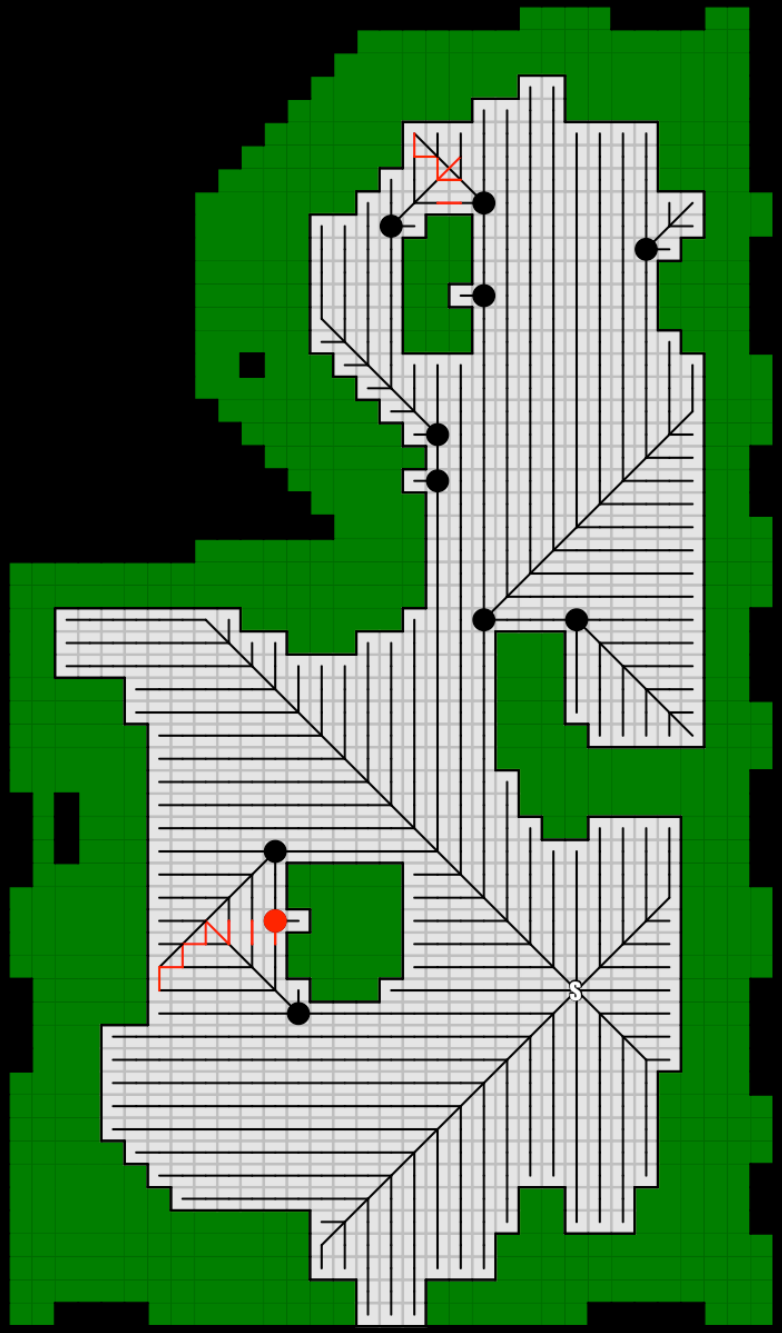
- Order all **optimal** paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to p_2





