The Deployment of Fast A* Search
CPAIOR Master Class

Prof. Nathan Sturtevant
University of Denver

UNIVERSITY OF DENVER

Outline

- Fundamentals
  - Best-first search & heuristics
- Abstraction
  - Deployment in commercial software
- Heuristics
  - Heuristics and Maximum Variance Unfolding

Best-First Search

UNIVERSITY OF DENVER

Problem Size Comment

- Generally referring to interesting class of problems:
  - The faster search the better
  - Small enough to fit in memory
  - Not enough memory for all-pairs shortest paths
  - *Solving different instances on same graph each time*
- Examples:
  - Maps / navigation
  - Video games

UNIVERSITY OF DENVER

State Space Search

- We assume:
  - A start state
  - A successor function
  - A goal state or a goal test function

UNIVERSITY OF DENVER

Algorithm Measures

- Complete
  - Is it guaranteed to find a solution if one exists
- Optimal
  - Is it guaranteed the find the optimal solution

UNIVERSITY OF DENVER

Best-First Algorithms

- Choose a metric of best
  - Expand states in order from best to worst

- Requires:
  - Sorted OPEN list/priority queue
  - CLOSED list

UNIVERSITY OF DENVER

Definitions

- Node is *expanded* when taken off queue
- Node is *generated* when put on queue

- *g*-cost is the cost from the start to the current node
- *c*(*a*, *b*) is the edge cost between *a* and *b*
  - Sometimes also designates optimal path cost

Best-First Algorithms

- Best-First Pseudo-Code
  1. Put start on OPEN
  2. While(OPEN is not empty)
     1. Pop best node *n* from OPEN
     2. if (*n* == *goal*) return path(*n*, *goal*)
     3. for each child of *n* // generate children
        1.put/update value on OPEN/CLOSED
  3. return NO PATH

UNIVERSITY OF DENVER

Best-First Algorithms: Updating children

- Best-First child update
  1. If child on OPEN, and new cost is less
     1. Update cost and parent pointer
  2. If child on CLOSED
     1. Ignore*
  3. Otherwise
     1. Add to OPEN list

UNIVERSITY OF DENVER

Uniform-Cost Search

- Dijkstra's algorithm
  - Best-first is the *g*-cost

Dijkstra - Detail

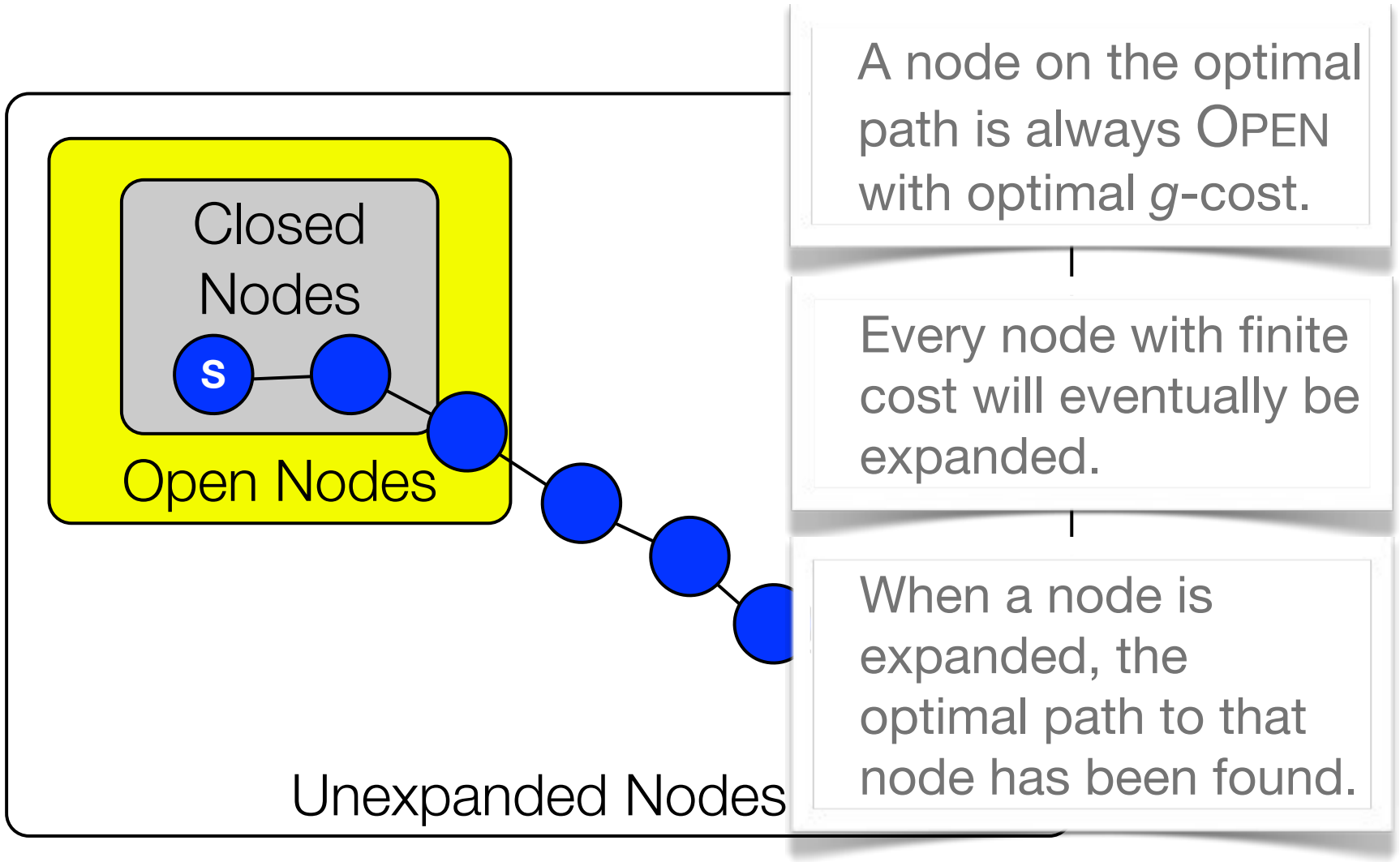
Dijkstra

UNIVERSITY OF DENVER

Dijkstra's Completeness

- Complete? / Will it find a solution?
  - Finite Graph: yes
  - Infinite Graph
    - Must have finite cost path to goal
    - Edge costs at least epsilon
    - Cannot have negative cost loops

Dijkstra's Optimality: High Level View



UNIVERSITY OF DENVER

Dijkstra's Time & Space Complexity

- In exponential domains assumed:
  - Branching factor *b*
  - Minimum edge cost *e*
  - $O(b^{c/e})$ -- *c* - solution cost
- In AI we ignore data structure overheads
  - Assume that costs can be bucketed
- Harder to analyze domains that fit in memory

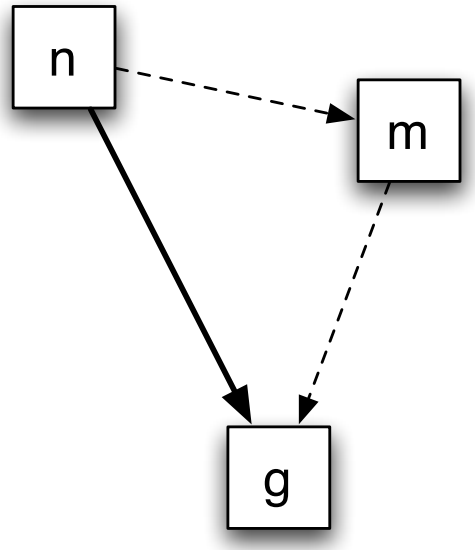
UNIVERSITY OF DENVER

Heuristic

- What is a heuristic?
  - An estimate of the cost from a given state to the goal
- Where do they [traditionally] come from?
  - Solutions to abstracted problem

