Lazy Validation of Experience Graphs

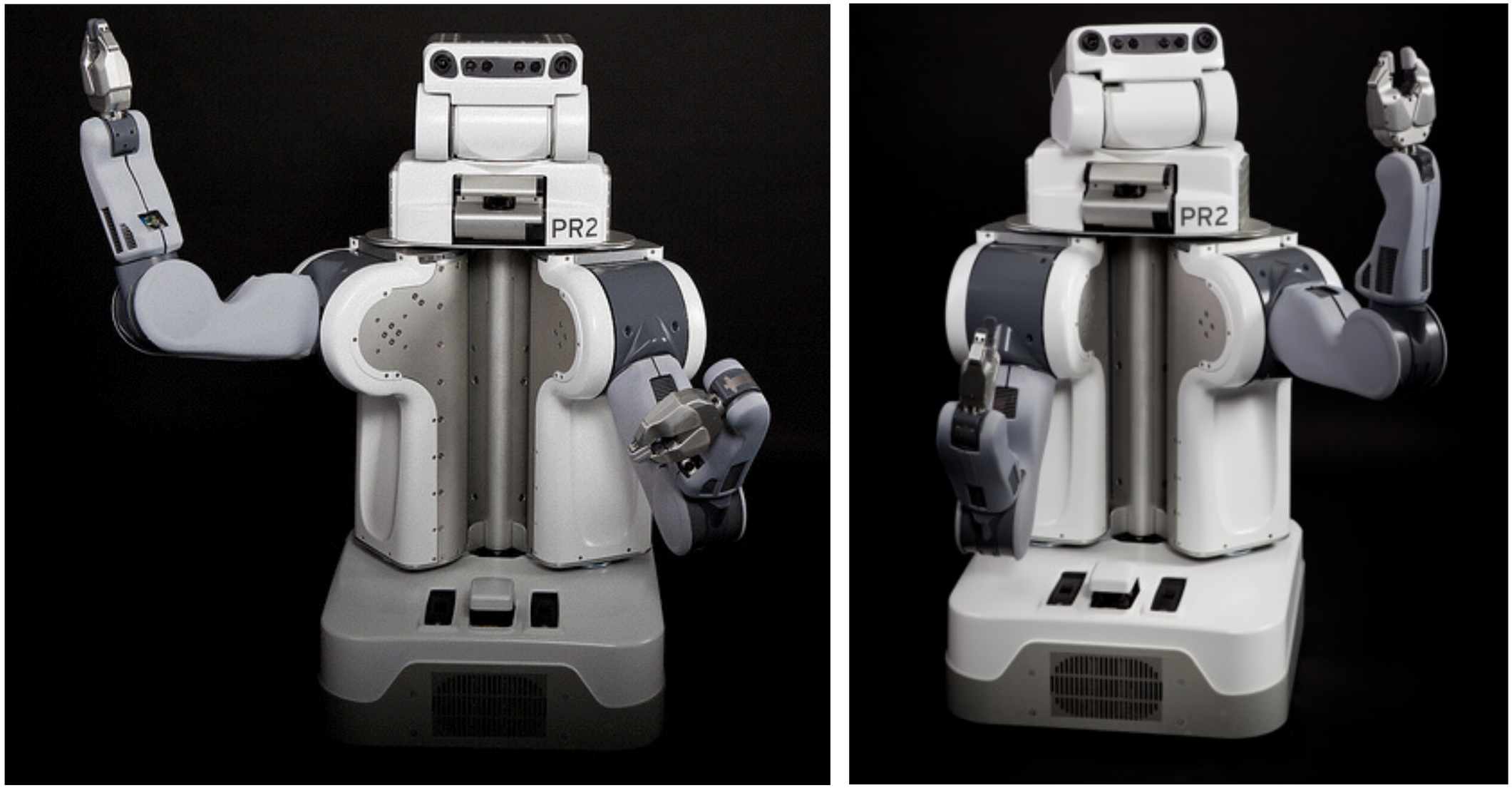
Victor Hwang

Speaking Qualifier

Committee:
Maxim Likhachev (chair)
Siddhartha Srinivasa
Michael Shomin

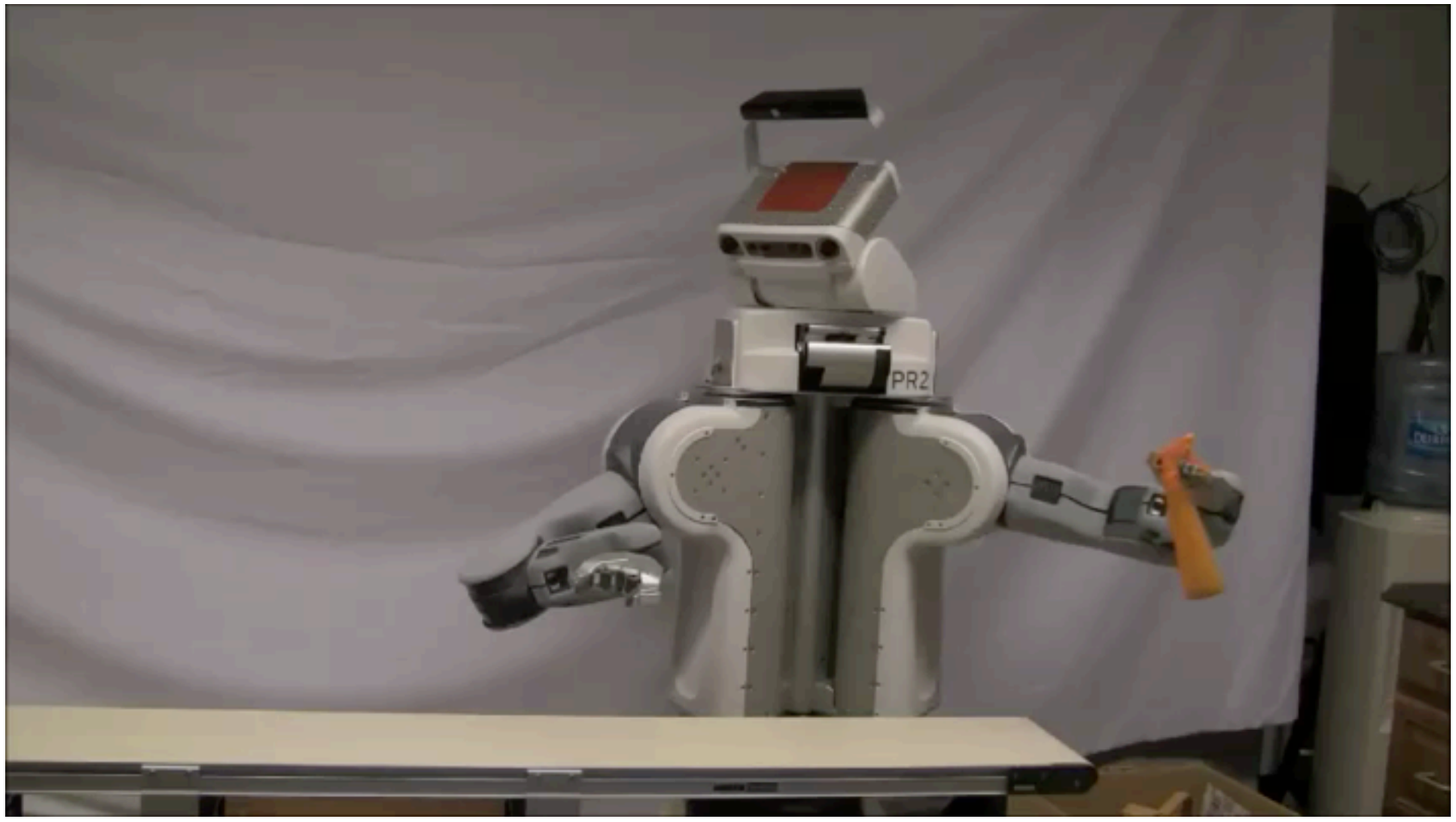
The Motion Planning Problem

- Find a high quality path from one state to another



The Motion Planning Problem

- In many scenarios, high dimensional problems need to be solved quickly with efficient paths



Cowley, et al., 2013

Robots should use prior experience to accelerate planning, even in changing environments

Lazy validation of Experience graphs is a method to efficiently plan with prior experiences in changing environments

Outline

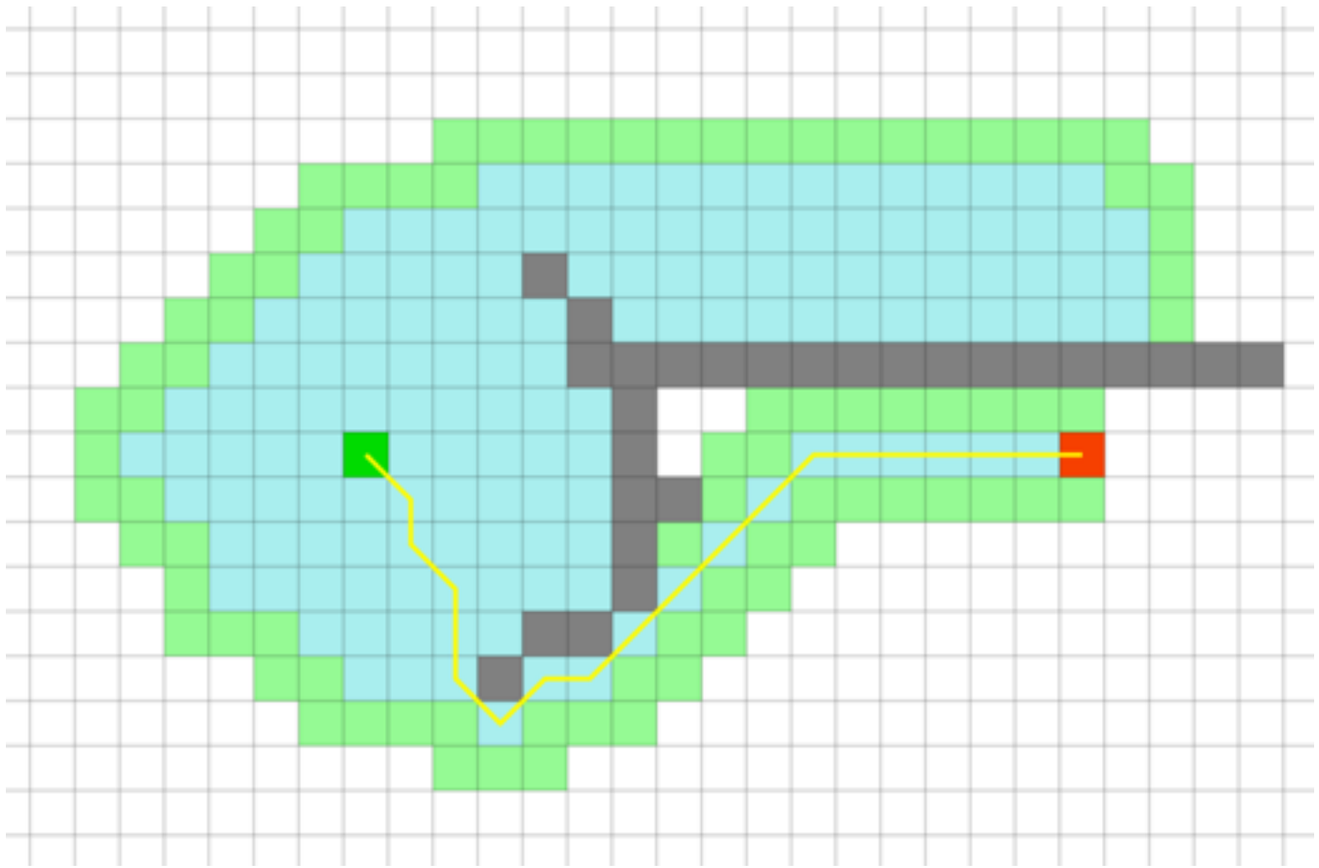
- Related work
- E-Graphs background
- Lazy validation
- Results

Related Work

- Sampling based planners
  - Lazy PRM (Bohlin, et al., 2000)
  - RRTConnect (Kuffner, et al., 2000)
- Trajectory Libraries - (Stolle, et al., 2006)
- Search-based methods
  - Weighted A* (Pohl, 1970)
  - Experience Graphs (Phillips, et al., 2012)

A* Search

- Construct a graph that represents the planning problem, search with A*
- Quality of the search-focusing heuristic function is extremely important



A* Search

- Construct a graph that represents the planning problem, search with A*
- Quality of the search-focusing heuristic function is extremely important

Experience Graphs

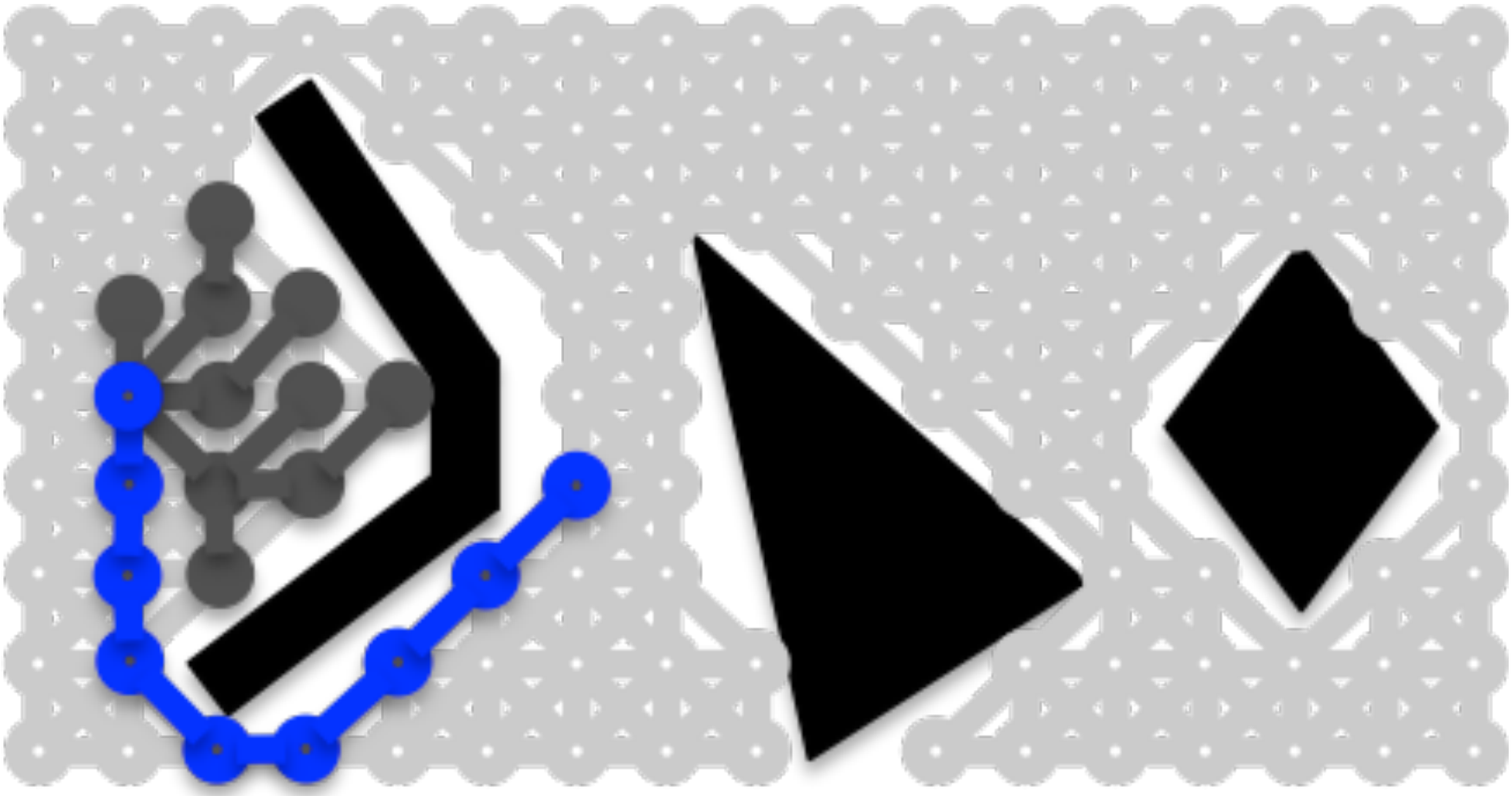
- Augments A* search with past experiences to improve performance for similar tasks