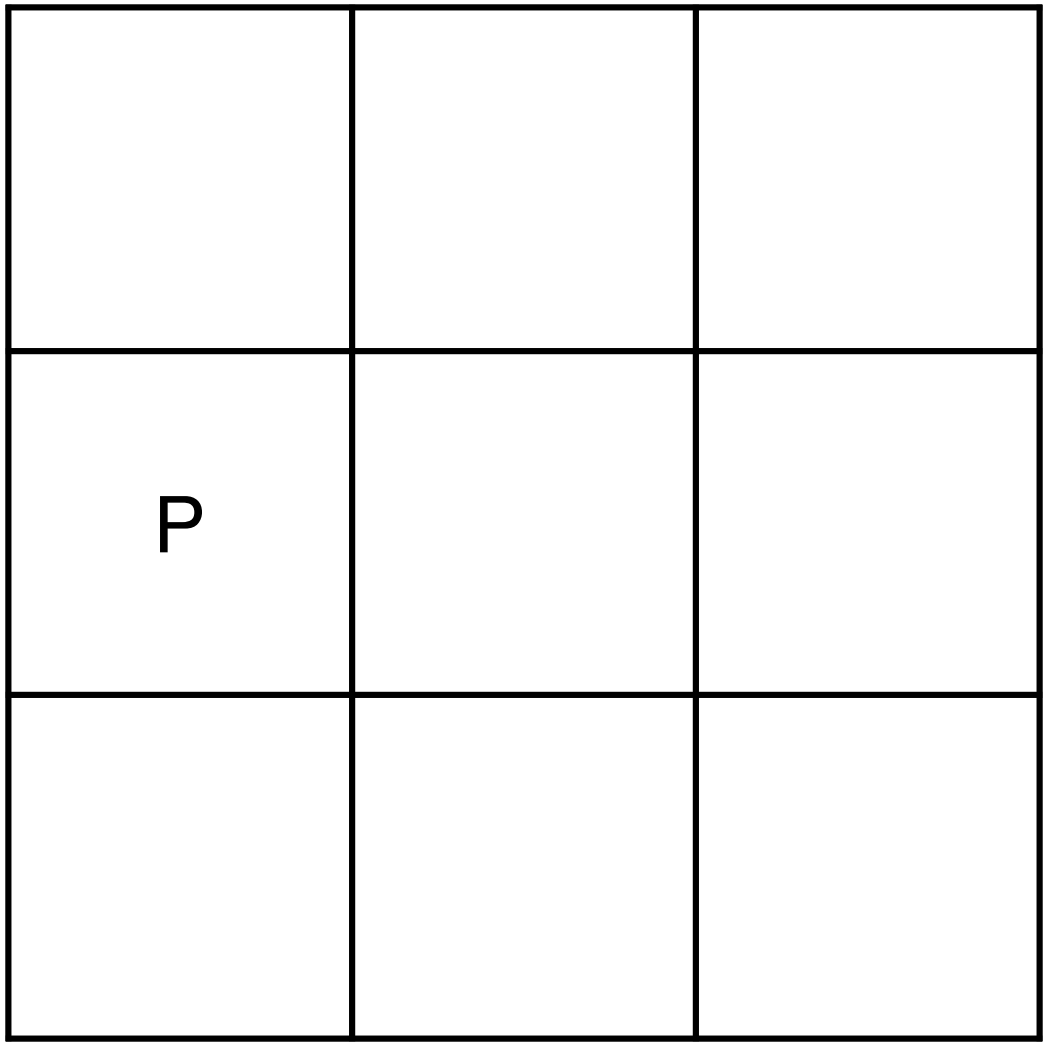


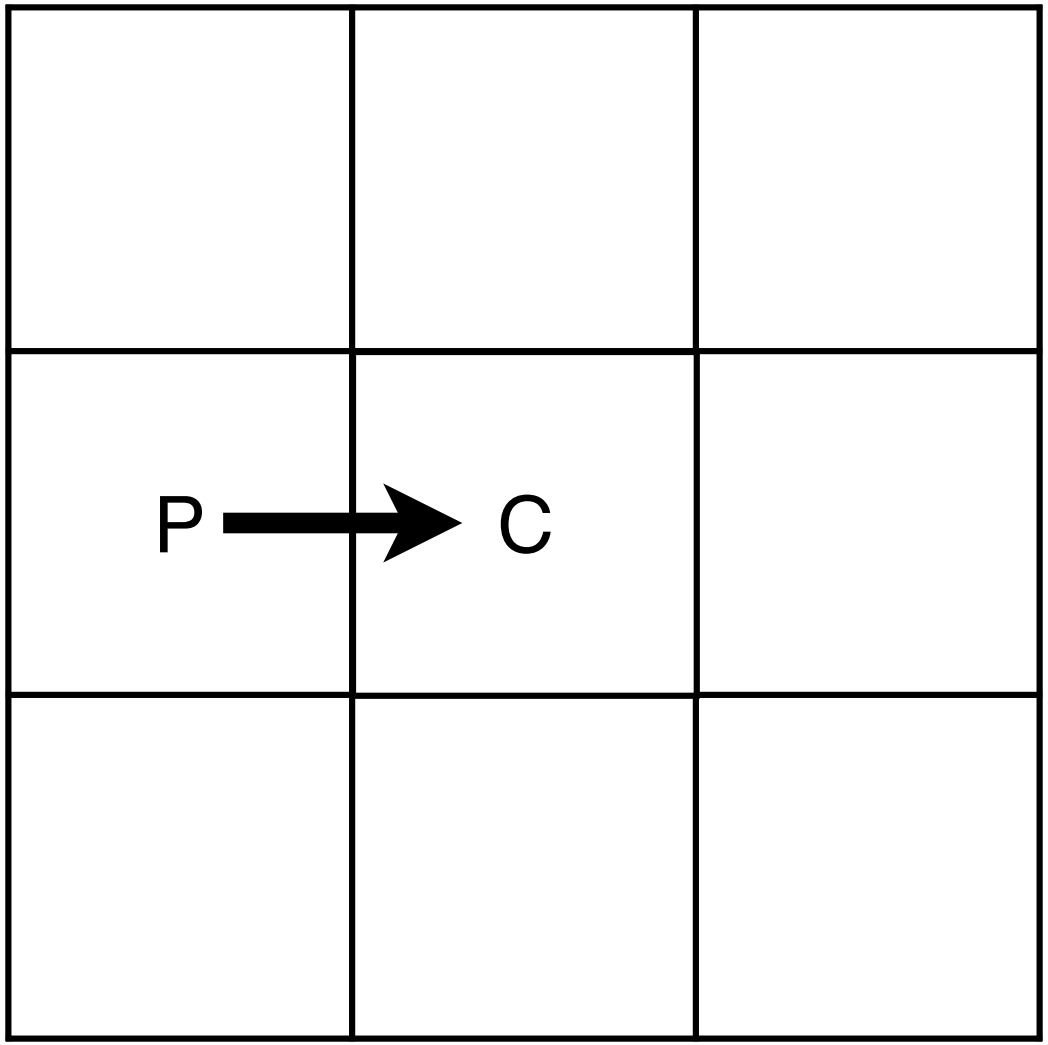

S

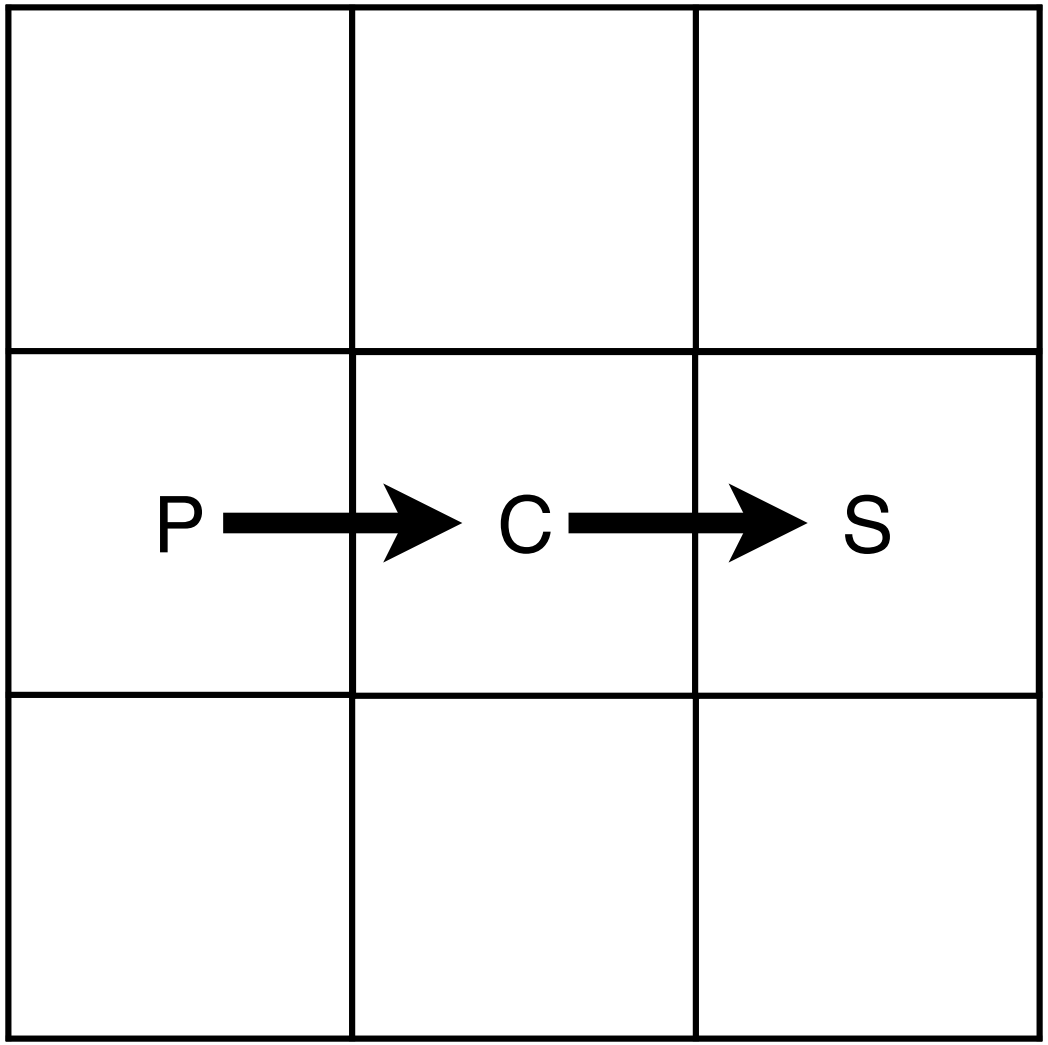

S

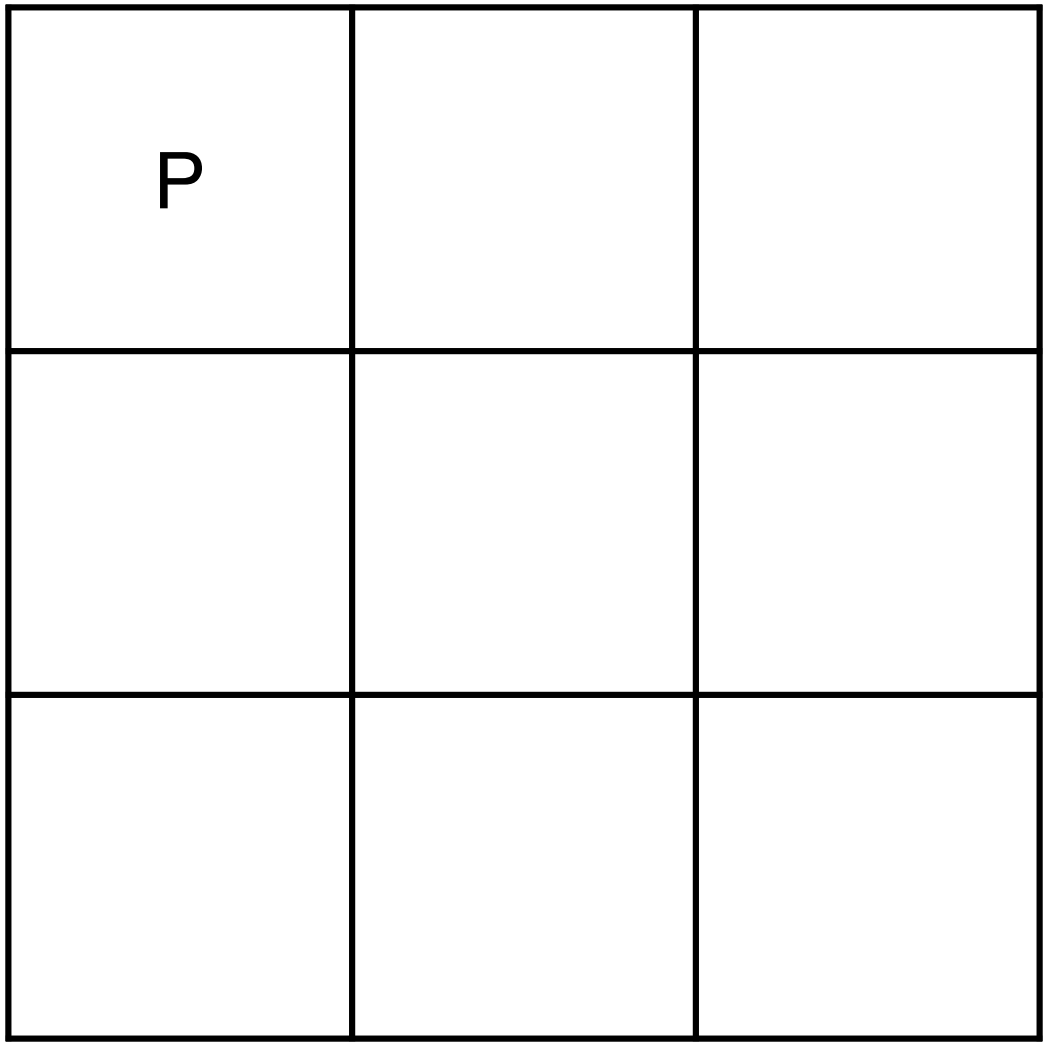
Canonical Neighbor Rule

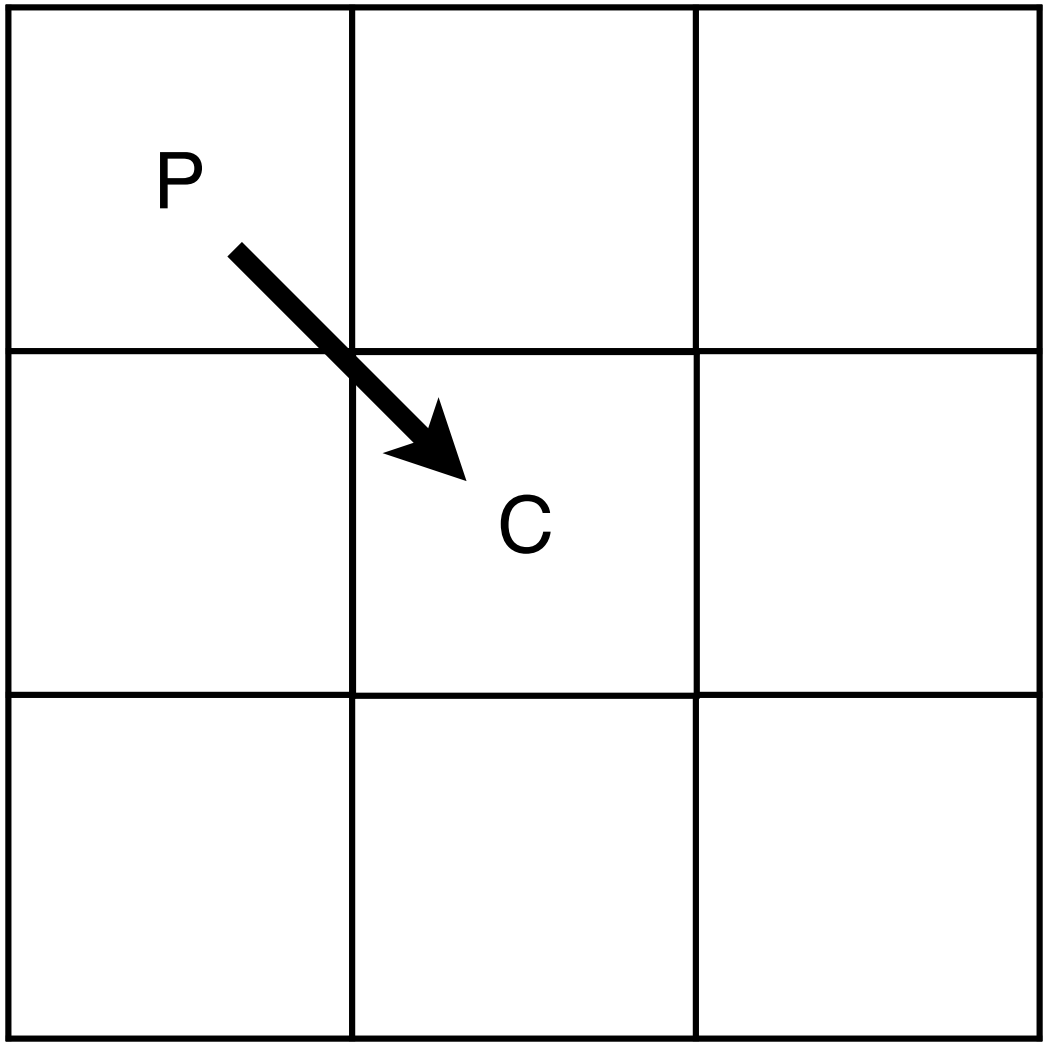
- If this state was reached from a parent via:
  - A cardinal action
    - Only 1 successor in the same cardinal direction
  - A diagonal action
    - 2 children from component cardinal actions
    - 1 diagonal child