Properties of Heuristics

- Perfect heuristic: h*(n)
- Admissible: h(n) ≤ h*(n) for all n
- Consistent:
  - h(n) ≤ c(n, m) + h(m) (directed)
  - c(n, m) ≥ |h(n) - h(m)| (undirected)



- Local consistency implies global consistency
- Consistency implies admissibility

UNIVERSITY OF DENVER

f-cost Monotonically Non-decreasing

- *f*-cost: *g-cost* + *h*-cost
- $h(n) \leq h(m) + c(n, m)$ [*consistency*]
- $g(n) + h(n) \leq g(n) + h(m) + c(n, m)$
- $f(n) \leq g(m) + h(m) = f(m)$

- *f*-cost is monotonic non-decreasing

UNIVERSITY OF DENVER

Pure Heuristic Search / Greedy Best-first Search

- Best-first is the *h*-cost
- Complete?
  - Only on finite graph
- Optimal?
  - No

Pure Heuristic Search

UNIVERSITY OF DENVER

A*

- Best-first is the *f*-cost
  - $f = g + h$
  - *f* is an estimate of the complete path length
- Optimality?
  - Depends on the heuristic

UNIVERSITY OF DENVER

A* Completeness

- Complete? / Will it find a solution?
  - Finite Graph: yes
  - Infinite Graph
    - Must have finite cost path to goal
    - Edge costs at least epsilon
    - Cannot have negative cost loops
  - Finite heuristic cost

UNIVERSITY OF DENVER

A* Optimality

- Same conditions as Dijkstra
- Plus one of:
  - Consistent Heuristic
  - Admissible heuristic
    - Need to change child update rule

UNIVERSITY OF DENVER

A* - Analysis

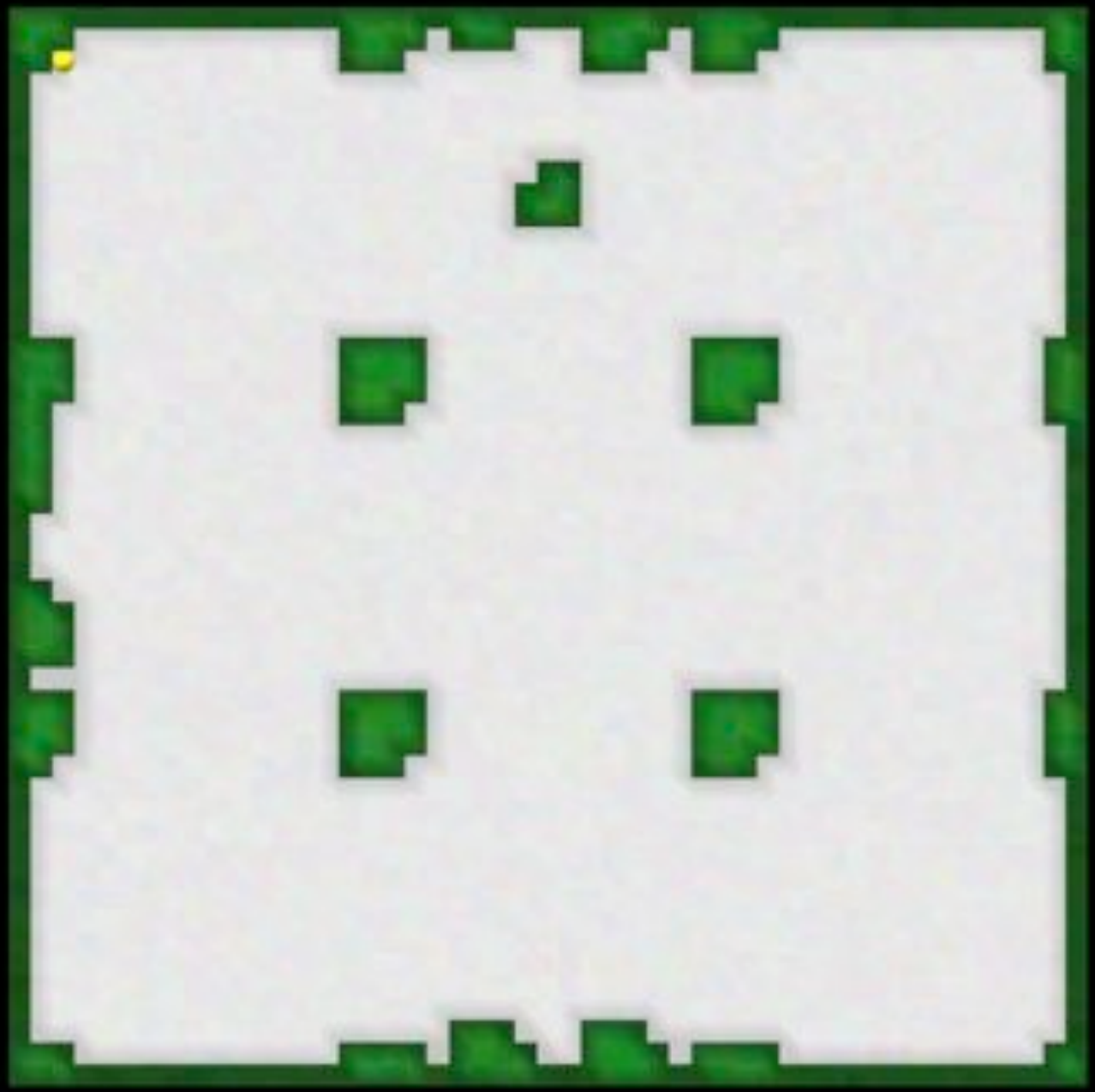
- Worst case: h(n) = 0 for all *n*
  - Same as Dijkstra $O(b^{c/e})$
- Best case: h(n) = h*(n) for all *n*
  - Will go straight to the goal(?)