Experience Graphs

- Store states and edges of prior planning episodes in a subgraph of the original graph
- Create a heuristic that guides the search towards states on the E-Graph

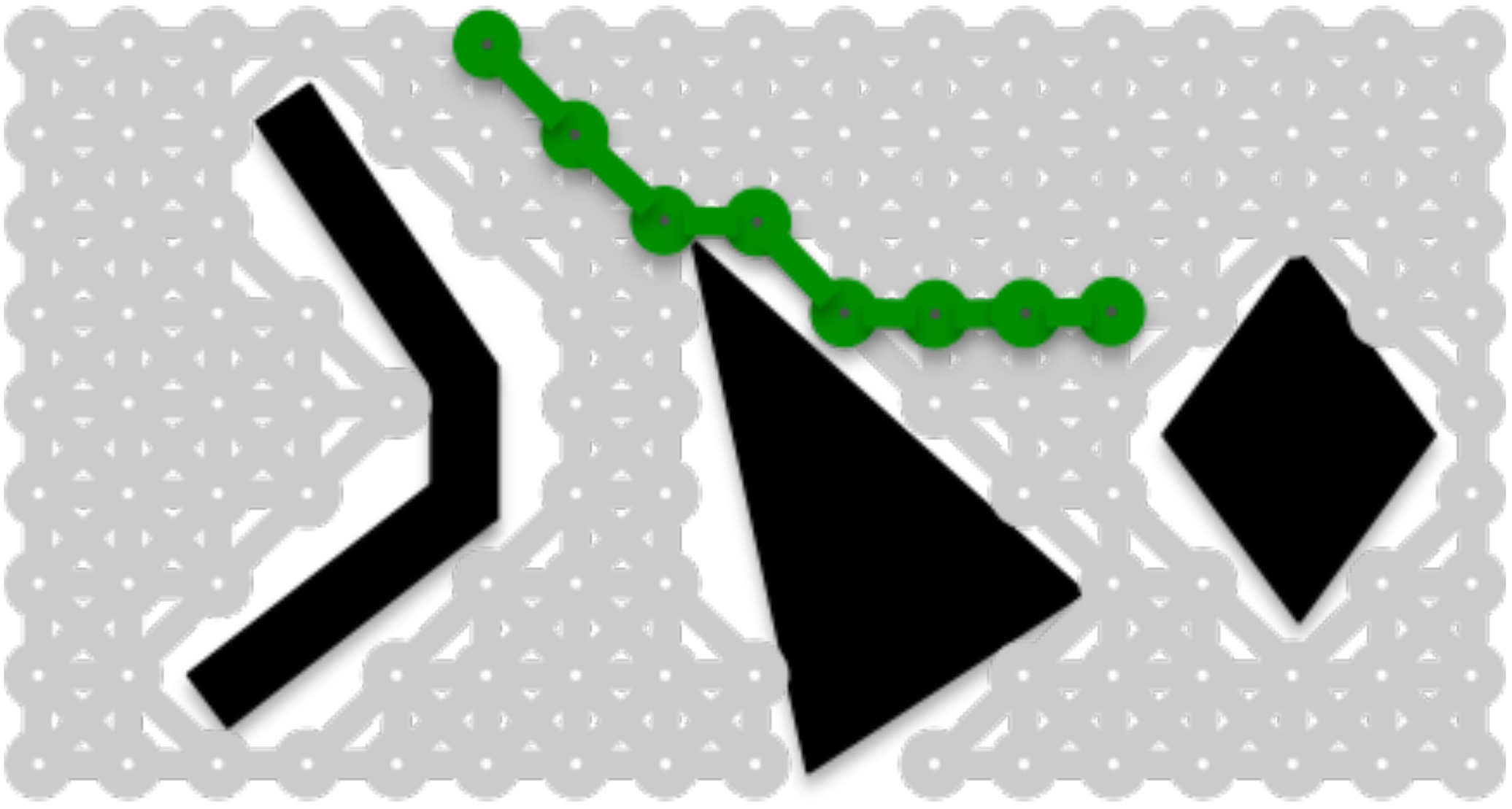
E-Graphs

A new planning scenario



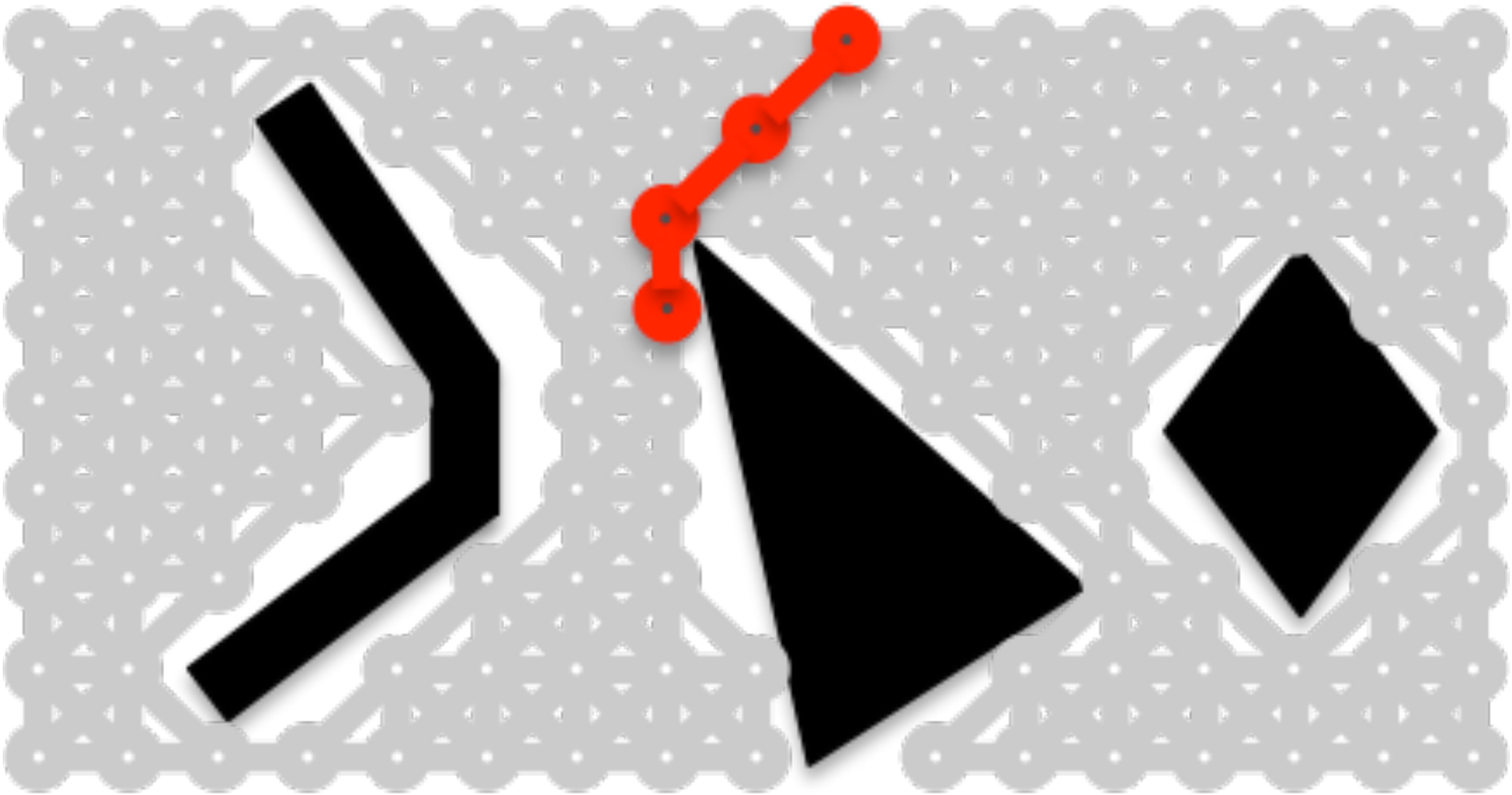
E-Graphs

A second query



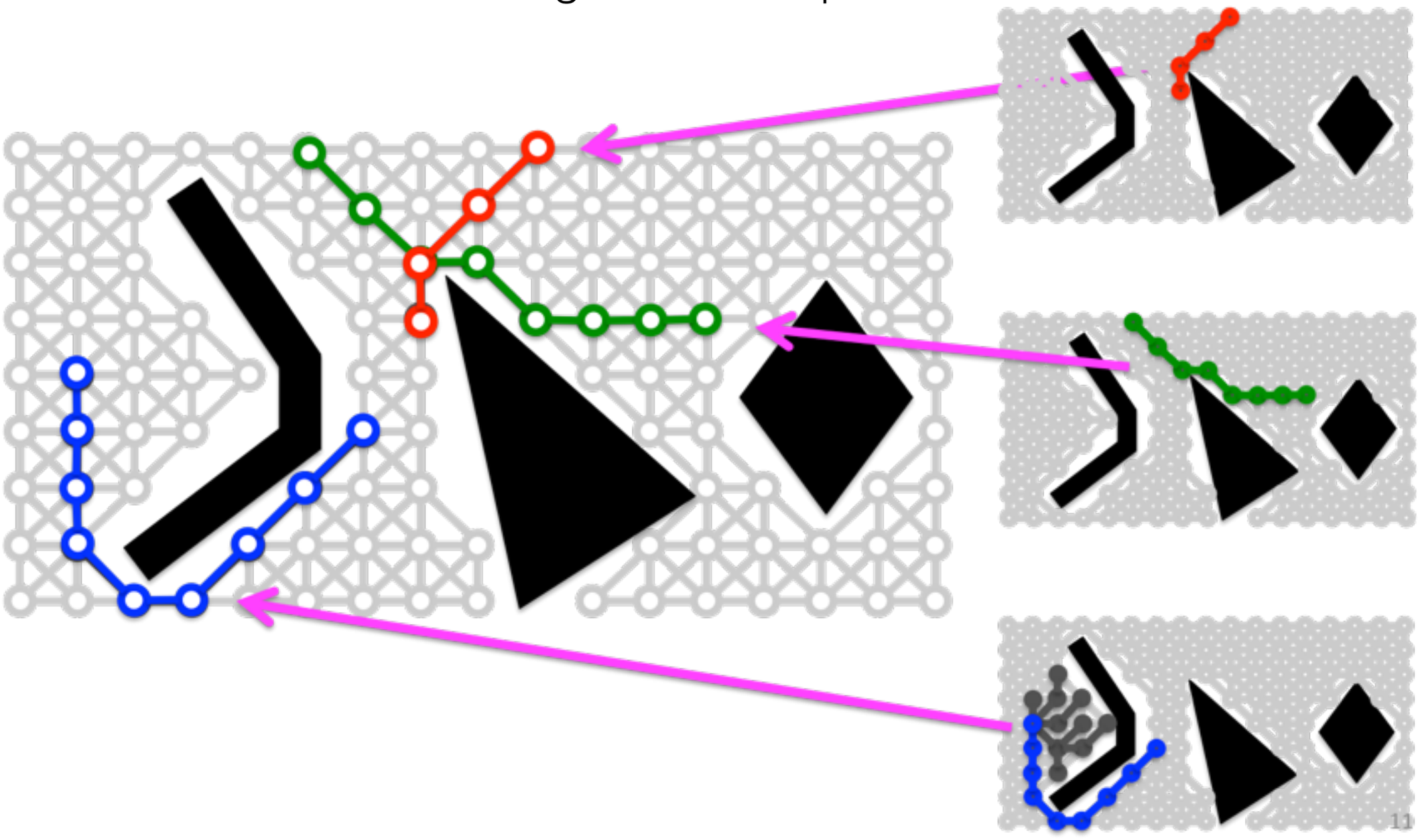
E-Graphs

…a third query



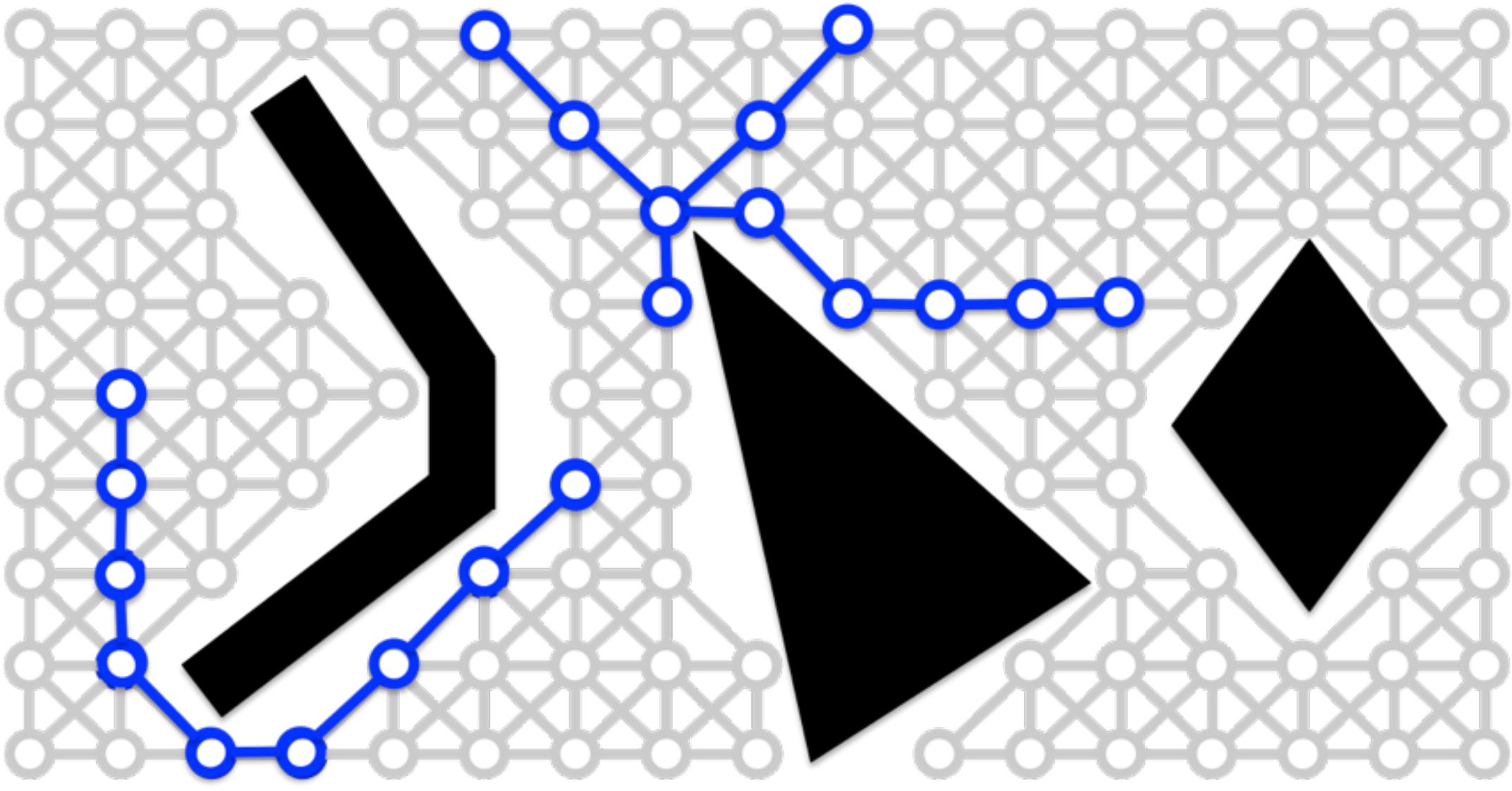
E-Graphs

Merge all three queries



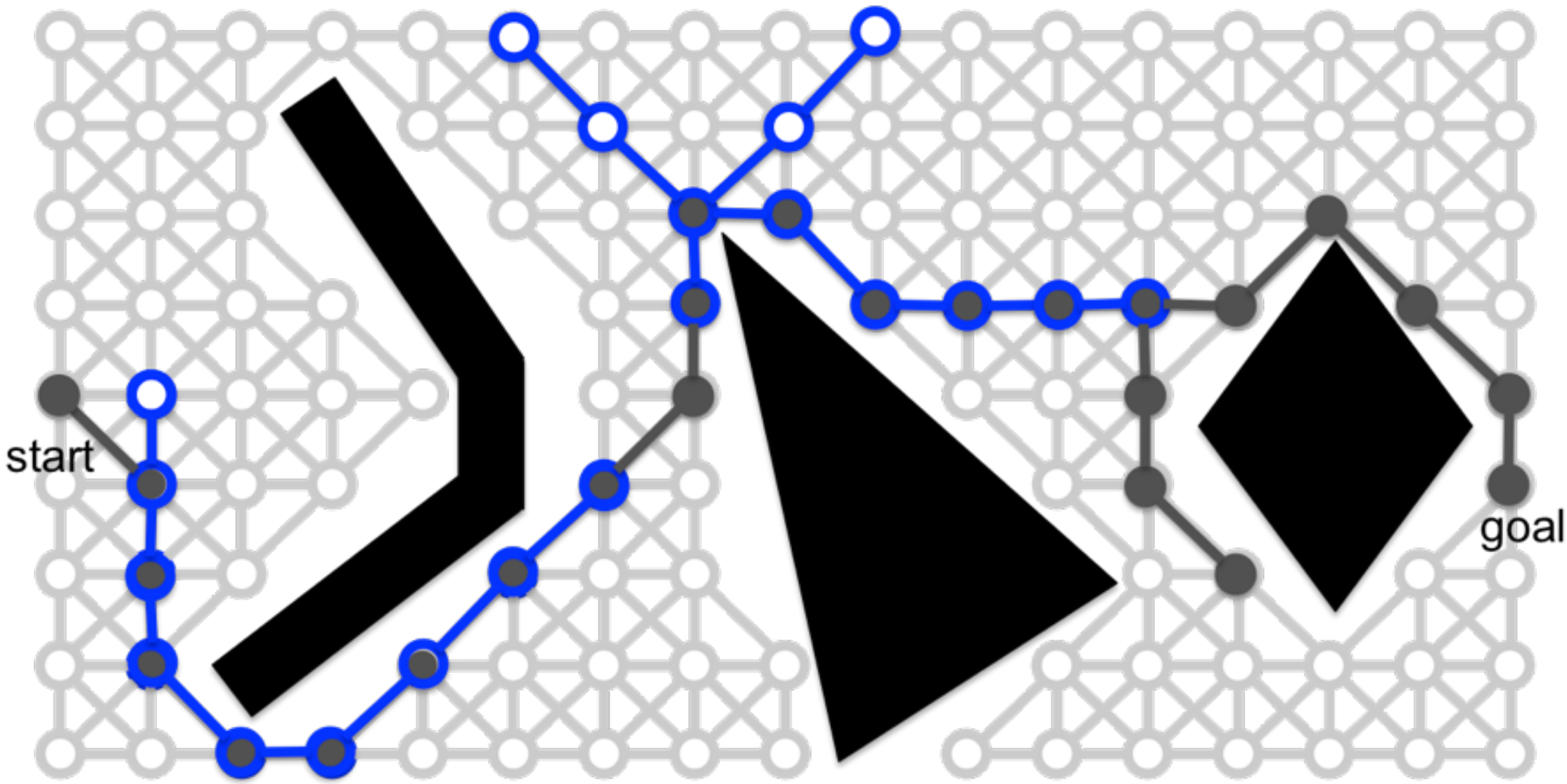
E-Graphs

The complete E-Graph after we merge all three trajectories



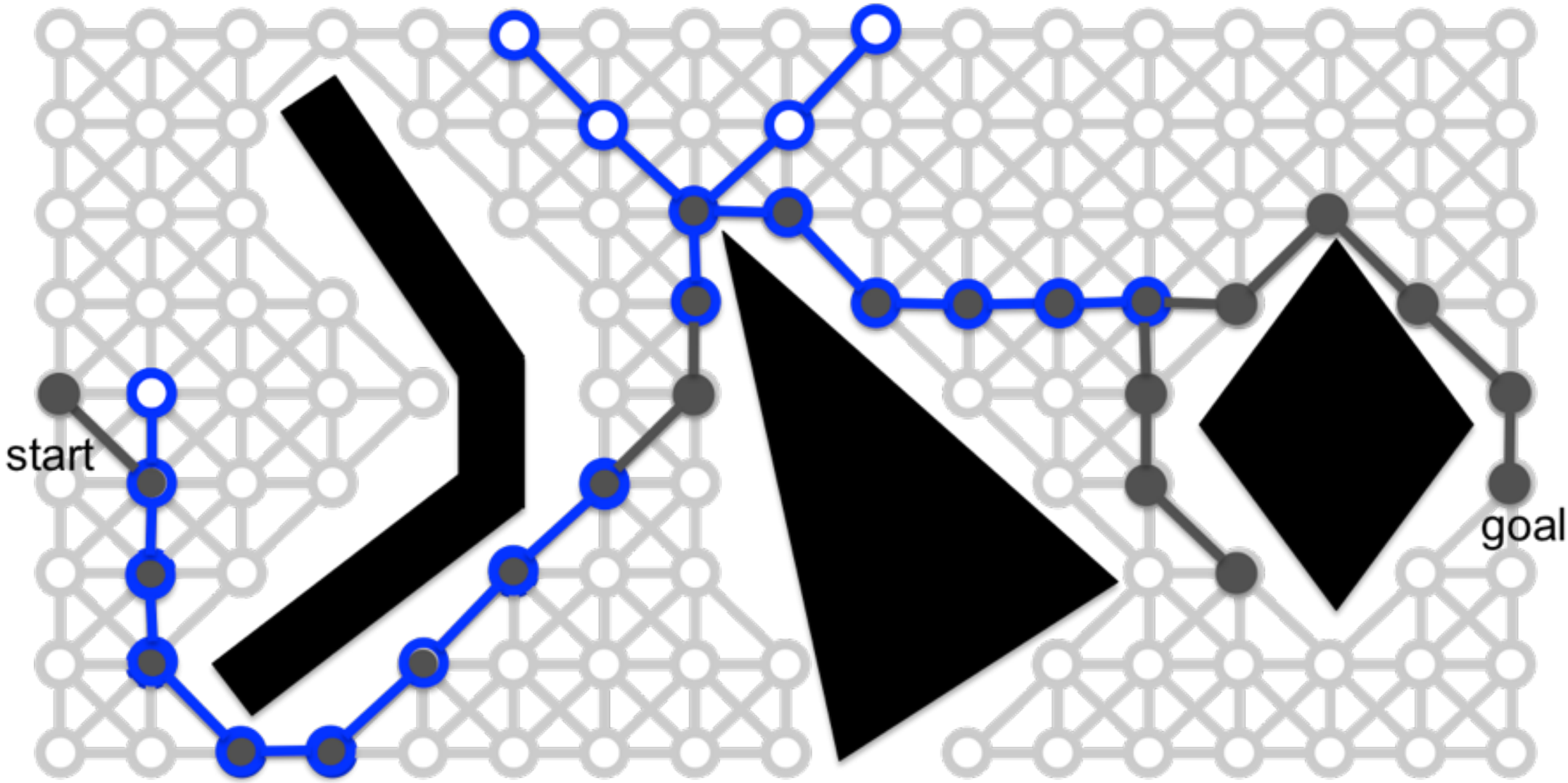
E-Graphs