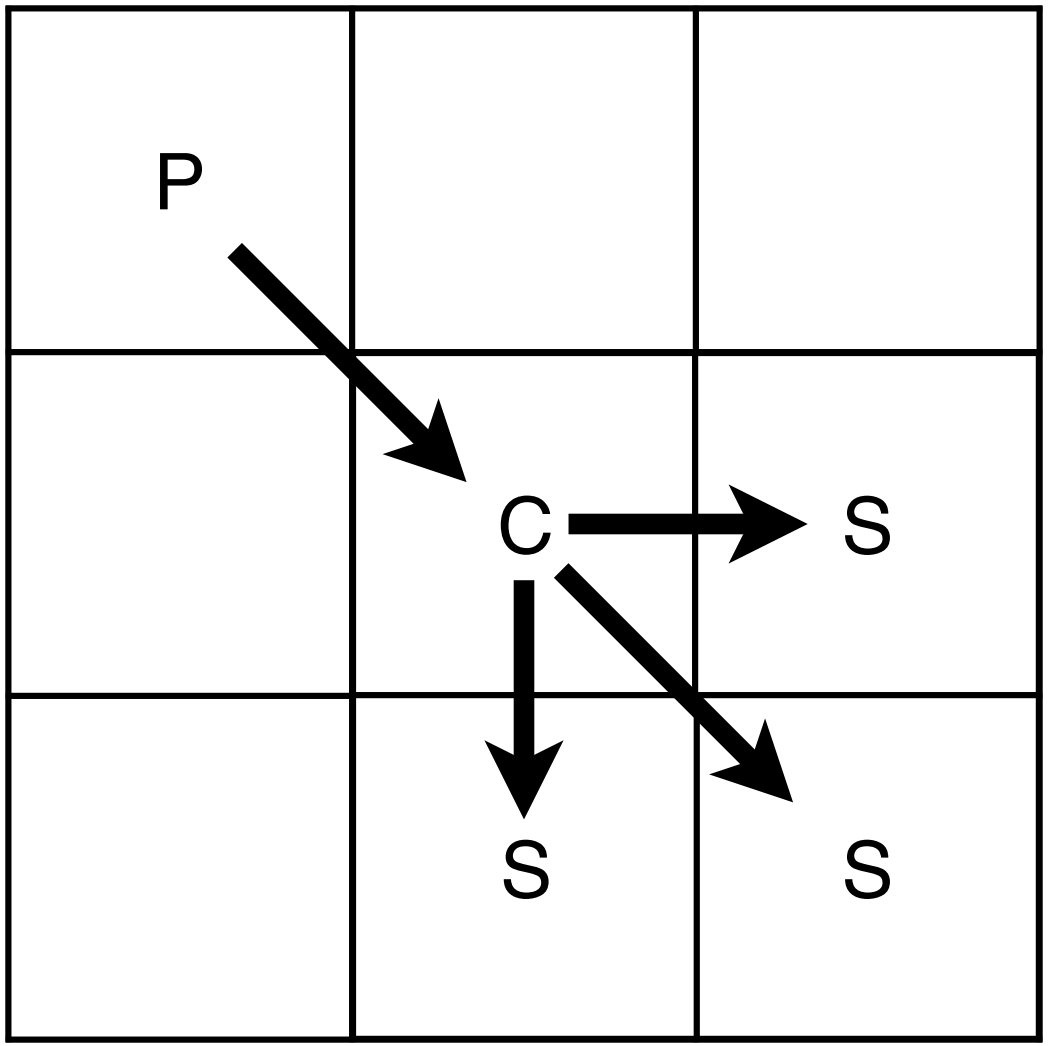

P

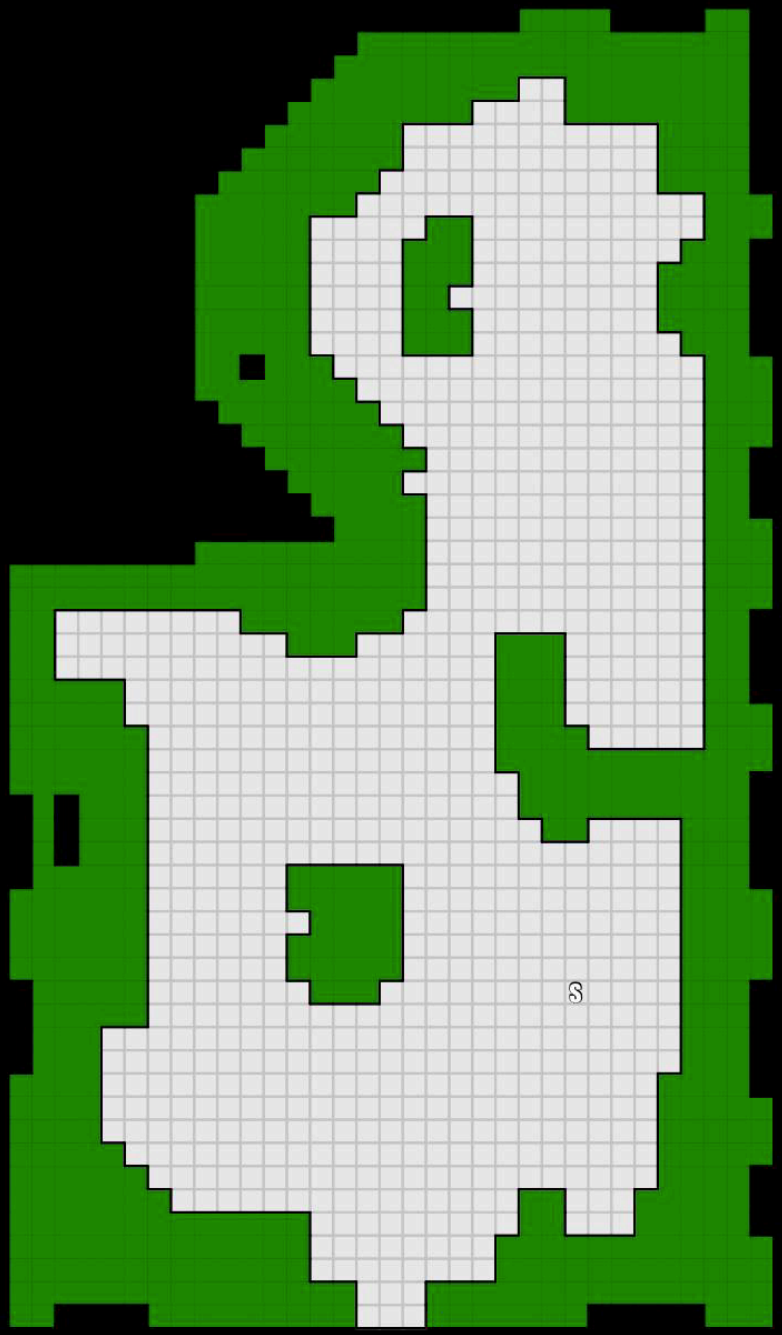

P
C

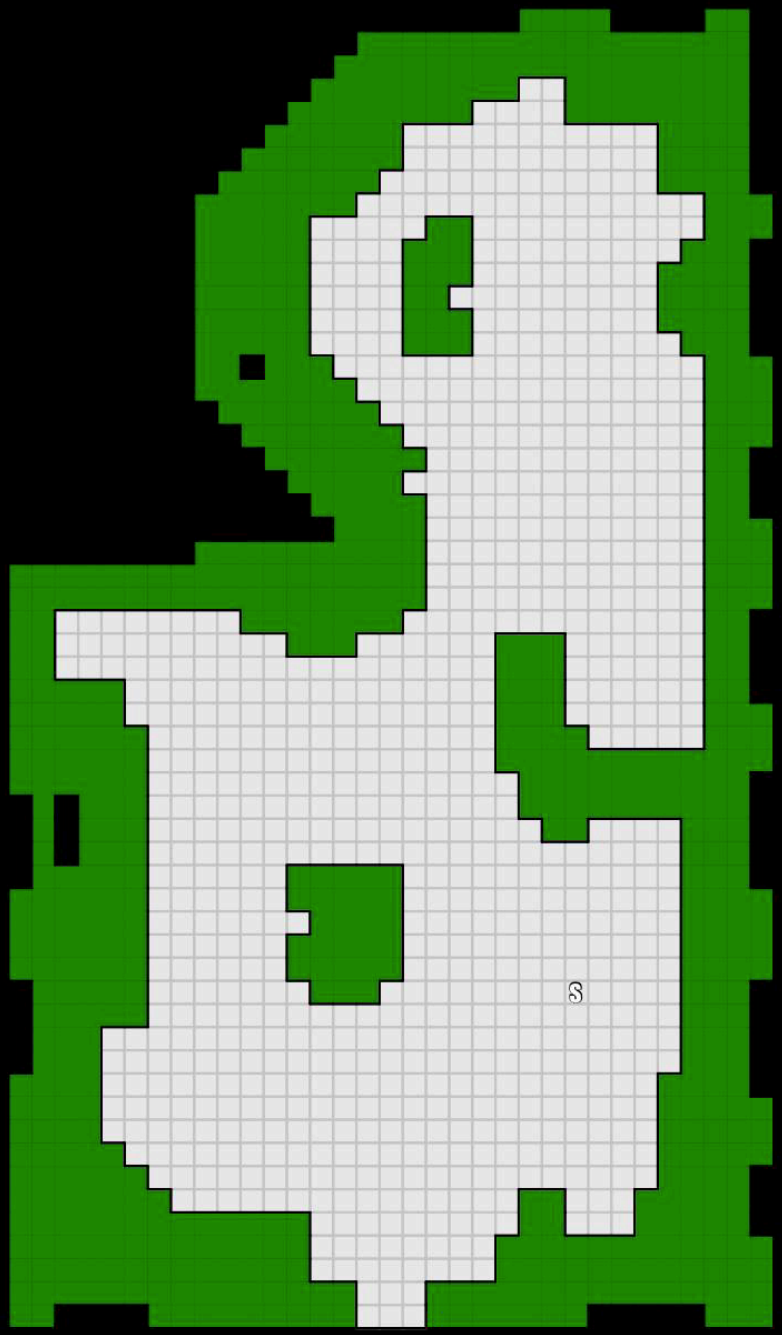

P
C
S

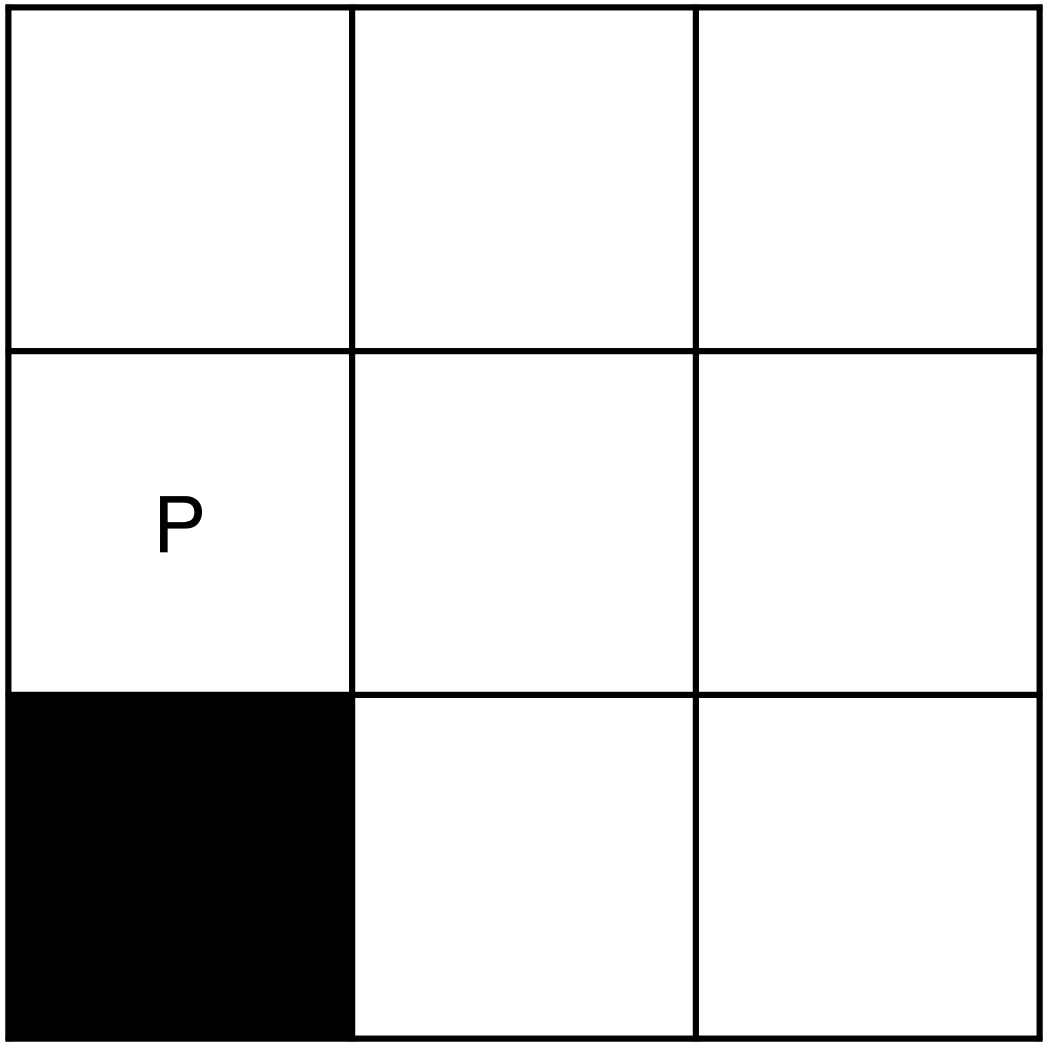

P

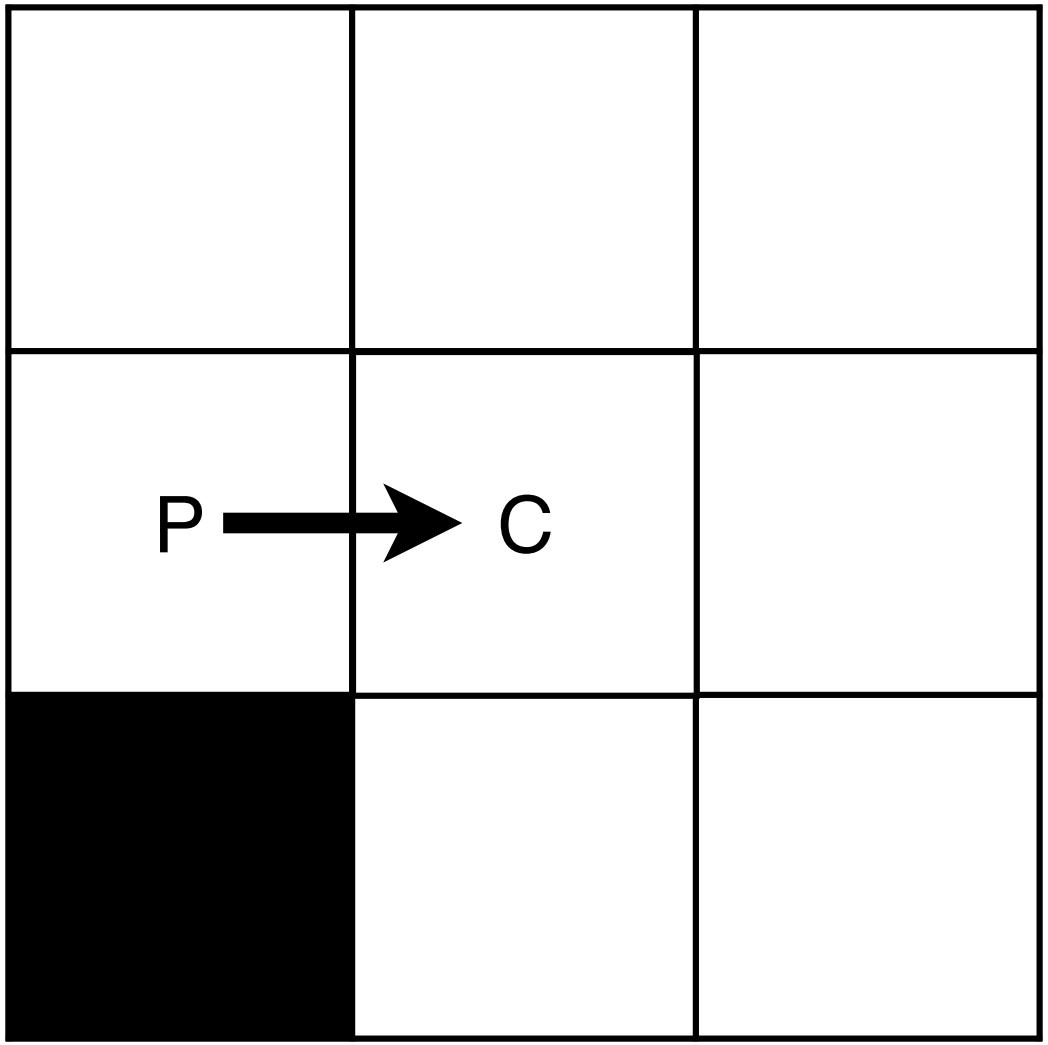

P
C

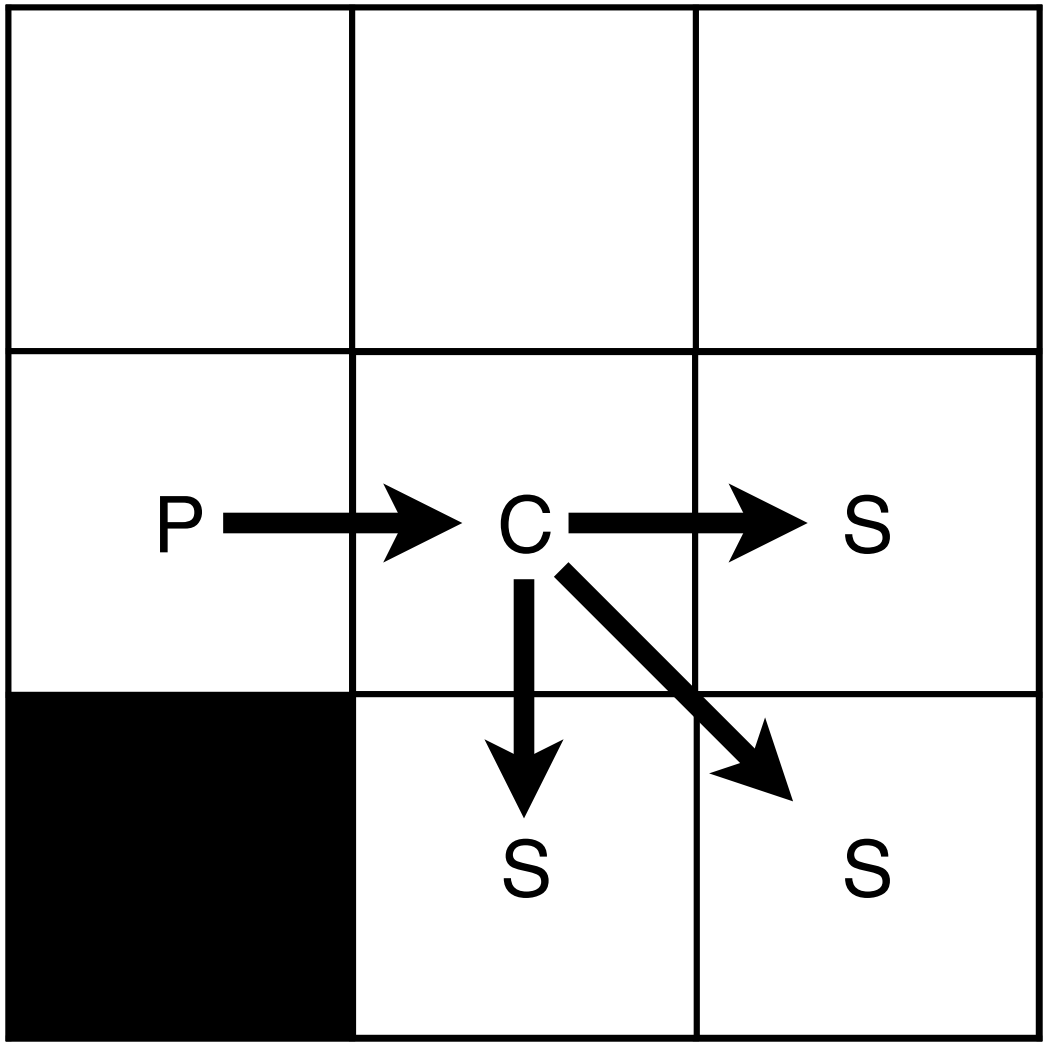

P
C
S
S
S

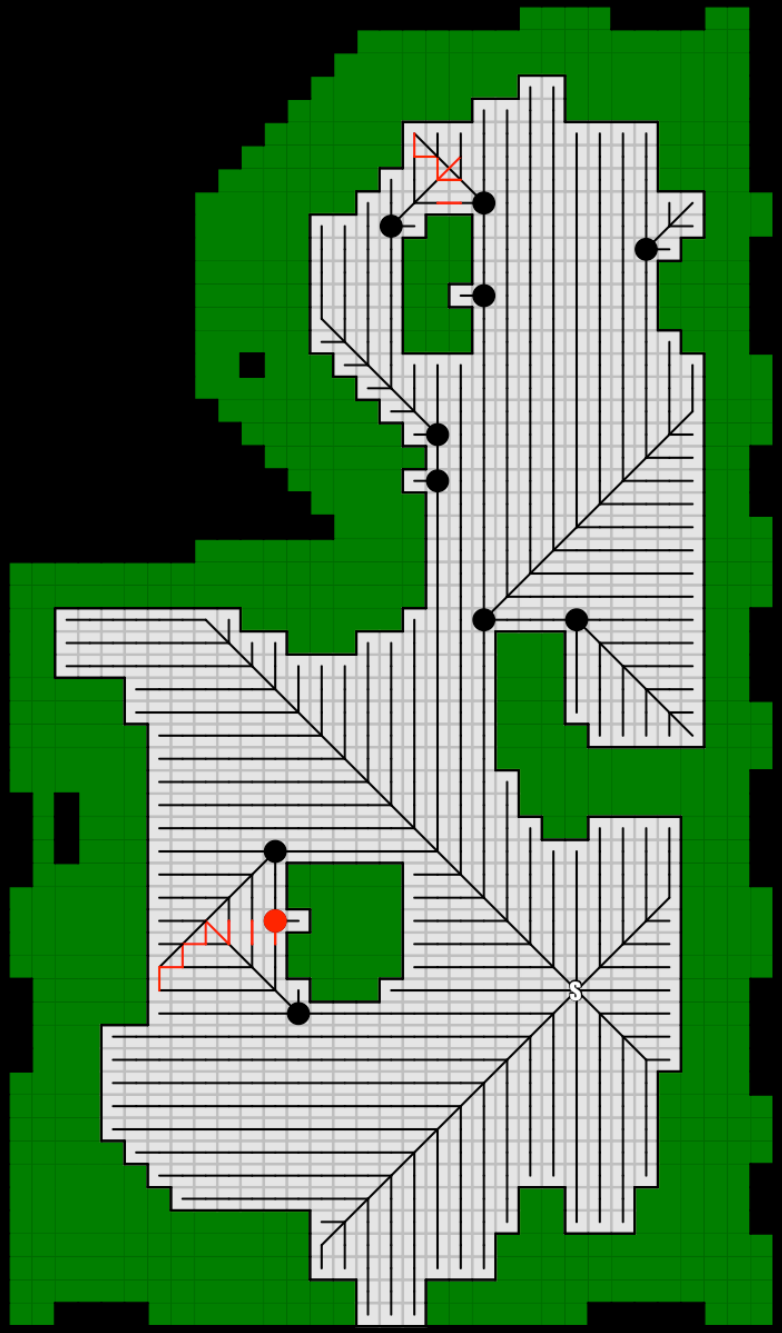


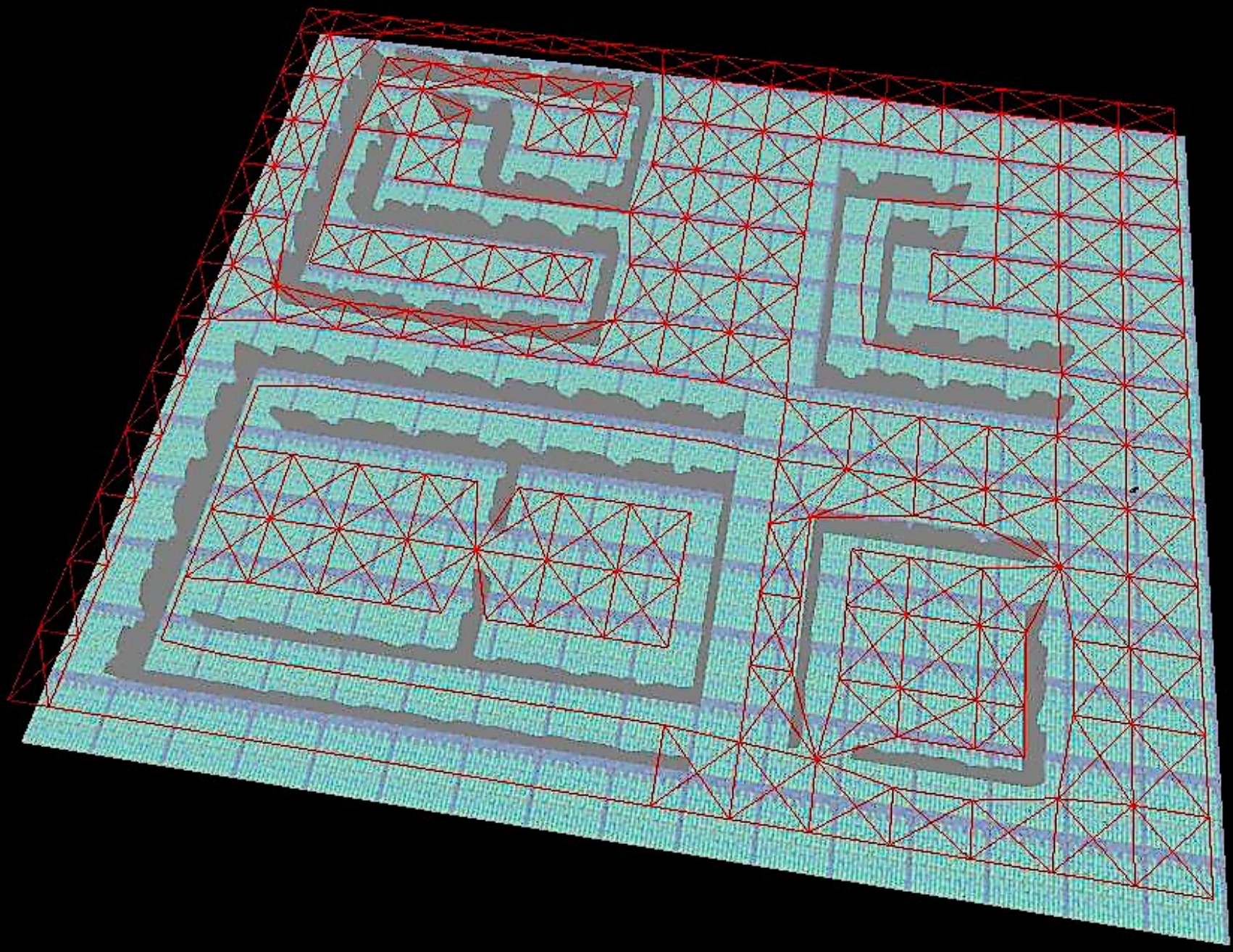



P


P
C


P
C
S
S
S

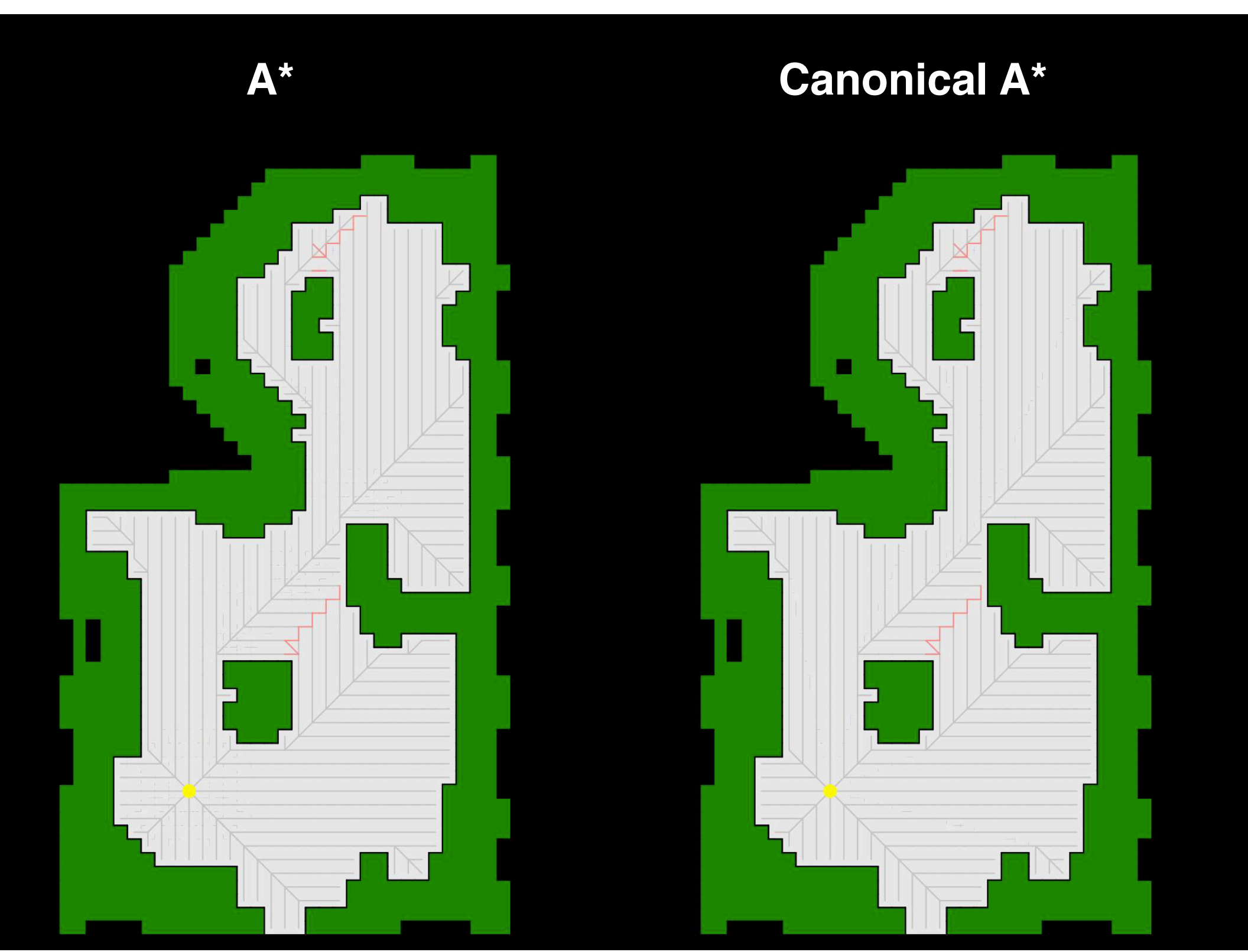