UNIVERSITY OF DENVER

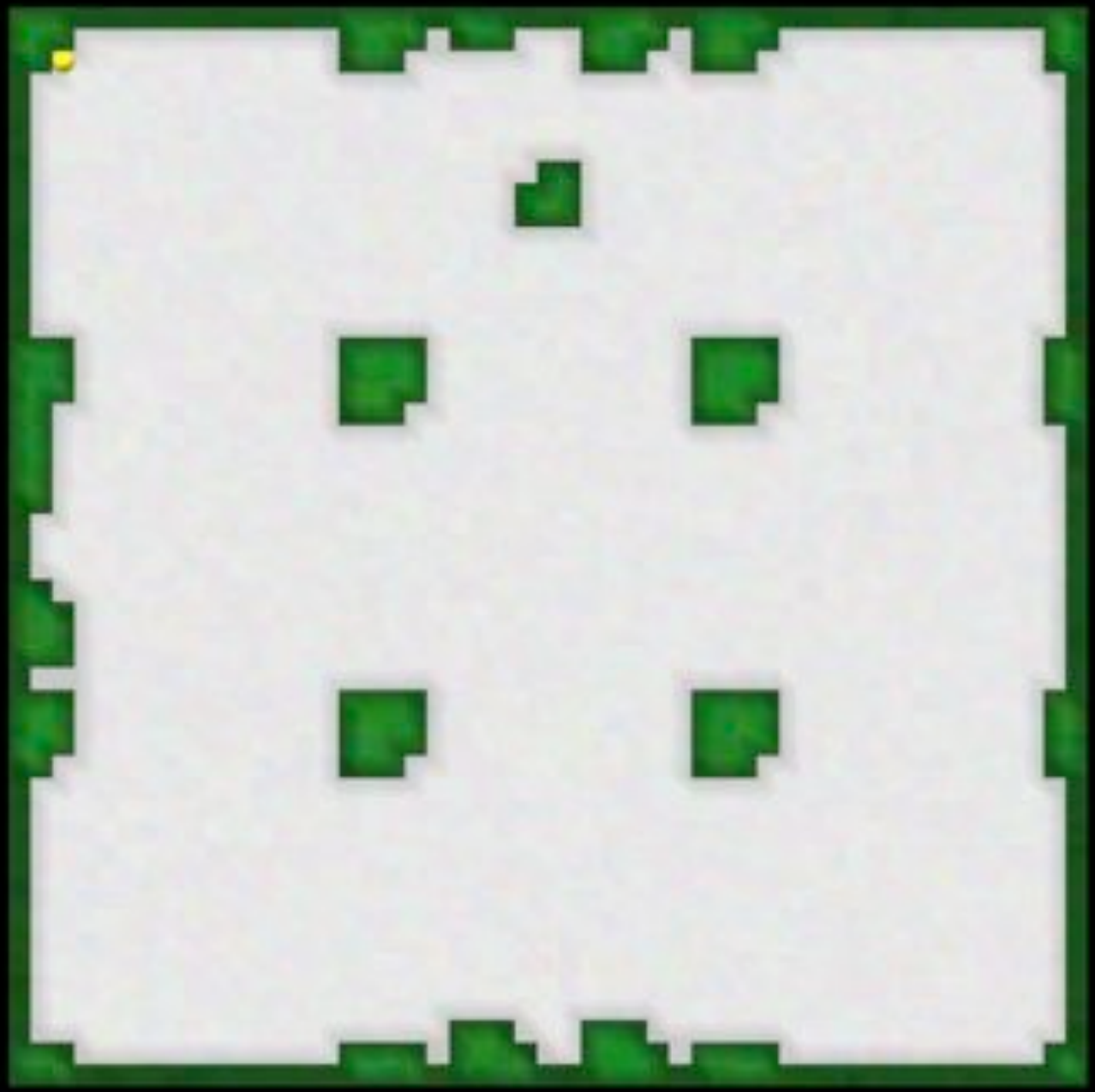
A* implementation details

- Break ties towards states with higher g-cost
- If a successor has *f*-cost as good as the front of OPEN
  - Avoid the sorting operations
- Make sure heuristic matches problem representation
  - With 8-connected grids don't use straight-line heuristic

