Improved Collaborative Pathfinding

Nathan Sturtevant
Michael Buro
University of Alberta

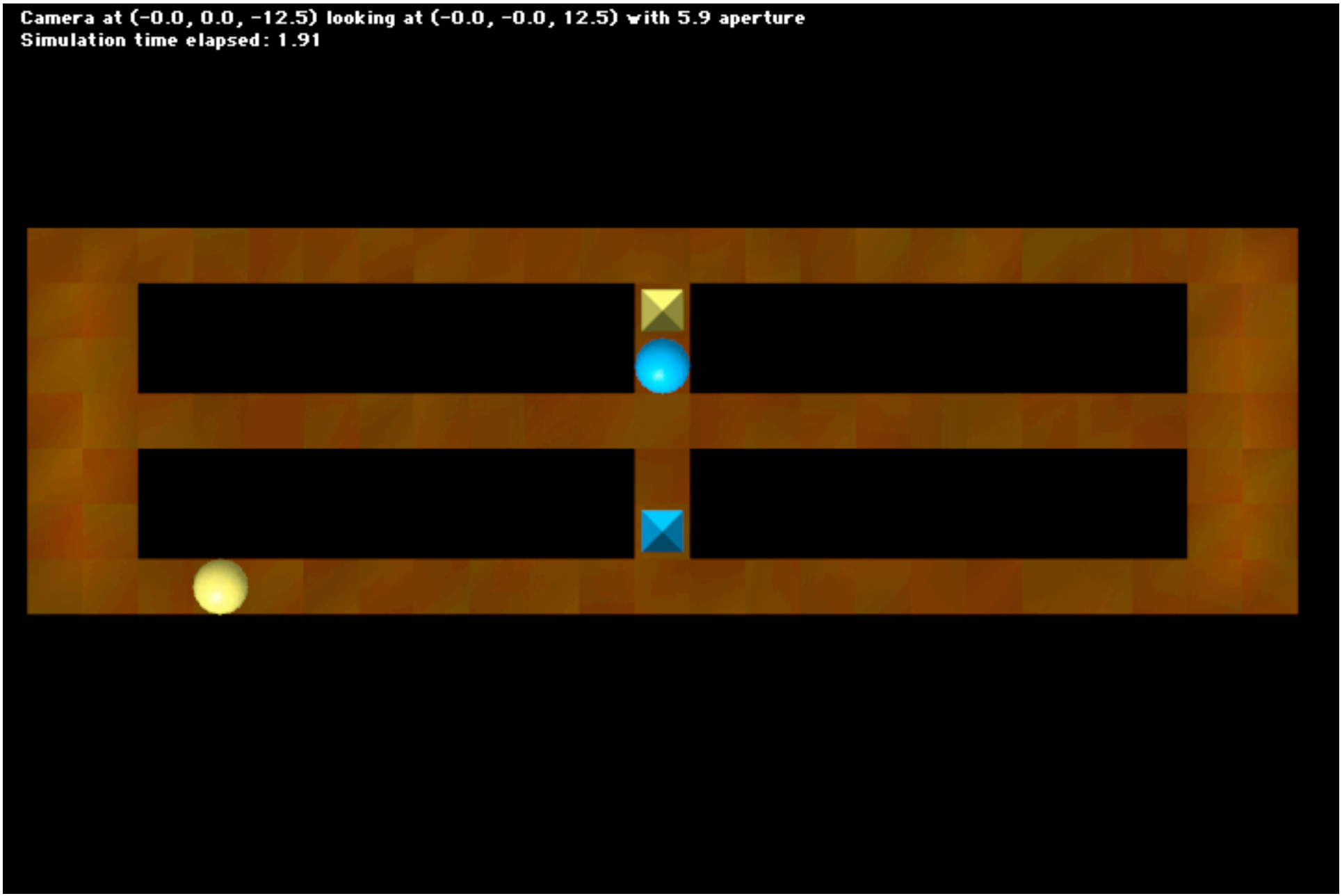
AIIDE - June 23, 2006

Cooperative Pathfinding

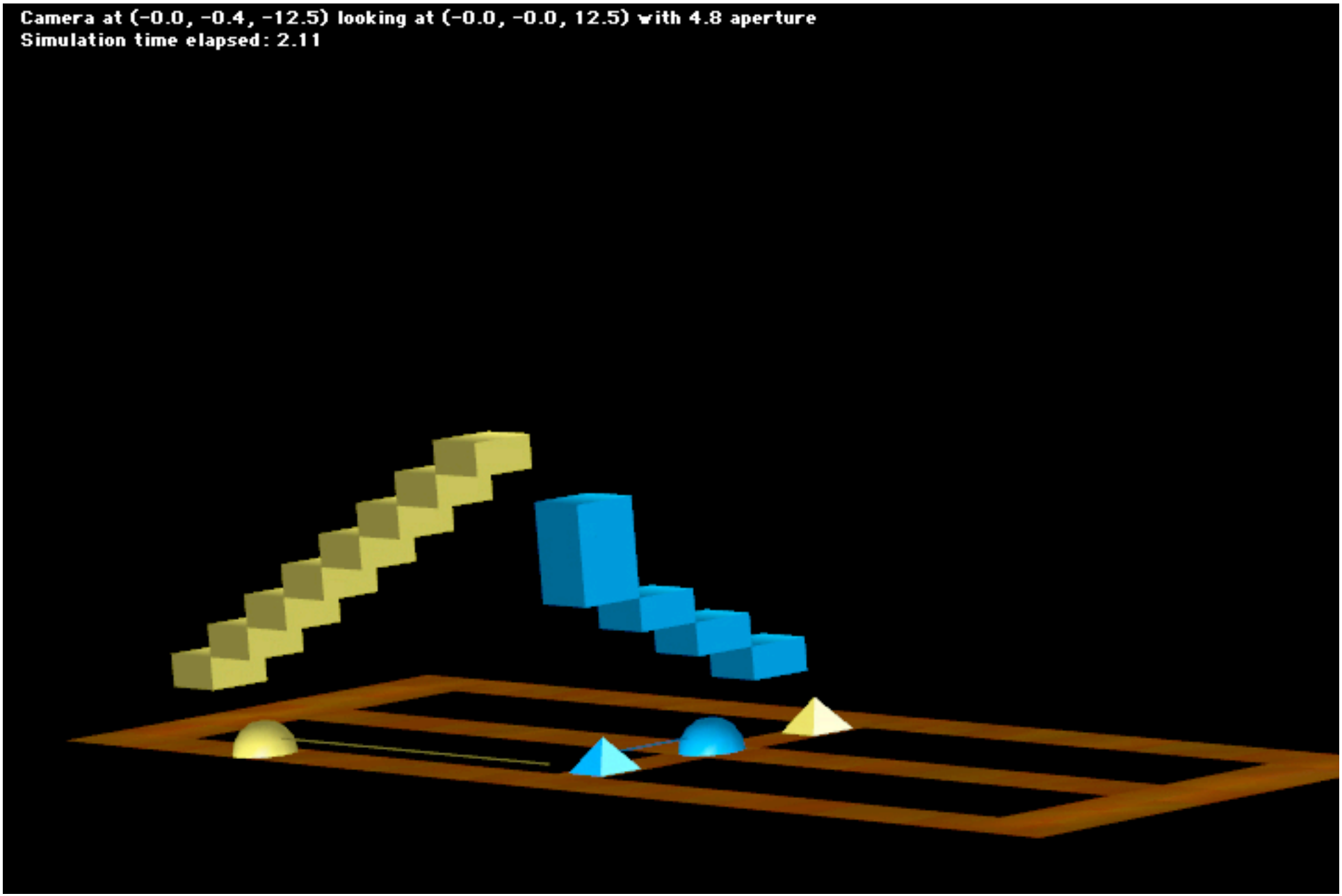
- Goal: Multiple agents cooperate during path planning and execution
- Different applications than flocking
  - Coordinate movement to avoid collisions
  - User may control the main character
  - Grid-based worlds

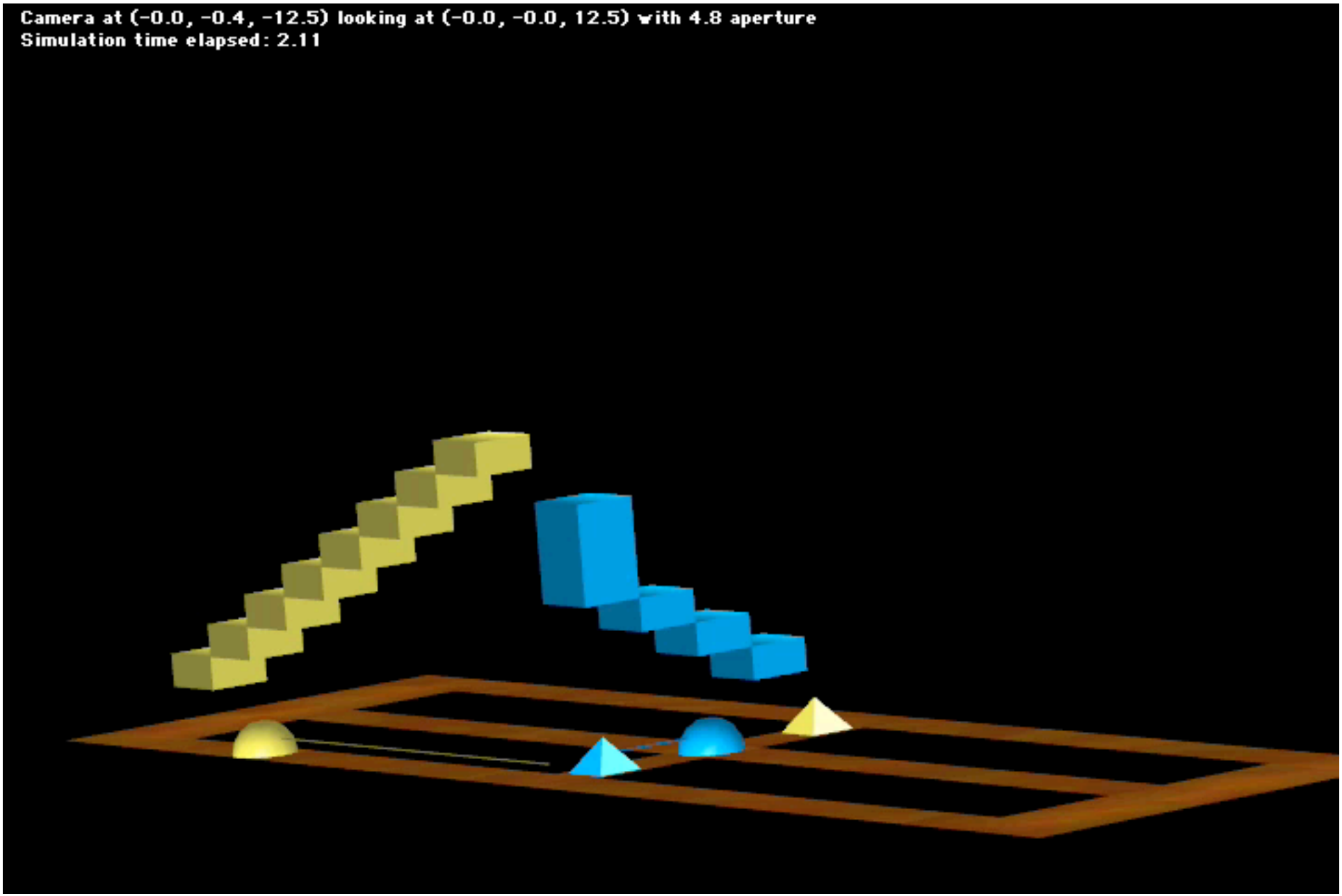

Quests
Menu (F10)
Allies
Log (F12)
750
200
15/12
No Upkeep
Local Player was victorious.


Quests
Menu (F10)
Allies
Log (F12)
750
200
15/12
No Upkeep
Local Player was victorious.


Camera at (-0.0, 0.0, -12.5) looking at (-0.0, -0.0, 12.5) with 5.9 aperture
Simulation time elapsed: 1.91


Camera at (-0.0, 0.0, -12.5) looking at (-0.0, -0.0, 12.5) with 5.9 aperture
Simulation time elapsed: 1.91


Camera at (-0.0, -0.4, -12.5) looking at (-0.0, -0.0, 12.5) with 4.8 aperture
Simulation time elapsed: 2.11


Camera at (-0.0, -0.4, -12.5) looking at (-0.0, -0.0, 12.5) with 4.8 aperture
Simulation time elapsed: 2.11

Possible Strategies

- Plan all units simultaneously
  - Computationally intractable
  - $(units \times actions)^{depth}$
- Plan individual units
  - Not complete
  - A lot of techniques needed to be practical

Overview

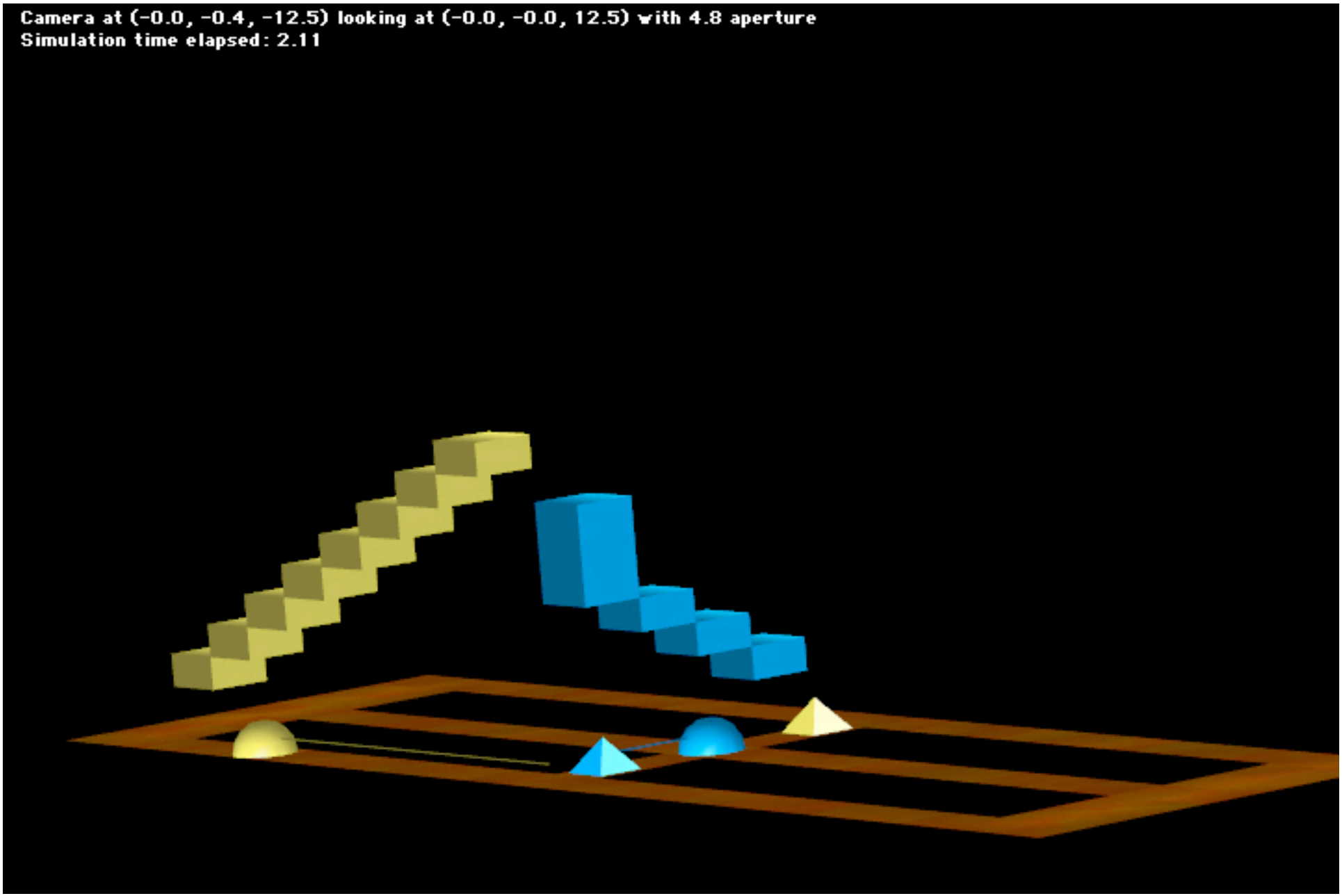
- Previous Work
  - Why problem is hard
  - What techniques simplify the problem
  - Drawbacks of current approach
- New Techniques
- Evaluation

WHCA*(*w*)

- Windowed Hierarchical Cooperative A*
  - Cooperative A*
  - Hierarchical Heuristic
  - Windowed cooperation
- Silver, 2005

WHCA*

- Use a hash table to store time-space indexed reservations
  - Constant time acces
  - Is a space/time cell free?
  - Reserve a space/time cell
  - Free a space/time cell

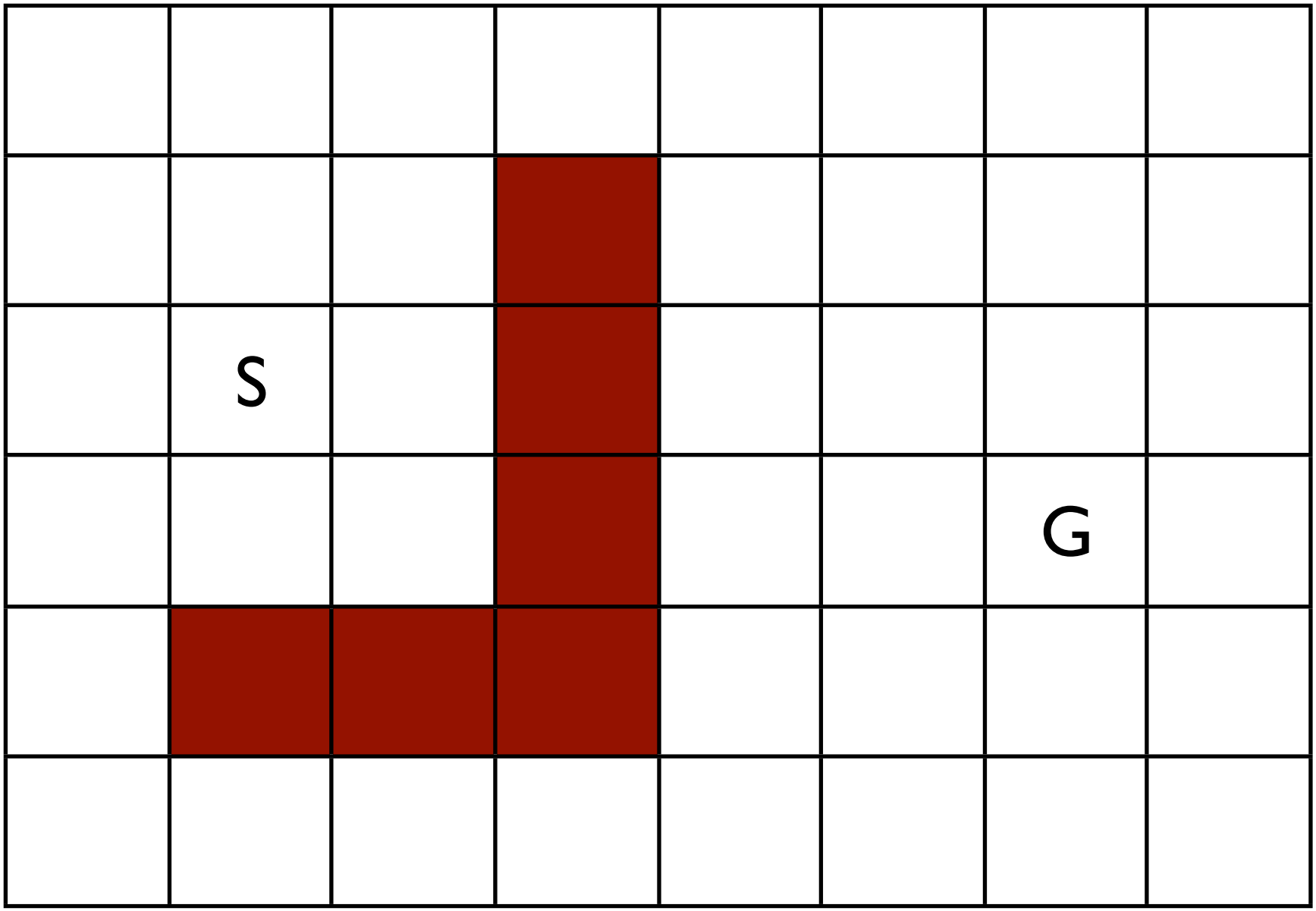

Camera at (-0.0, -0.4, -12.5) looking at (-0.0, -0.0, 12.5) with 4.8 aperture
Simulation time elapsed: 2.11