- Using E-Graph now expands fewer states



E-Graphs

- E-Graph planner complete with respect to the original graph
- E-Graph planner maintains guarantees on solution cost



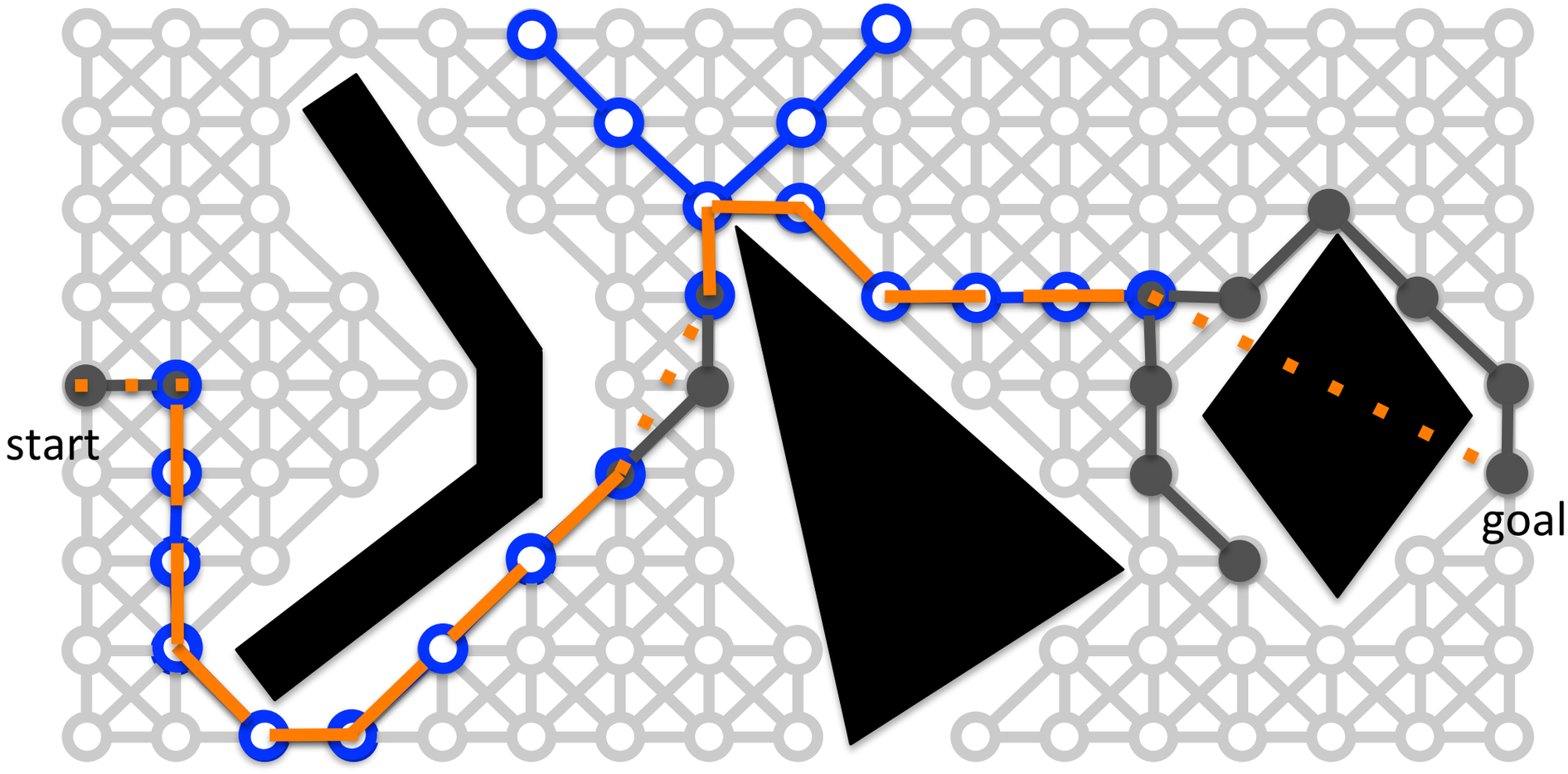
E-Graph Heuristic

- One standard heuristic
  - Cost of shortest path for an “easier” version of the problem (3D heuristic space)
- E-Graph heuristic
  - Cost of shortest path where the path includes 2 kinds of edges
    - Heuristic space edges
    - E-Graph edges

E-Graph Heuristic

$$h^{\mathcal{E}}(s_0) = \min_{\pi} \sum_{i=0}^{N-1} min\{\varepsilon^{\mathcal{E}} h^{G}(s_i, s_{i+1}), c^{\mathcal{E}}(s_i, s_{i+1})\}$$