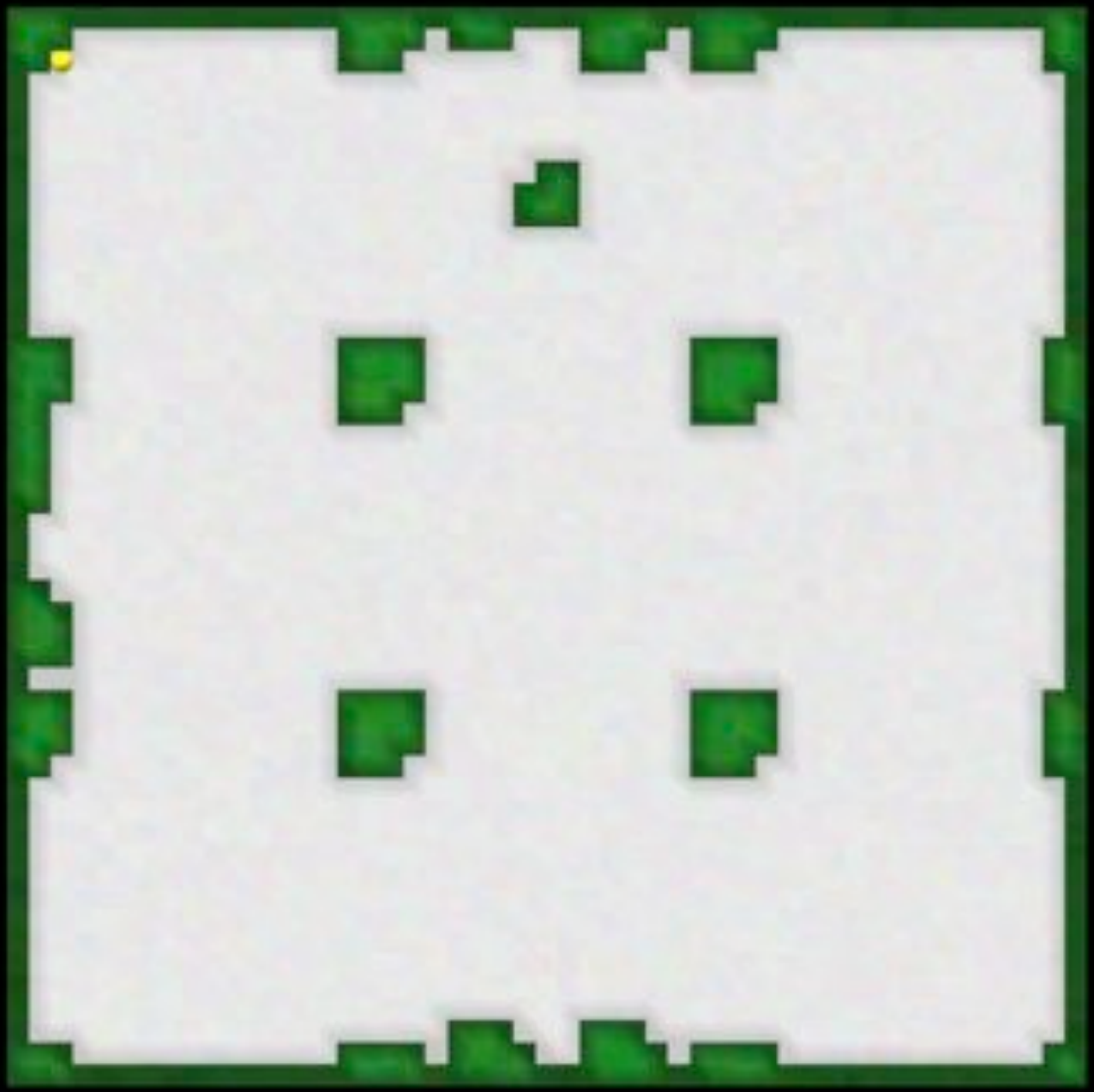
A* - Ties towards high g-costs



A* - Ties towards low g-costs



Heuristic: Euclidean Distance

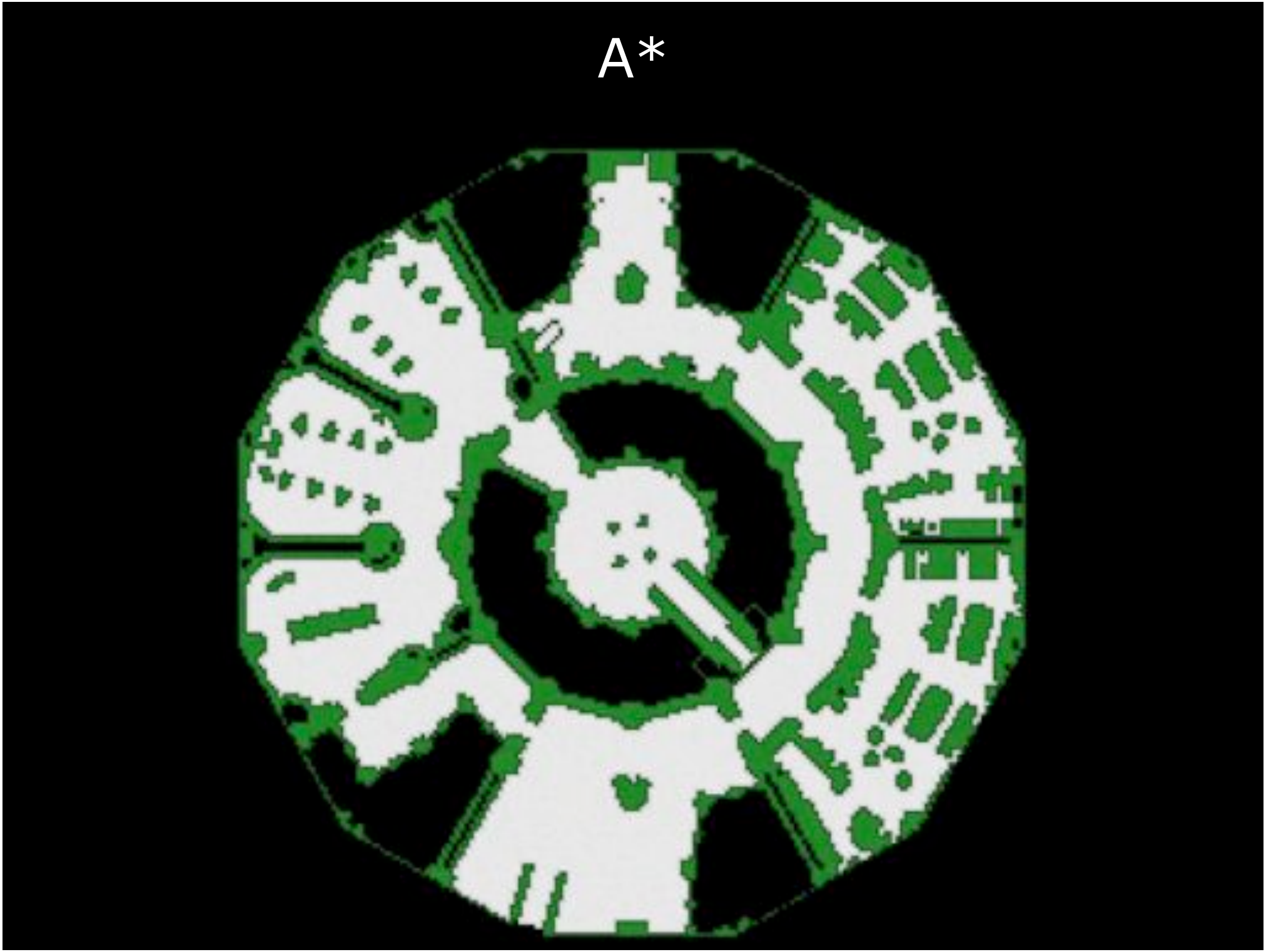


UNIVERSITY OF DENVER

A* is Optimal in Unique Node Expansions

- Suppose there was a better algorithm, E
  - There must be some node *n* which A* expands, but E doesn't
  - Re-arrange the problem and put the goal after *n*
  - E cannot be optimal/complete

A*

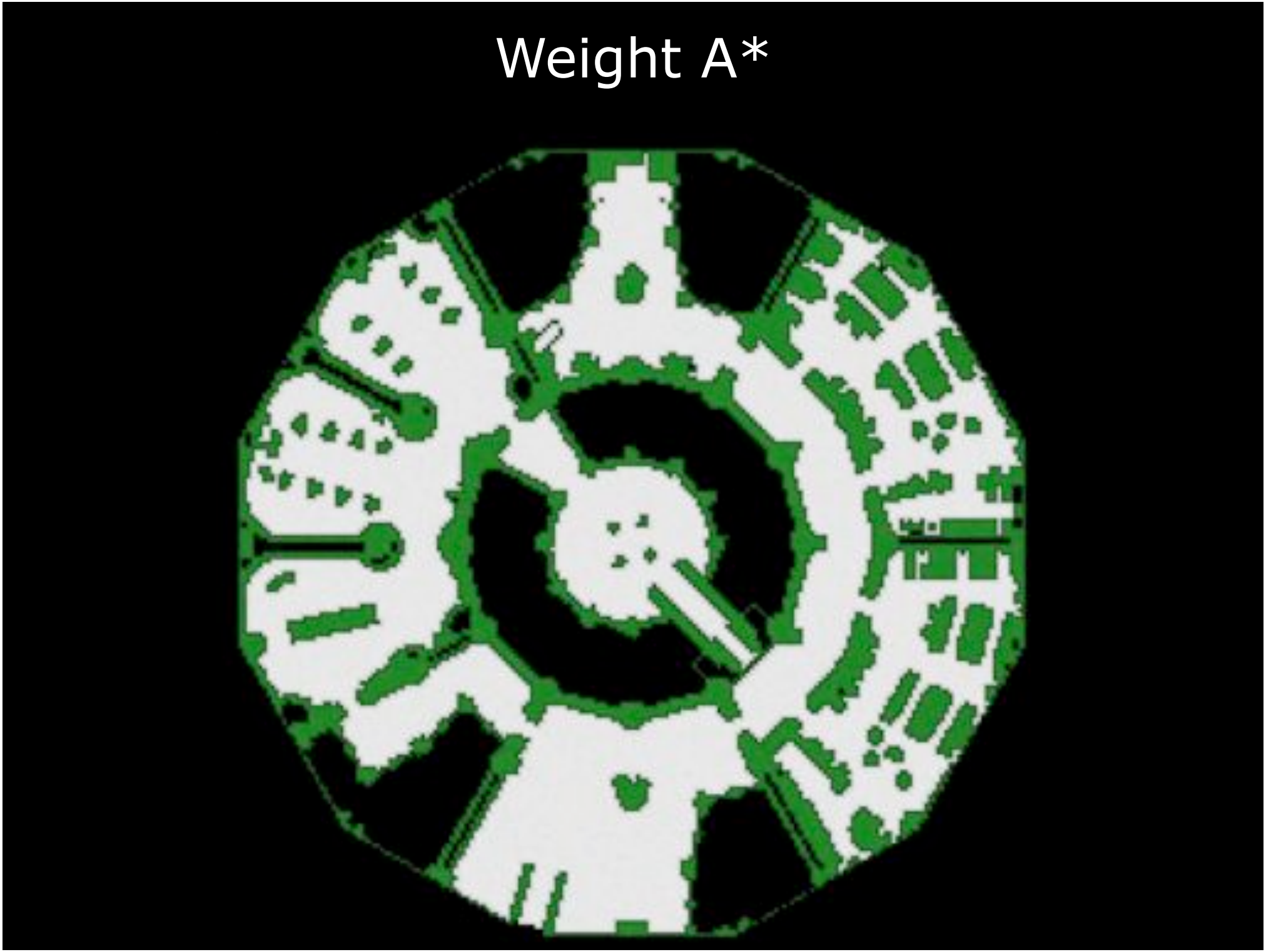


UNIVERSITY OF DENVER

Can we make A* go faster?