S

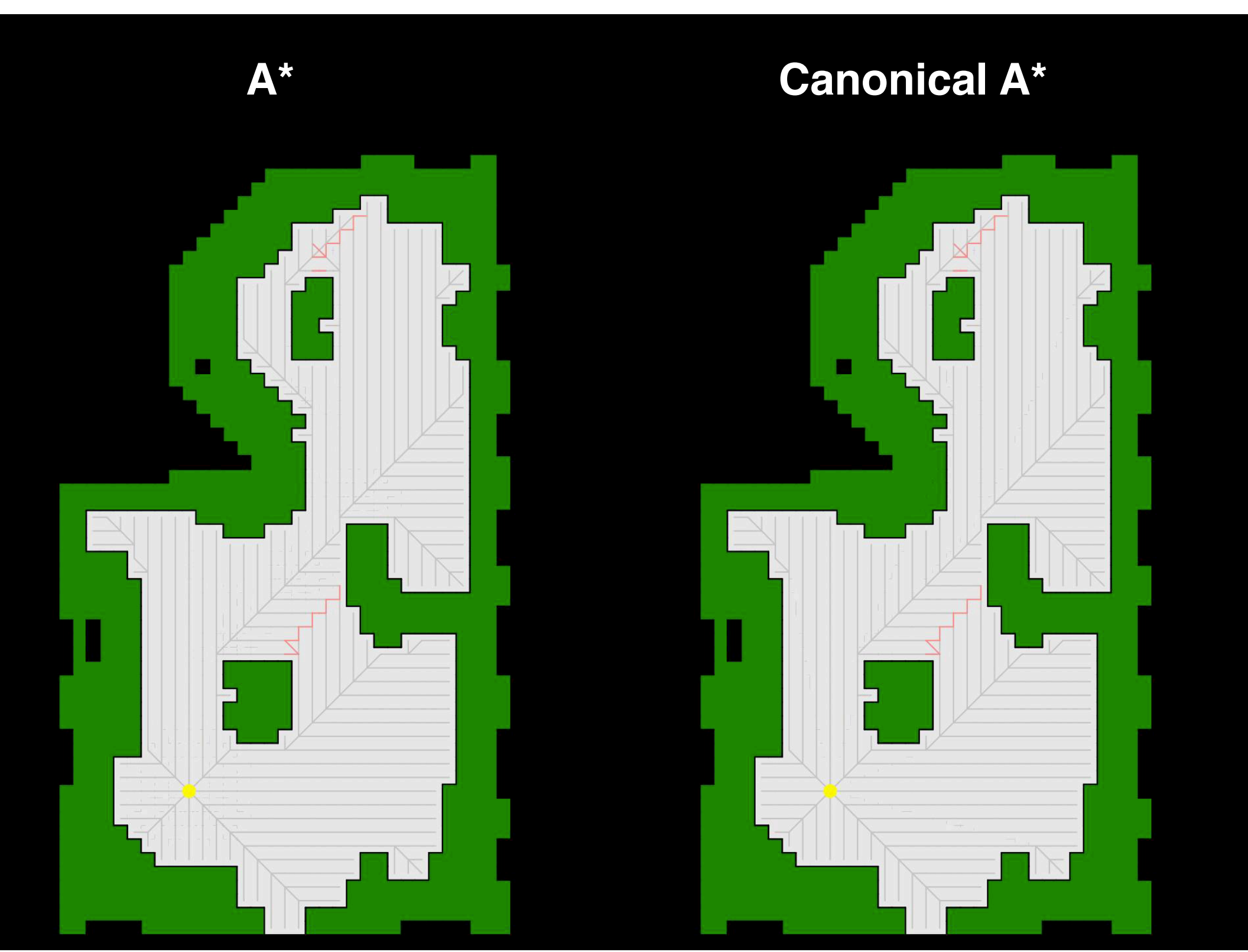


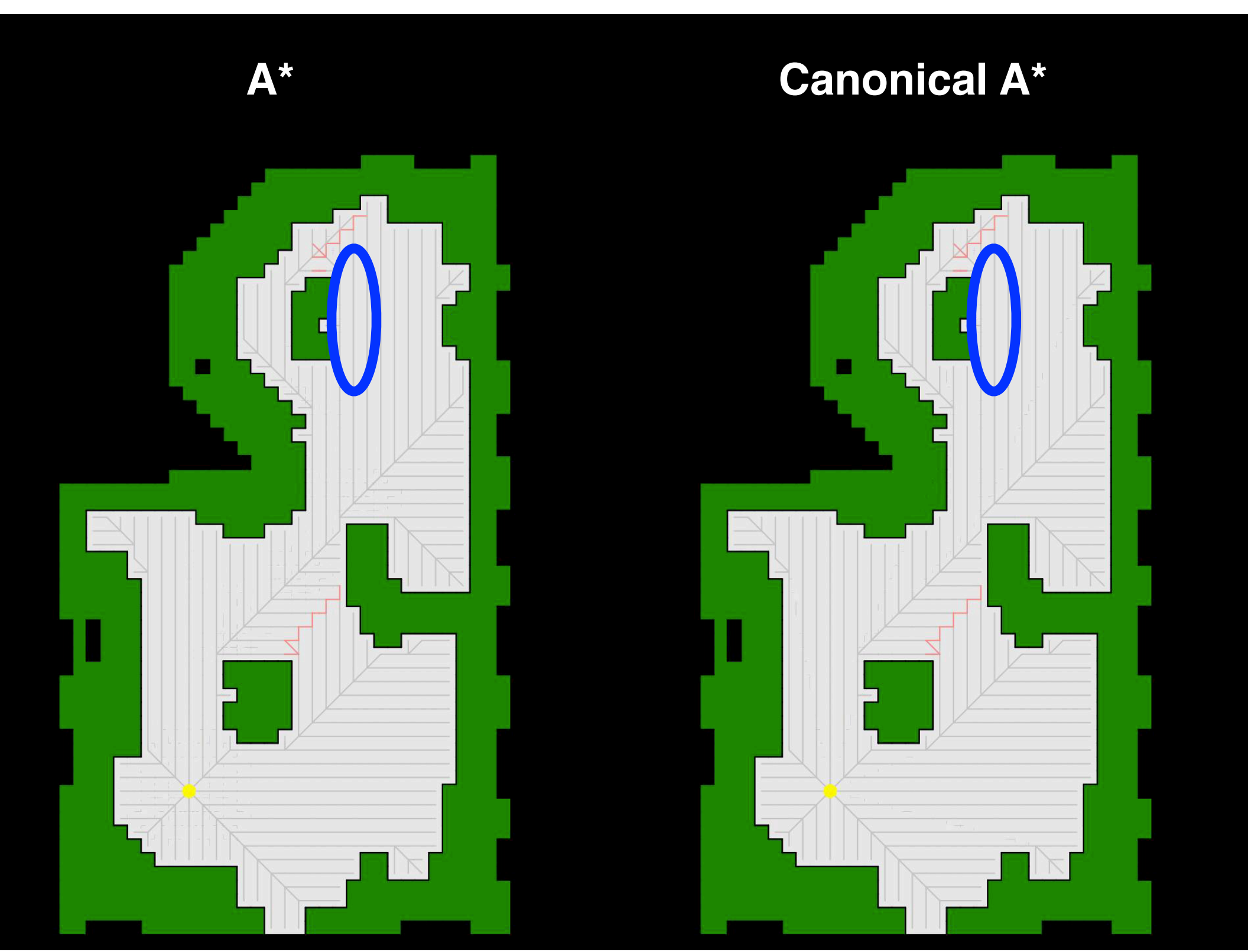
Canonical Orderings in Graphs

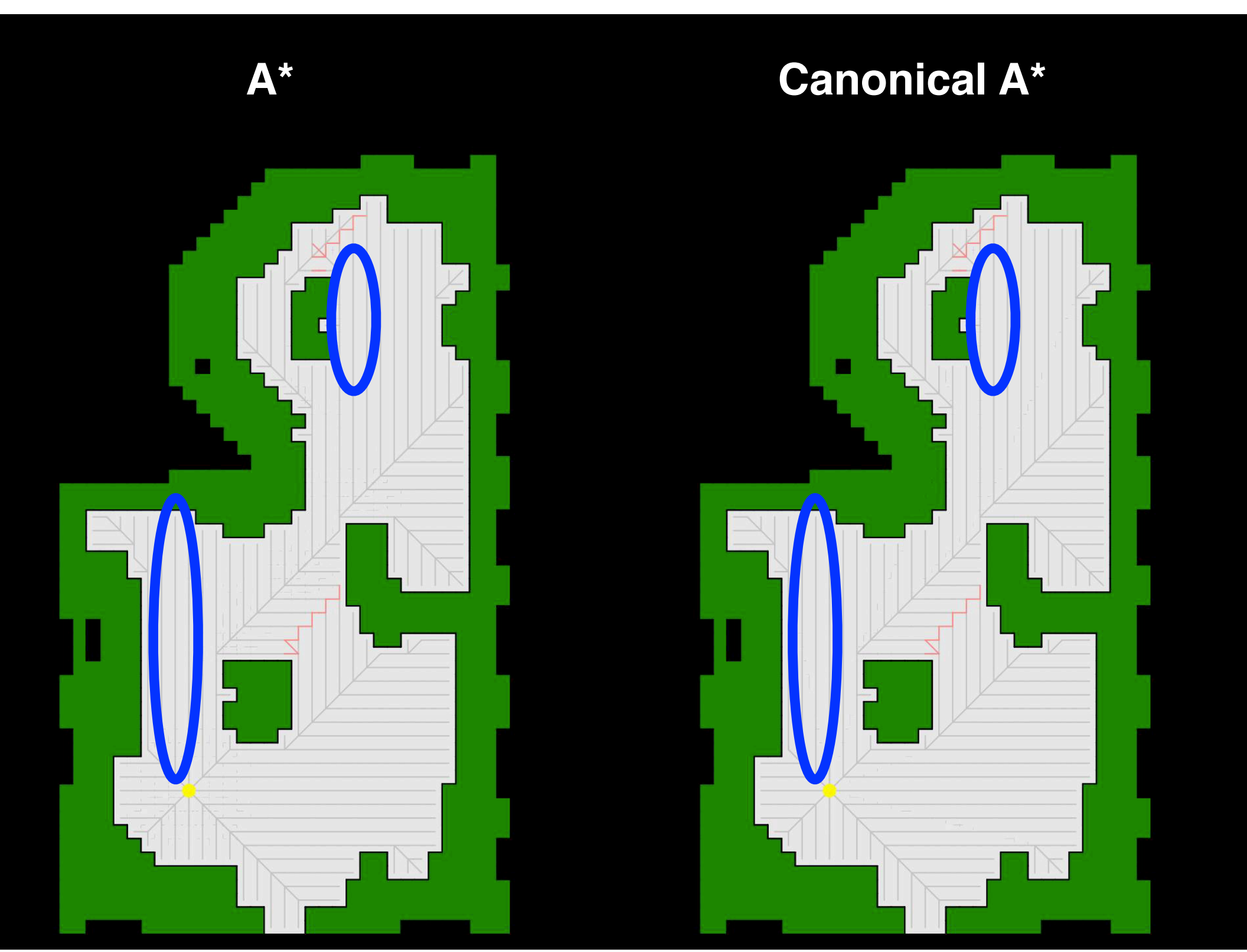
- Used to improve reach computation
  - (Goldberg, Kaplan and Werneck, 2006)
- Used to reduce redundant paths in transit routing
  - (Antsfeld, Harabor, Kilby, and Walsh, 2012)

- Can we use it to prune successors like in JPS


A*
Canonical A*


A*
Canonical A*