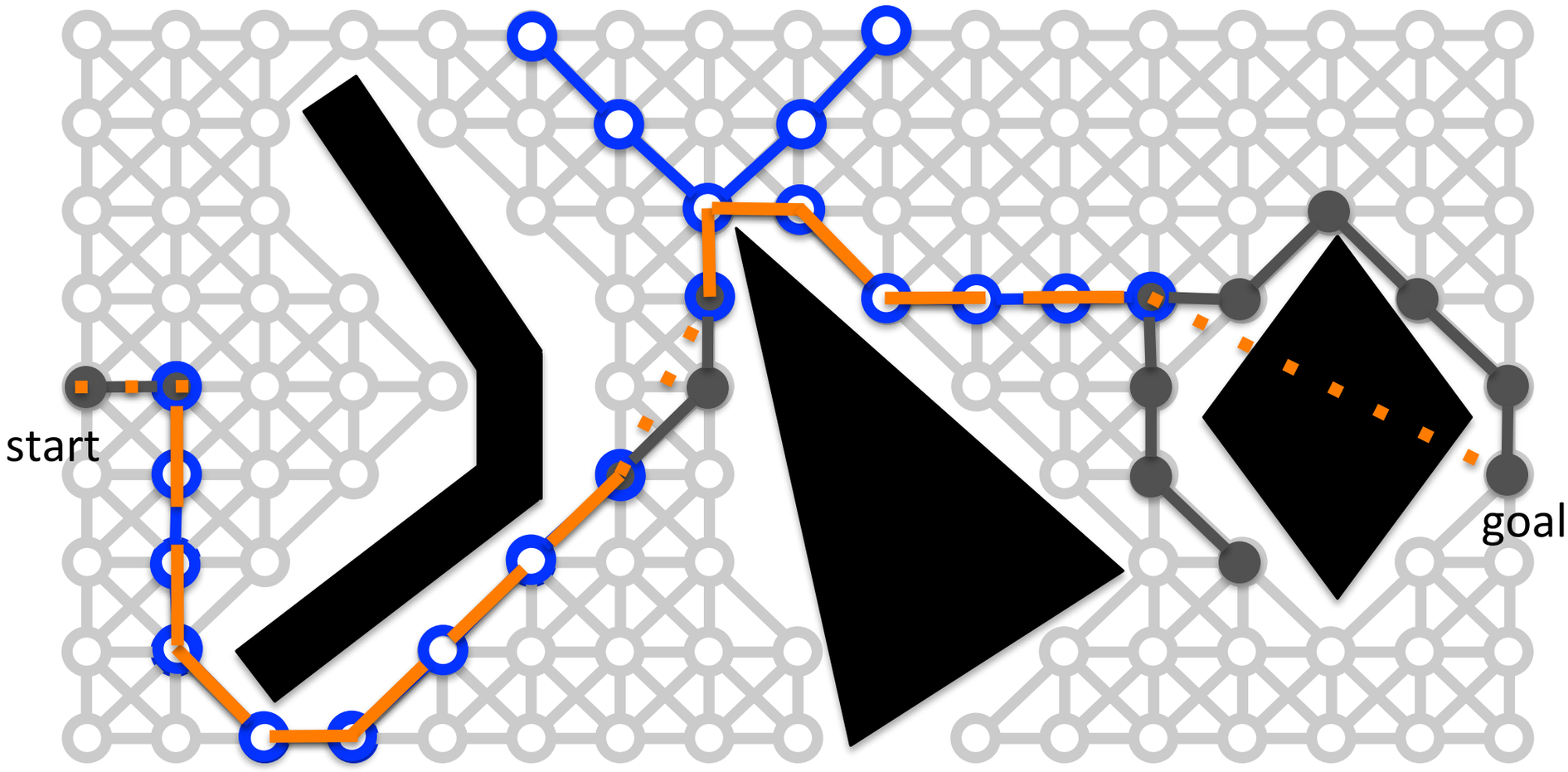
Phillips, et al., 2012



E-Graph Heuristic

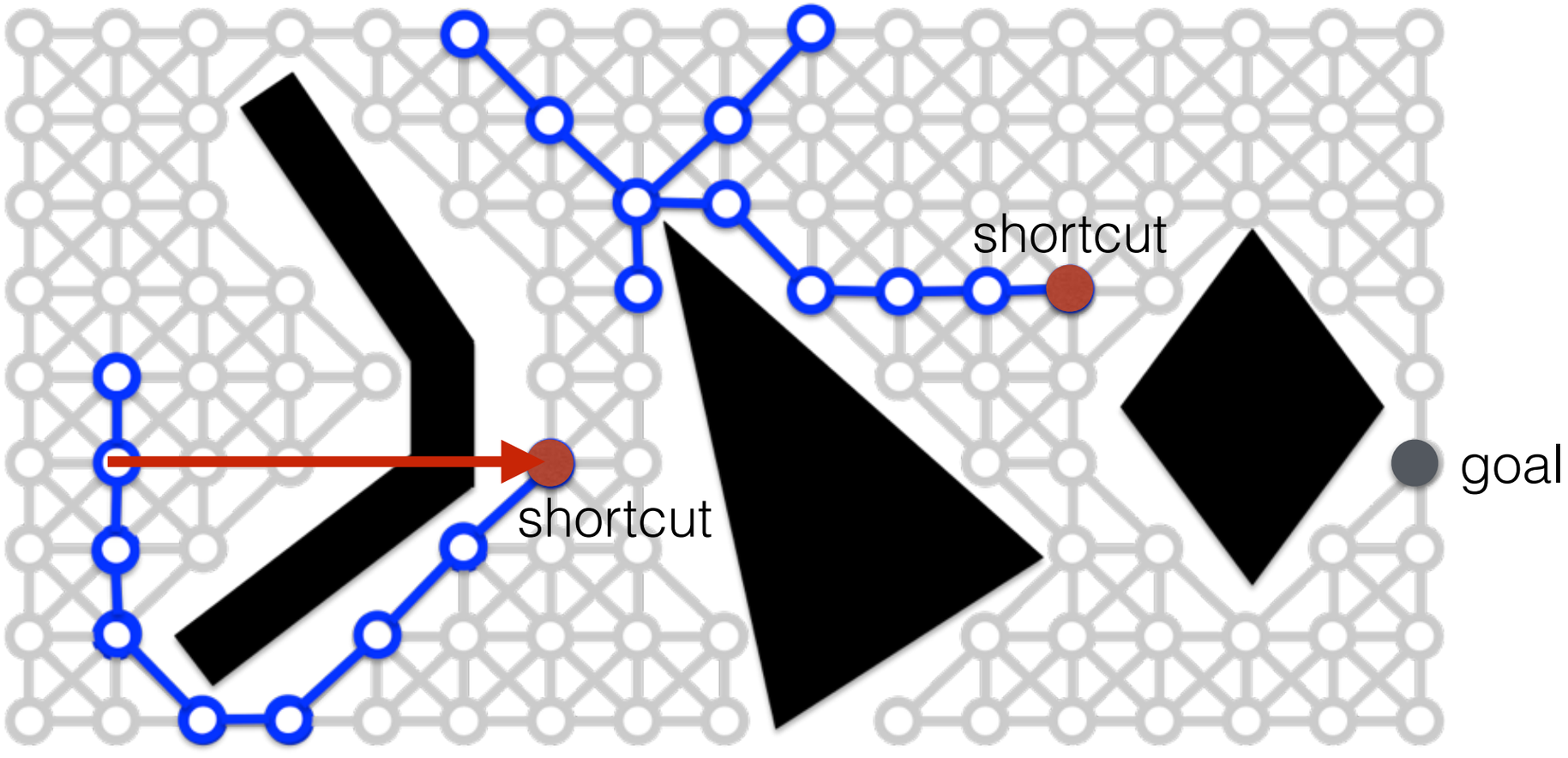
Heuristic(s) $= \min_{\pi} \sum_{i=0}^{N-1} min\{$ ε * Heuristic edge cost , E-Graph edge cost $\}$

Phillips, et al., 2012



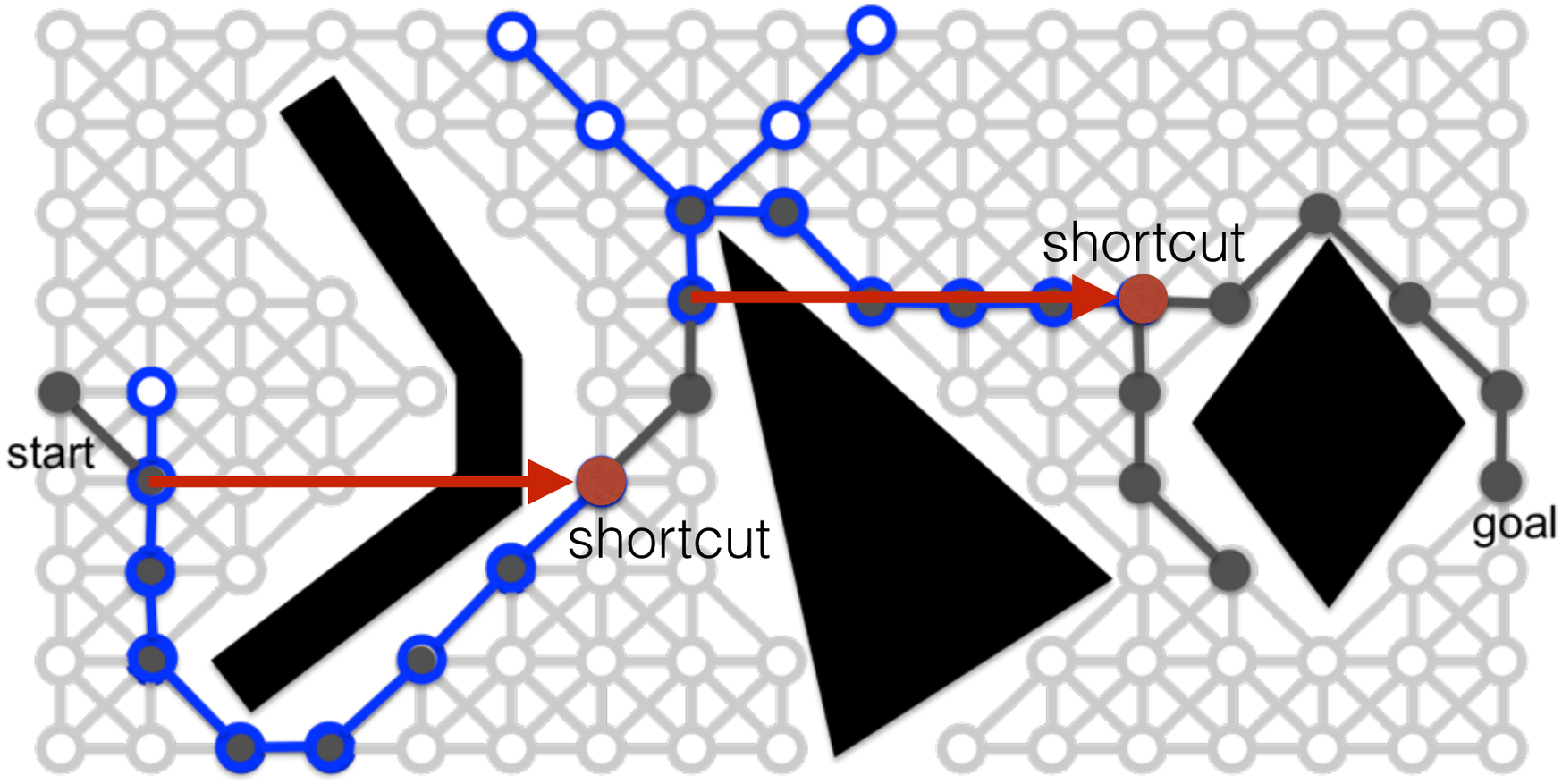
E-Graph Shortcut

Avoids expanding states by generating successors closest (in heuristic value) to the goal on a component



E-Graph Shortcut

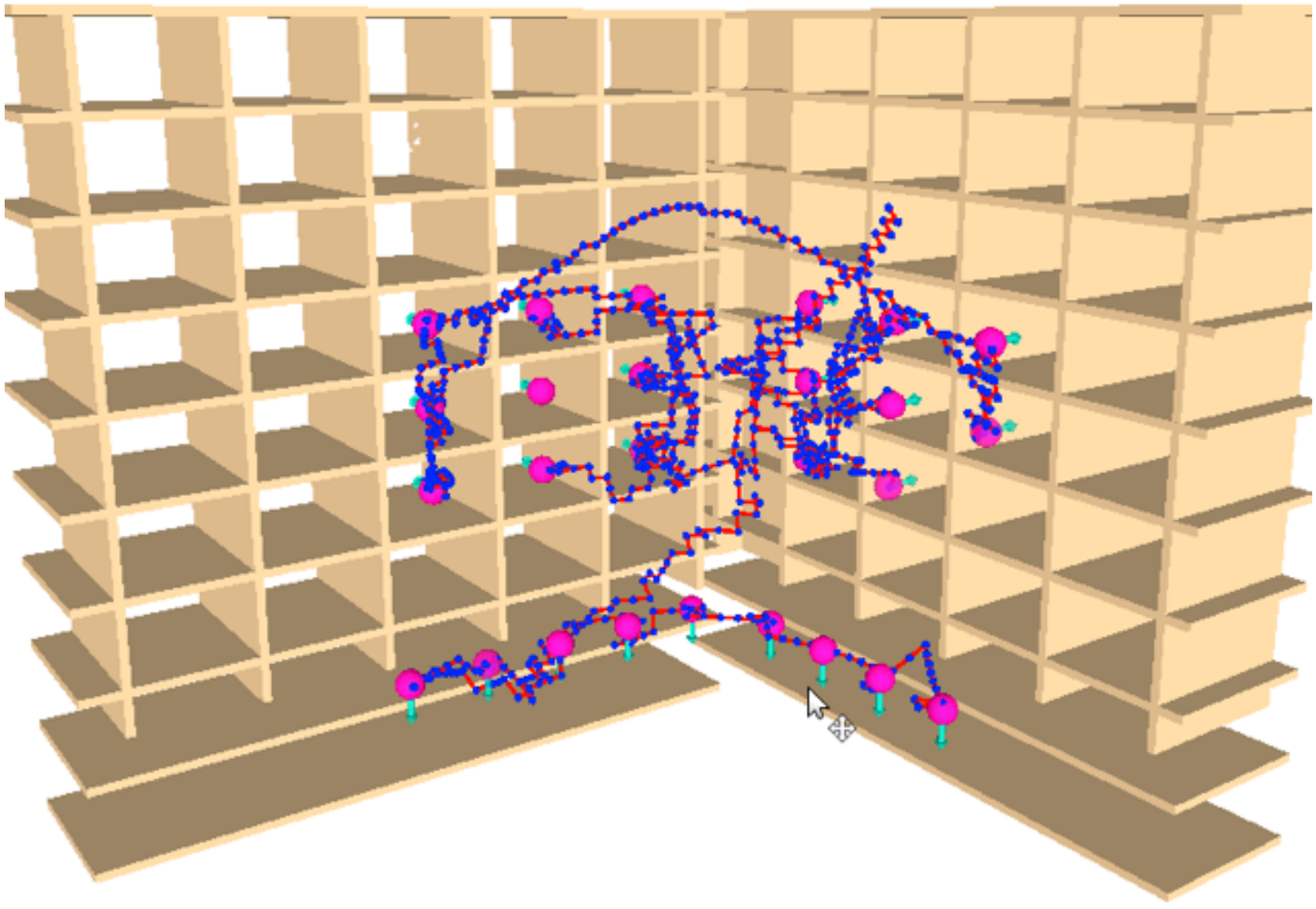
Avoids expanding states by generating successors closest (in heuristic value) to the goal on a component