- A*: f(n) = g(n) + h(n)
- Weighted A*: f(n) = (1-*w*)·g(*n*) + *w*·h(*n*)
- If *w* = 1?
  - Pure Heuristic Search
- If *w* = 0?
  - Dijkstra's
- Similar to depth-bounded discrepancy search

Weight A*



UNIVERSITY OF DENVER

IDA*

- IDA* is an iterative deepening version of A*
  - Very useful in exponential domains
  - Cost of iterations is completely amortized in search
- Not used on problems with many cycles
- Similar to Limited Discrepancy Search

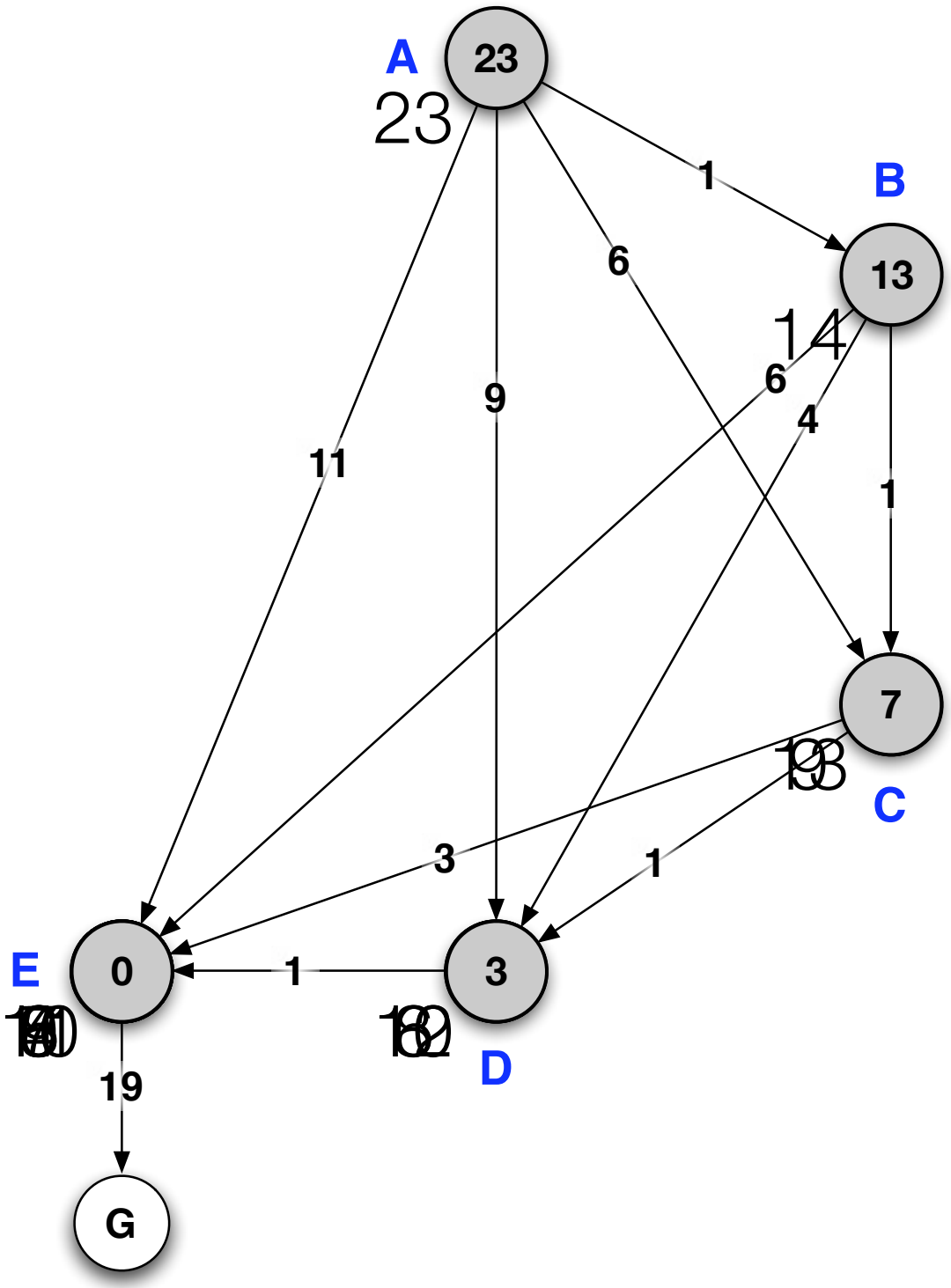
Inconsistent Heuristics

UNIVERSITY OF DENVER

When A* doesn't work well

- Inconsistent heuristics ignored for ~20 years
  - Not thought to occur in practice

- What happens when we have inconsistent heuristics?
  - Optimal path to goal will still be found
  - First path to a node is not necessarily optimal
  - Can re-open closed nodes



node	f	g	h
A	23	0	23
E	11	11	0
D	12	9	3
E	10	10	0
C	13	6	7
E	9	9	0
D	10	7	3
E	8	8	0
B	14	1	13
E	7	7	0
D	8	5	3
E	6	6	0
C	9	2	7
E	5	5	0
D	6	3	3
E	4	4	0
G	23	23	0

UNIVERSITY OF DENVER

Definition

- Let N be the number of nodes expanded by A*
  - N is the number of *nodes* not the number of *expansions*
- A* can re-expand nodes
  - Express in terms of N