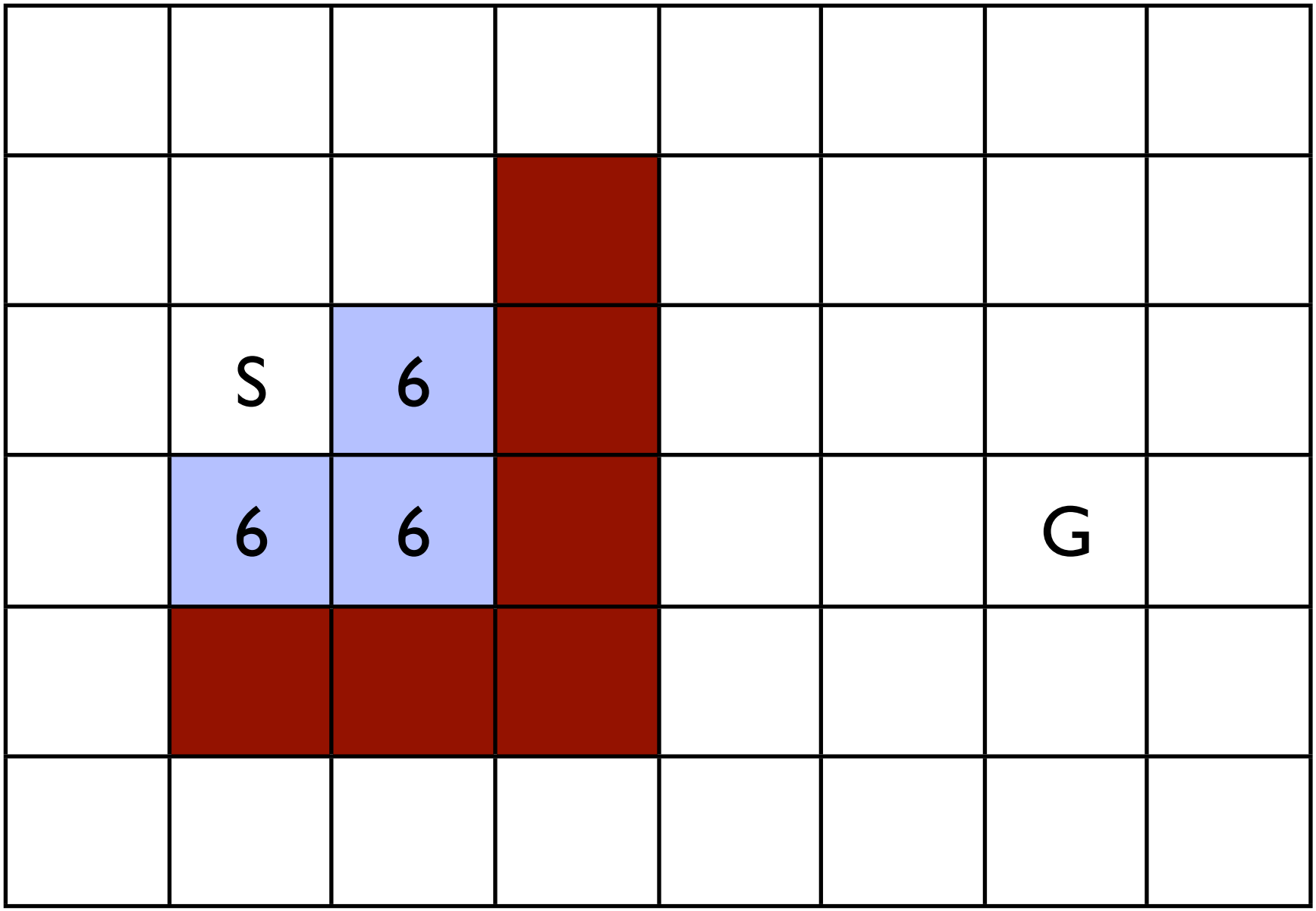
A*

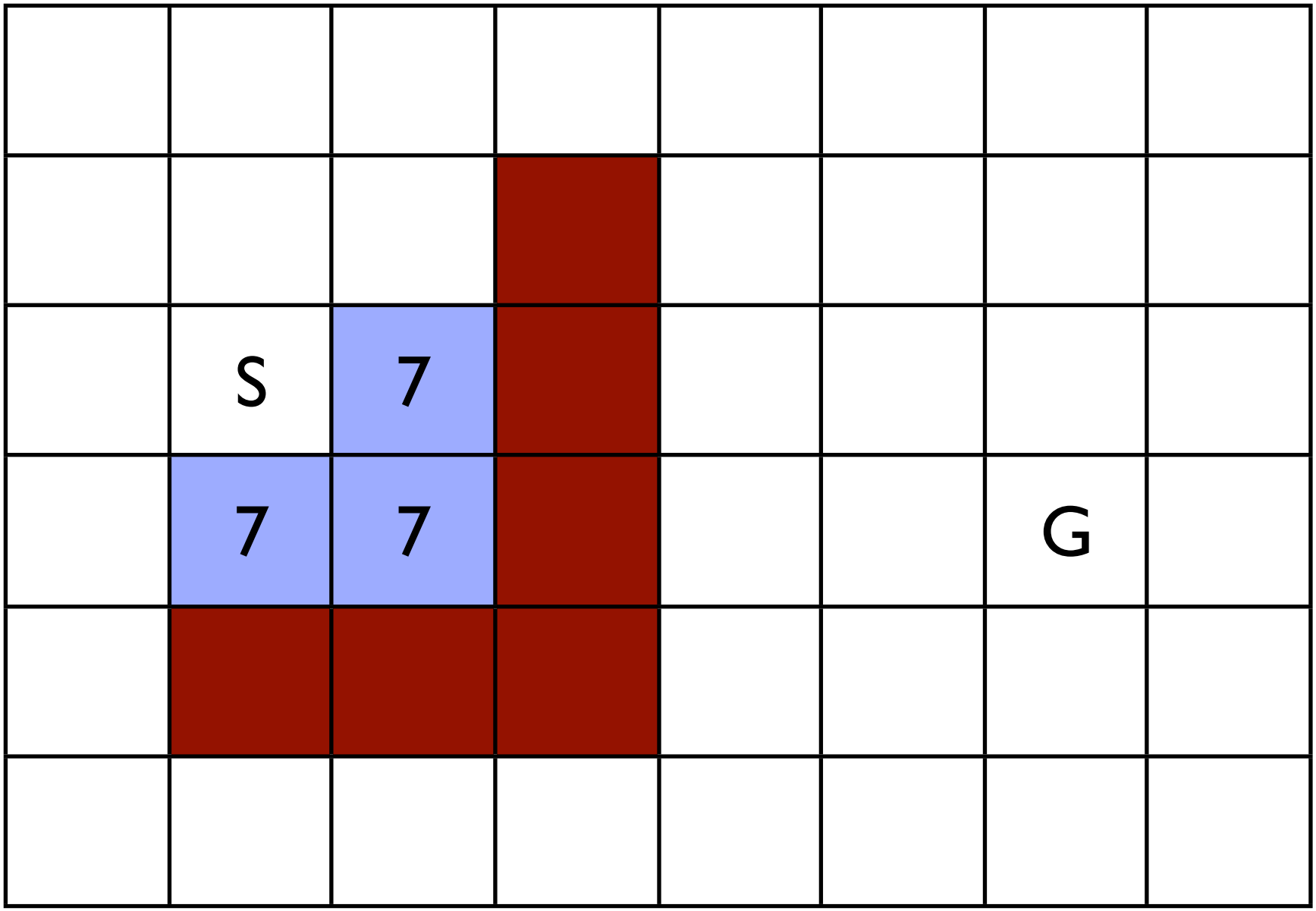
- A* relies on a heuristic to guide search
  - Poor heuristics cause extra node expansions
  - Cost is the **area** in which the heuristic is poor

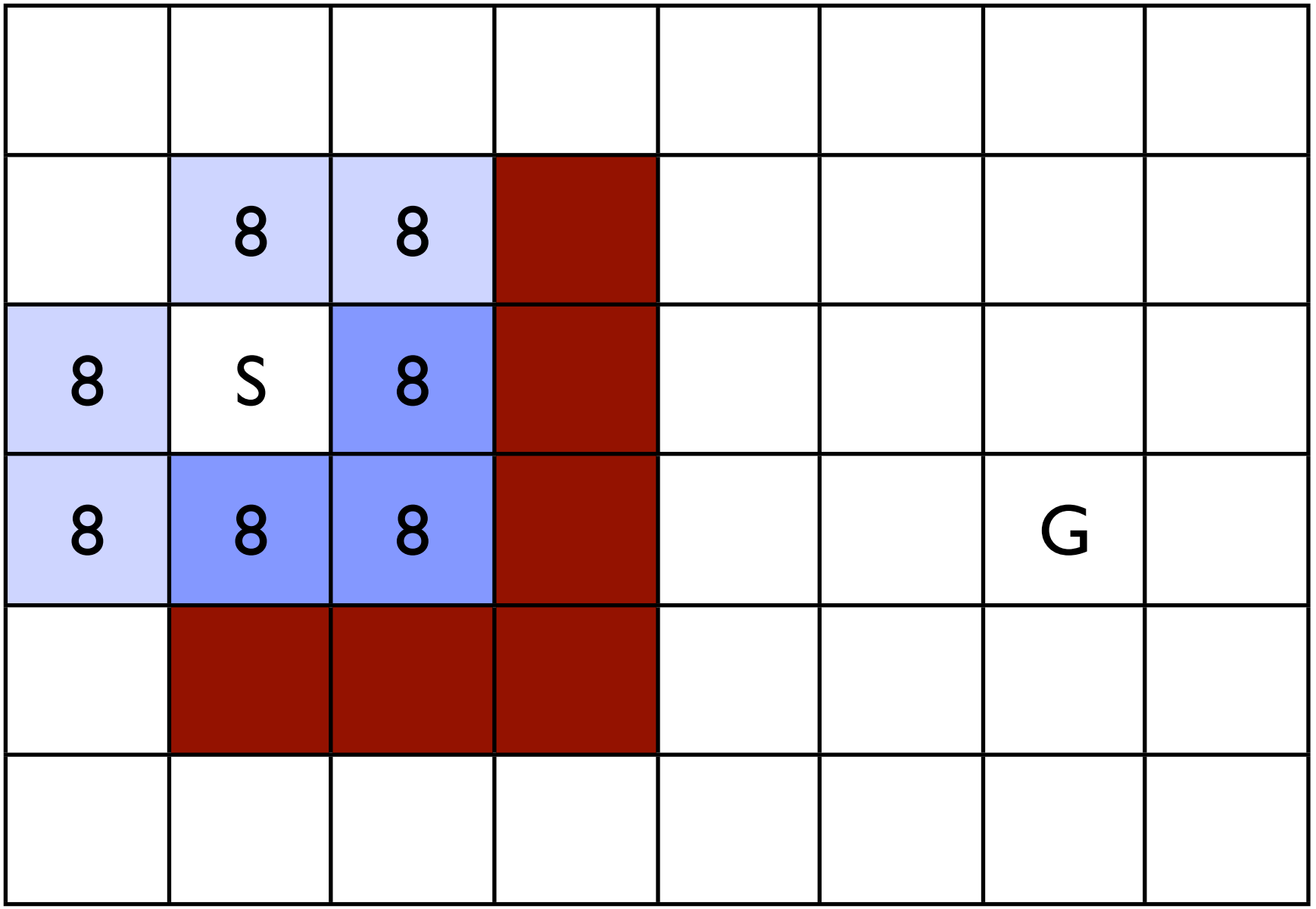
Cooperative A*

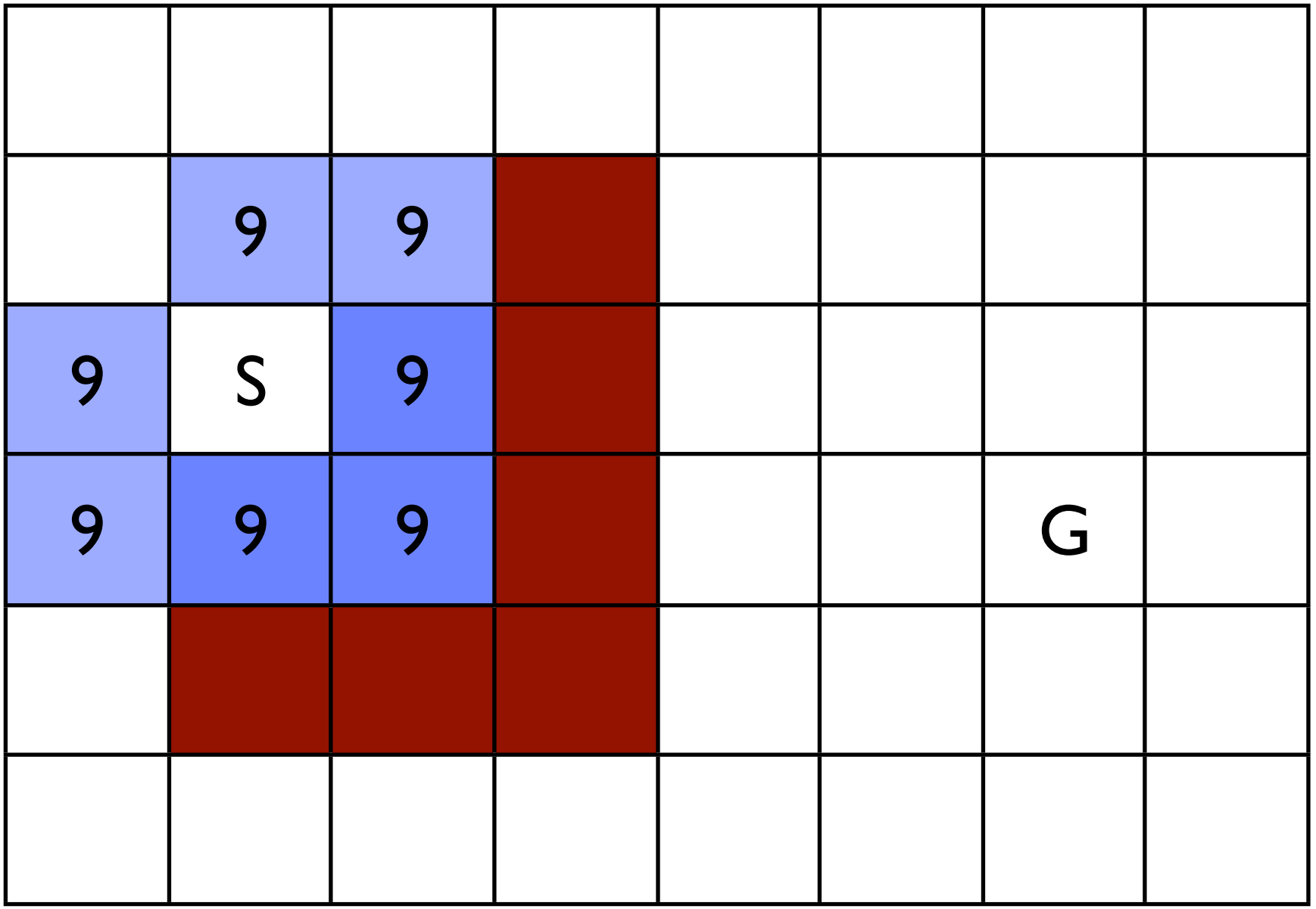
- 3-dimensional search problem
  - x-location, y-location, time
  - Still need a heuristic
    - Cost is the **area** in which the heuristic is poor *times* the time to get out of that area = **volume**