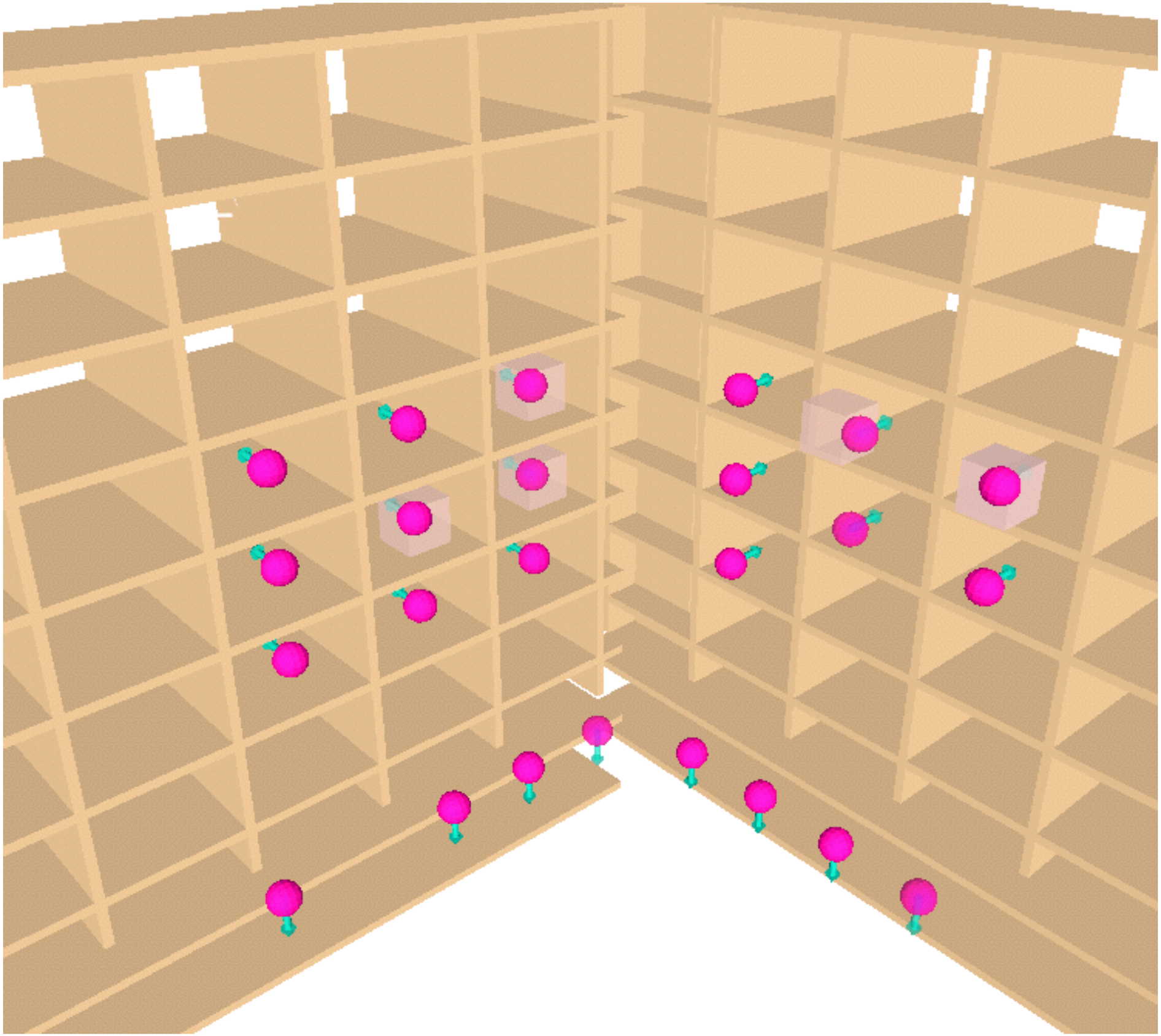
The E-Graph framework uses a heuristic to guide a search towards previously planned paths that are relevant

Single arm manipulation (7D) in a mailroom environment

With enough experience, planning between two points is fast

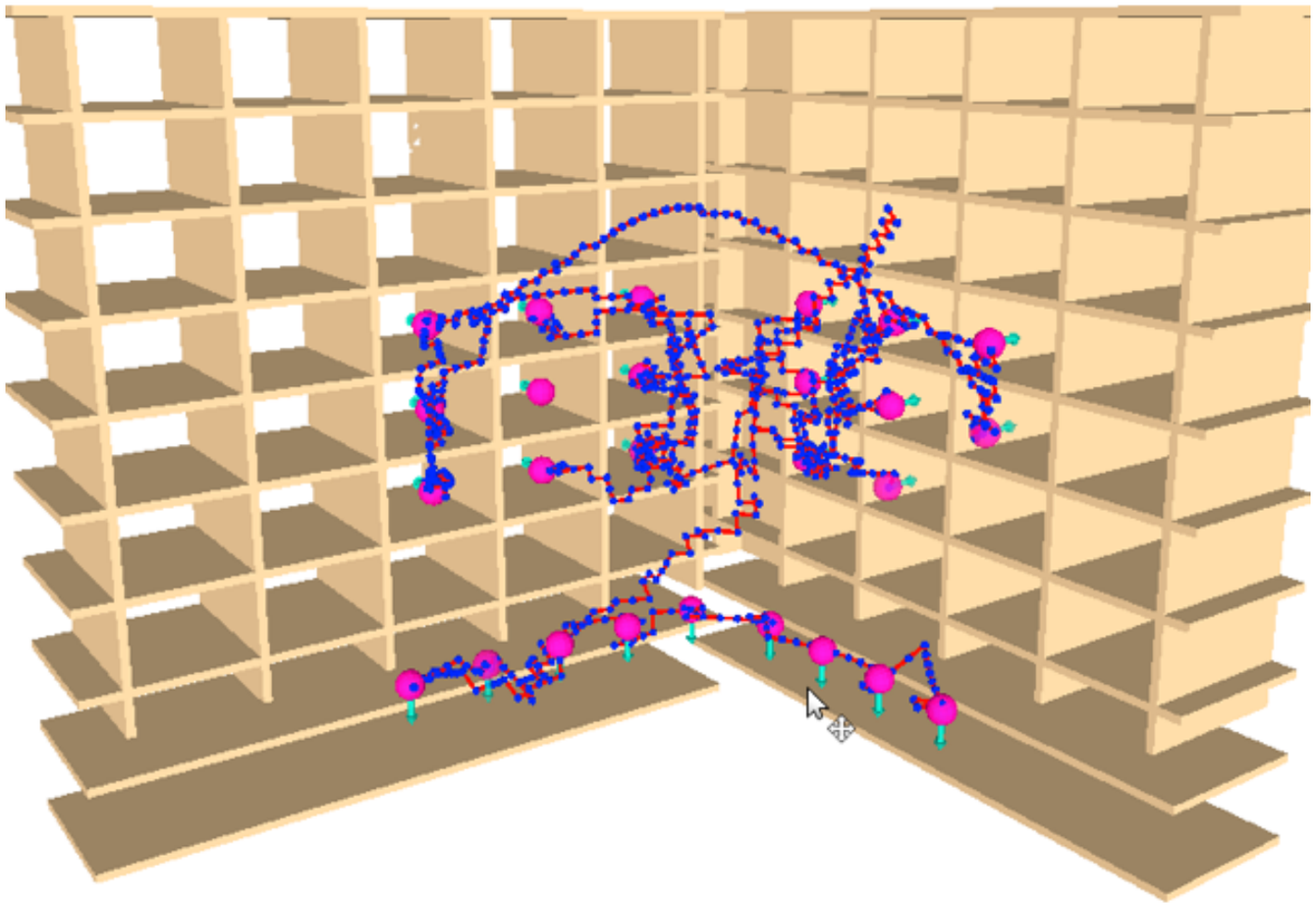


How is this framework affected by small changes in the environment?

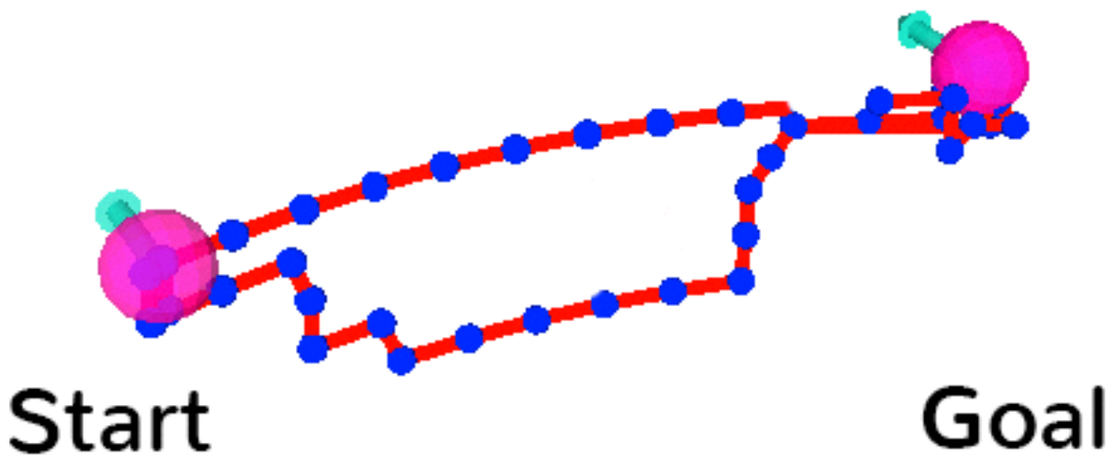


- With small changes in the environment, we need to ensure our experience is valid

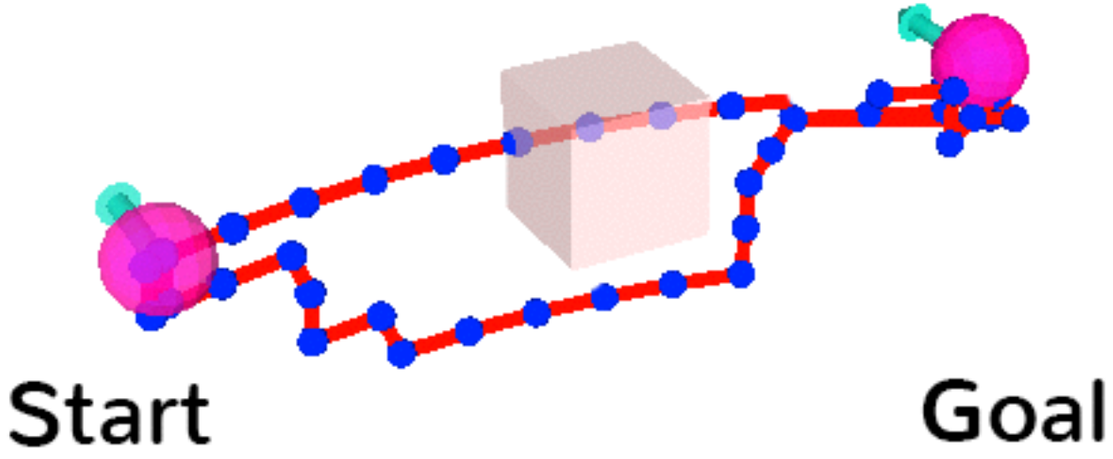
- We could fully validate the E-Graph for every planning request, but this is slow



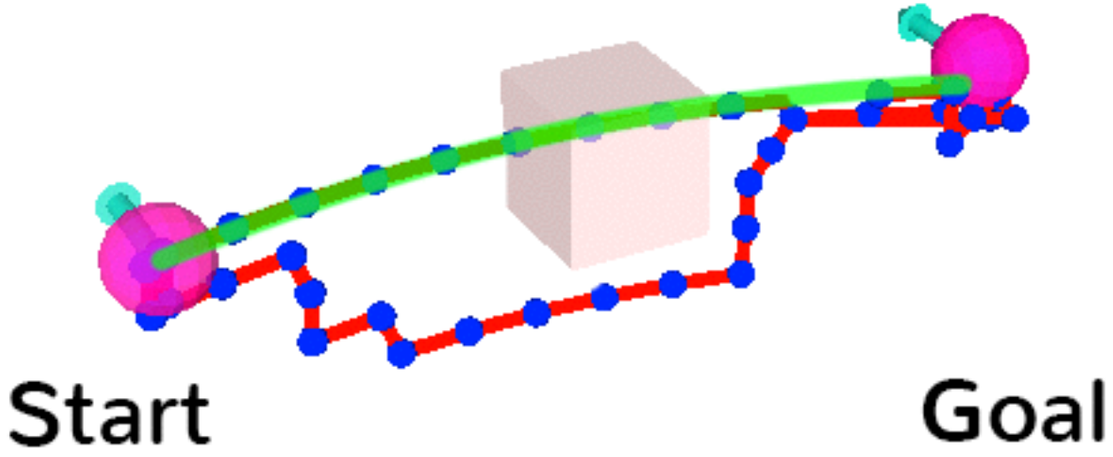
Lazy validation: Only validate E-Graph edges that lie along a solution path (rather than the entire E-Graph)