UNIVERSITY OF DENVER

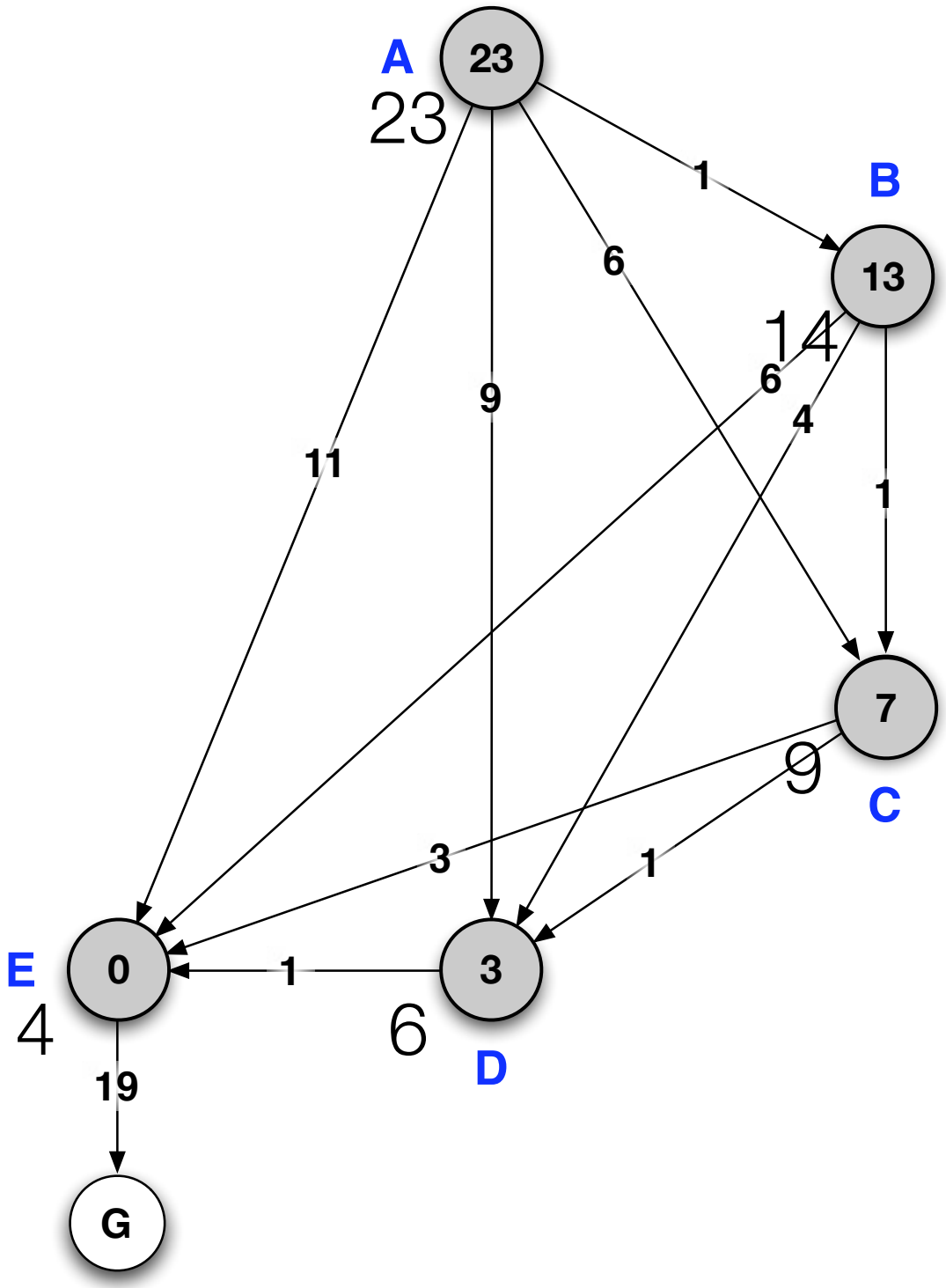
Analysis

- With N nodes, $(1 + 2^{N-2})$ nodes expanded!
  - 6 nodes
  - “E” expanded 8 times
  - “D” expanded 4 times
  - “C” expanded 2 times
  - “A”, “B”, “G” expanded once

UNIVERSITY OF DENVER

Martelli, 1979

- Suggested Algorithm “B”
  - Maintain global “F” value
  - Maximum f-value opened so far
  - If there are nodes on OPEN with f < F
    - Open in order of increasing g-cost
    - Dijkstra’s algorithm

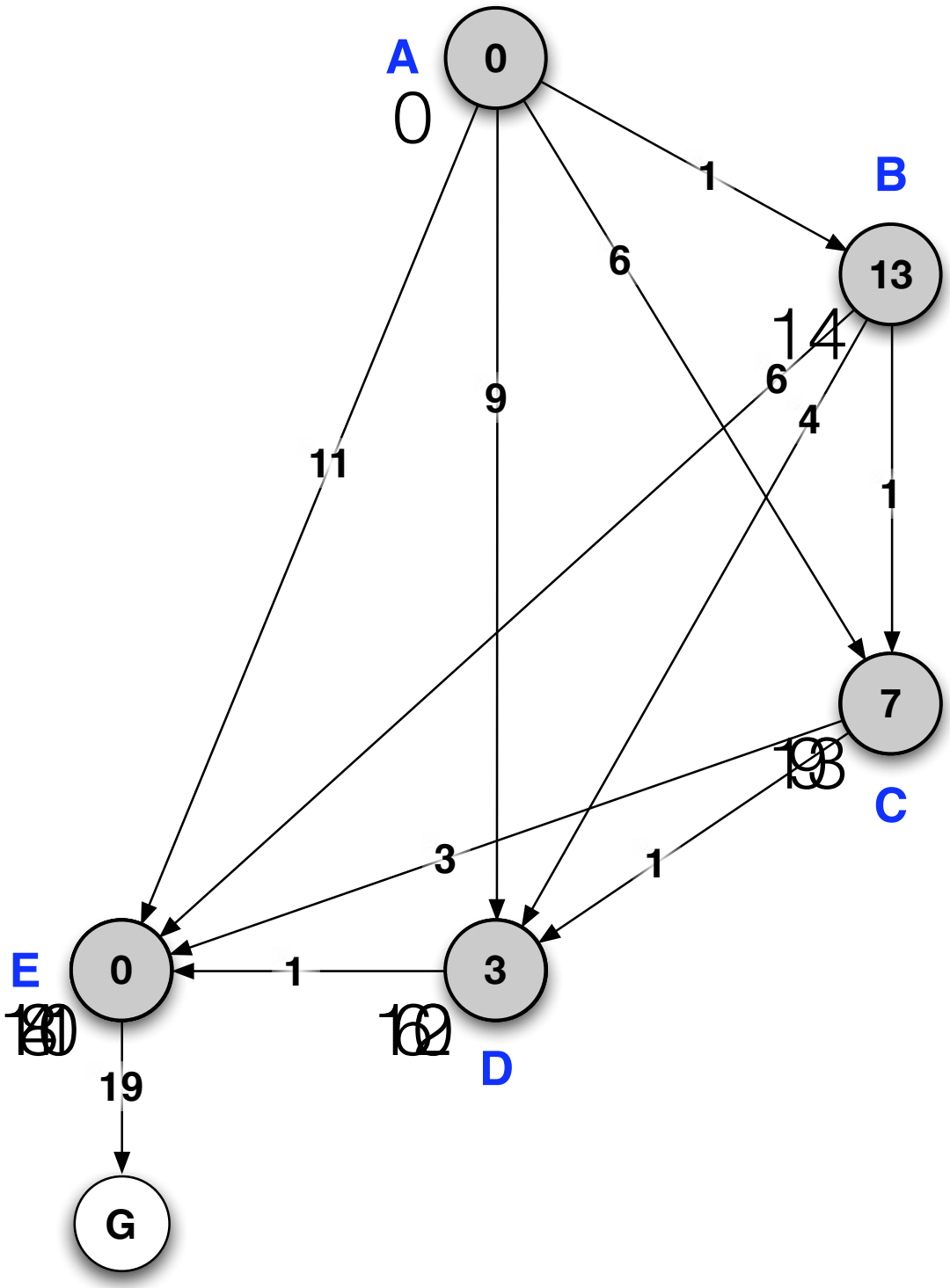


node	f	g	h
A	23	0	23
B	14	1	13
C	9	2	7
D	6	3	3
E	4	4	0
G	23	23	0

UNIVERSITY OF DENVER

Martelli

- Algorithm works well, does it fix everything?
  - No -- worst case still $O(N^2)$
  - Just lower cost of start heuristic to 0



node	f	g	h	F
A	0	0	0	0
E	11	11	0	11
D	12	9	3	12
E	10	10	0	12
C	13	6	7	13
D	10	7	3	13
E	8	8	0	13
B	14	1	13	14
C	9	2	7	14
D	6	3	3	14
E	4	4	0	14
G	23	23	0	23