A*
Canonical A*


A*
Canonical A*

Experimental Results

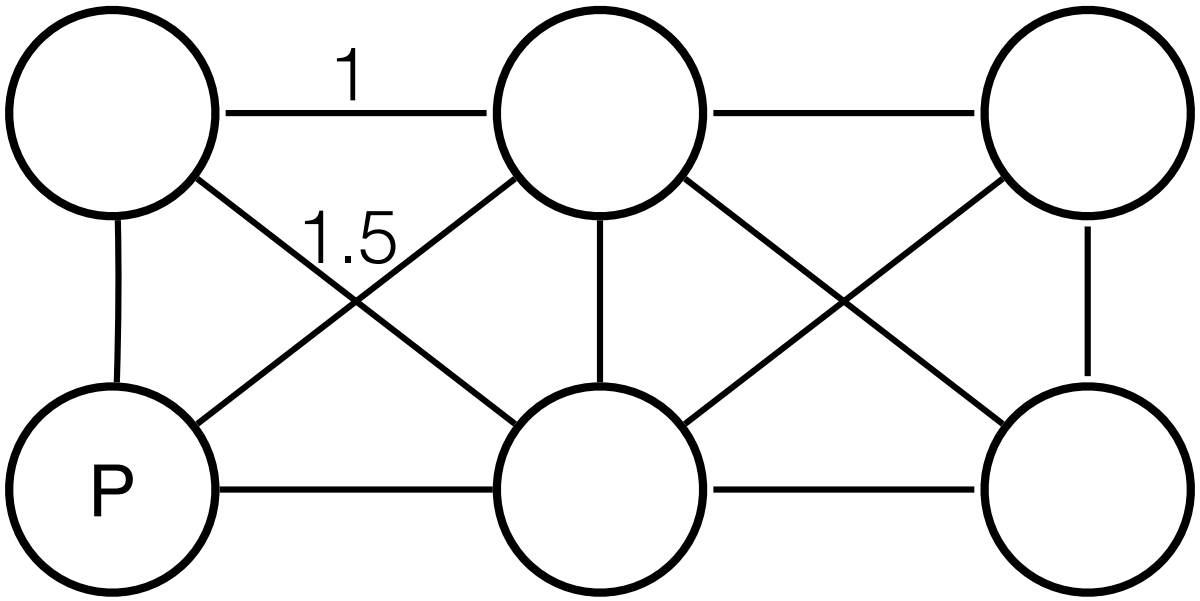
	A*	CA*
Expansions	13,295	13,302
Generations	99,483	13,654

Canonical Ordering Rules

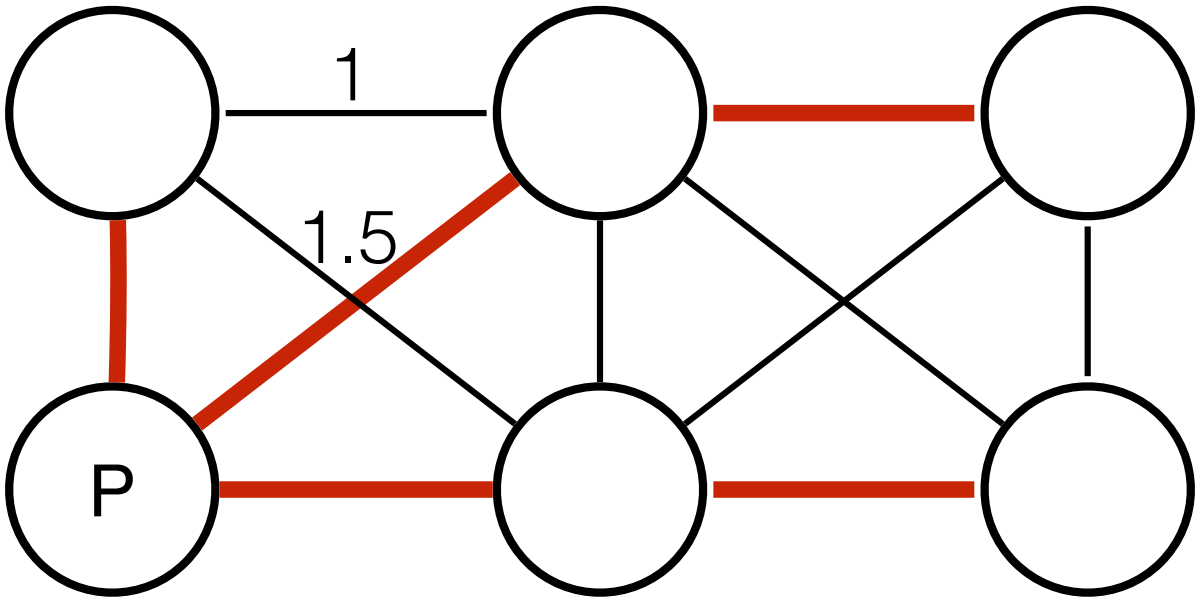
- In grids:
- The action to reach a state determines successors
  - This is generic to any state

- In graphs:
- Given the parent, determine the successors
  - Store specific information for every state