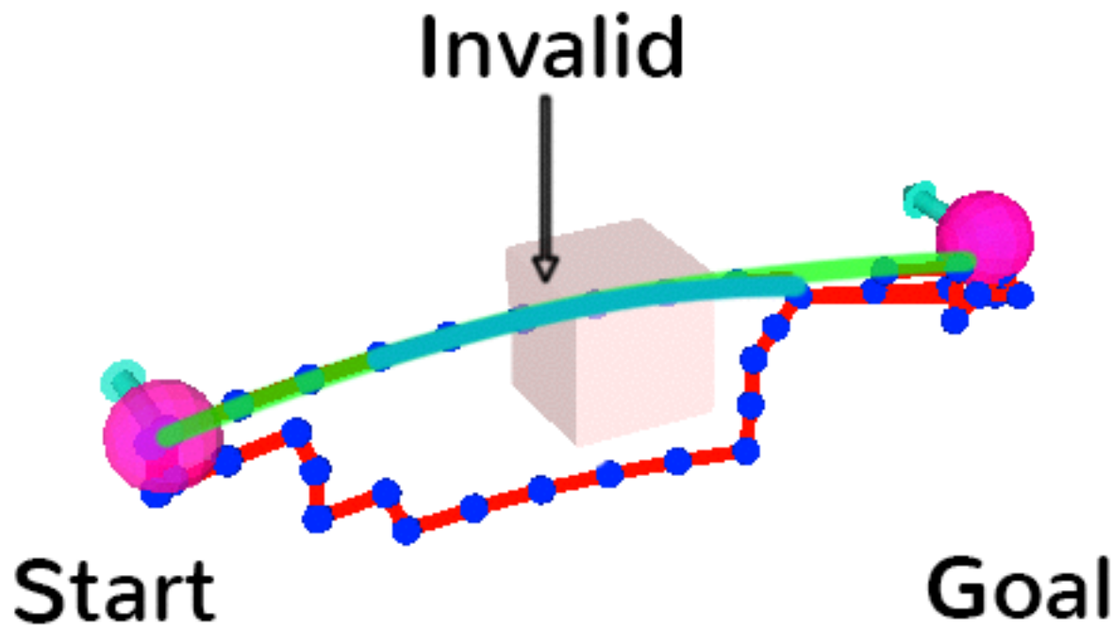
Initial E-Graph



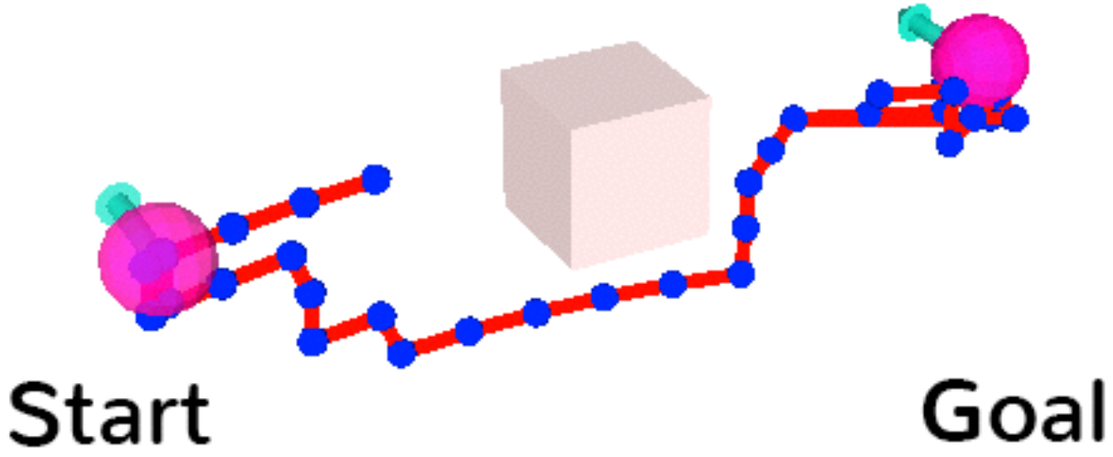
Environment changes



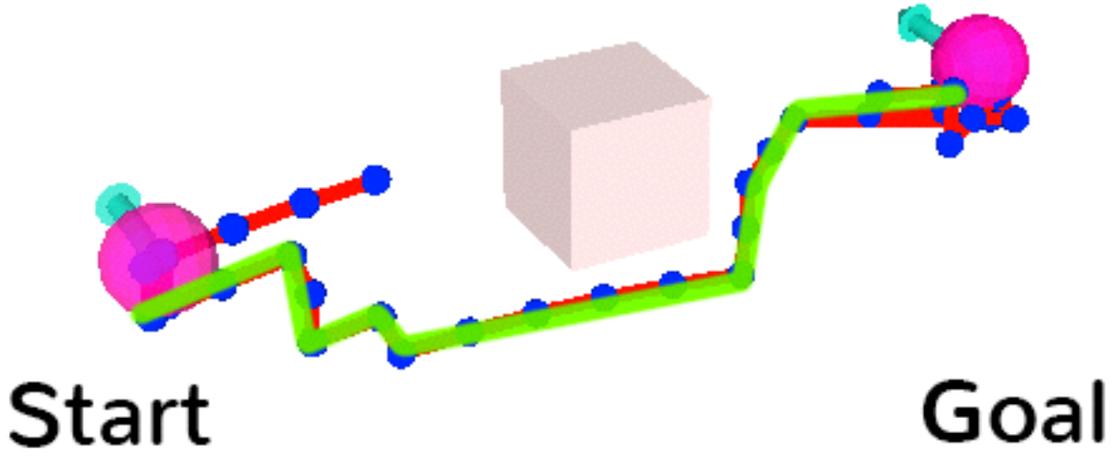
Generate a plan



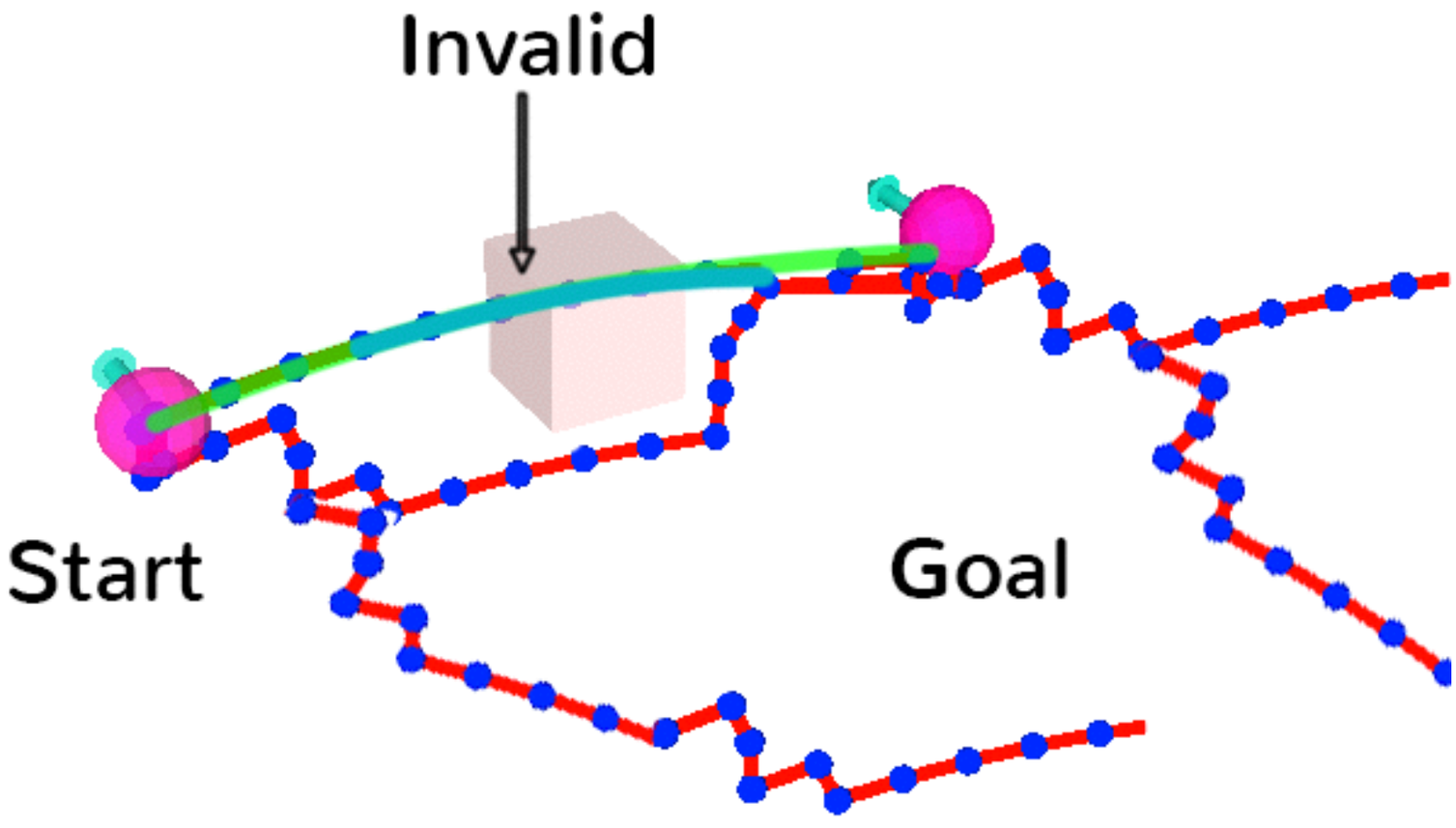
Validate the E-Graph Edges



Update E-Graph edges

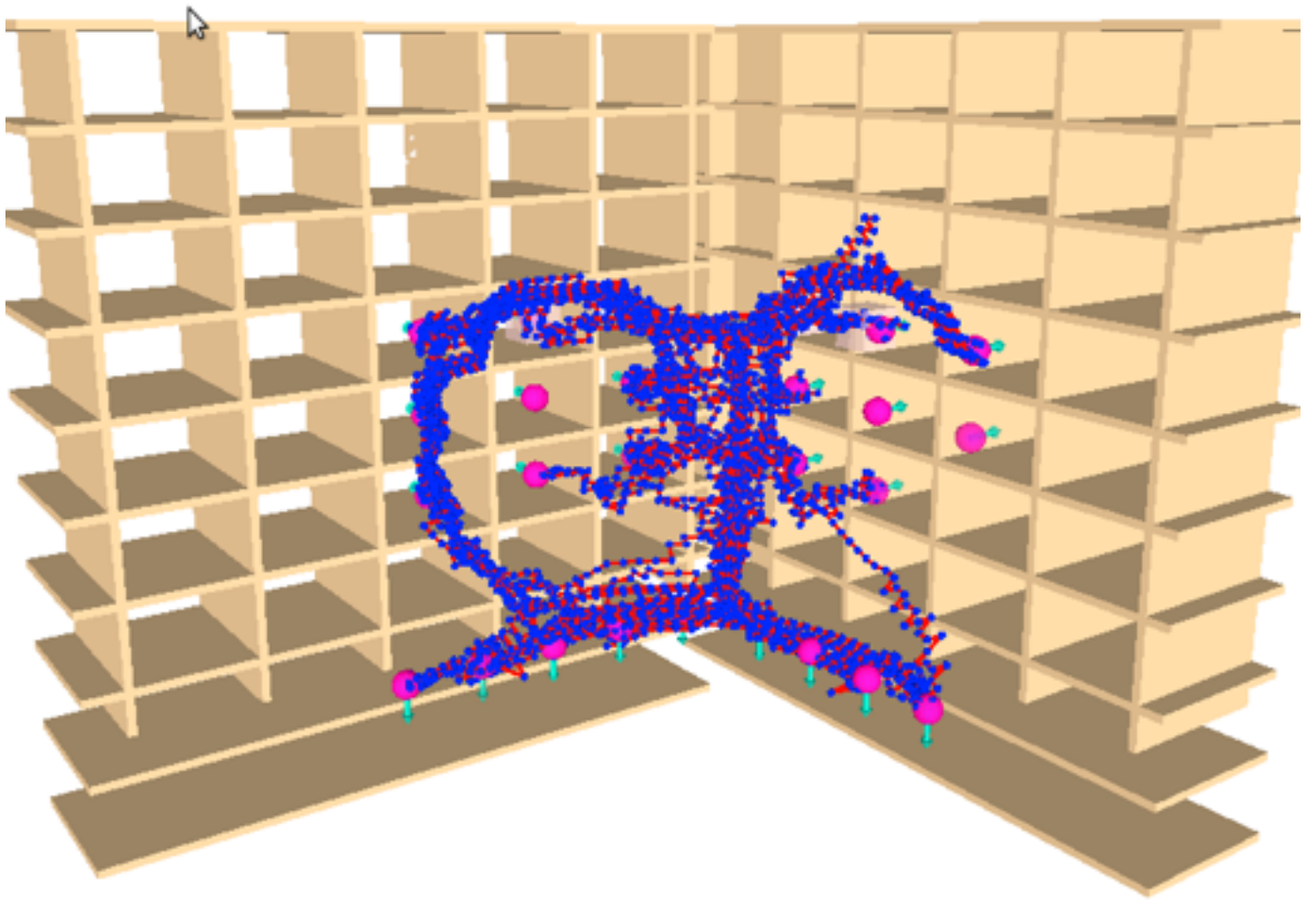


Replan


Invalid
Start
Goal

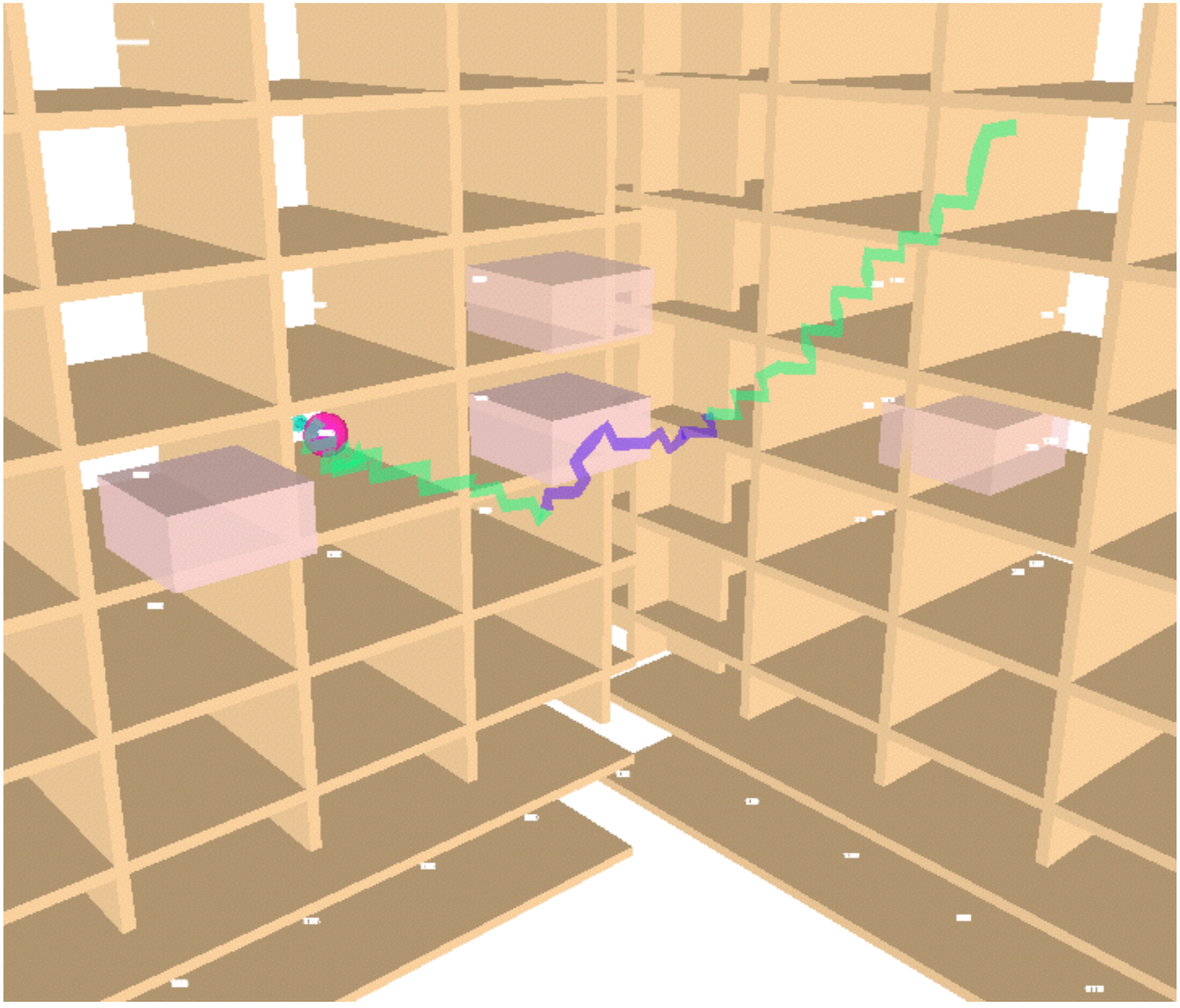
- Lazy validation of paths:
  - Generate a plan
  - Validate the E-Graph edges in the plan
    - If invalid, update the corresponding edges in E-Graph
  - Feedback path and replan