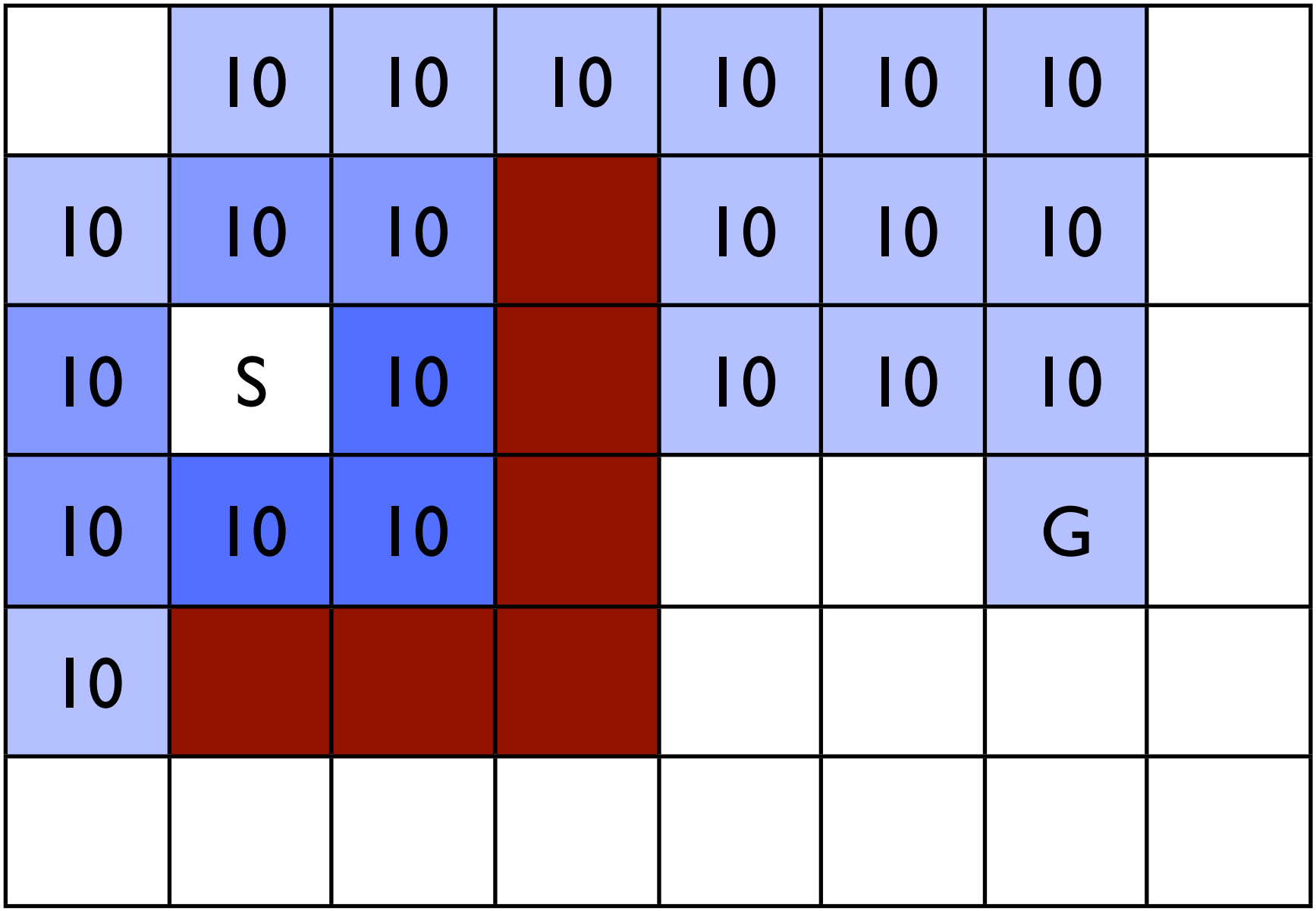

S
G


S
6
6
6
G


S
7
7
7
G

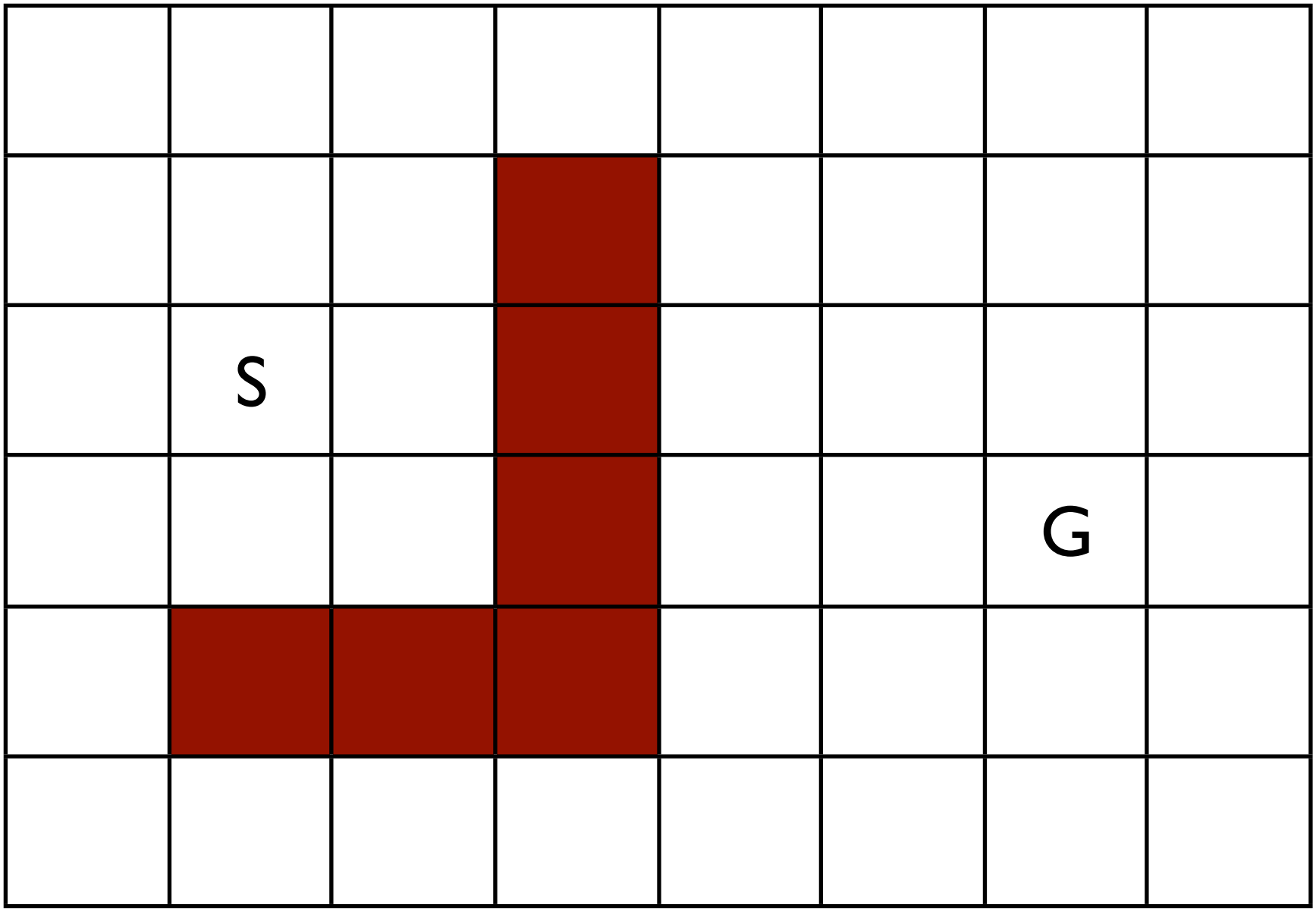

8
8
8
S
8
8
8
8
G


9
9
9
S
9
9
9
9
G

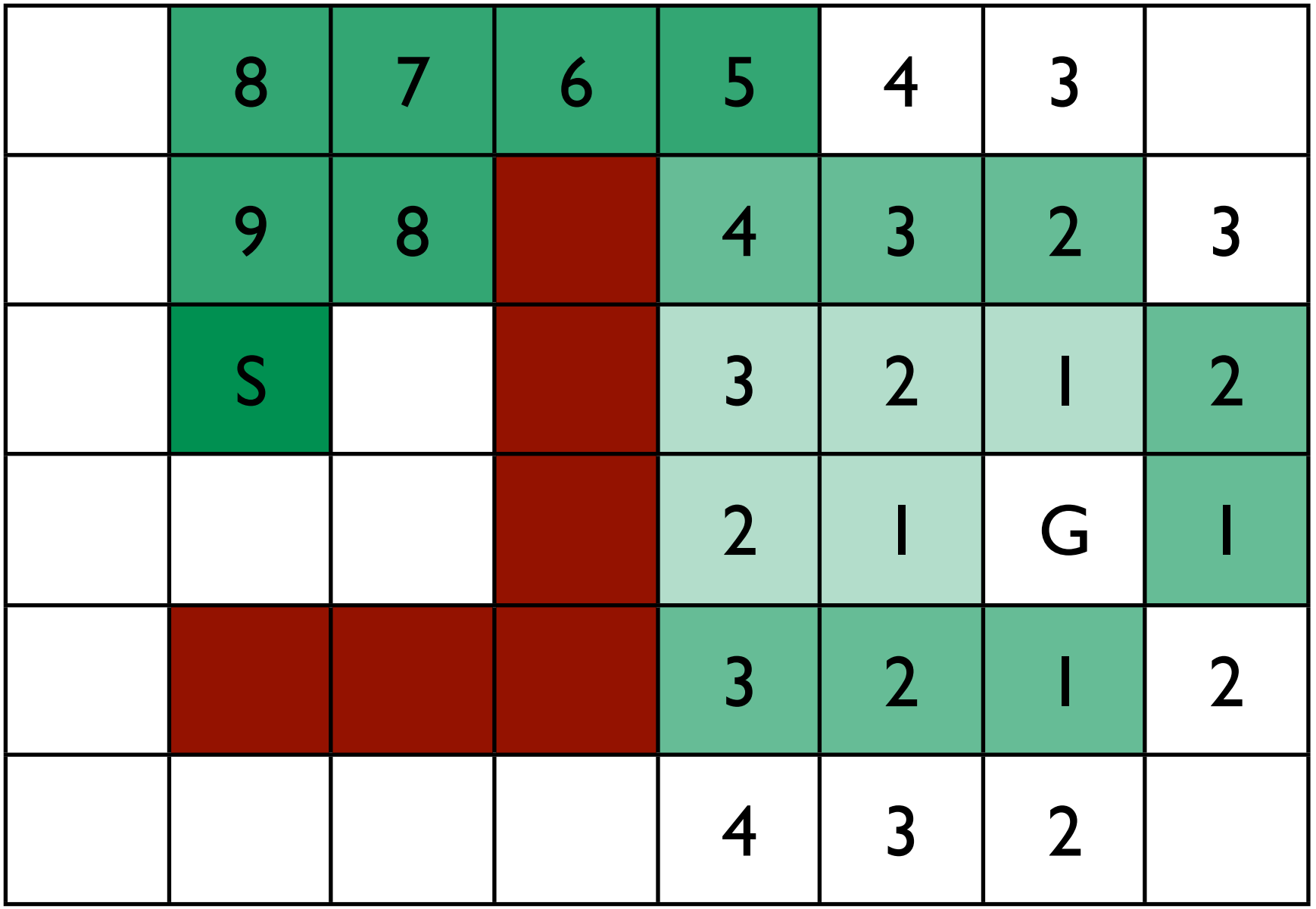

10 10 10 10 10 10
10 10 10 10 10 10
10 S 10 10 10 10
10 10 10 10 G
10

Heuristics

- Need a very accurate heuristic
- Where can we get a heuristic?
  - Run A* from the goal to the start state to get h() value for many states


S
G

	8+2	7+3	6+4	5+5	4+6	3+7	
	9+1	8+2		4+4	3+5	2+6	3+7
	S			3+3	2+4	1+5	2+6
				2+4	1+5	G	1+7
				3+5	2+6	1+7	2+8
				4+6	3+7	2+8	


8 7 6 5 4 3
9 8 4 3 2 3
S 3 2 1 2
2 1 G 1
3 2 1 2
4 3 2

Windowed Search

- We now have a perfect heuristic
  - With a perfect heuristic only 1-step lookahead is needed
  - Stop search at any time and be guaranteed to be on a path to the goal
- Do *k*-step lookahead in cooperative space

WHCA*(*k*)

- Do single A* search from goal to start
  - Do *k*-step forward cooperative search
  - Expand original search if new heuristic values needed

WHCA* Drawbacks

- First step is expensive
  - Compute complete reverse A* search
  - Compute forward CA* search
- Memory per unit is expensive
  - Keep whole search frontier in memory
- Goal State can't change

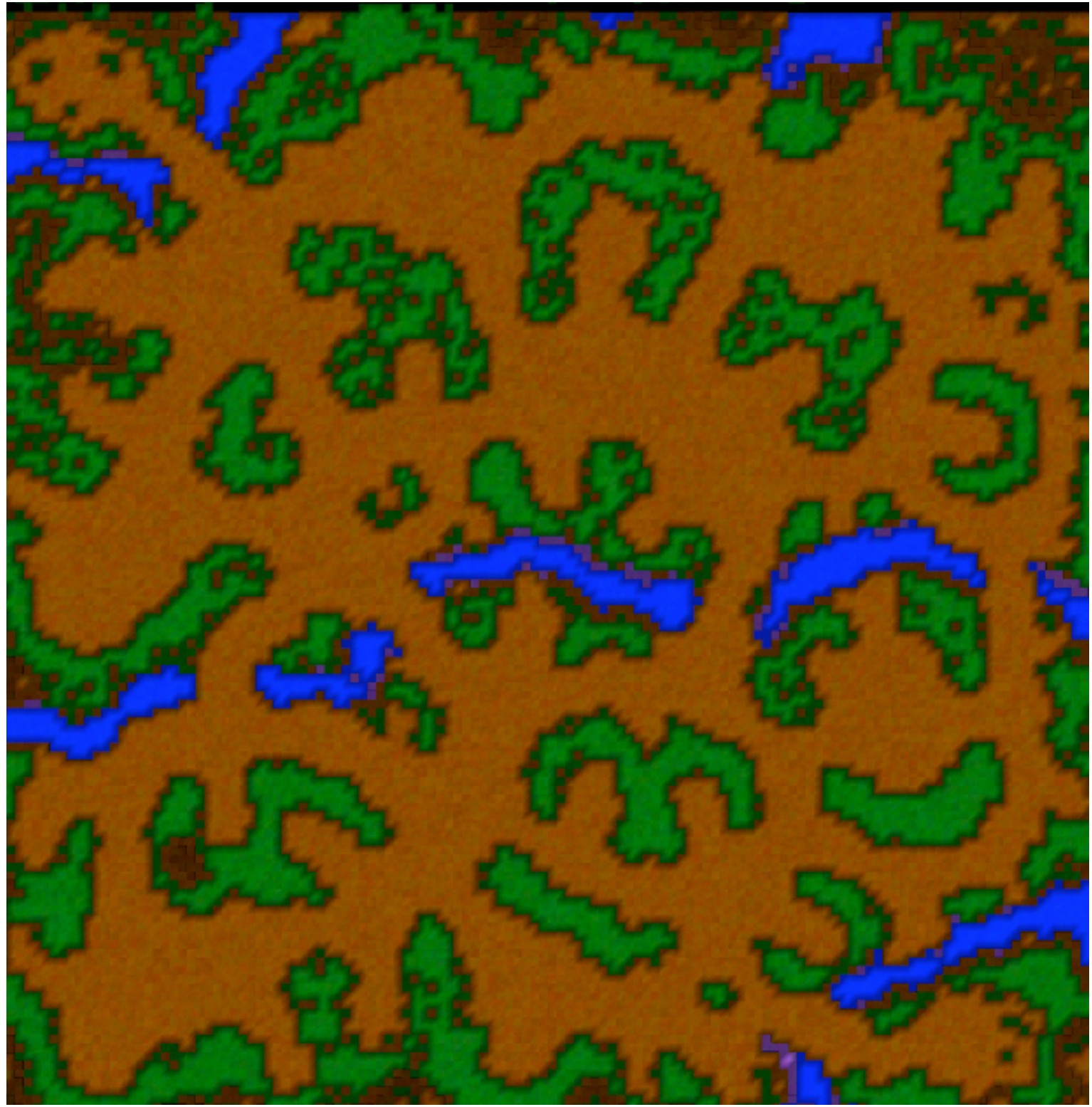
Improving WHCA*

- Abstraction
  - Widely used idea (eg Holte, 1996)
- Two proposed usages
  - WHCA*(*w*, *a*)
  - CPRA*(*k*)

Abstraction

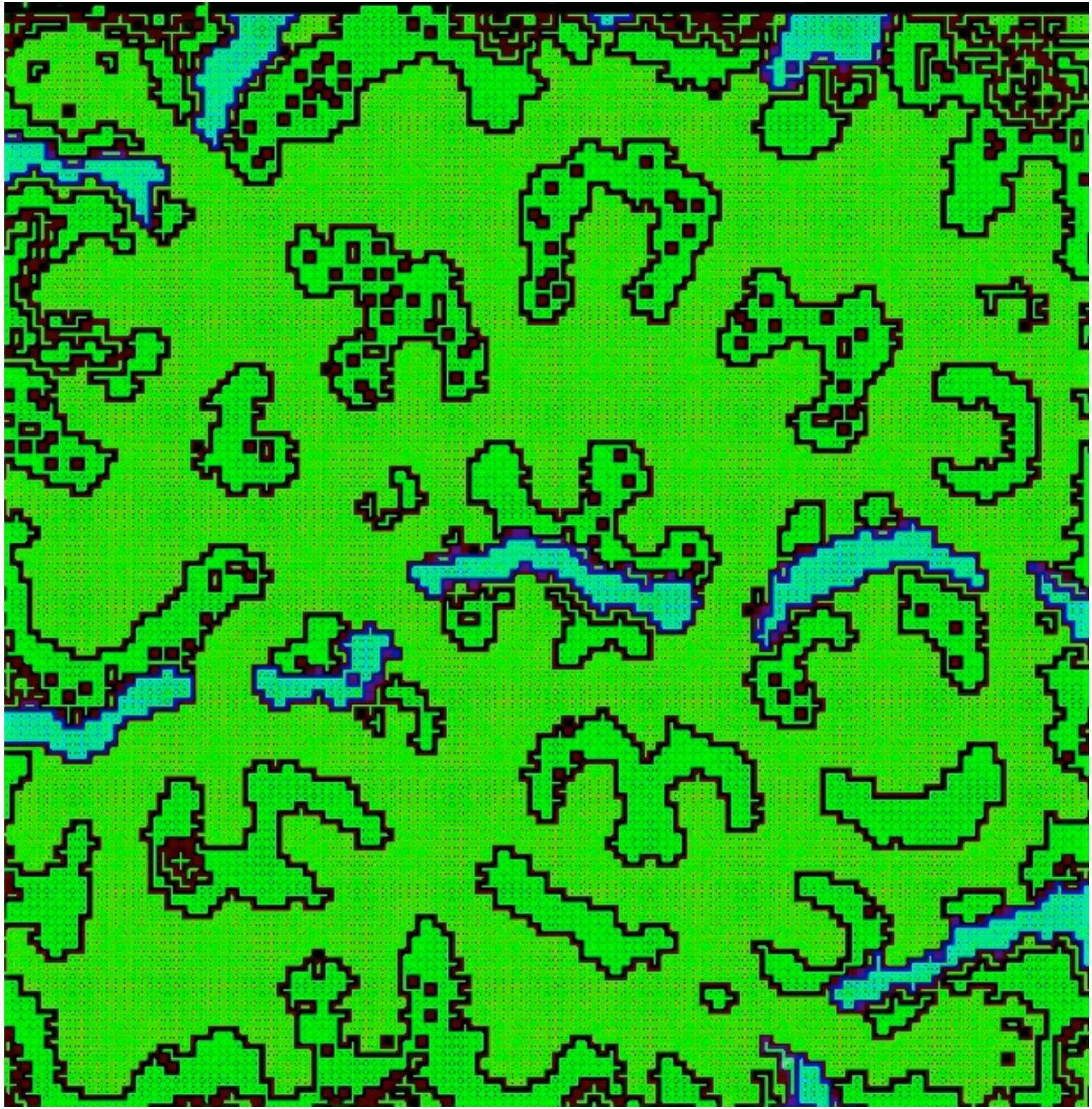
- Build “abstract” version of map
  - Lower resolution map
  - Automatic abstraction techniques
- Use one or more map abstractions to speed cooperative pathfinding
  - Clique reduction