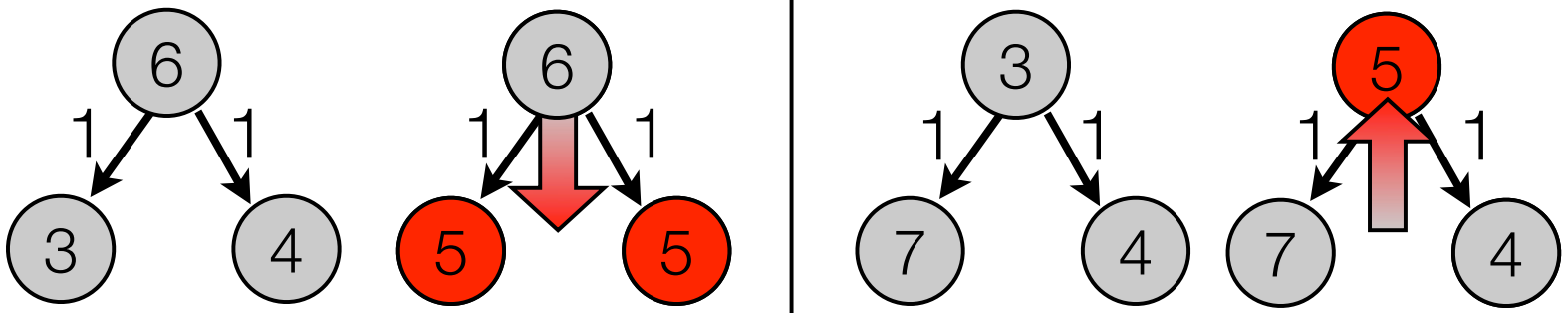
UNIVERSITY OF DENVER

Solution - Mero 84 - B’

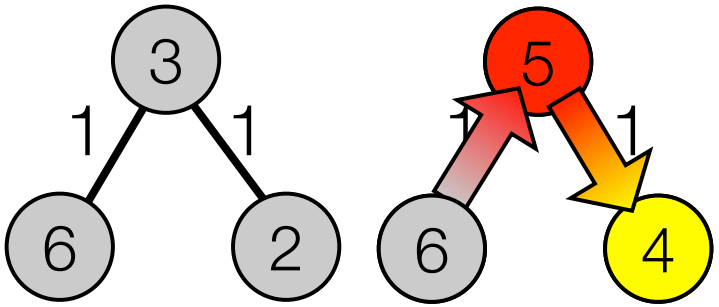
- Pathmax
  - When generating a node:
  - h’(n) = max(h(p) - c(n, p), h(n))
  - h’(p) = max(h(p), min(h(c) + c(c, p))) over all children *c*

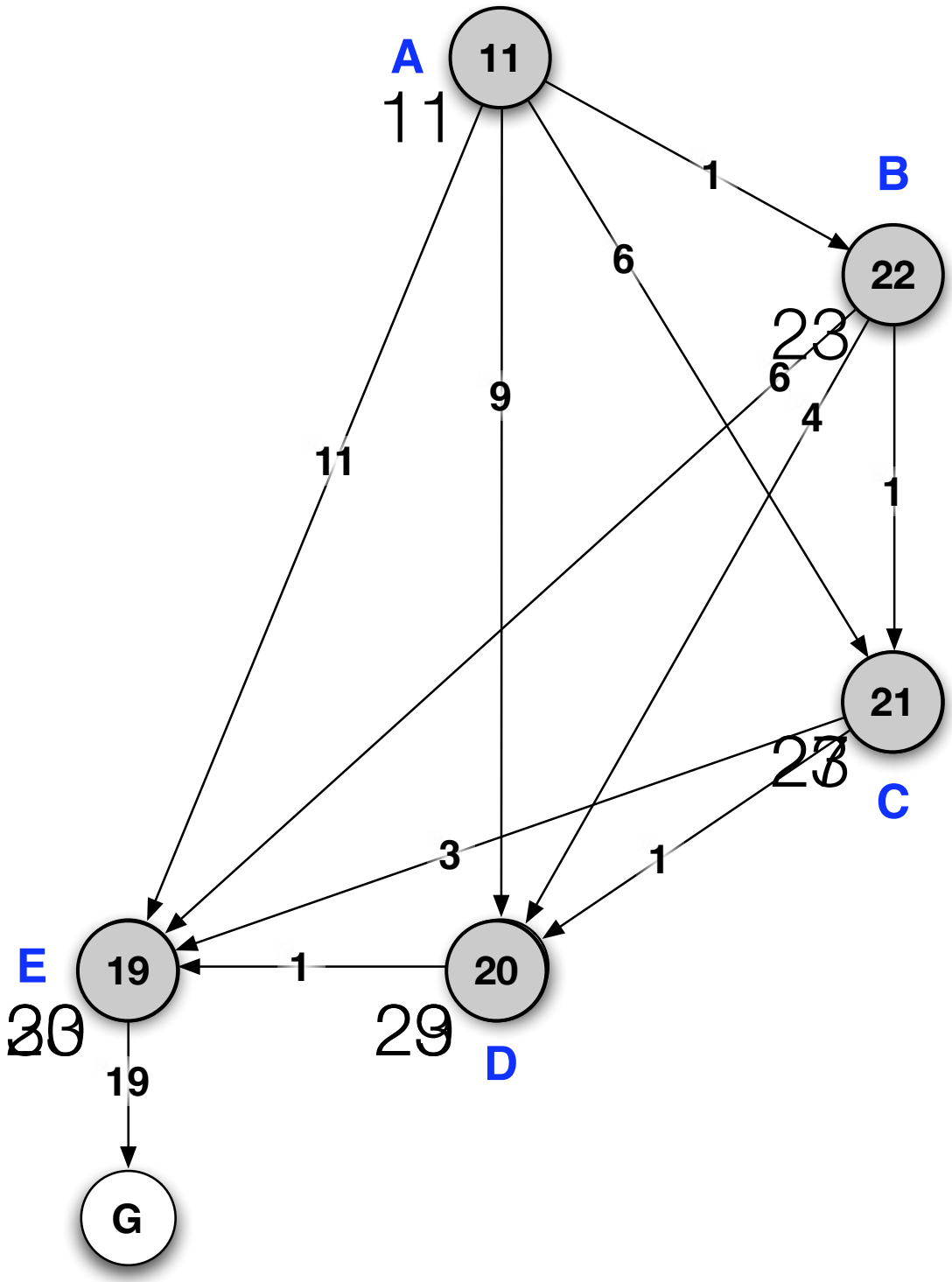
Pathmax rules

- Pathmax:



- BPMX [Felner, et al, 2005]