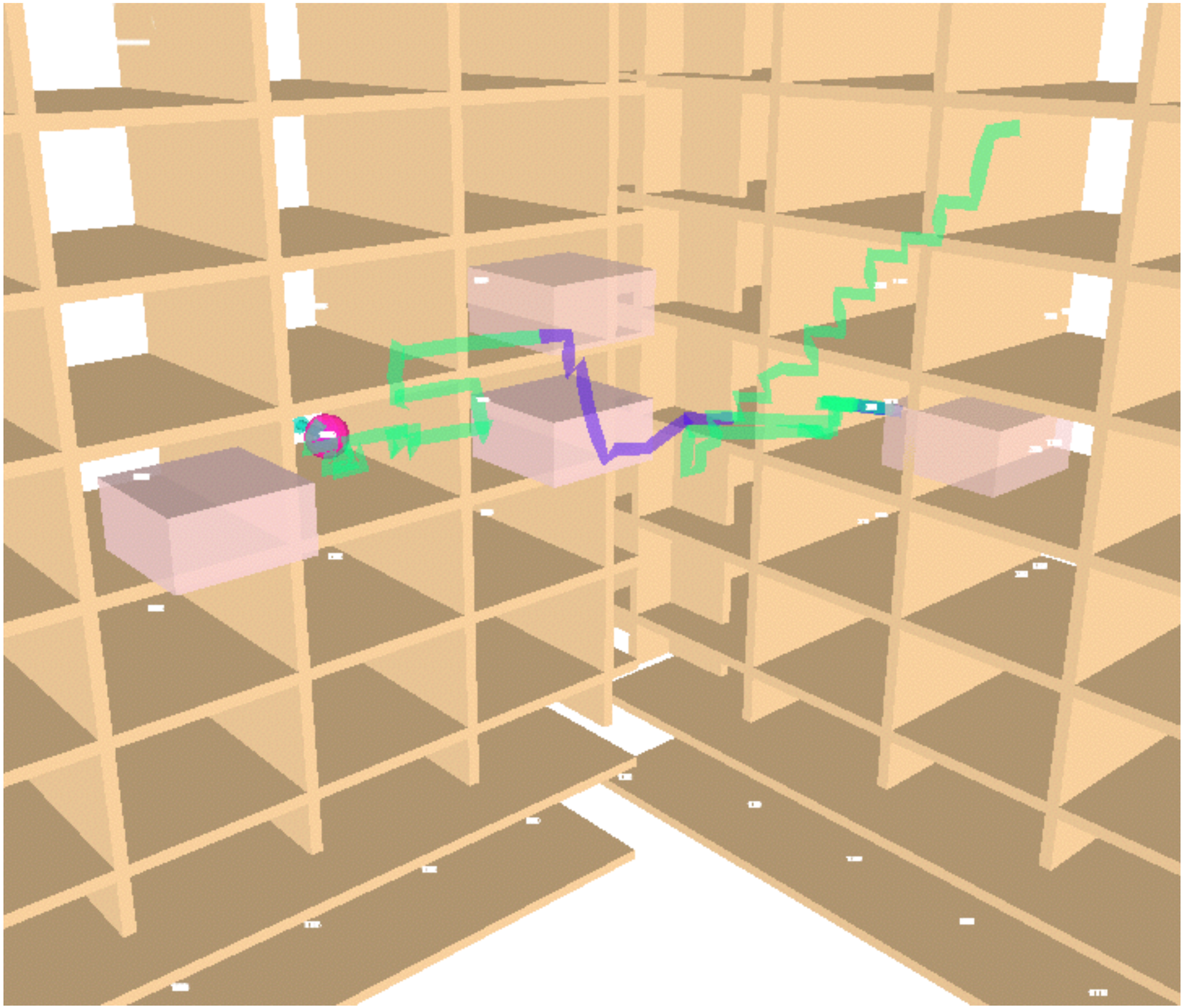
- With lazy validation, we avoid validating an E-Graph with ~4000 vertices
- Because the E-Graph is dense, replans are fast (0.1 sec)

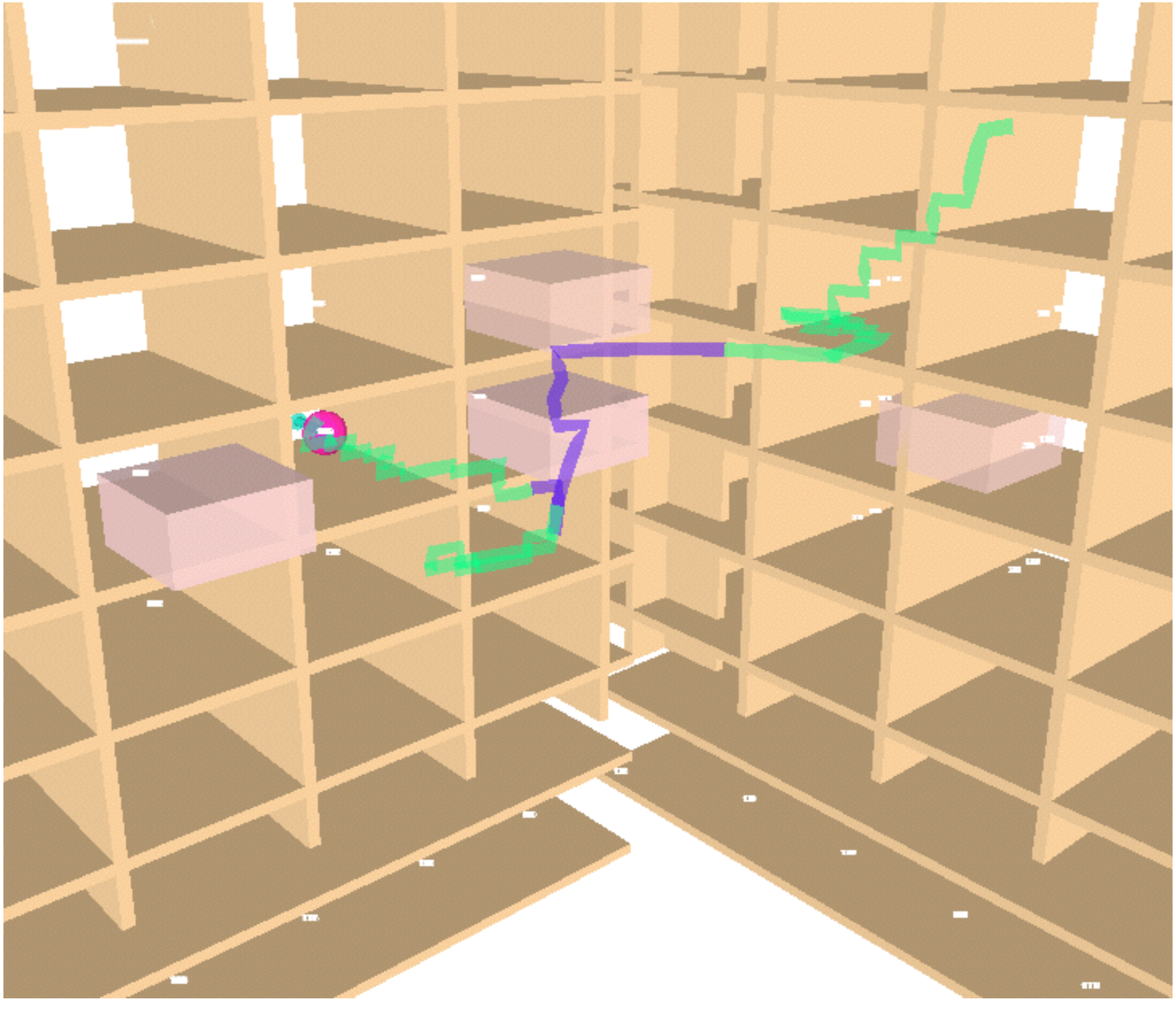


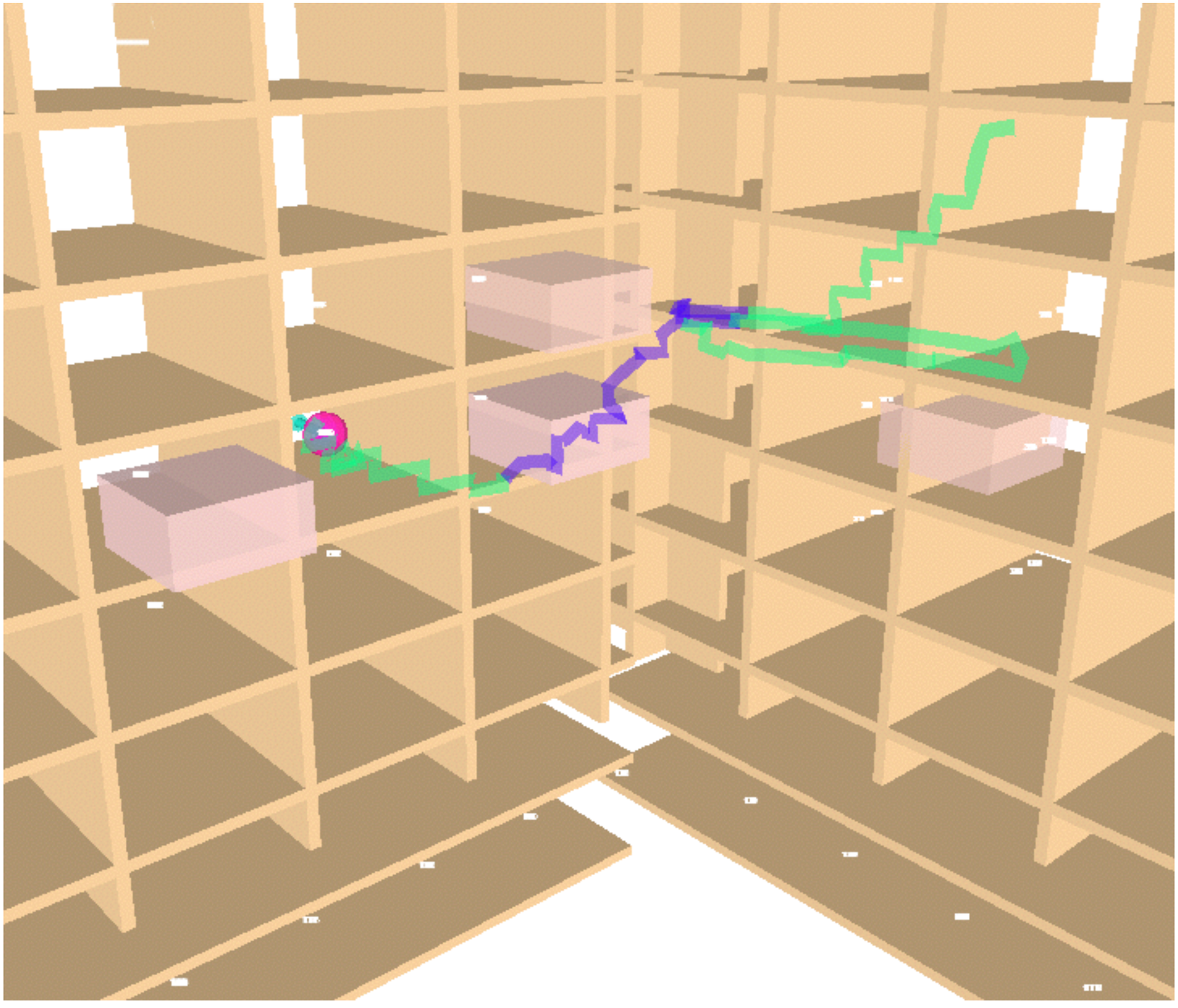
Experimental Setup

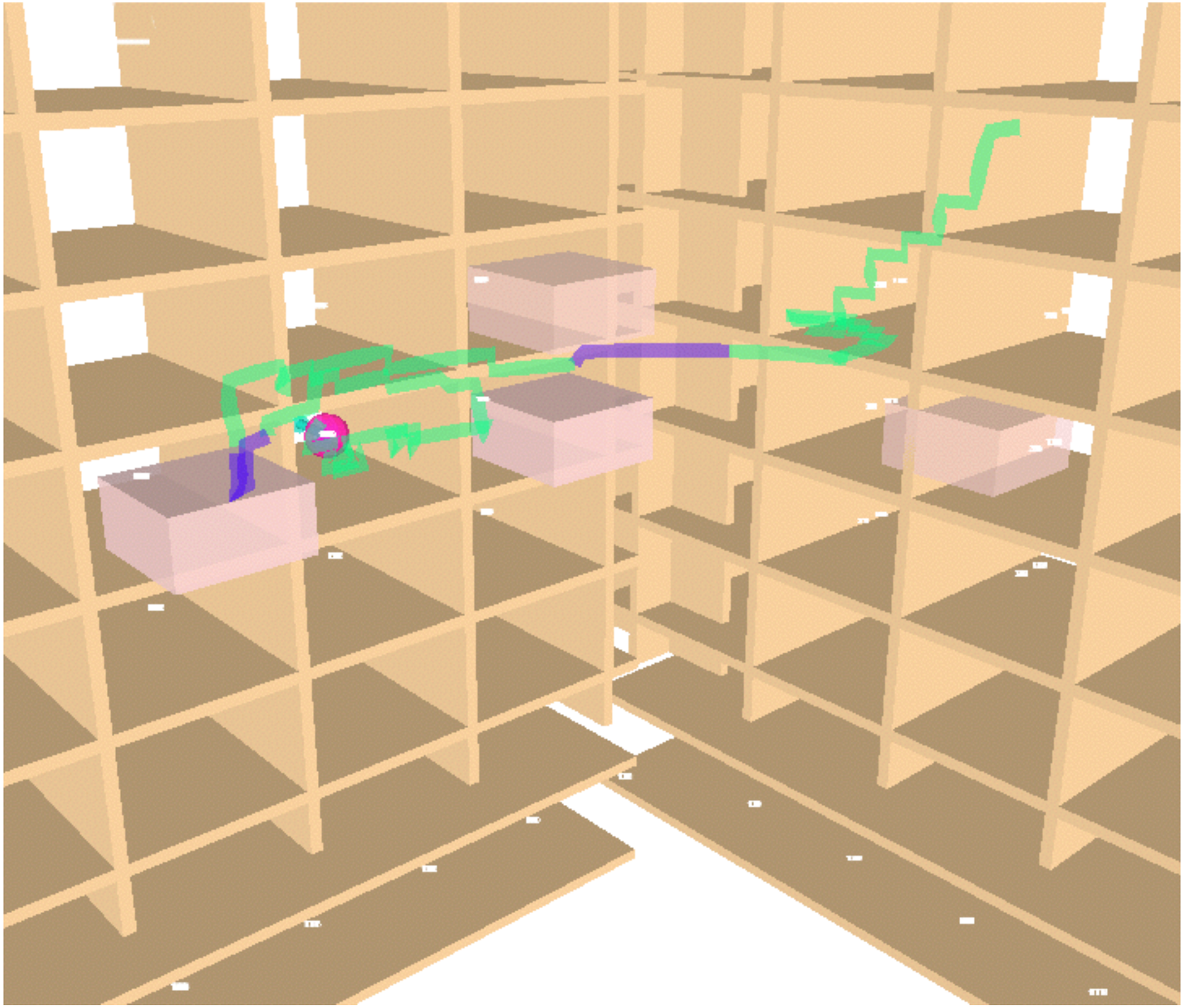
- Cubby scenario
  - 200 trials between 18 cubby hole locations to build E-Graph
  - 80 trials with 6 obstacles randomly placed in cubbyholes