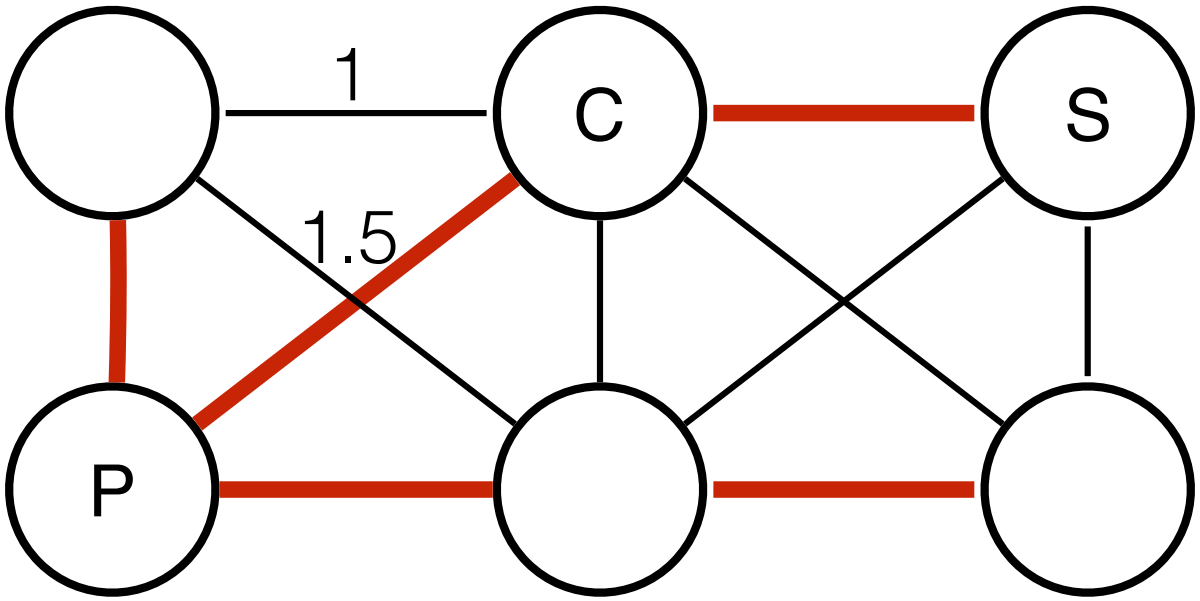
Local Search Method



Local Search Method

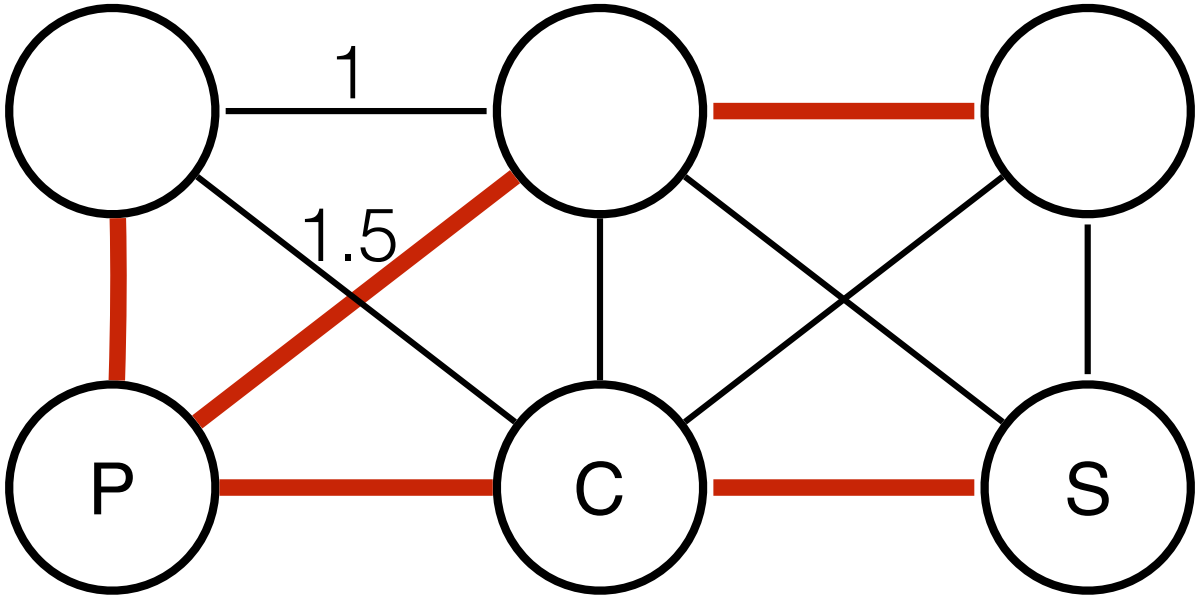


Local Search Method

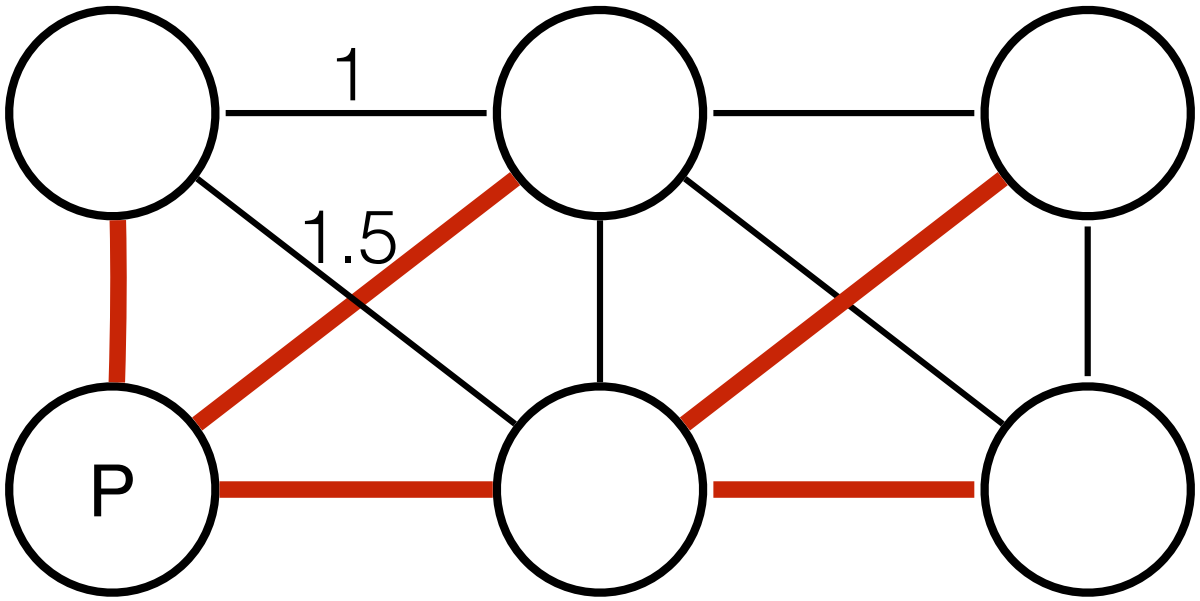


Local Search Method

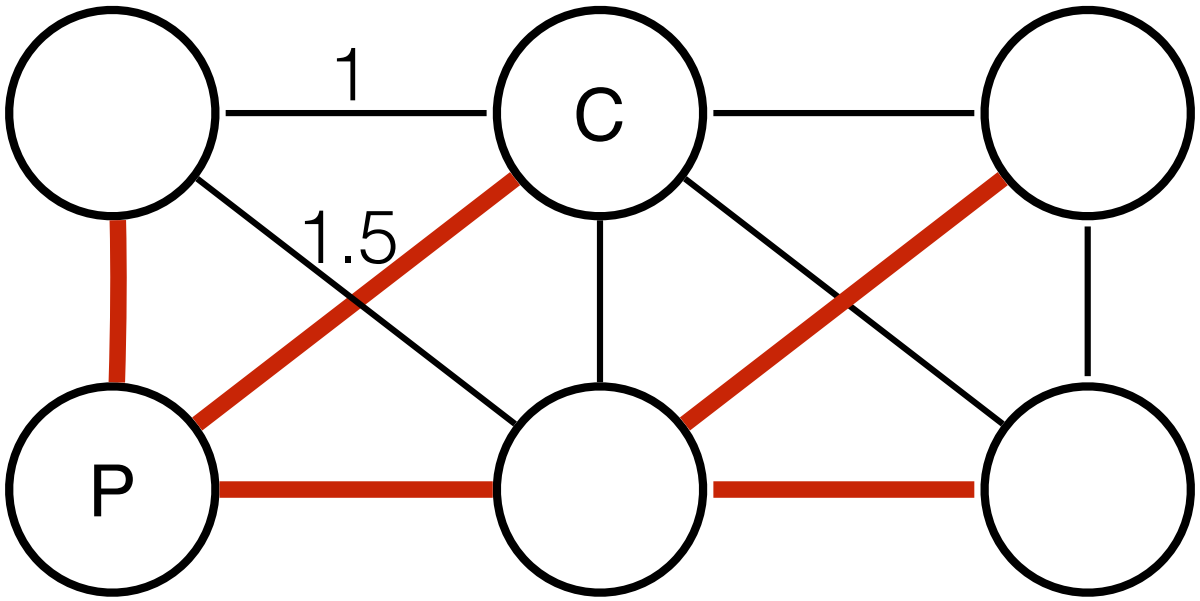
Note that we also prune neighbors even when there isn't a single canonical ordering. Even if all path costs are unique, it still will help.