Sample Map



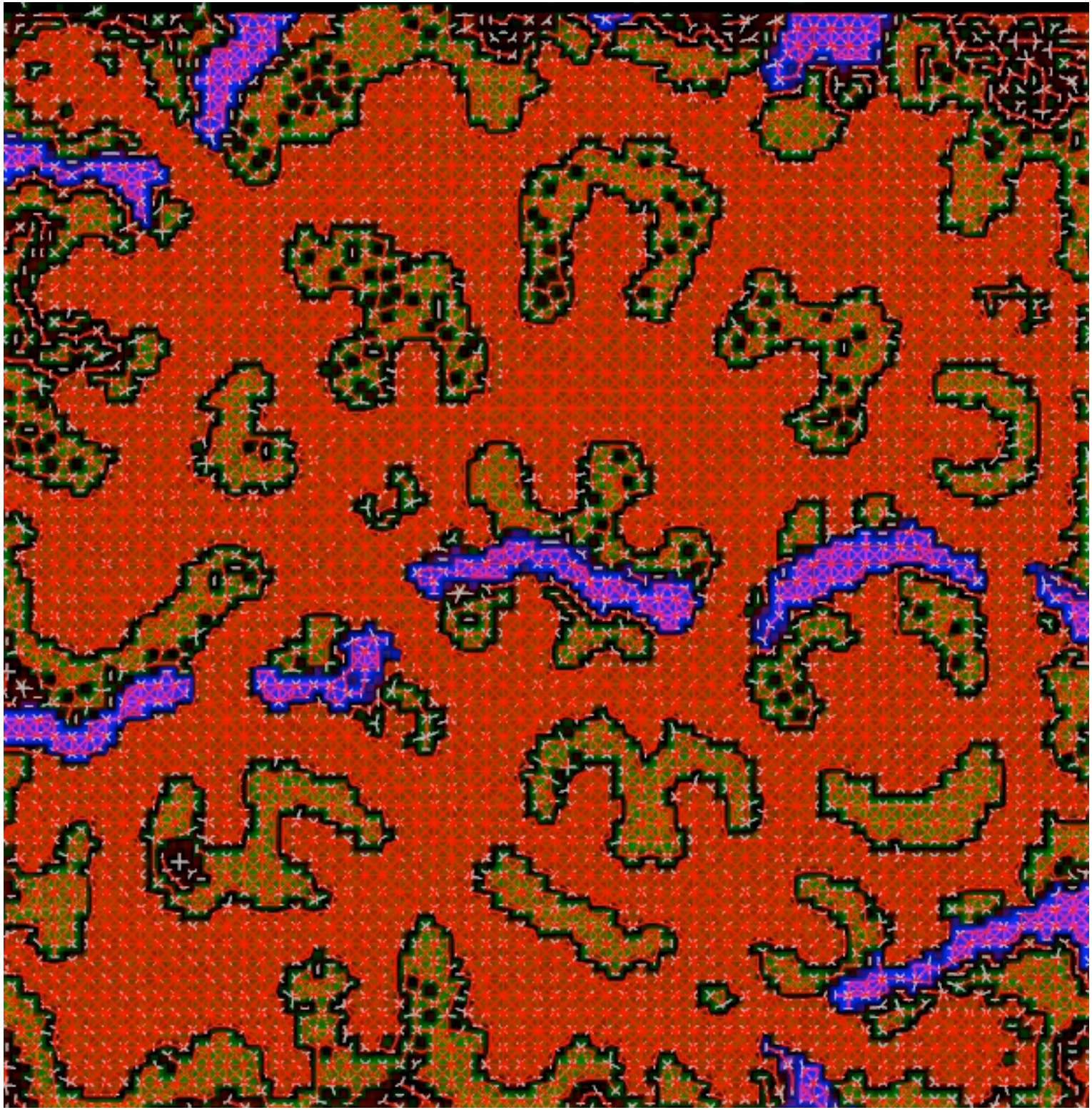
Base Graph

16,807 nodes



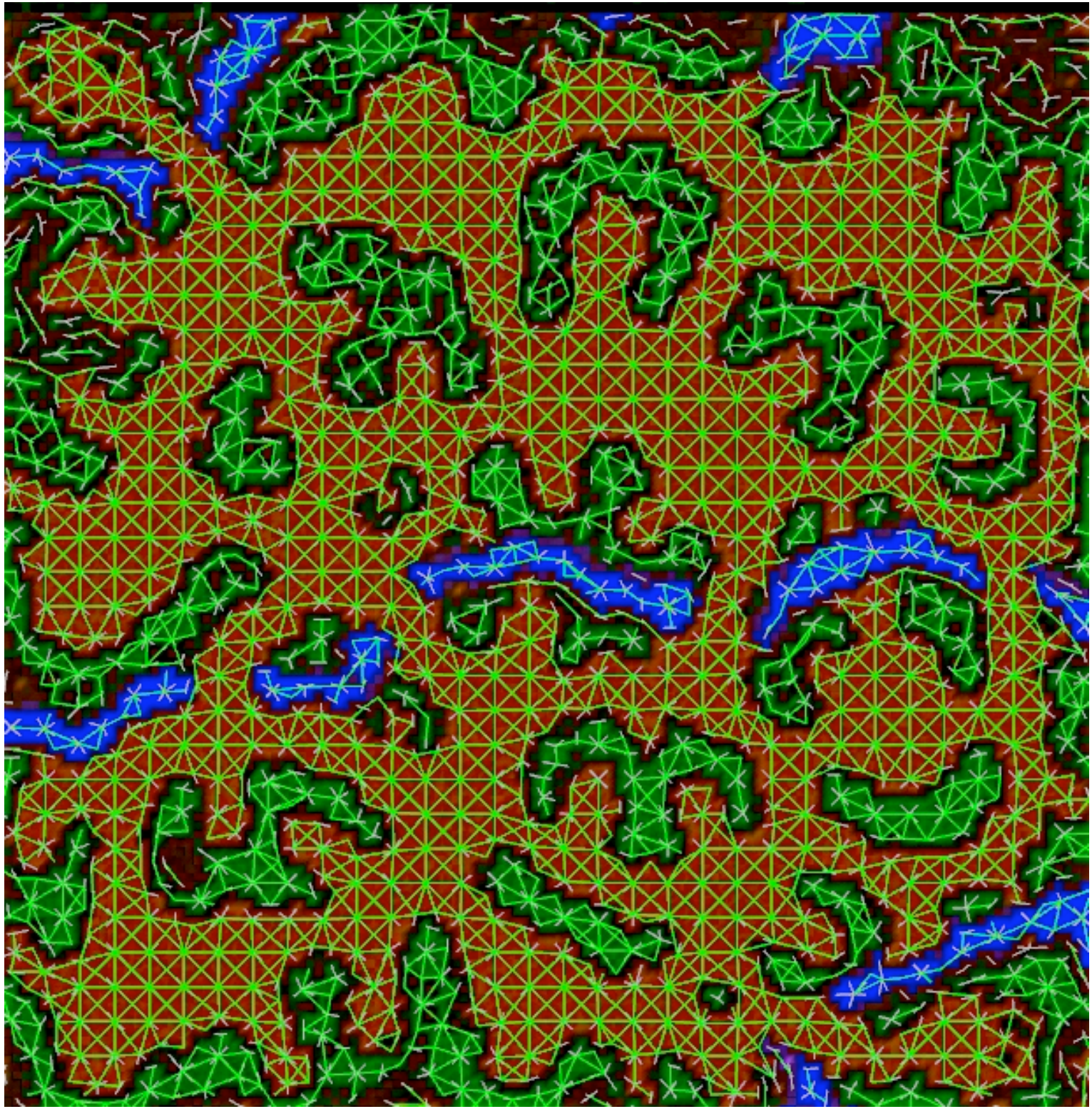
Abstraction 1

5,212 nodes



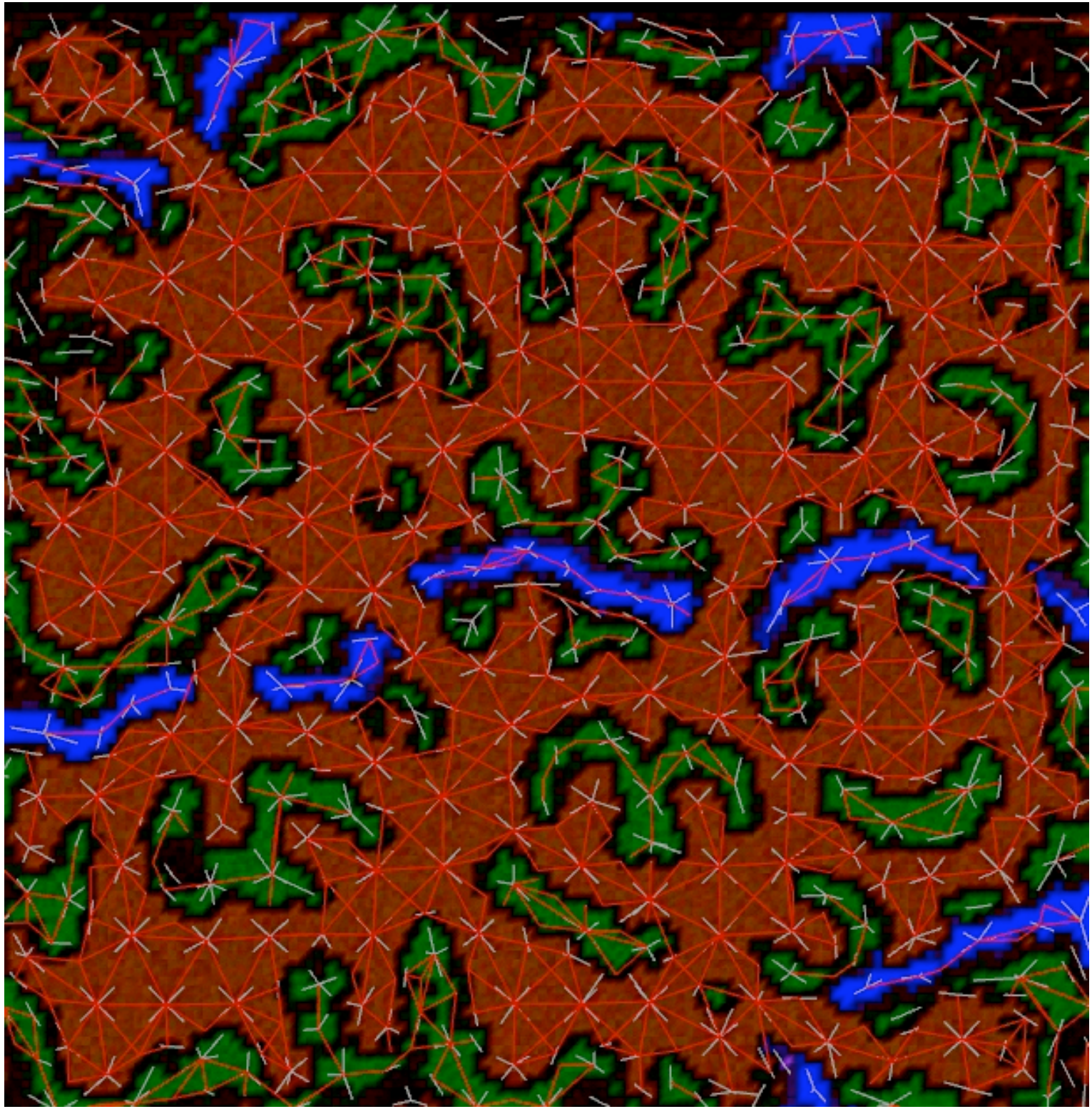
Abstraction 2

1,919 nodes



Abstraction 3

771 nodes



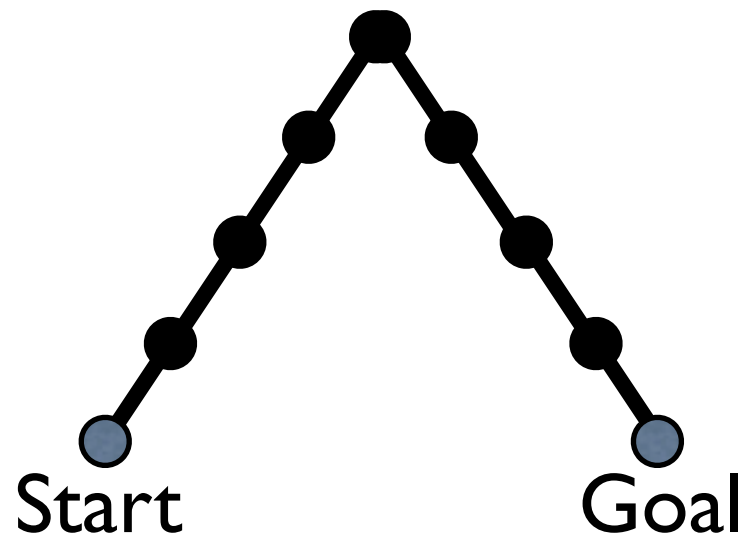
WHCA*(*k*, *a*)

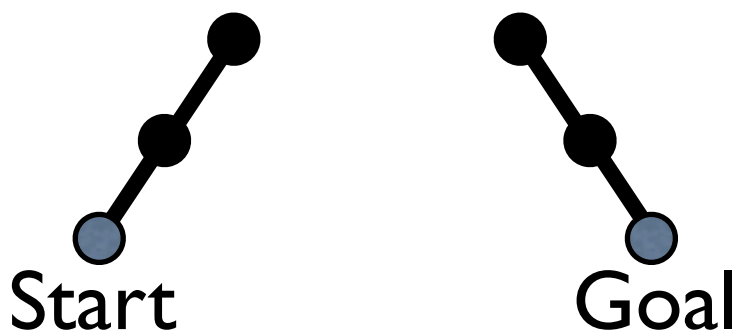
- Same as WHCA*(*k*) but do reverse A* search at level *a*
- Keep smaller A* open/closed list in memory
- Faster A* computation
- Eventually less accurate

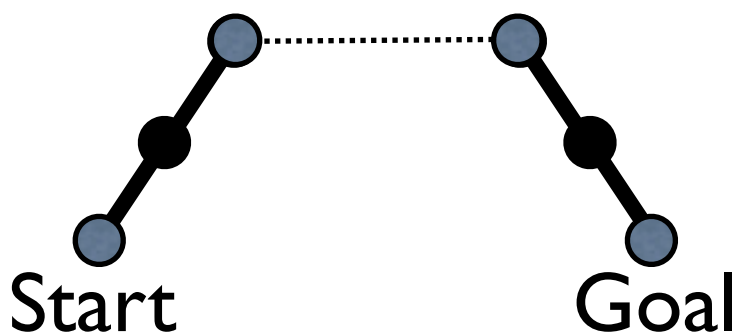
PRA*

- Partial-Refinement A*
  - Use multiple abstraction levels
  - Refine abstract paths using A*


Start
Goal


Start
Goal

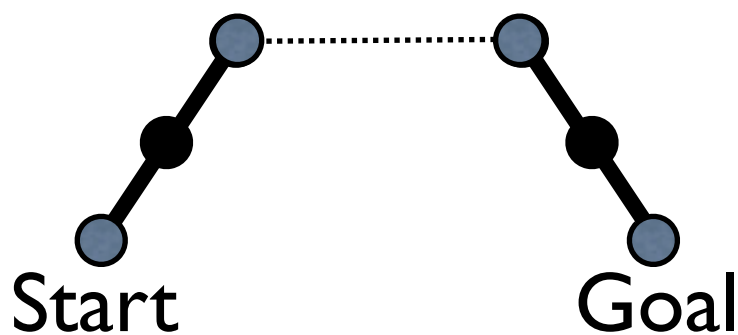

Start
Goal


Start
Goal

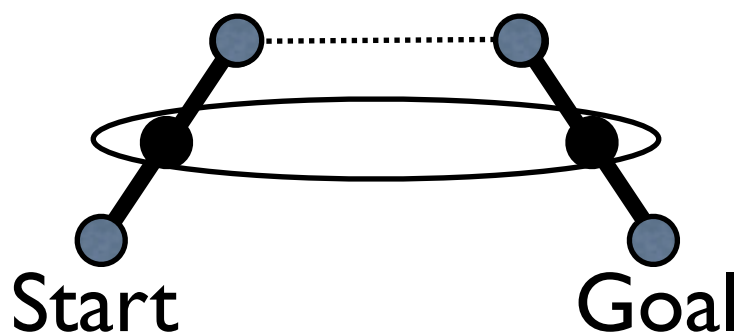
Pathfinding

- Given abstract path:
  - Path defines a *corridor* in the lower level of abstraction
  - Run A* in this corridor to find next path
  - Repeat until done

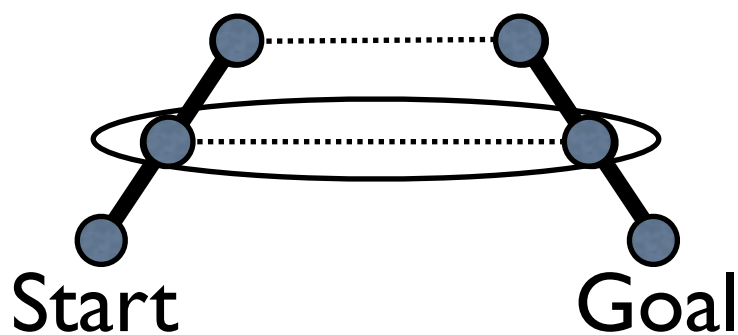
PRA*(∞)



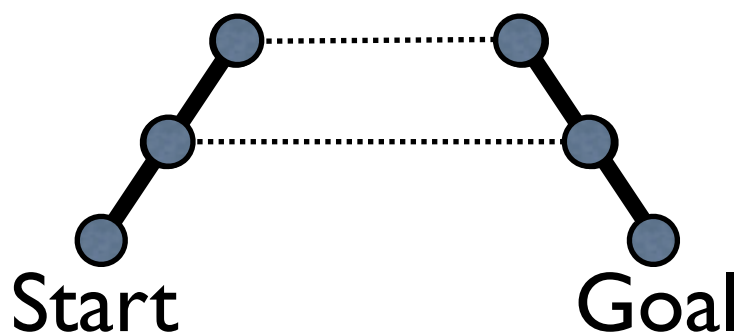
PRA*(∞)



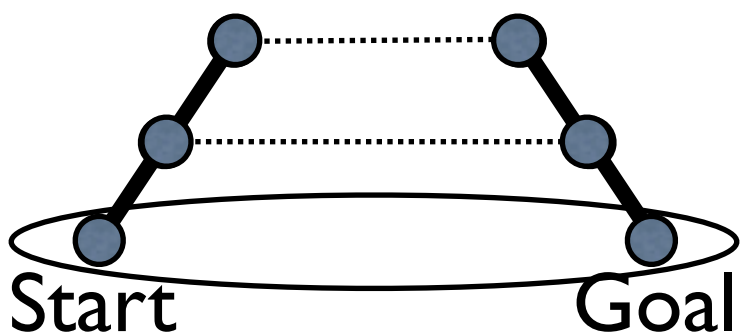
PRA*(∞)



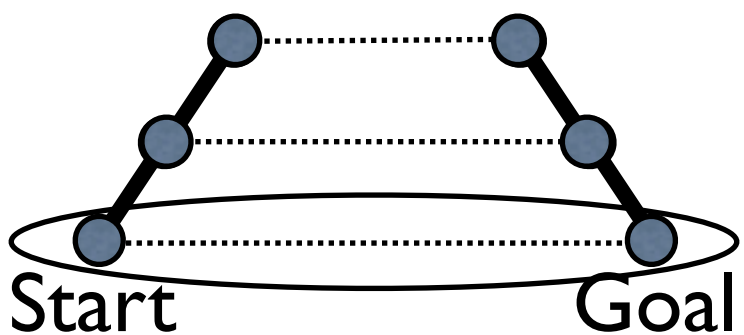
PRA*(∞)



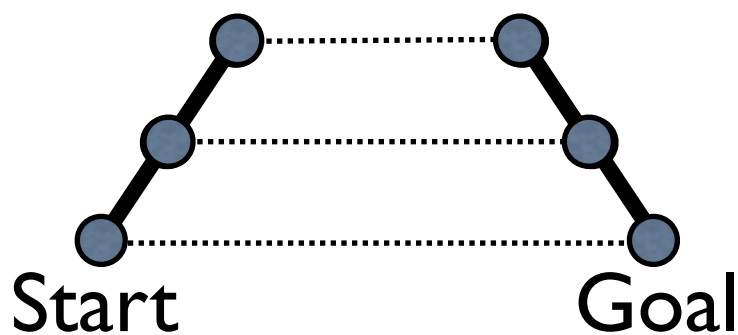
PRA*(∞)



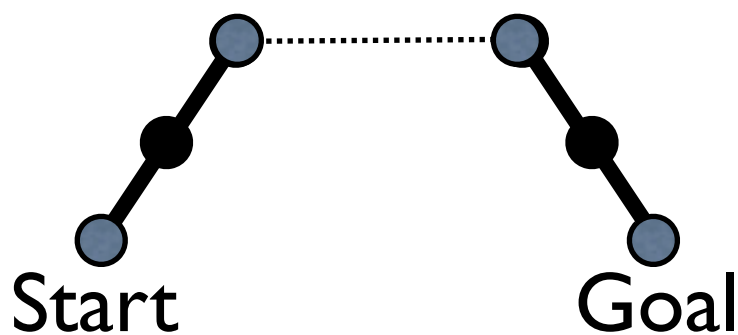
PRA*(∞)



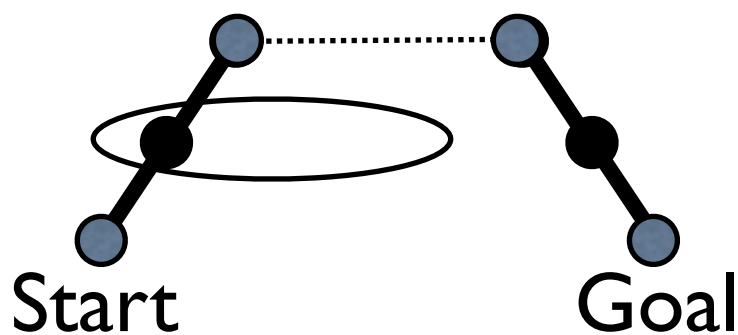
PRA*(∞)