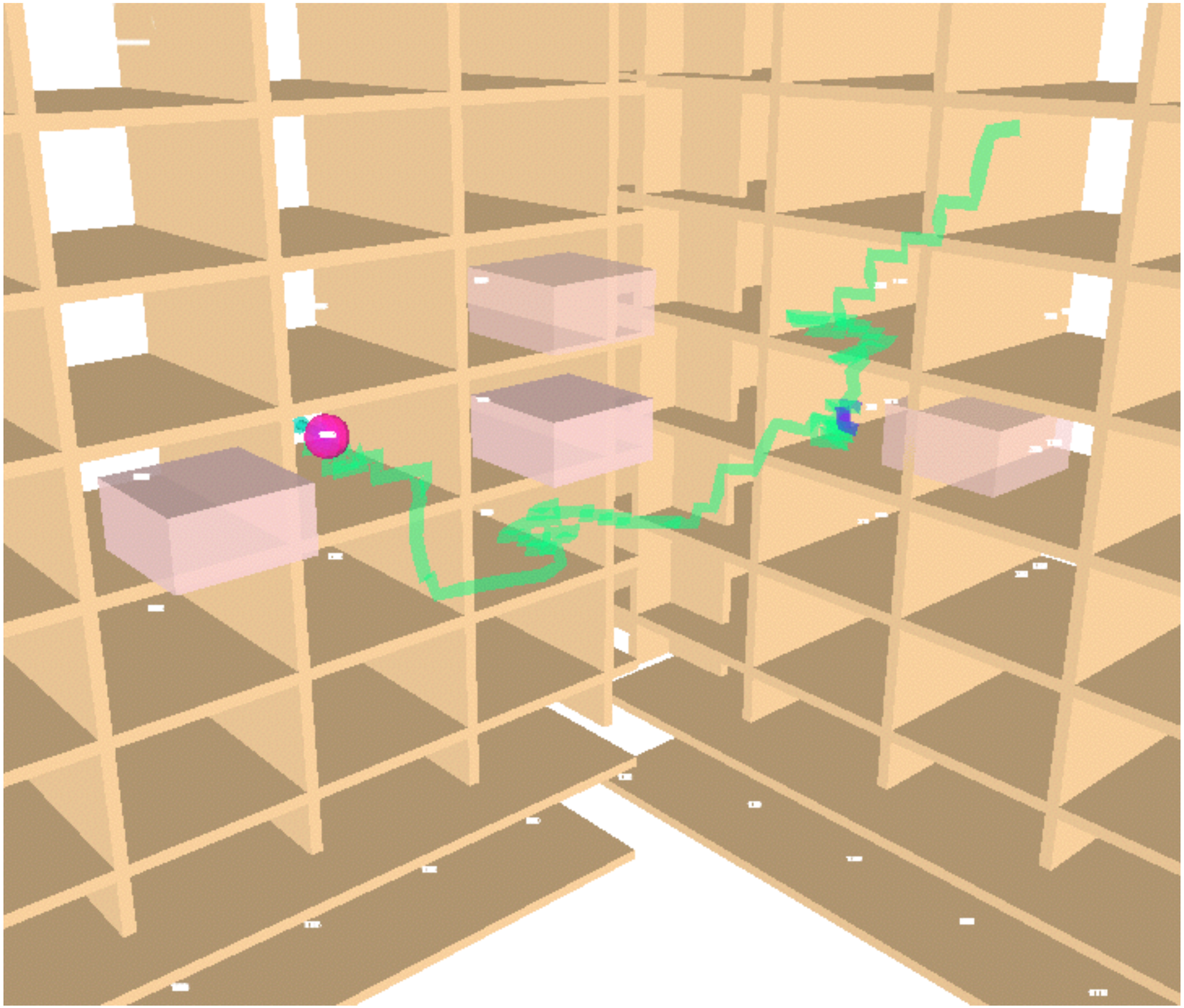


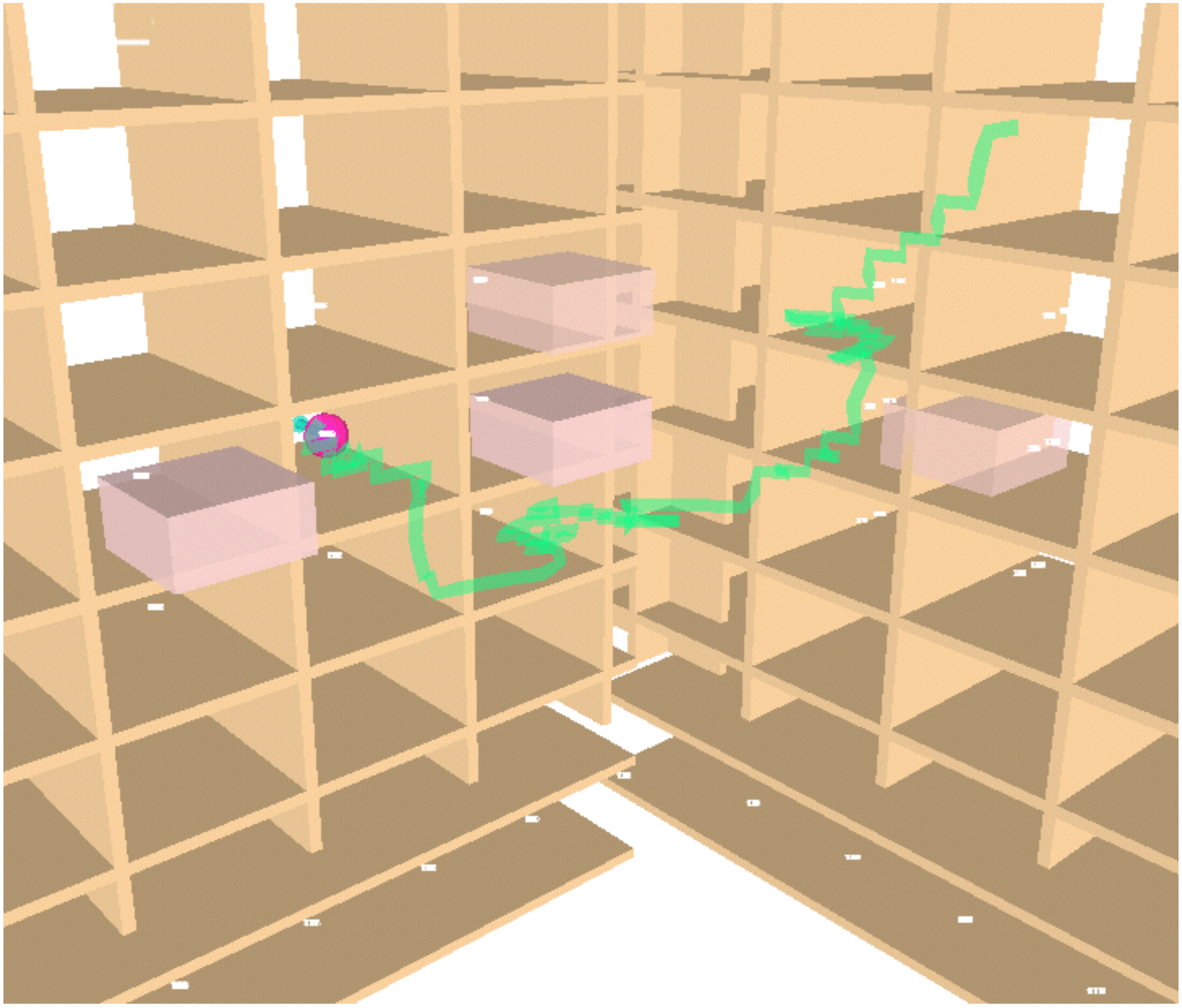








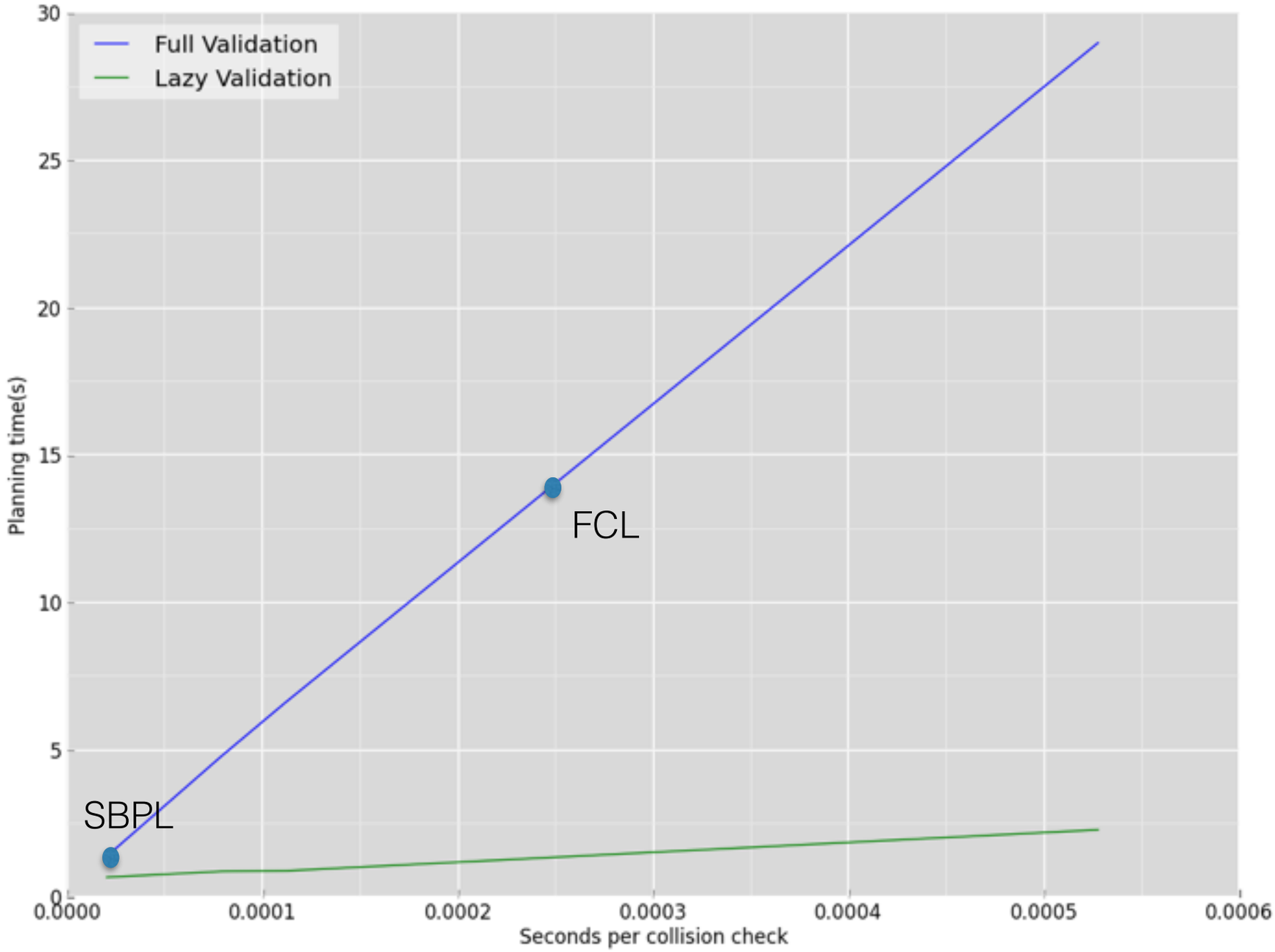




Collision Check Comparison

	Lazy Validation	Full Validation
Median CC/request	5181	36749
Avg CC/ request	18680	36769

Planning time vs CC time



Pan, et al., 2012

Planning Time Comparison

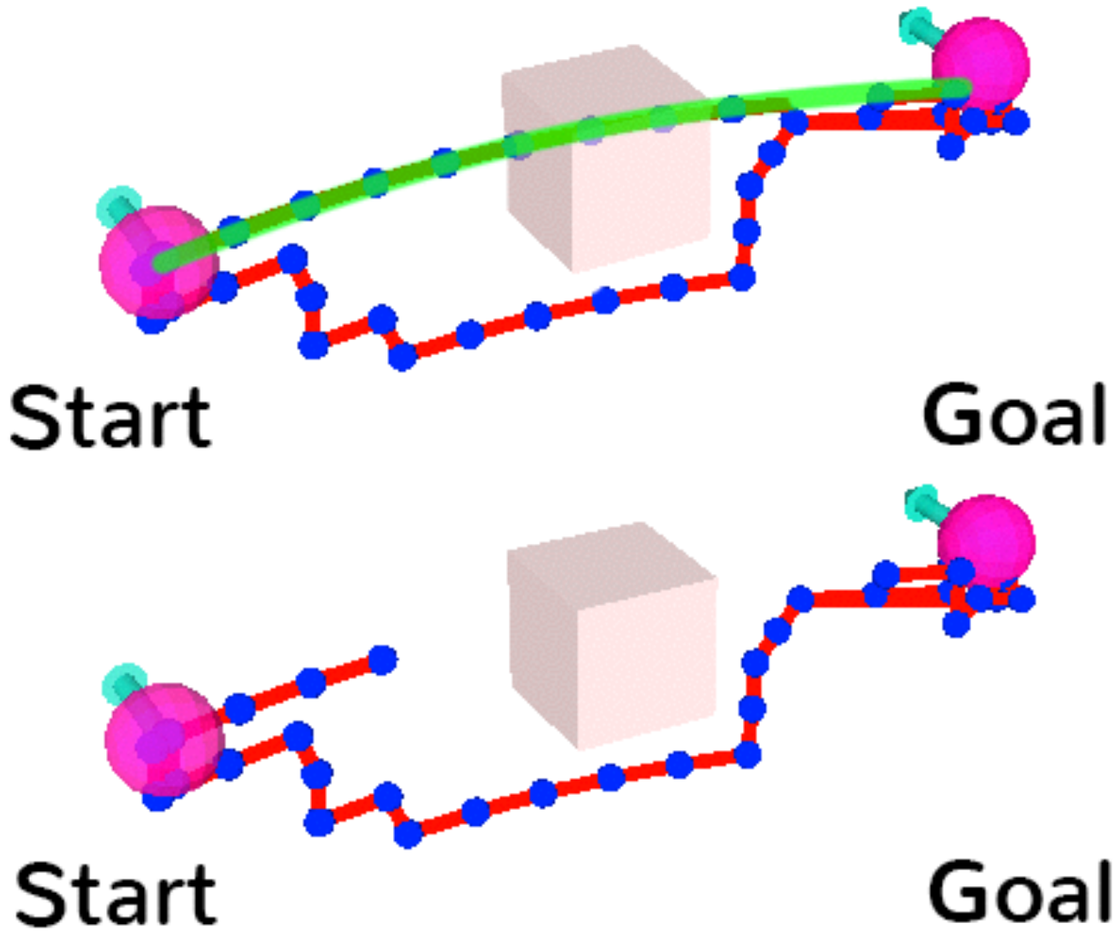
	Avg Planning Time (s)
Lazy Validation	1.04
Full Validation	1.20
RRTConnect	4.28
PRM	-
RRT*	-

Path Quality Comparison

	Path Quality with Shortcutting (End Effector)
Lazy Validation	1.55
Full Validation	1.58
RRTConnect	2.15

Lazy vs. On-the-fly Comparison

- On-the-fly: validates E-Graph edges during the search



Lazy vs. On-the-fly Comparison

	Lazy Validation	On-the-fly
Avg Planning Time (s)	1.20	1.26
Path Quality (EE)	1.55	1.53

- Similar performance in mailroom environment, but has drawbacks
  - Does not recompute heuristic
  - High overhead when E-Graph is uninformative

Future Work

- Reduce number of replans with better obstacle modeling
- Potential offline computations to improve the path quality

Conclusion

- Lazy validation enables efficient use of large E-Graphs in a changing environment
- Allows us to reuse experience to speed up planning

Thank you!