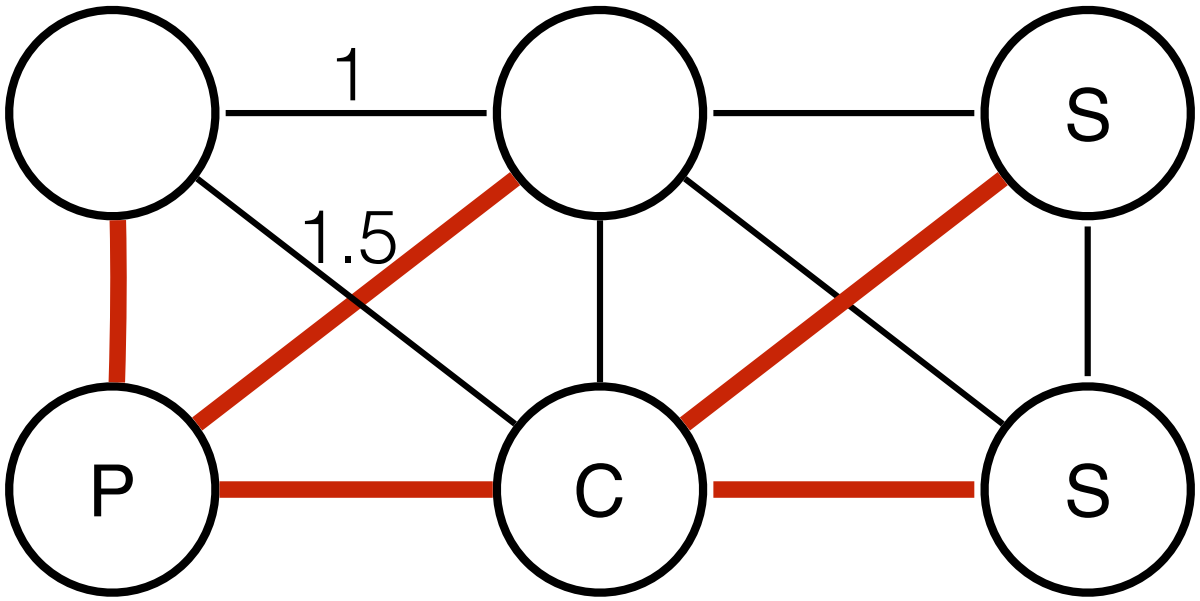
Local Search Method



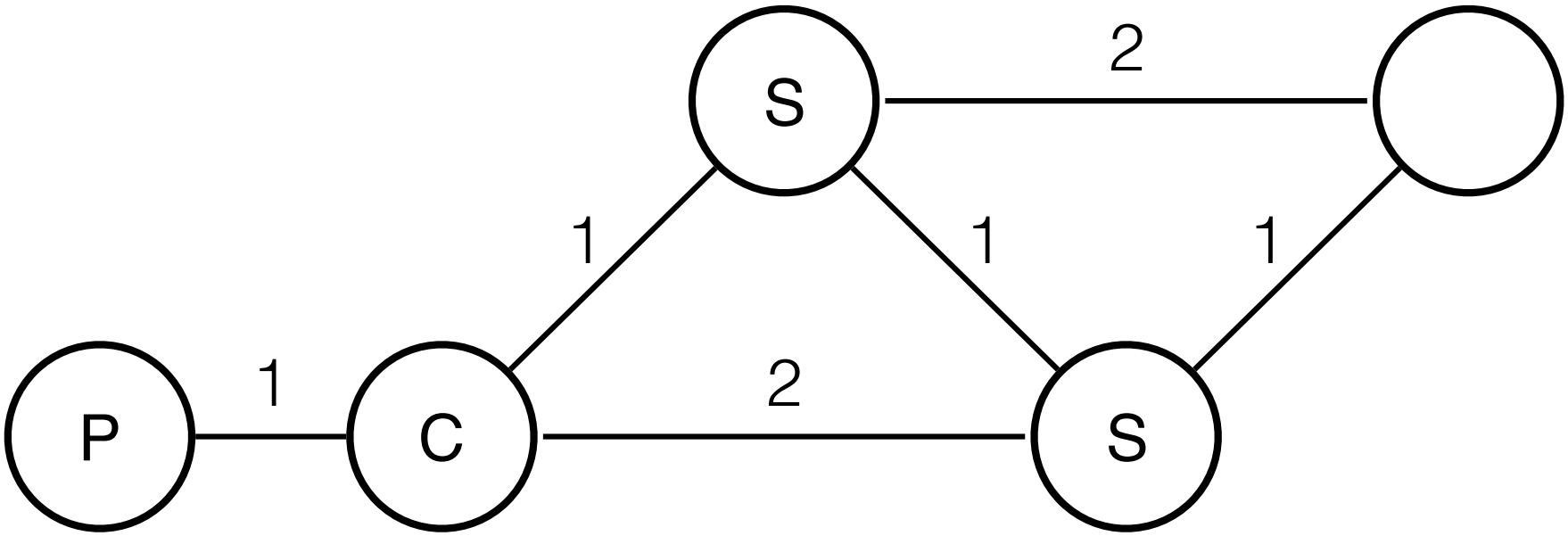
Local Search Method



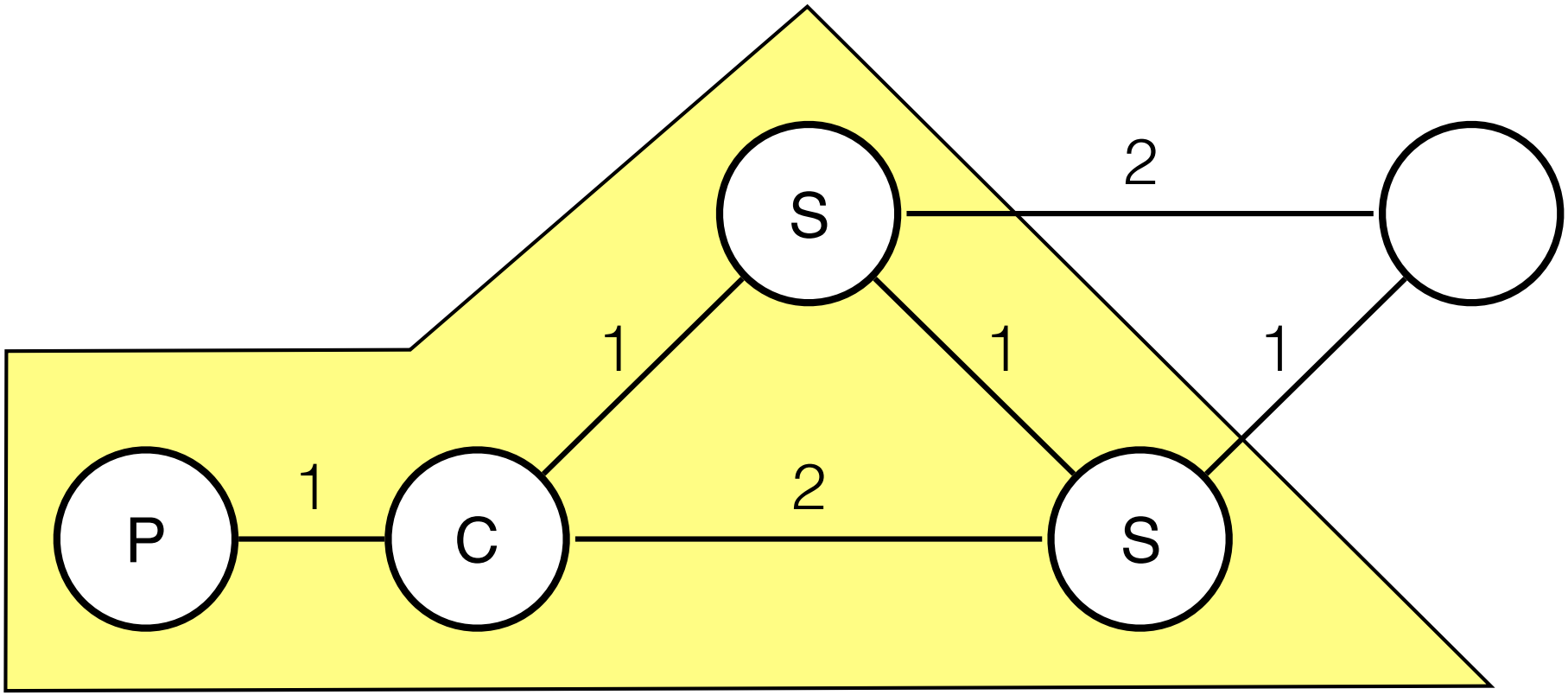
Local Search Method



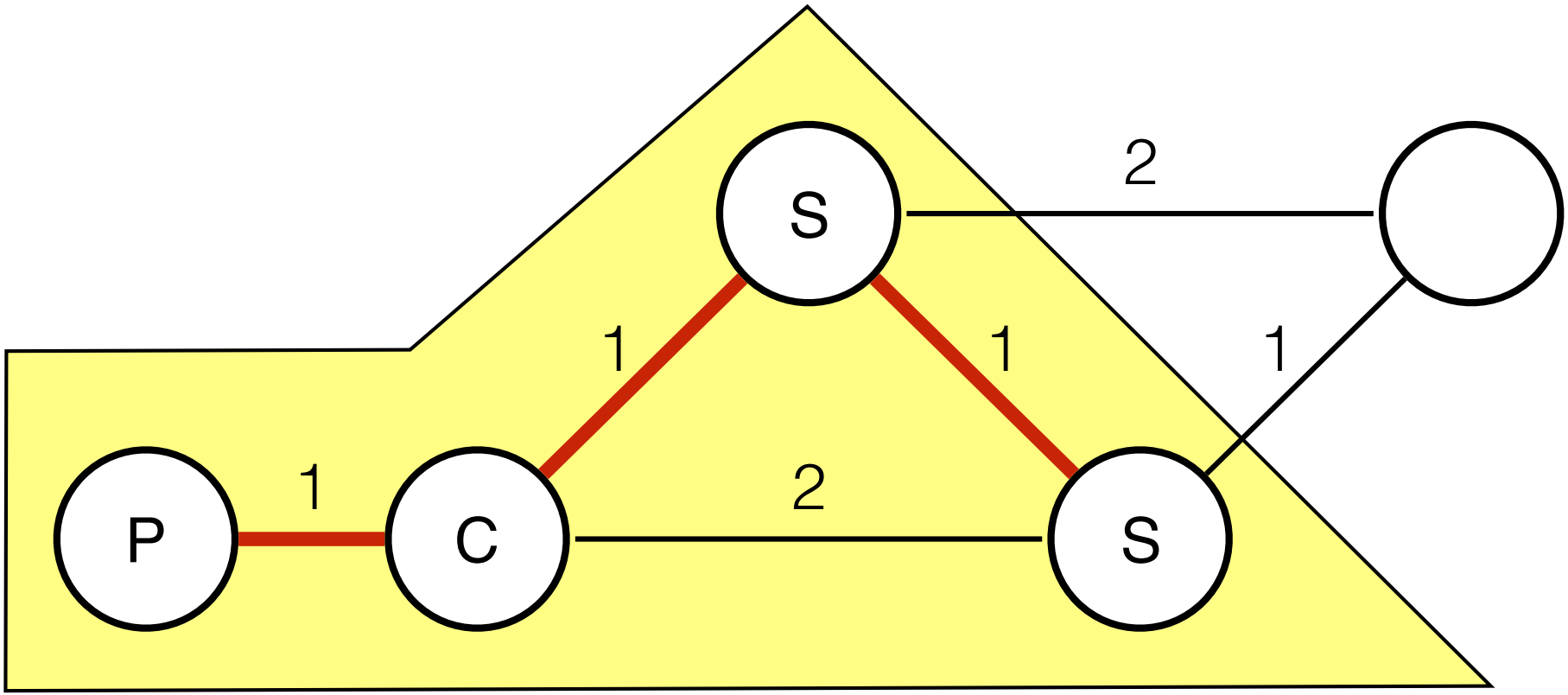
Correctness



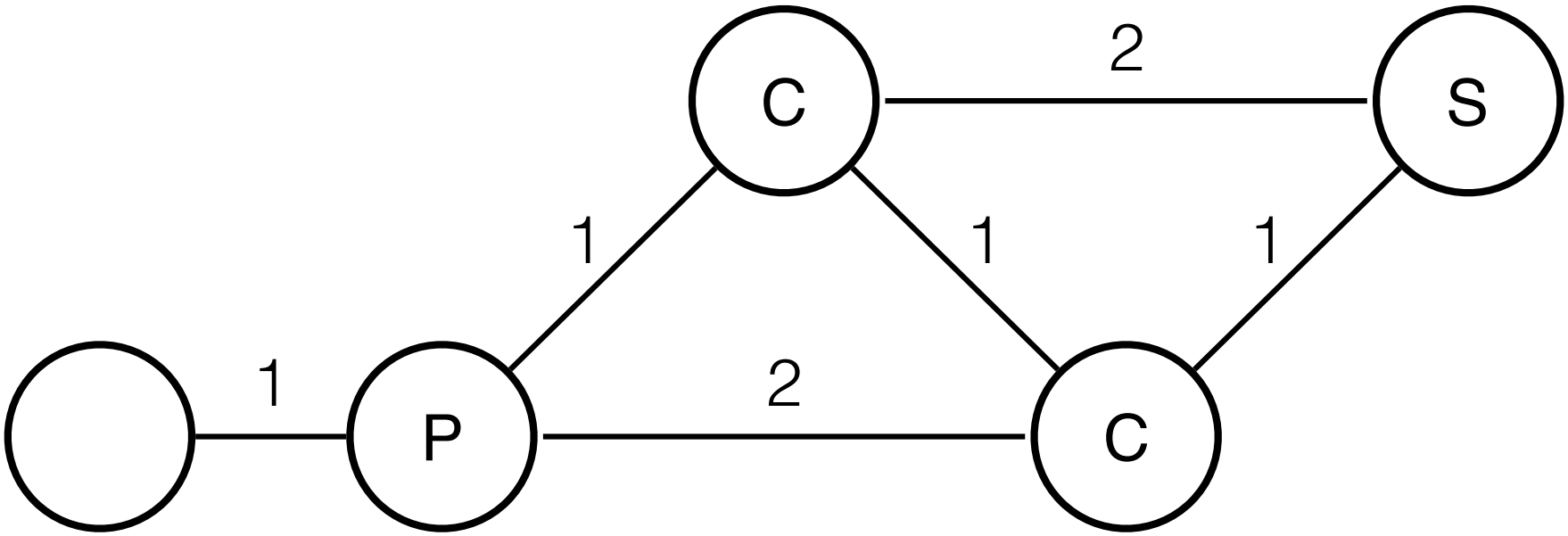
Correctness



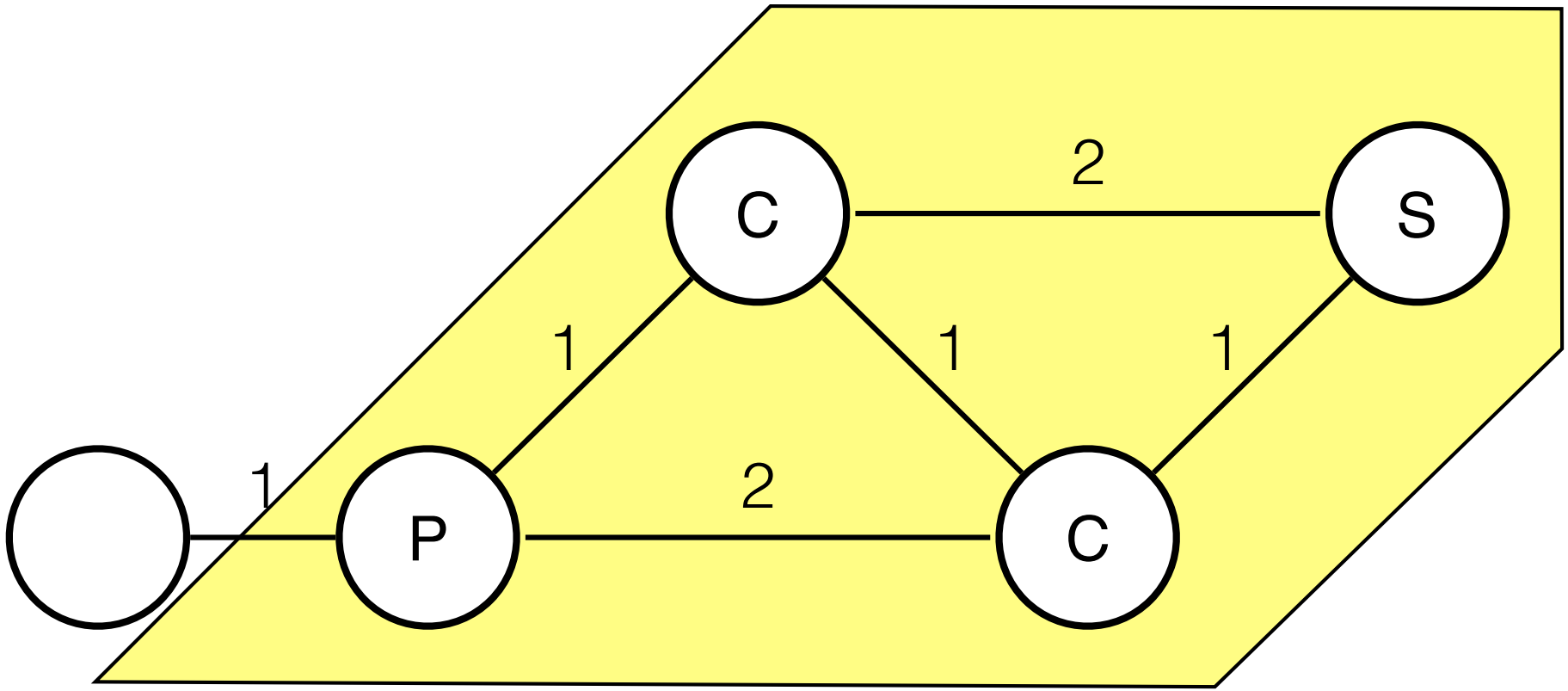
Correctness



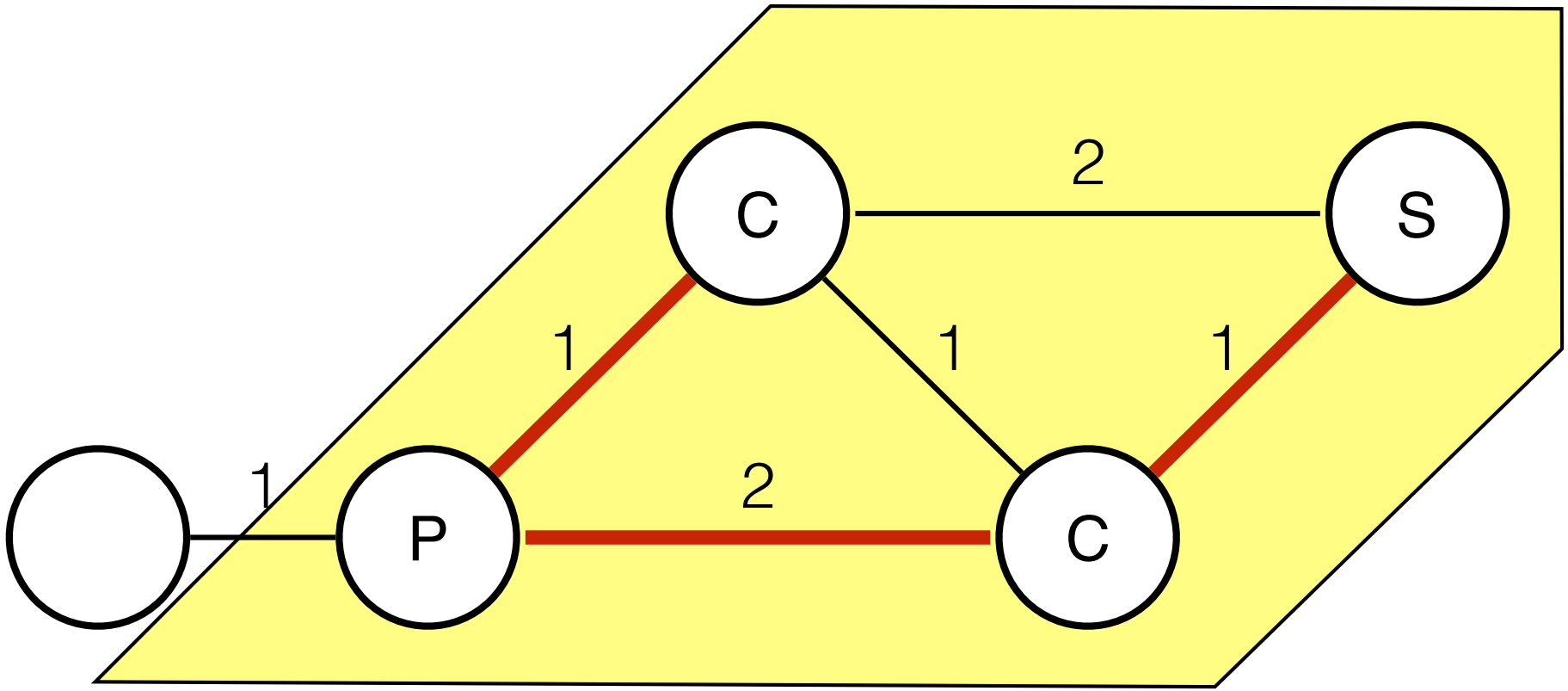
Correctness



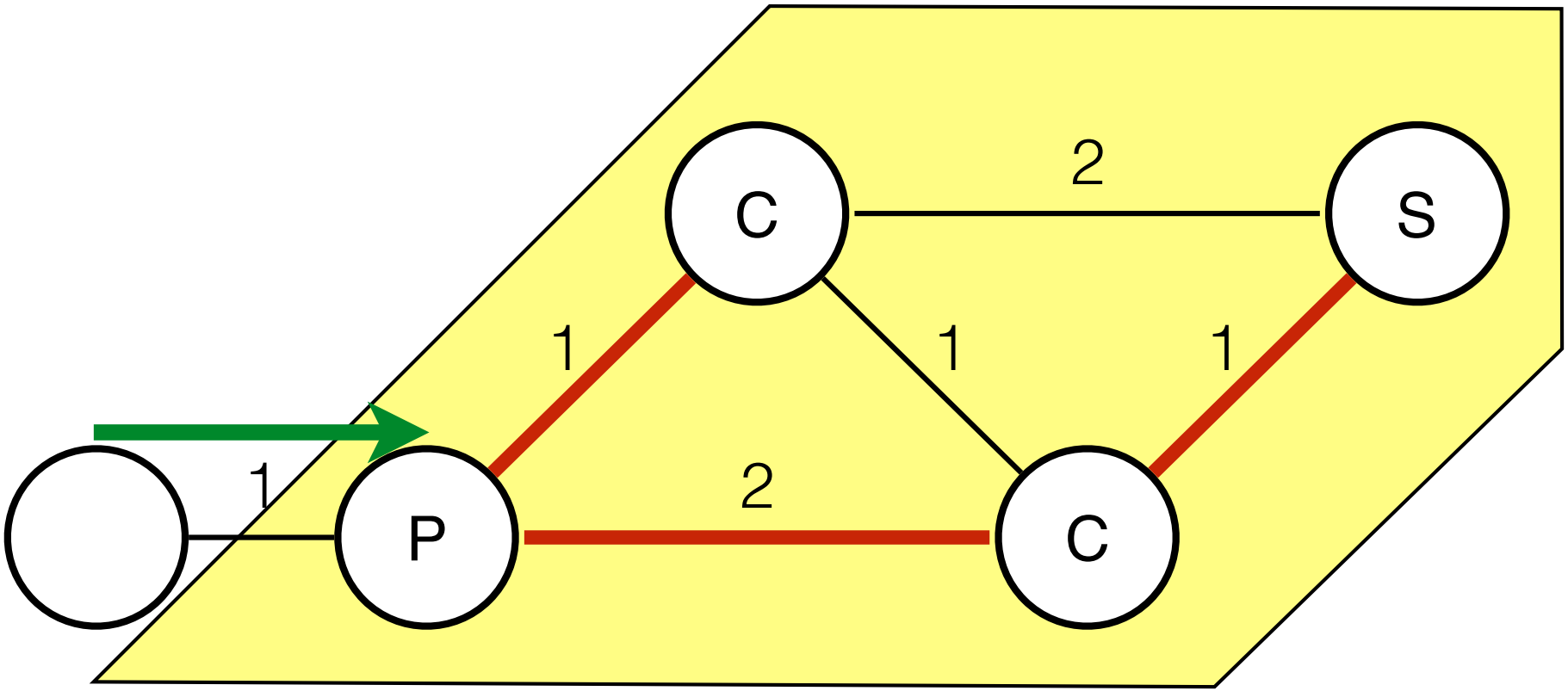
Correctness



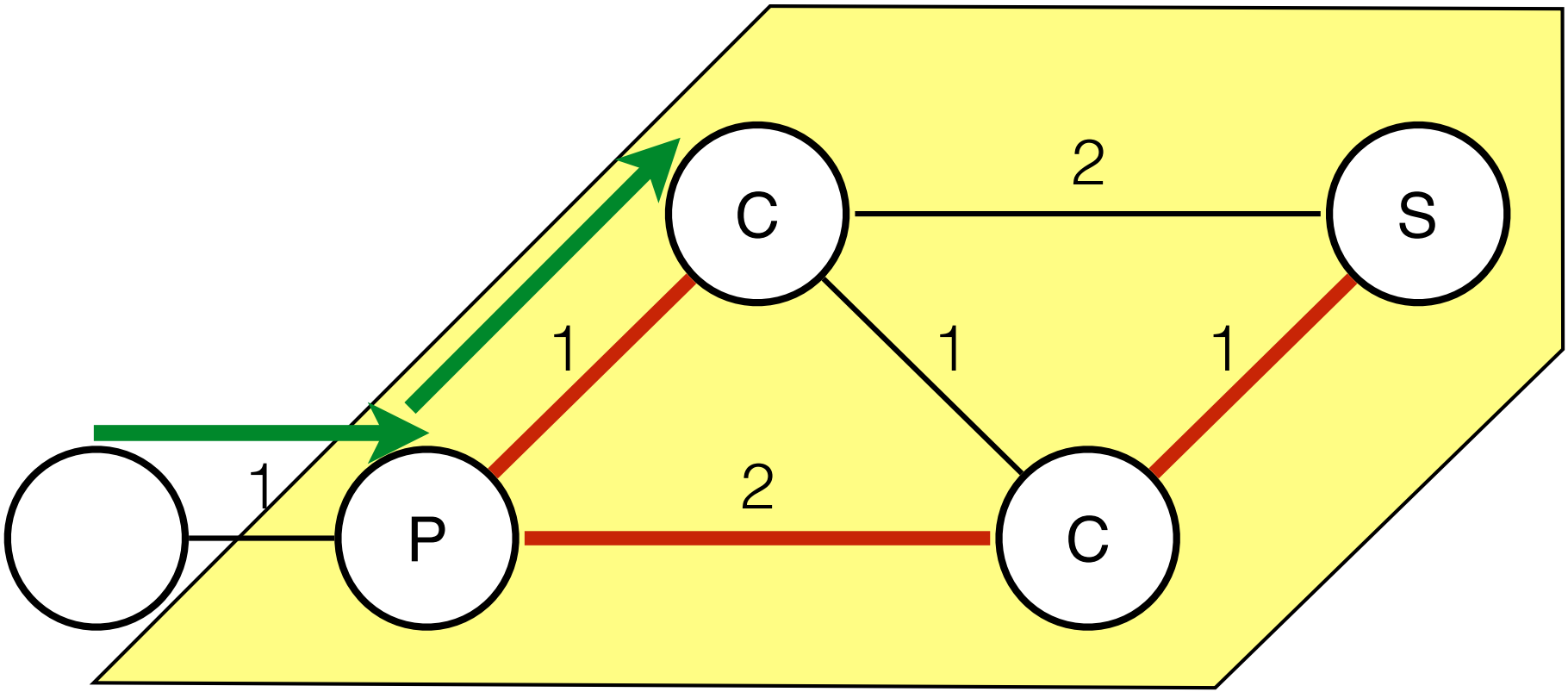
Correctness



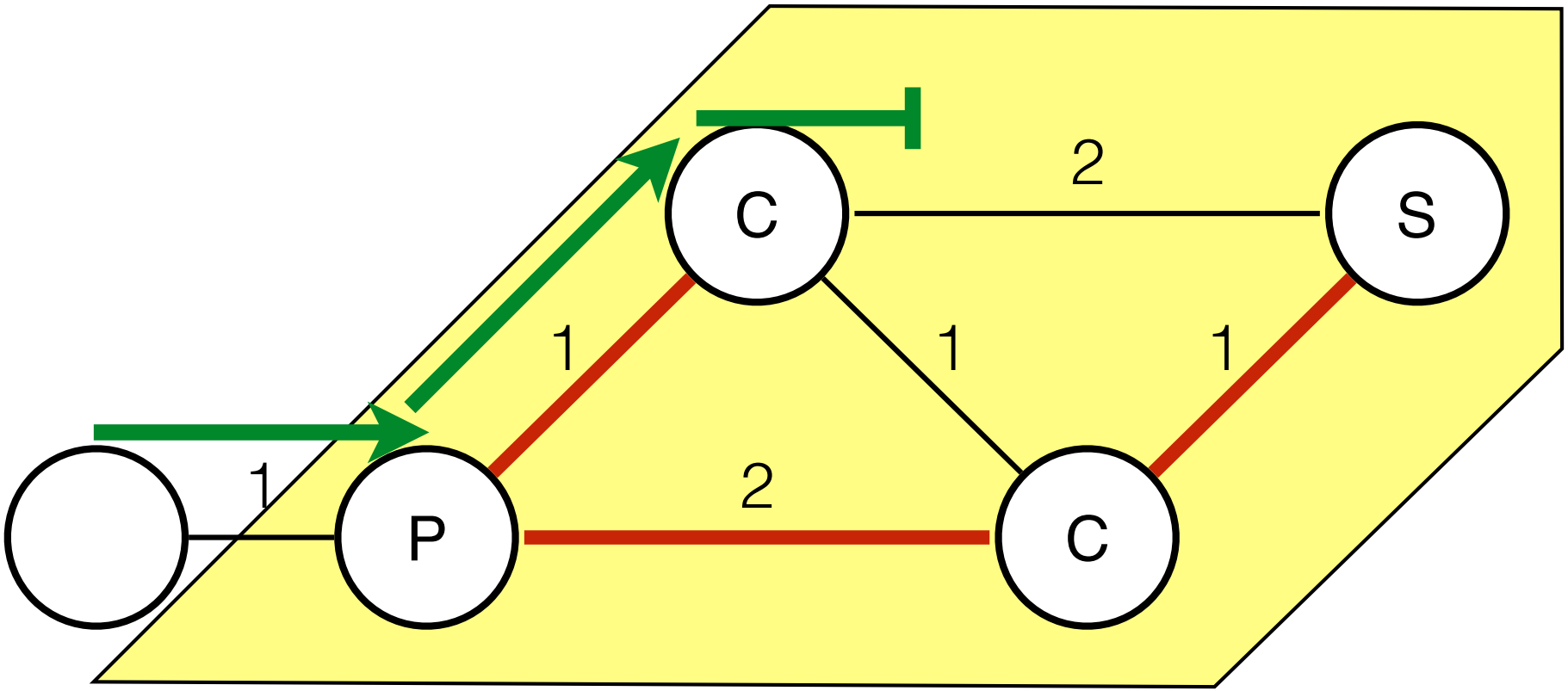
Correctness



Correctness



Correctness



Results

		Time	Expanded	Generated	Open
Graph	A*	3.27	3,382	25,059	229
	Canonical A*	2.93	3,521	3,731	152
Grid	A*	2.36	3,382	25,059	229
	Canonical A*	1.14	3,387	3,386	91

Results

		Time	Expanded	Generated	Open
Graph	A*	3.27	3,382	25,059	229
Graph	Canonical A*	2.93	3,521	3,731	152
Grid	A*	2.36	3,382	25,059	229
Grid	Canonical A*	1.14	3,387	3,386	91

Results

		Time	Expanded	Generated	Open
Graph	A*	3.27	3,382	25,059	229
	Canonical A*	2.93	3,521	3,731	152
Grid	A*	2.36	3,382	25,059	229
	Canonical A*	1.14	3,387	3,386	91

Results

		Time	Expanded	Generated	Open
Graph	A*	3.27	3,382	25,059	229
	Canonical A*	2.93	3,521	3,731	152
Grid	A*	2.36	3,382	25,059	229
	Canonical A*	1.14	3,387	3,386	91

Future Work

- More efficient implementation
  - Time and Memory
- Optimize Choice of Canonical Ordering
- Add JPS Jumping
- Test with existing graph-pruning approaches
- Correctness