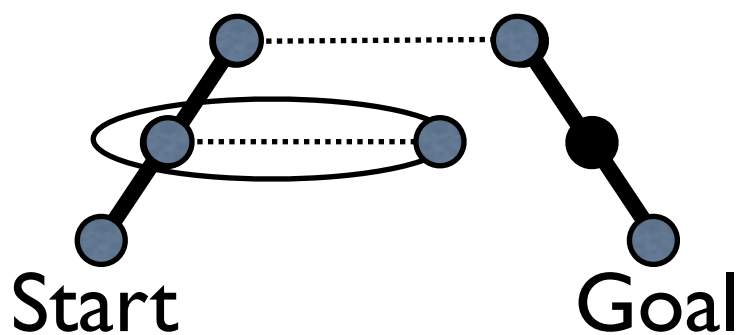
PRA*(*k*)



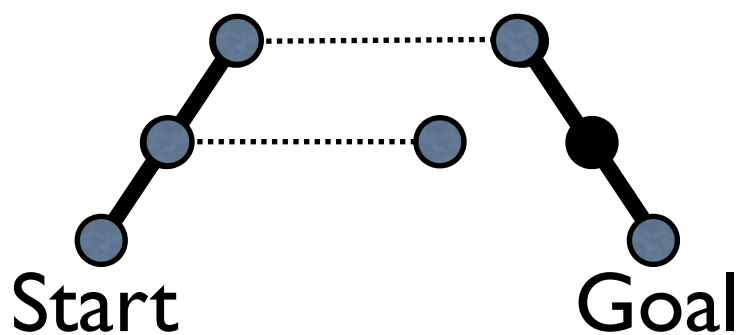
PRA*(*k*)



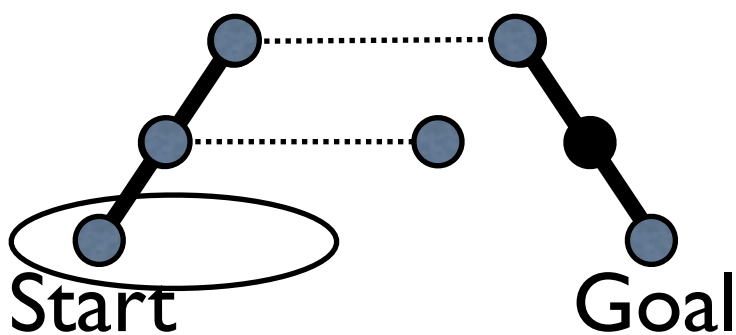
PRA*(*k*)



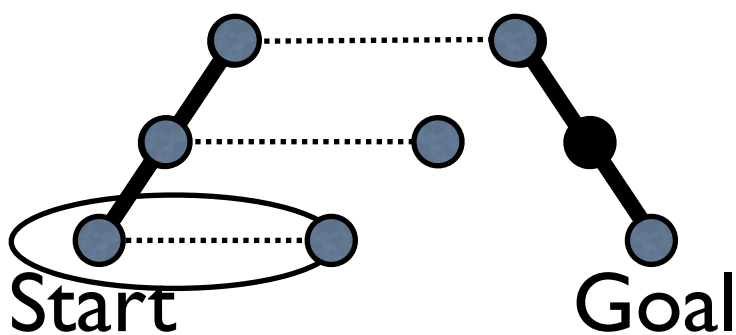
PRA*(*k*)



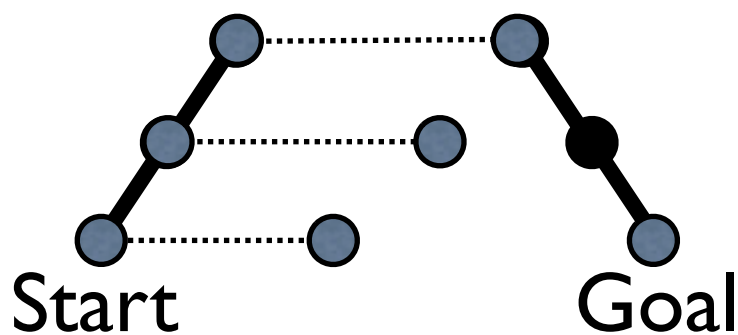
PRA*(*k*)



PRA*(*k*)



PRA*(*k*)

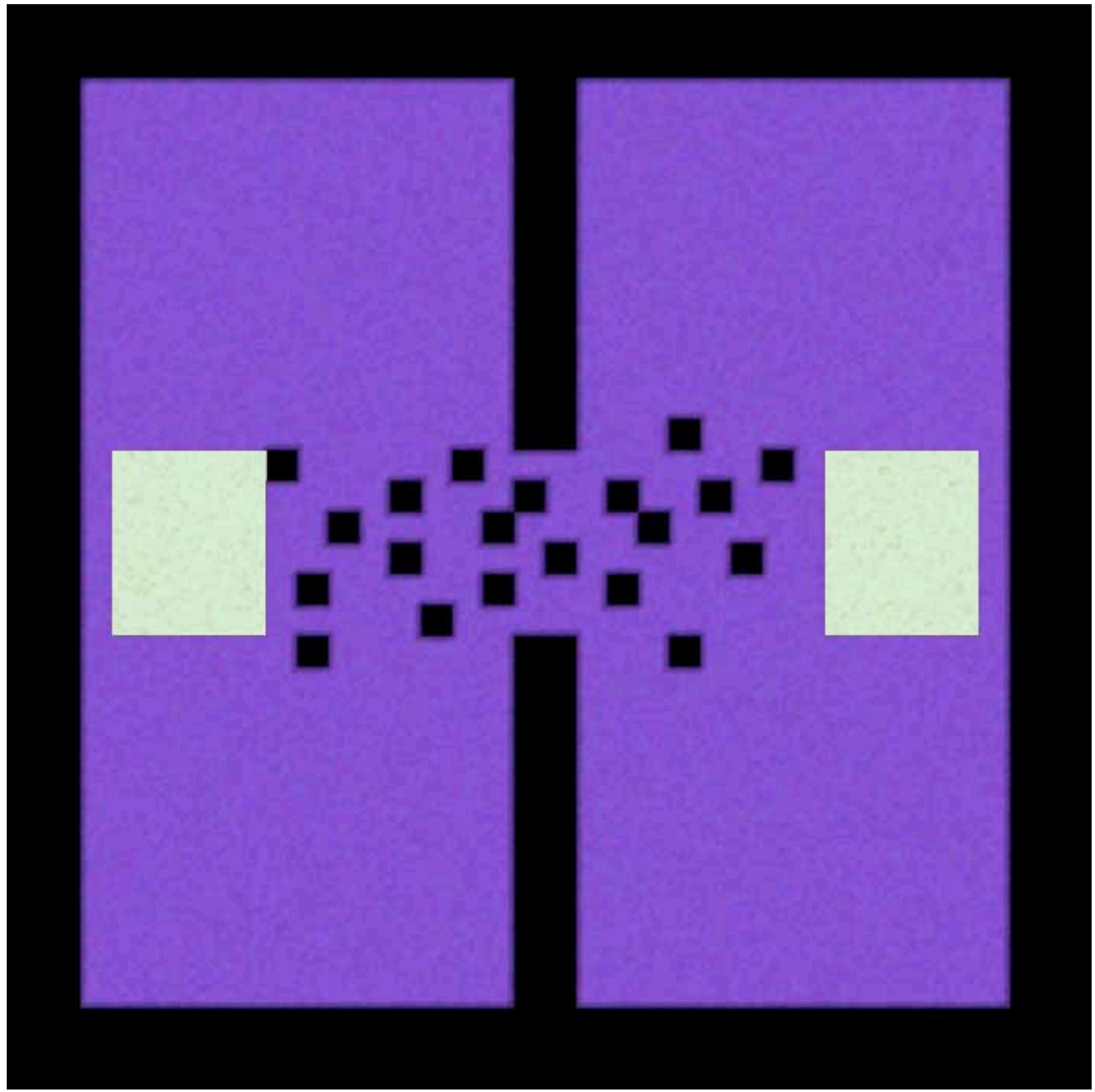


CPRA*(*k*)

- Same as PRA*(*k*), but do WHCA*(*k*, 1) at last refinement level
- Only plan part of total path
  - Much lower first-step cost
- Repeated WHCA* calls after executing each path

Experiments

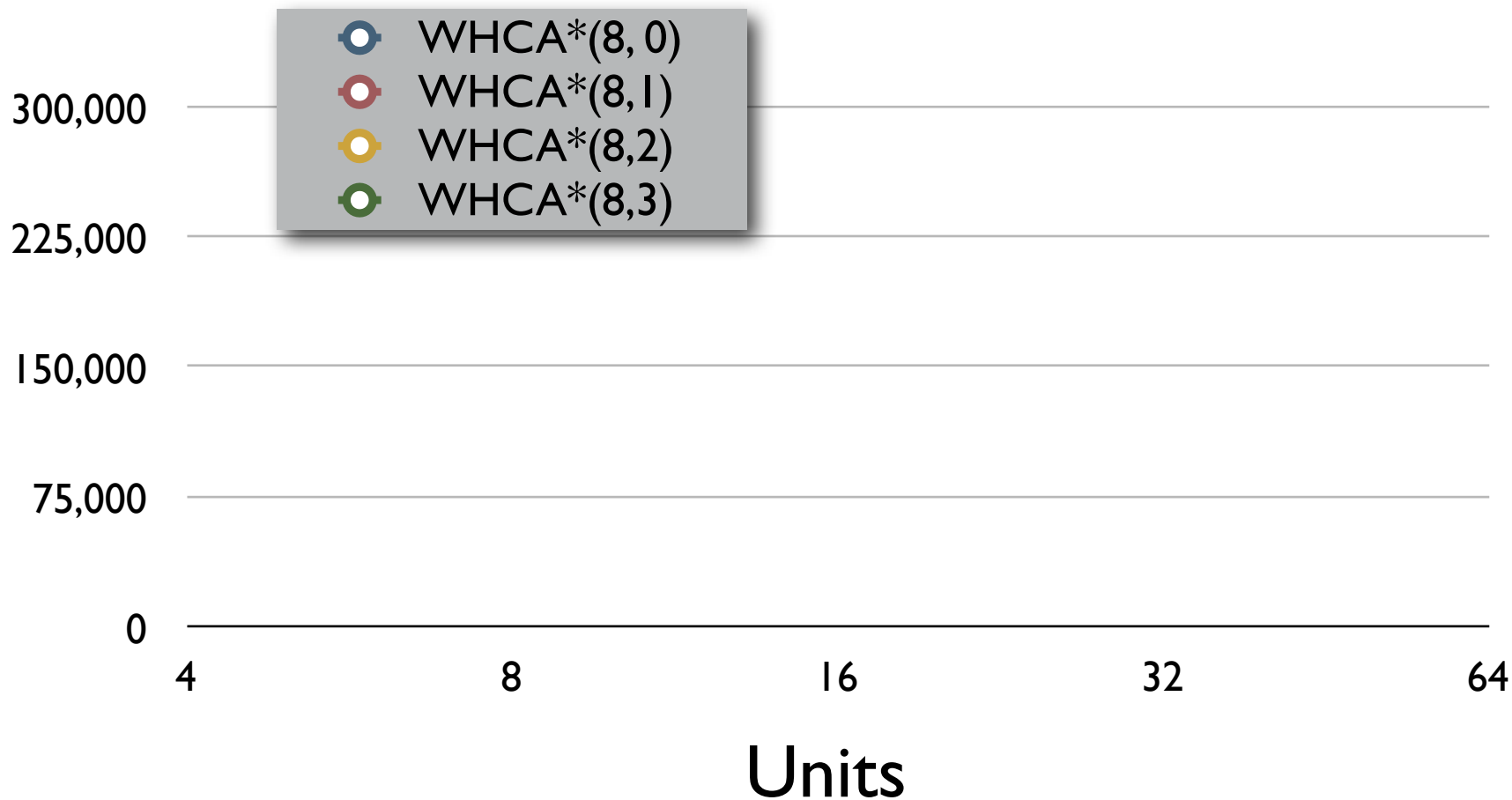
- Run algorithms on 256x256 map
  - Place units on opposite sides of map and ask them to cross sides
- Report 95%



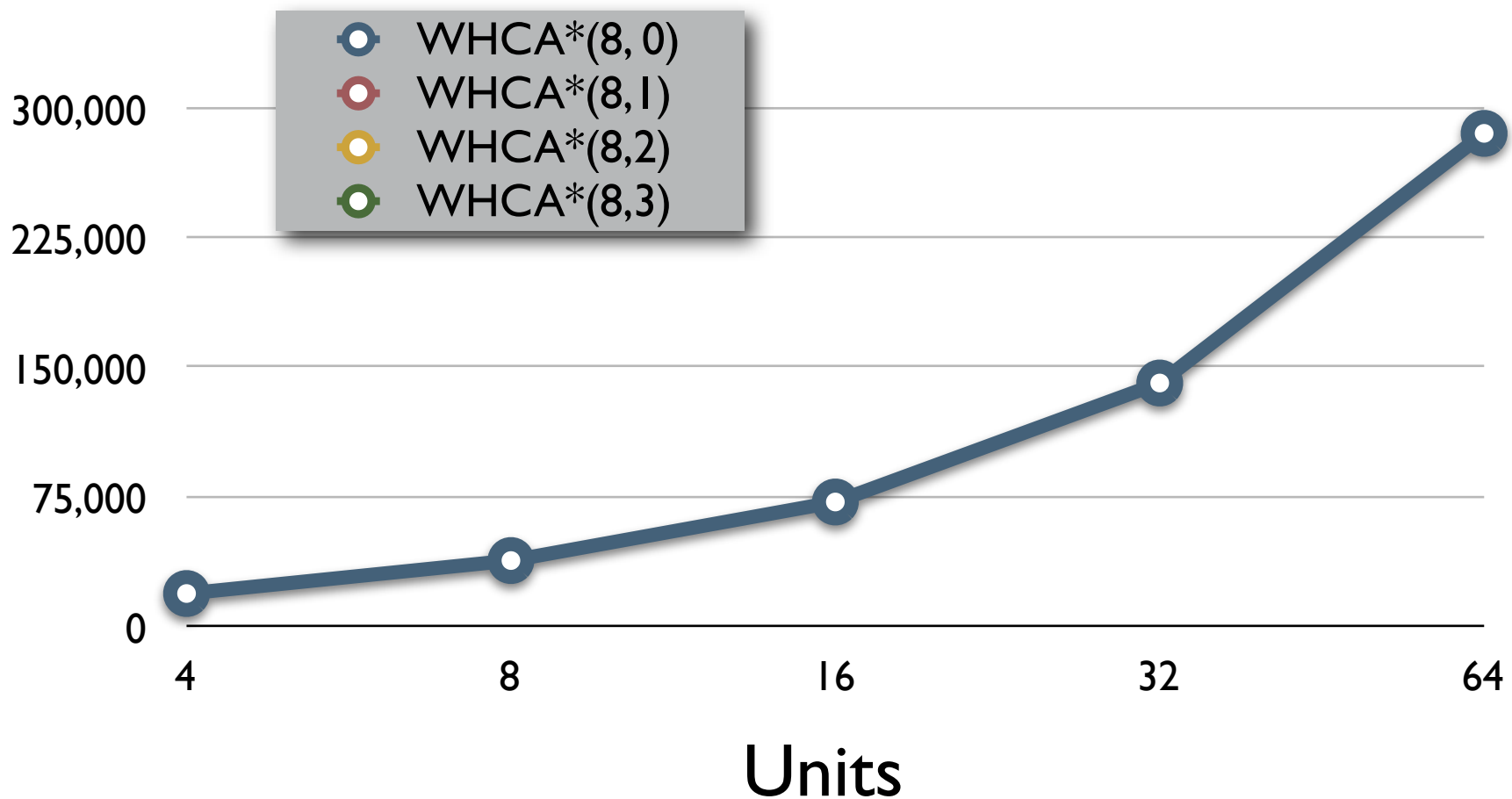
Memory Usage

Units

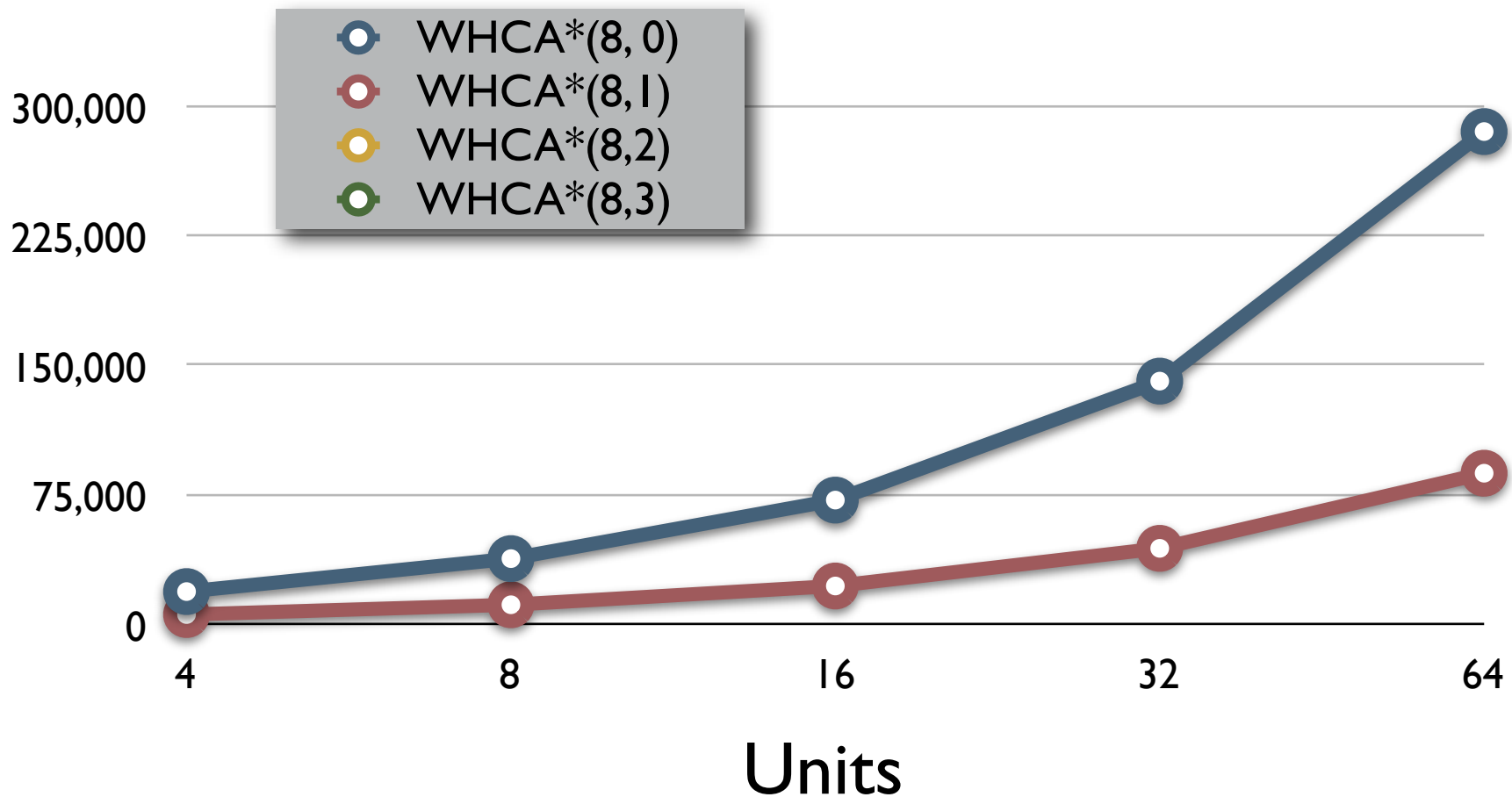
Memory Usage



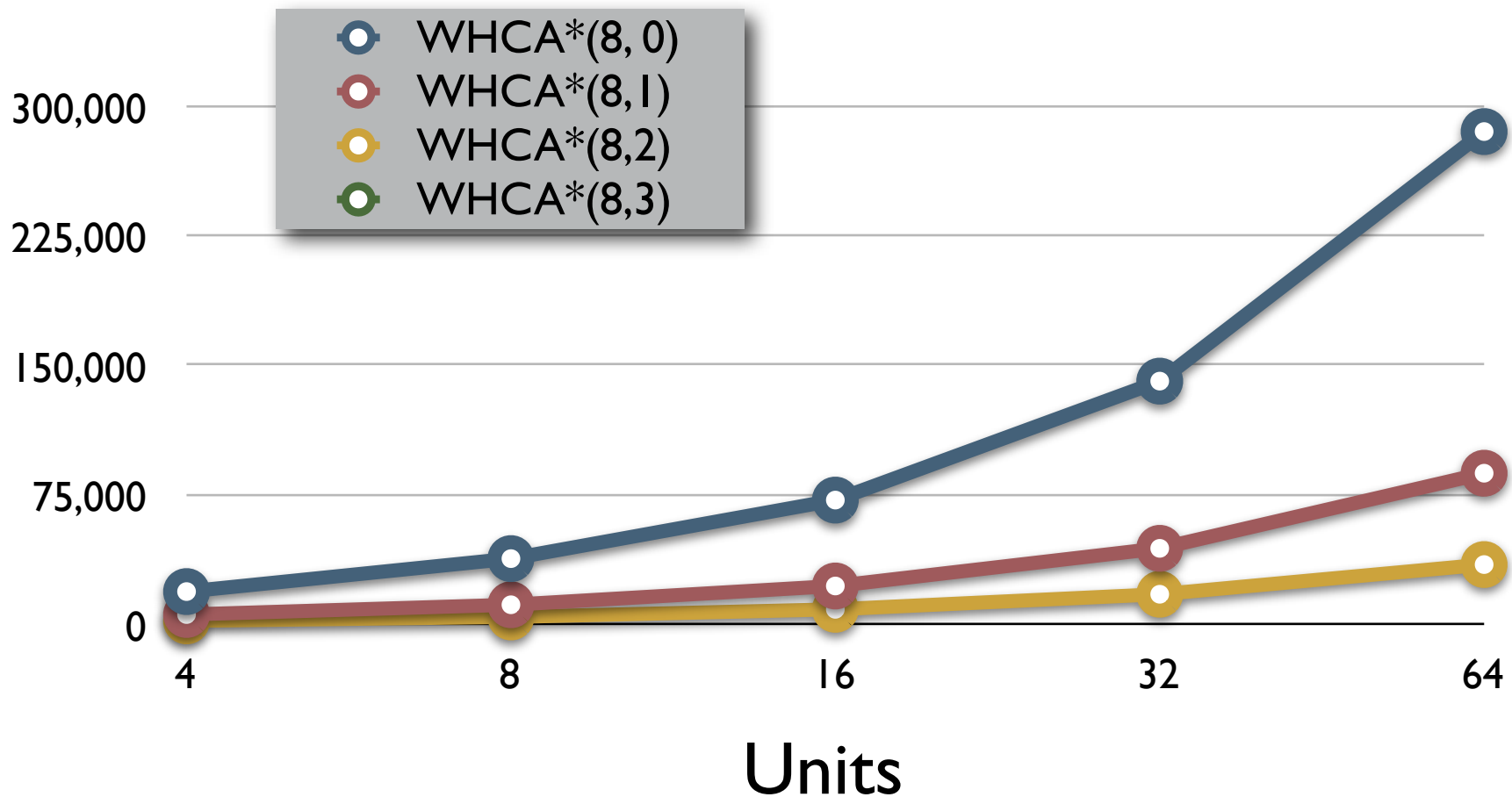
Memory Usage



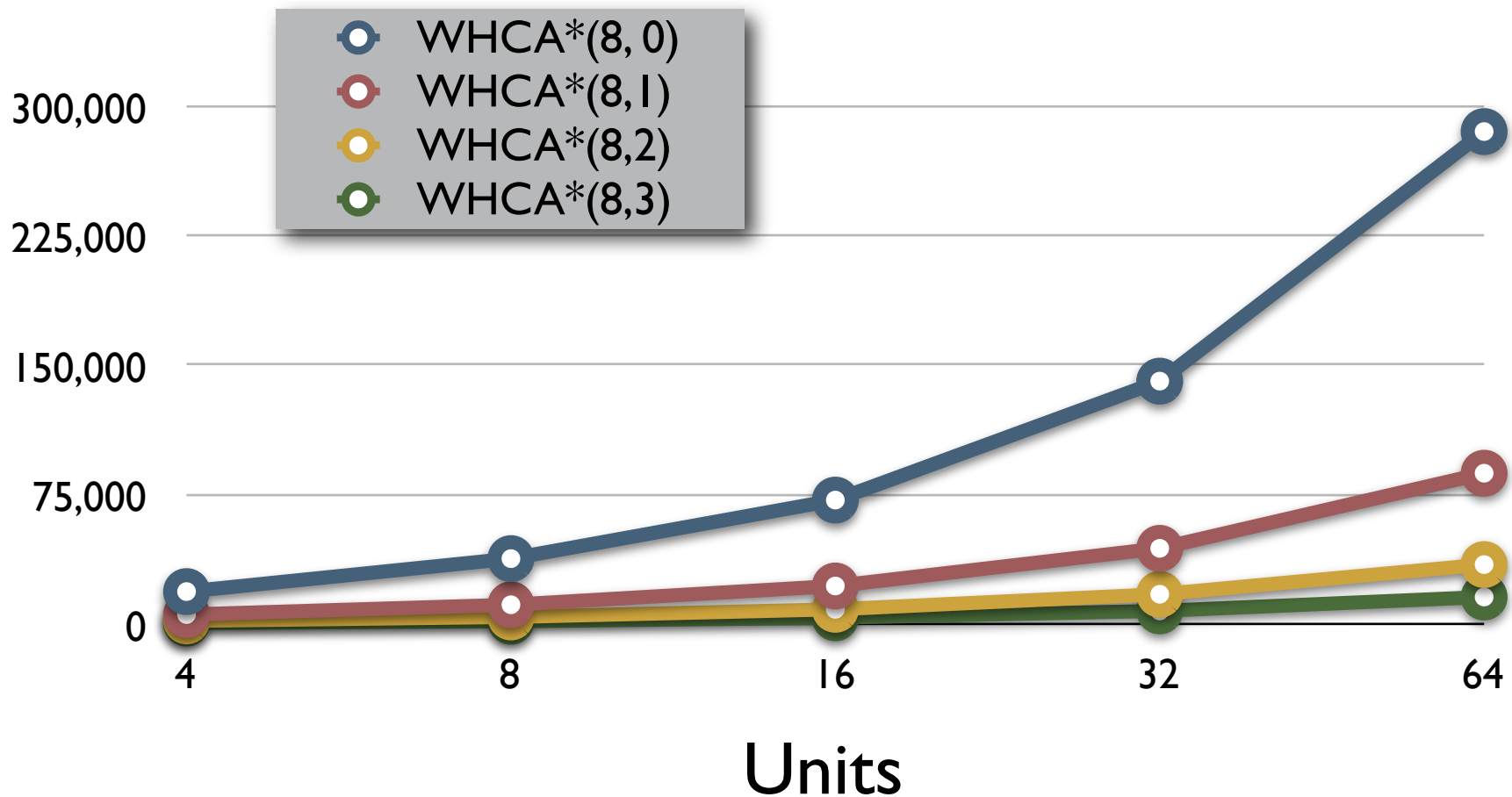
Memory Usage



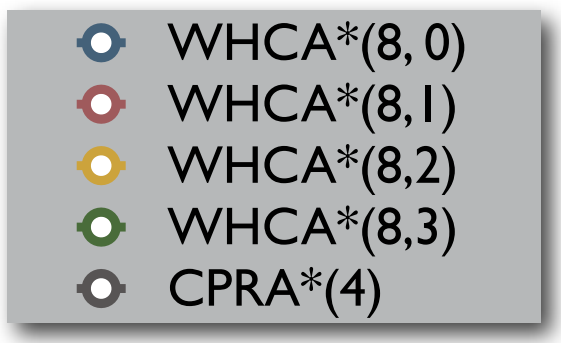
Memory Usage



Memory Usage

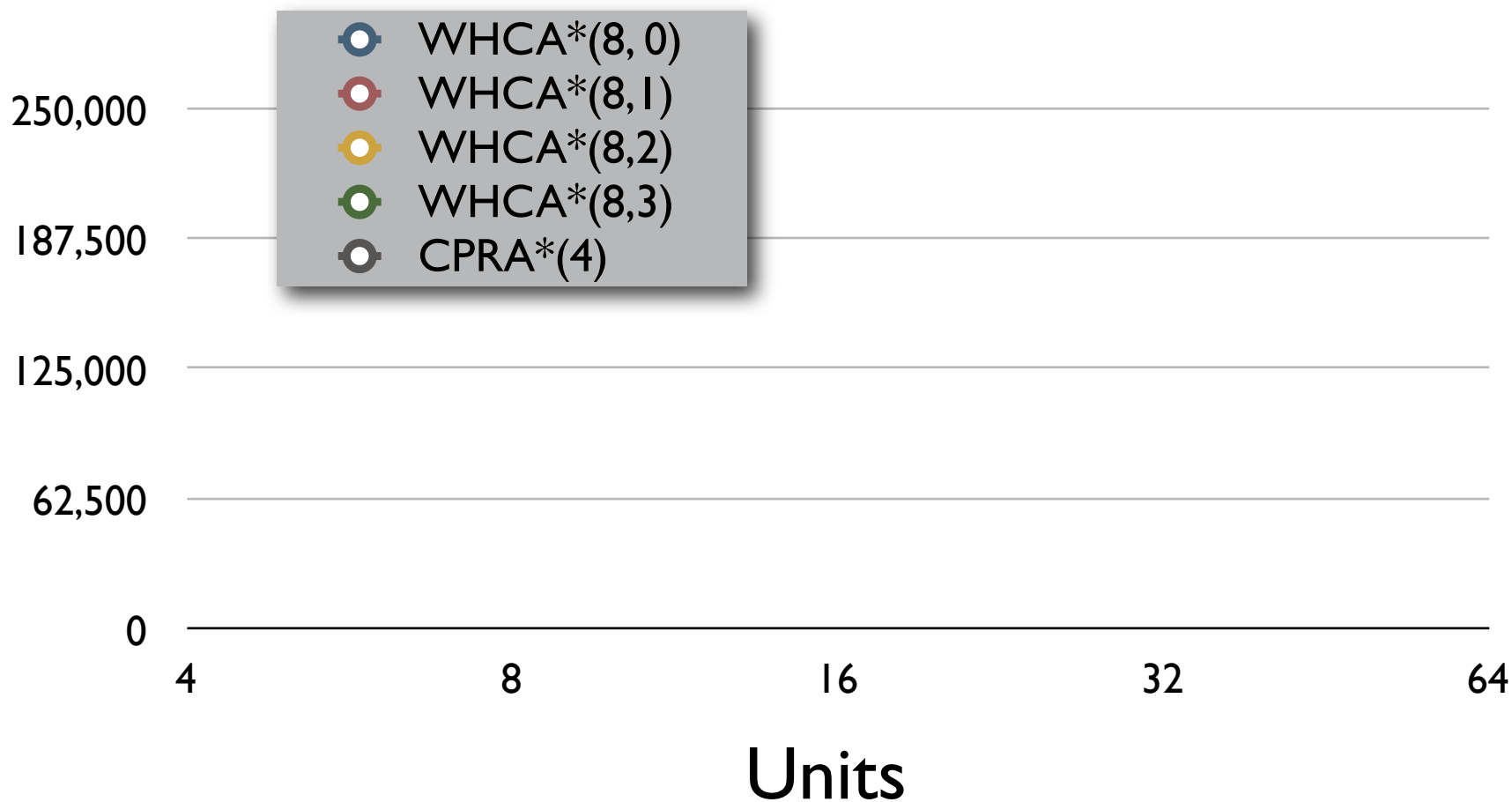


Nodes
First second

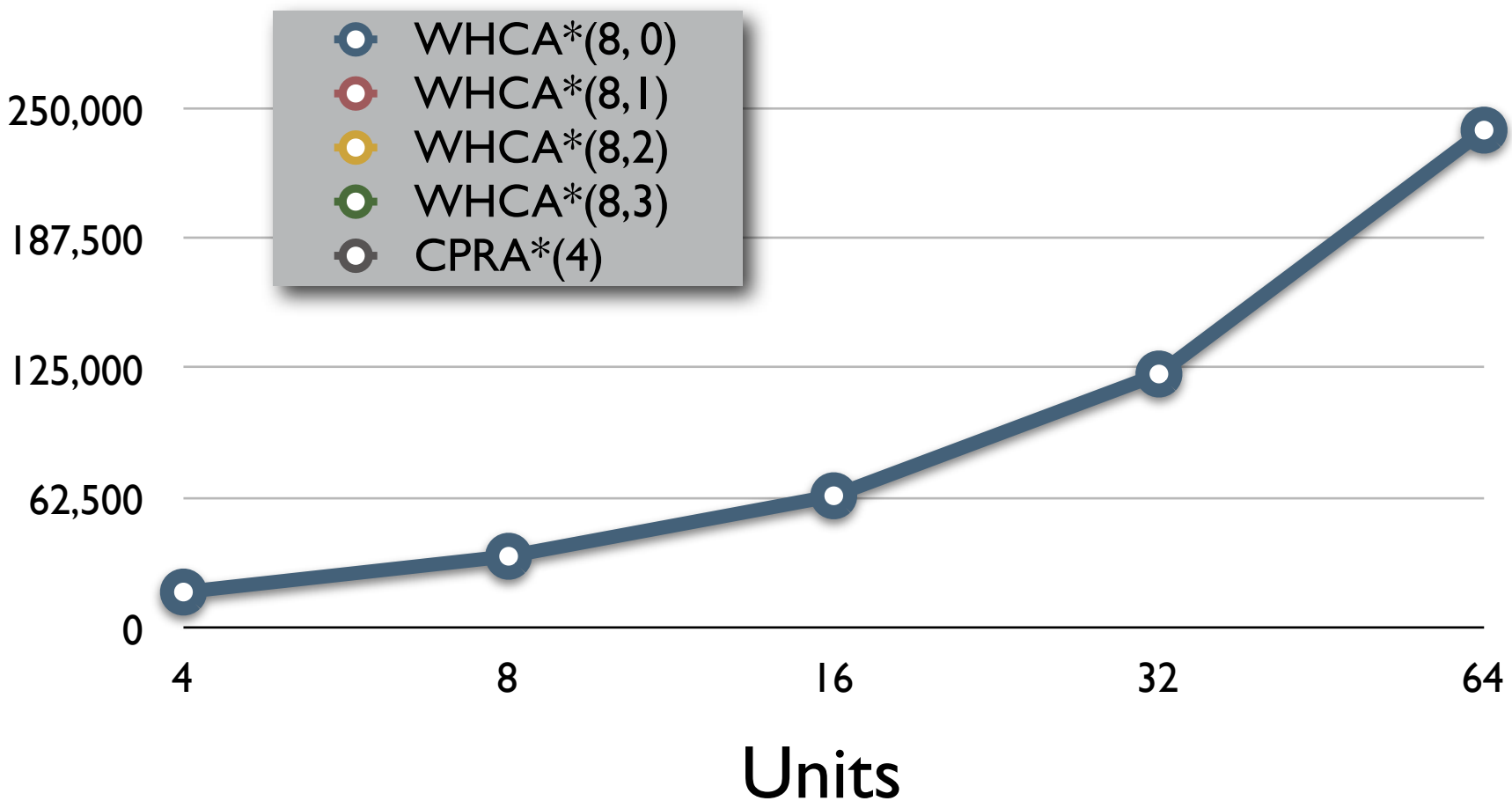


Units

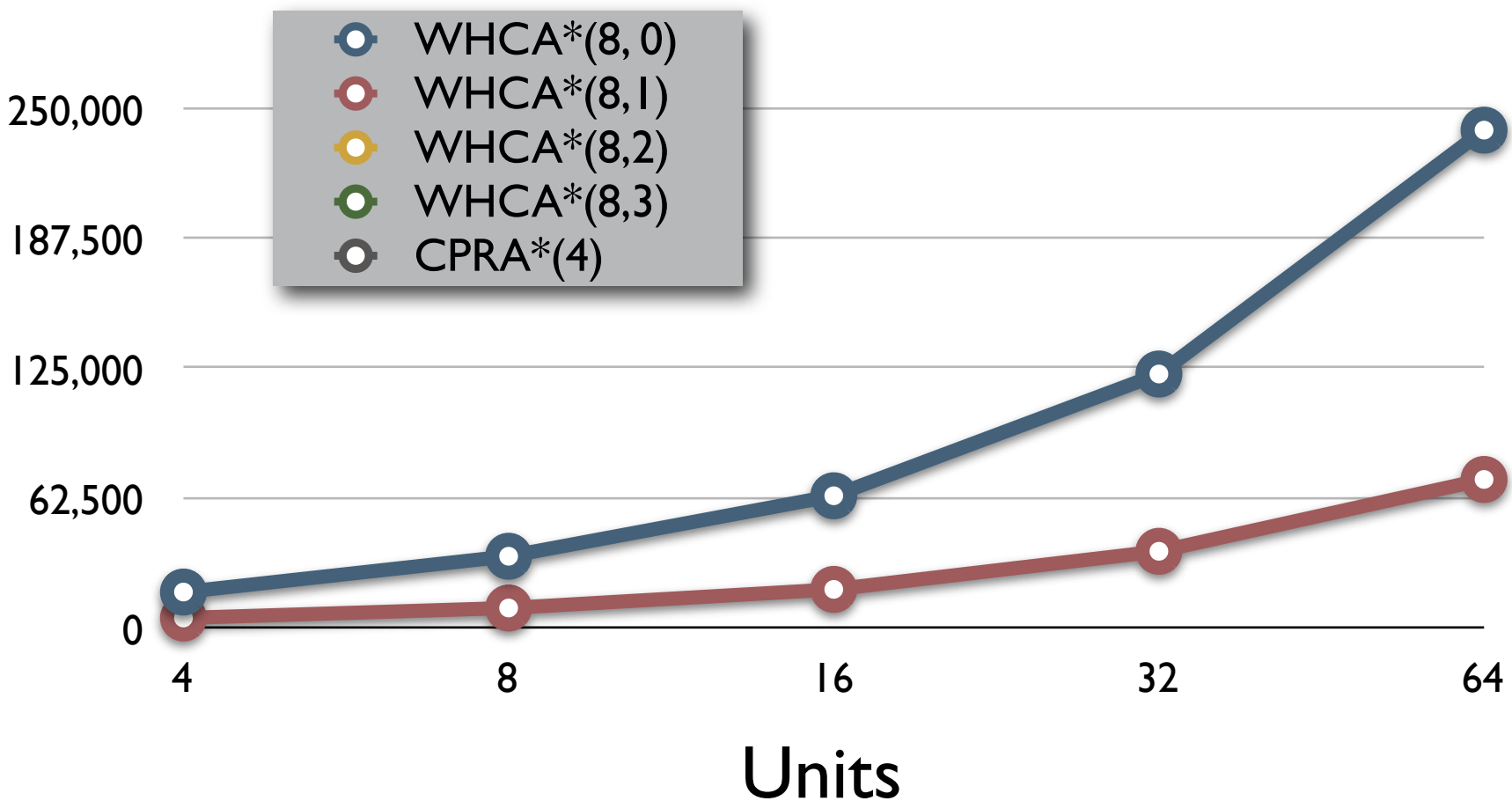
Nodes
First second



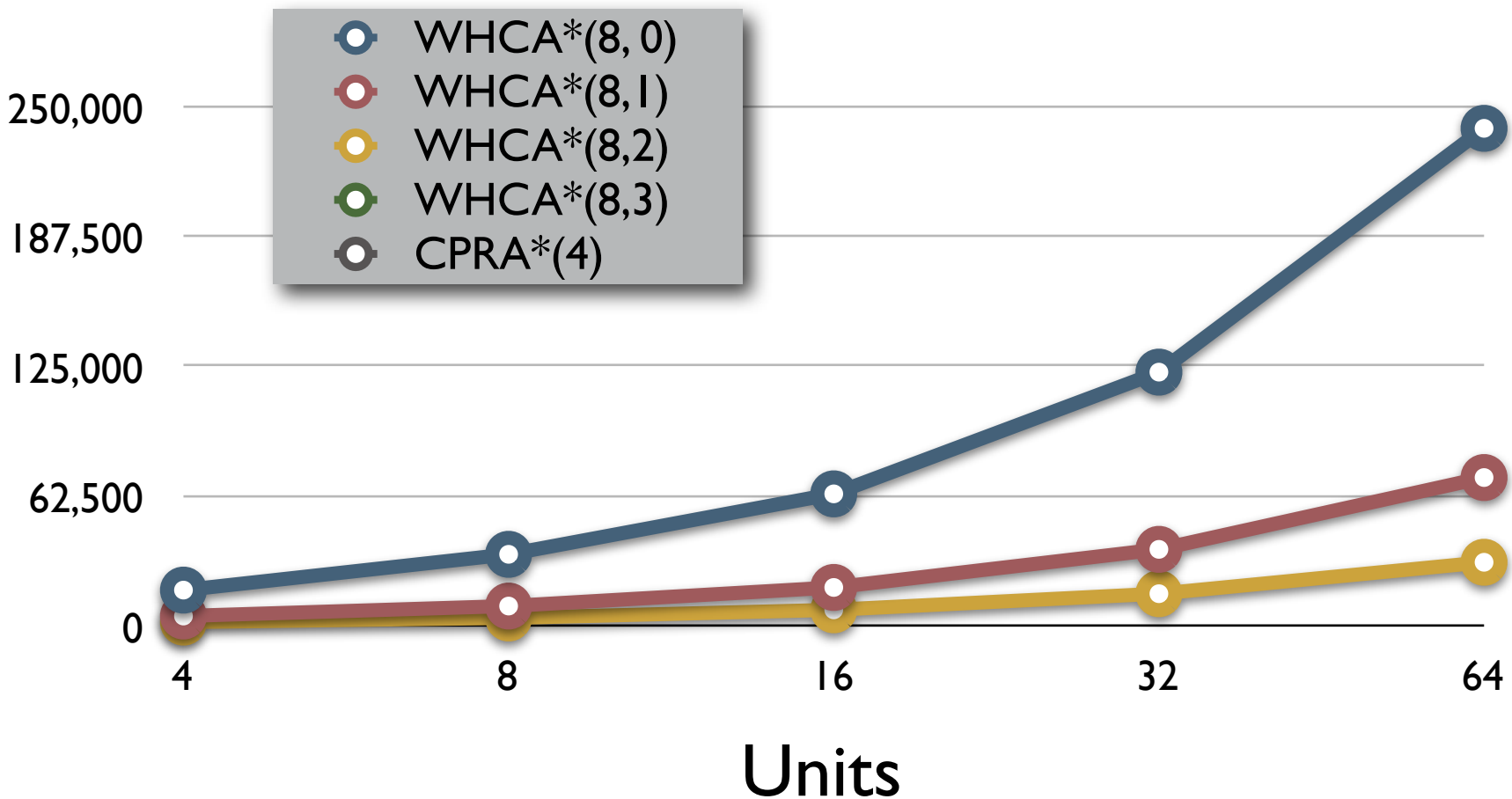
Nodes
First second



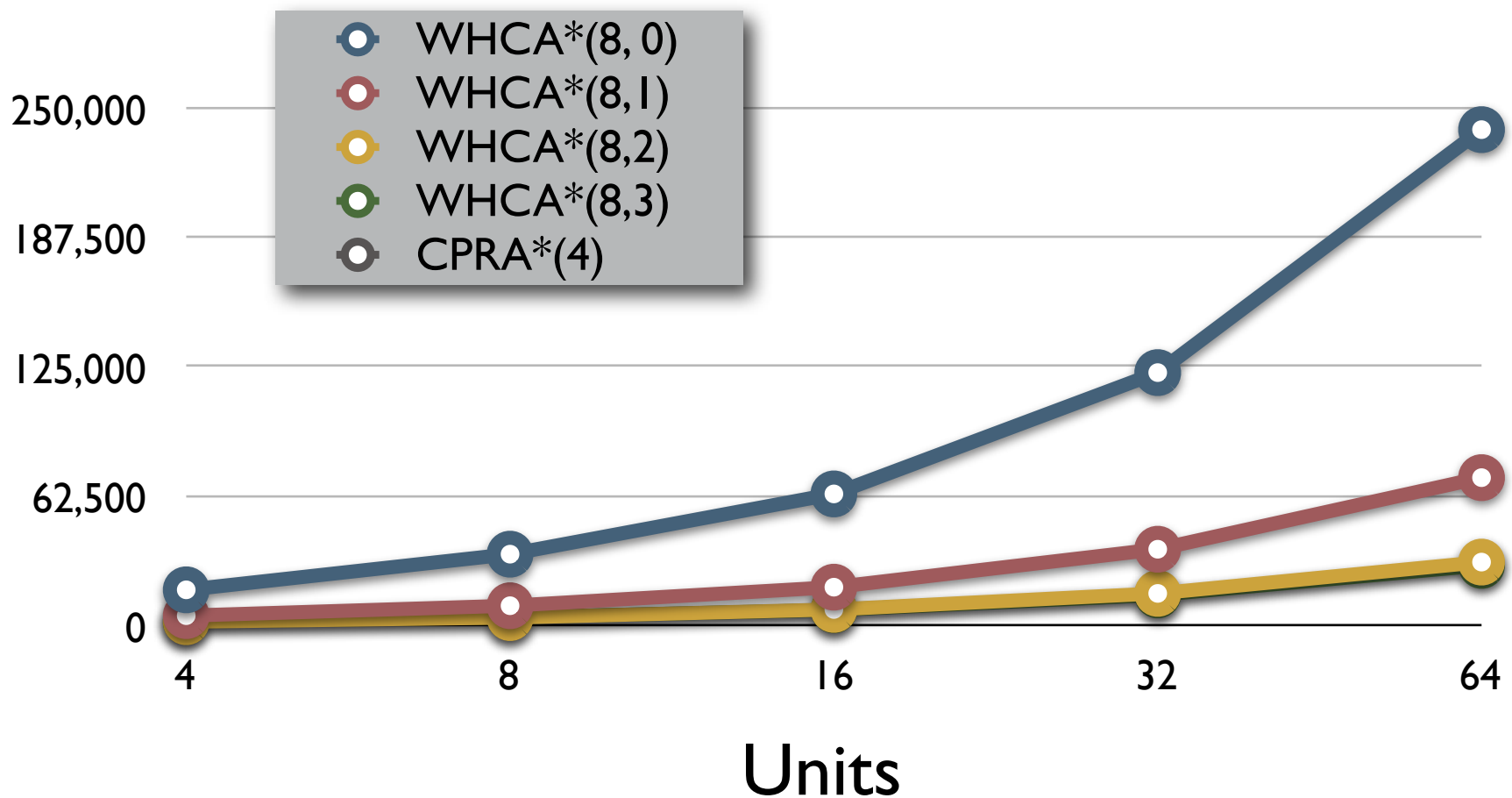
Nodes
First second



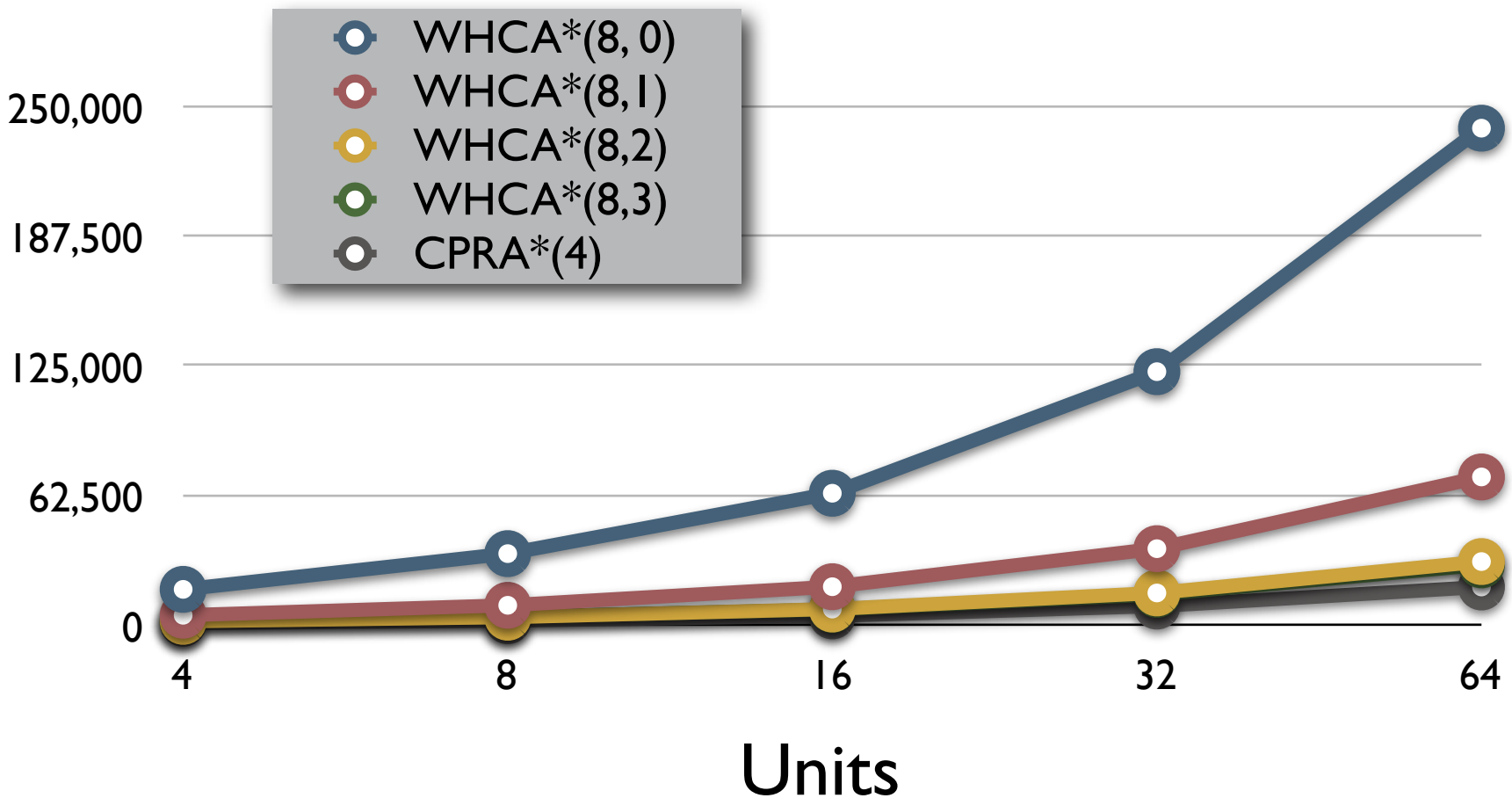
Nodes
First second



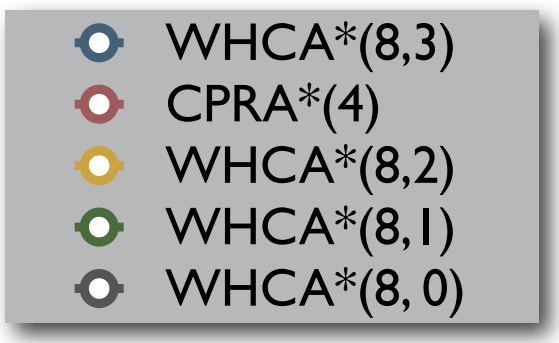
Nodes
First second



Nodes
First second

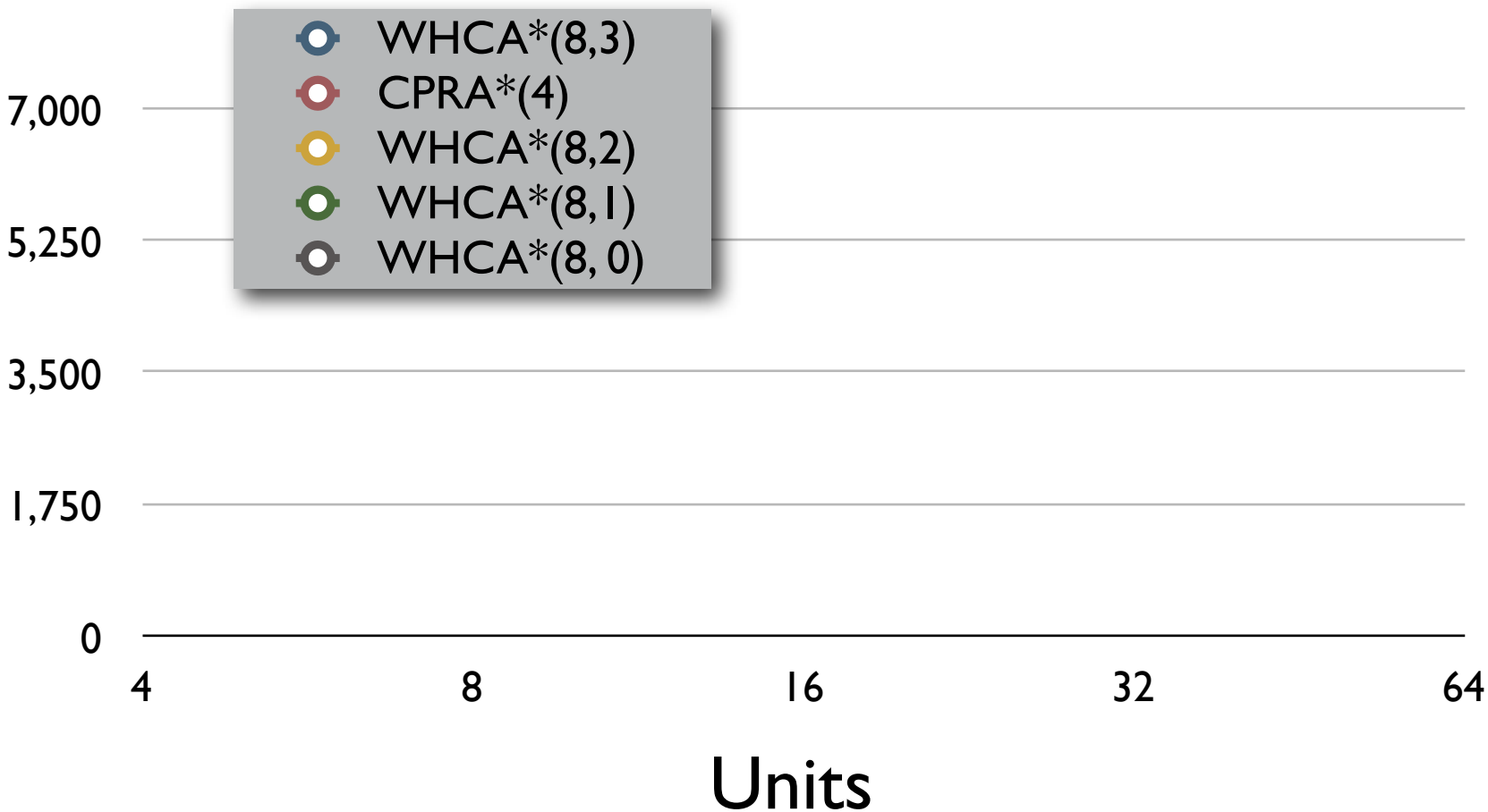


Nodes
Average per second

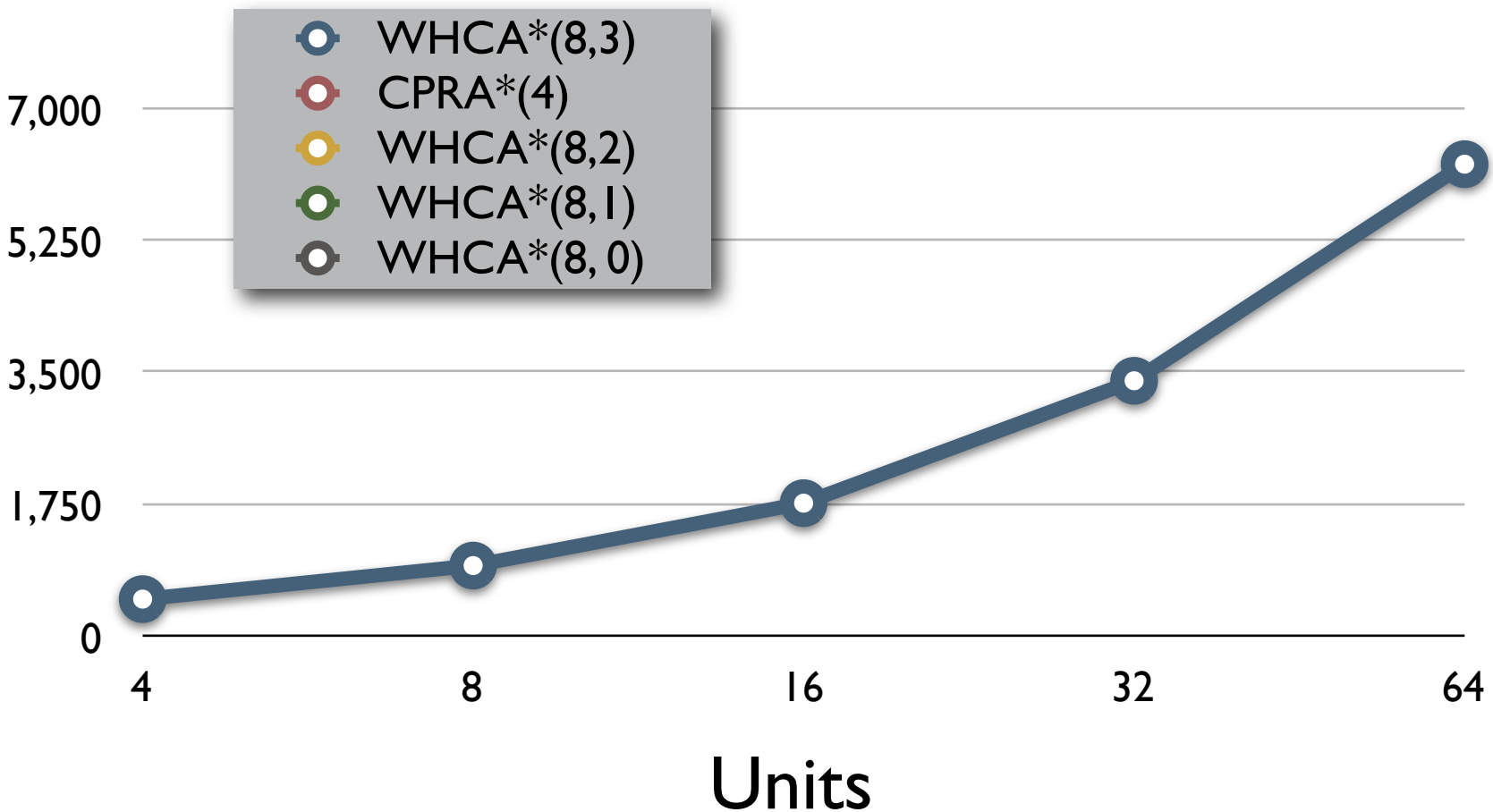


Units

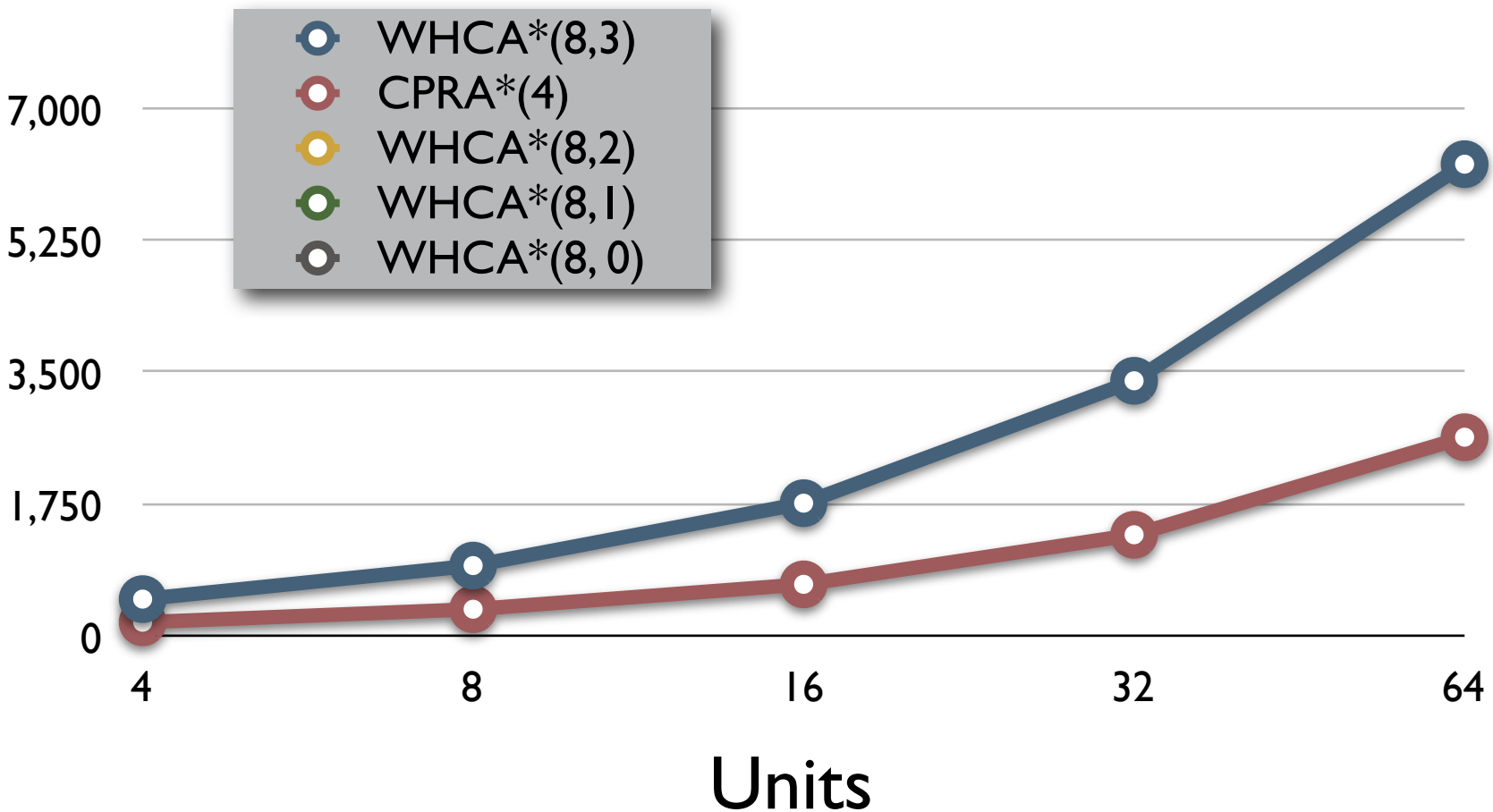
Nodes
Average per second



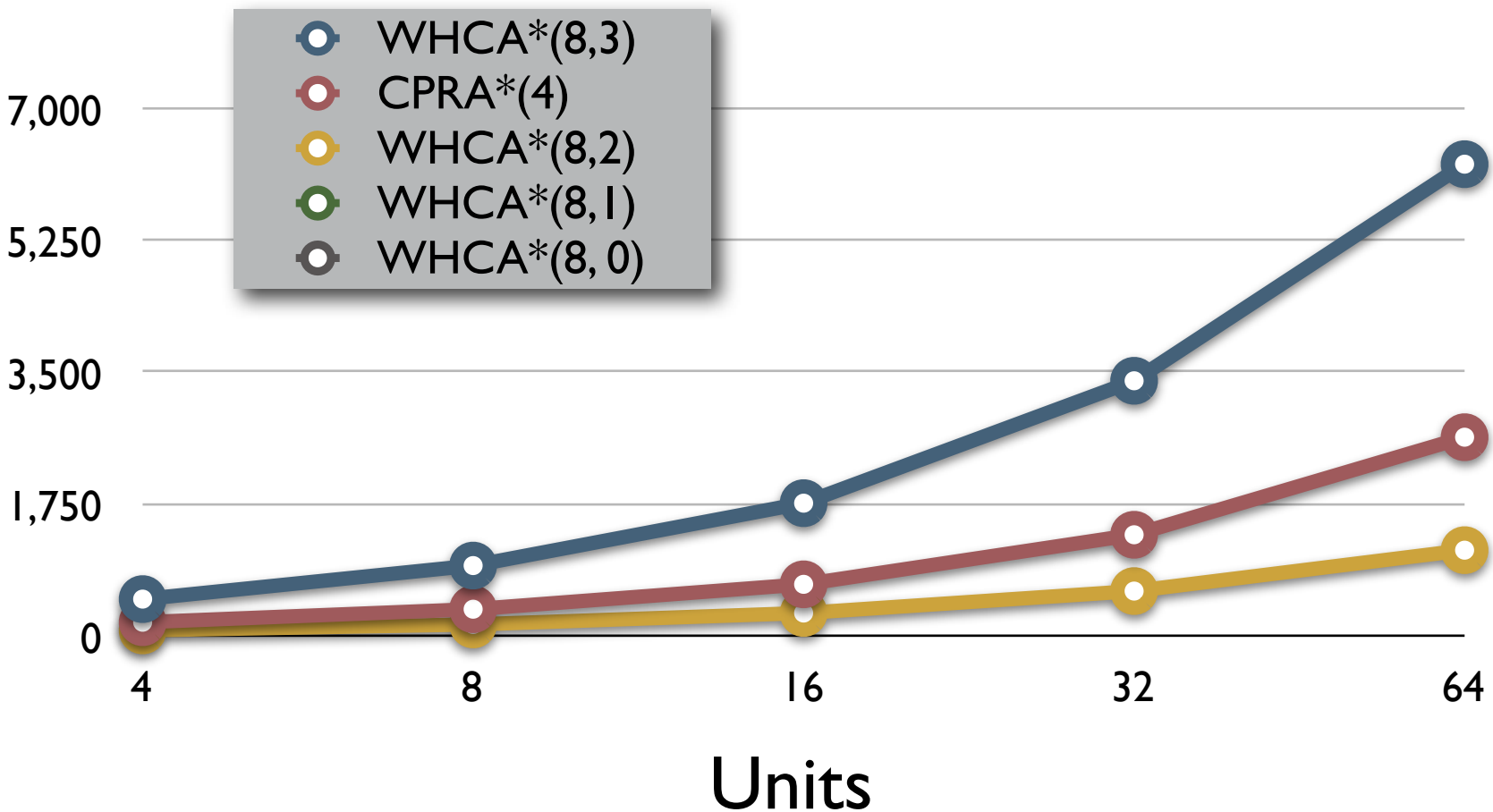
Nodes Average per second



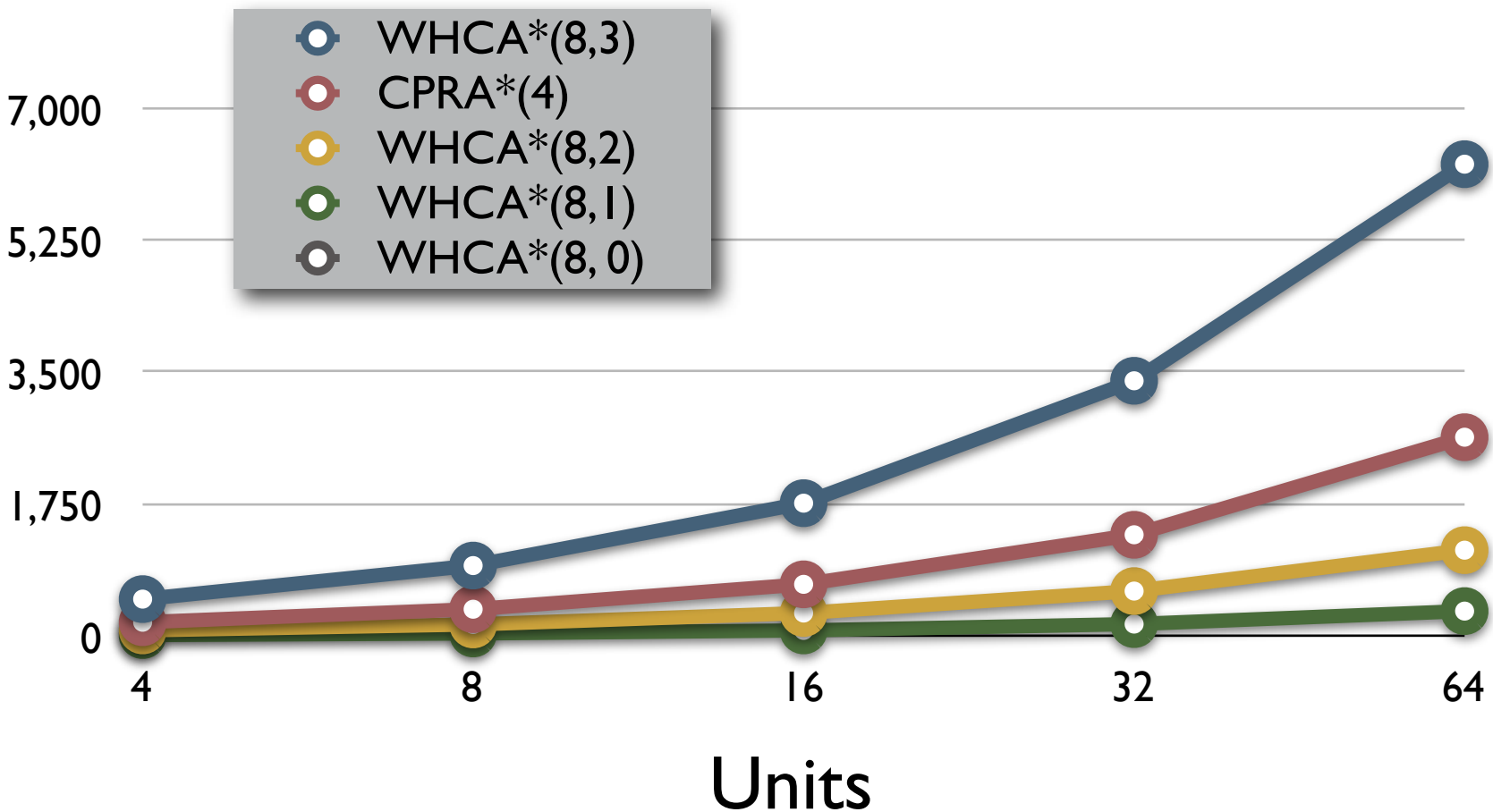
Nodes
Average per second



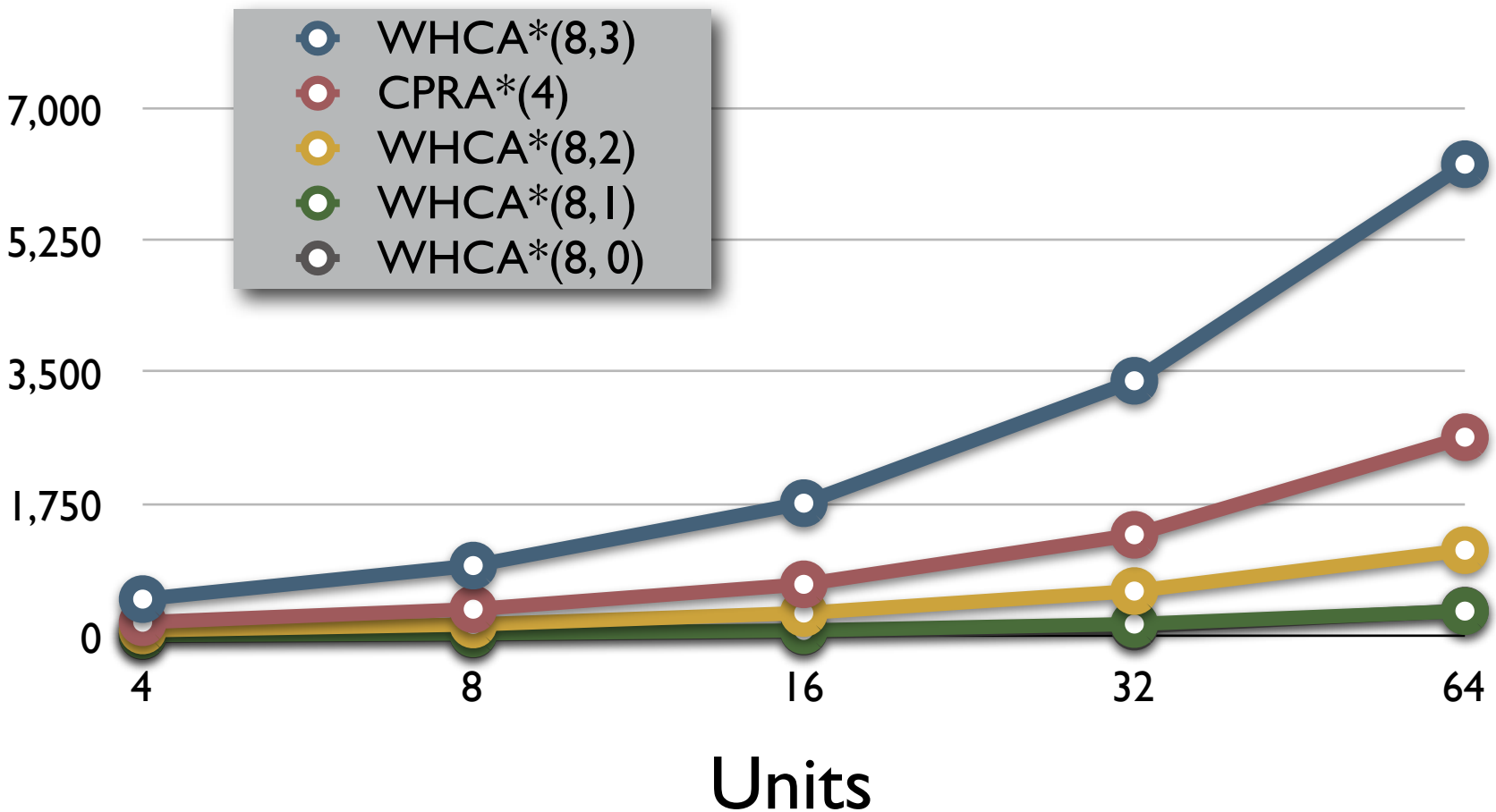
Nodes
Average per second



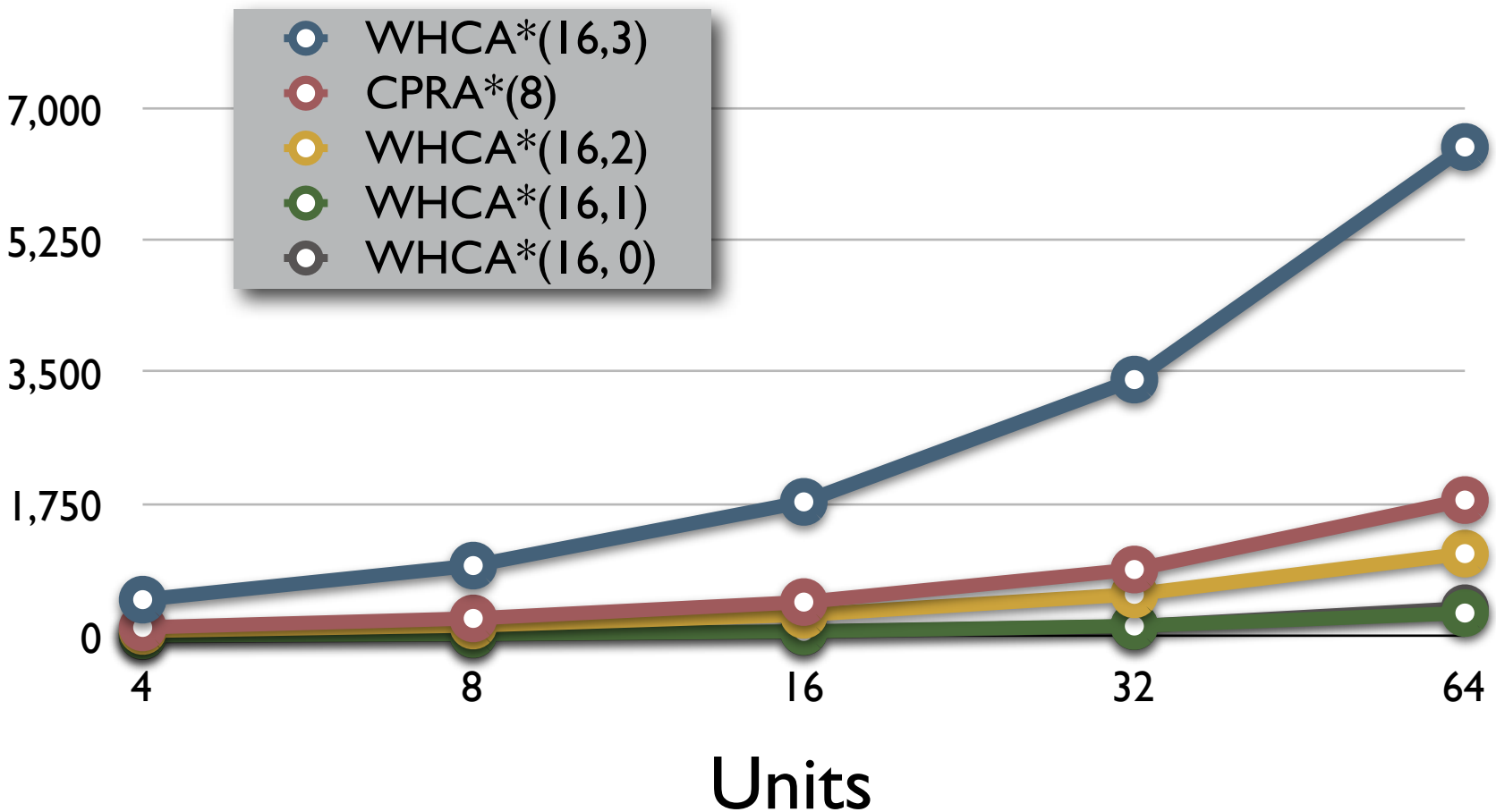
Nodes
Average per second



Nodes
Average per second



Nodes
Average per second



Summary

- Two new algorithms for cooperative pathfinding
  - Use abstract maps to reduce computational costs

Future Work

- Dynamically adjust reservation windows
  - Use congestion information
- Where does the user notice this?

Generalizing

- General technique for *n*-dimensional pathfinding problems
  - Solve problem in *n*-1 dimensional space
  - Use as heuristic in *n*-dimensional search
  - If possible use "lower resolution" version of *n*-1 dimensional problem

Thanks

- Markus Enzenberger

INFORMATICS iCORE CIRCLE OF RESEARCH EXCELLENCE