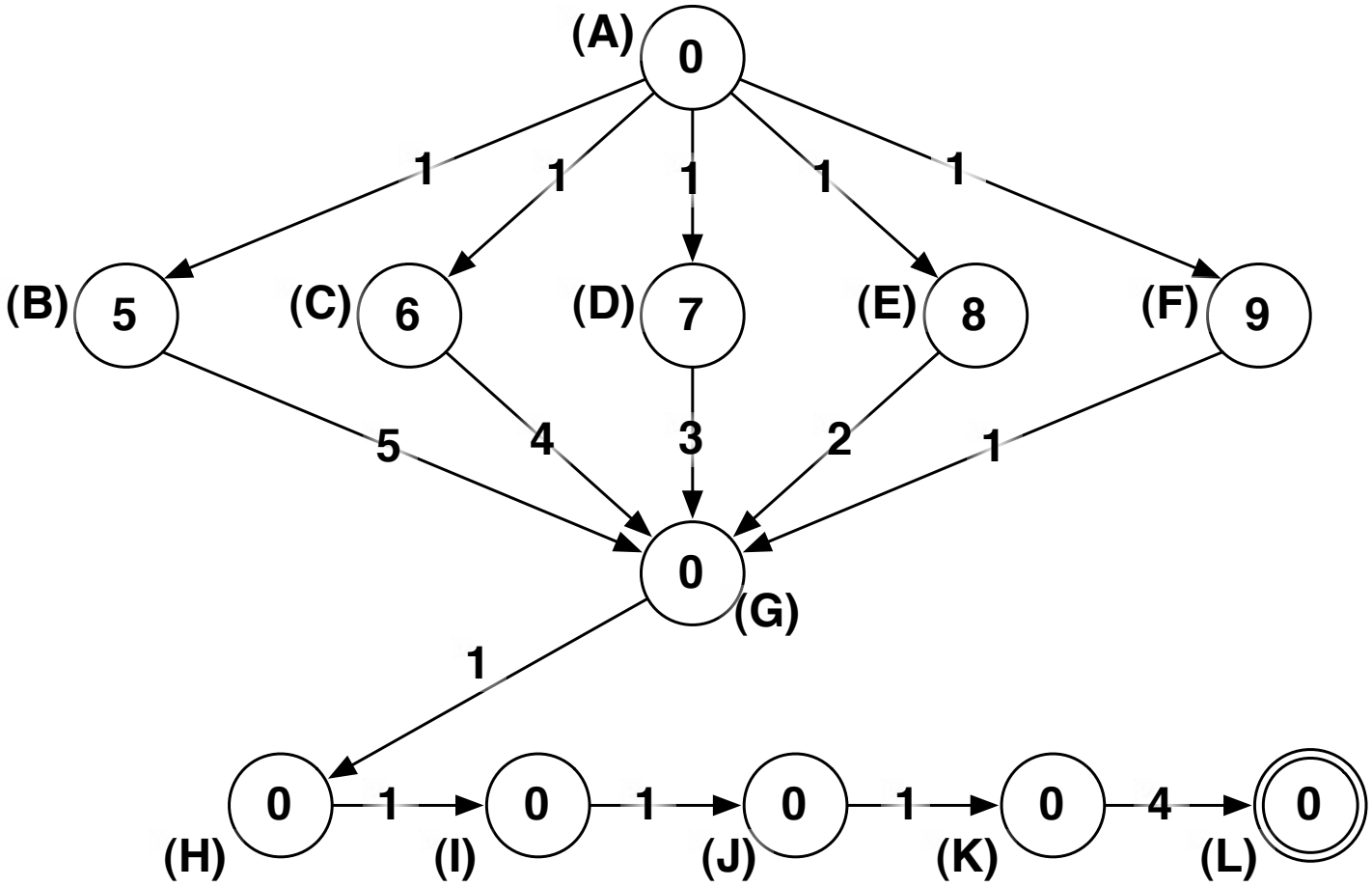




node	f	g	h	F
A	11	0	11	11
E	30	11	19	12
D	29	9	20	13
C	27	6	21	14
B	23	1	22	23
C	23	2	21	23
D	23	3	20	23
E	23	4	19	23
G	23	23	0	23

So...how bad?

- In the worst case, even B' can be $O(N^2)$



UNIVERSITY OF DENVER

General Inconsistency Bounds

- Suppose A* performs φ(N) > N expansions
  - Some node must be re-opened (φ(N) – N)/N times
    - Pigeon-hole principle
  - If Δ is the minimum change in h-cost for a node
    - h-cost is at least Δ·(φ(N) – N)/N
    - Solution cost is also at least Δ·(φ(N) – N)/N
      - We never open a node with f-cost > solution

UNIVERSITY OF DENVER

How do we get inconsistency in practice

- Special properties (duality)
- Max of multiple heuristics
  - Too expensive to use all heuristics, so use random subset of heuristics
- Compression

UNIVERSITY OF DENVER

What is good/bad inconsistency

- "Good" inconsistency
  - There are always good heuristics nearby
  - 1-step BPMX to 'fix' bad values
  - Improve the run-time distribution of h-values
- "Bad" inconsistency
  - Misleading values (worst path has lowest f-cost)
- Note: with no cycles, inconsistency isn't a problem

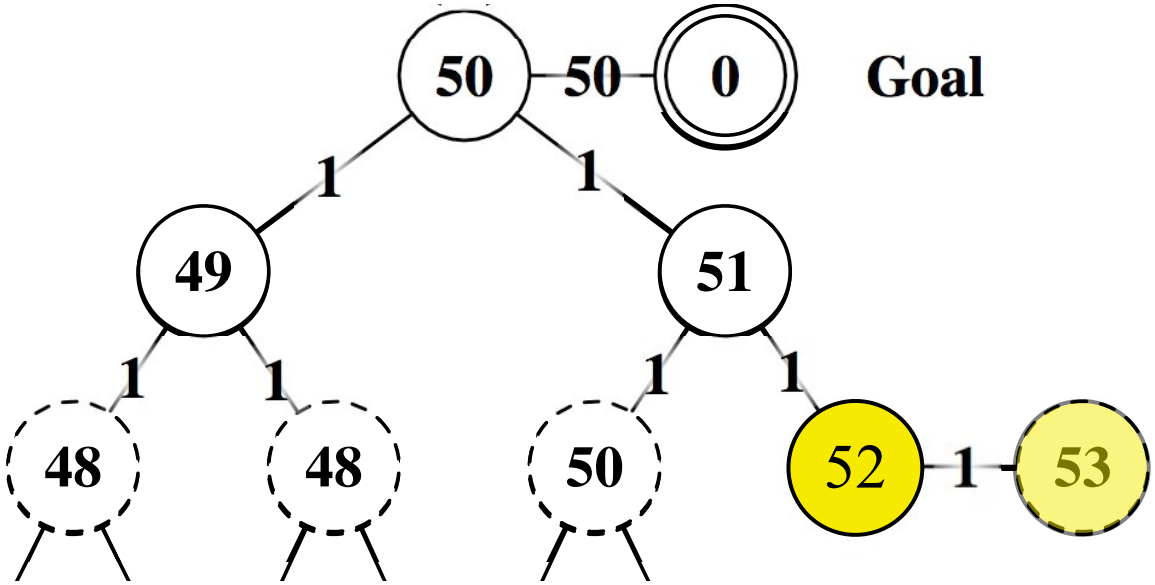
UNIVERSITY OF DENVER

BPMX in A*/IDA*

- BPMX is free in IDA*
- More expensive in A*
  - We don't naturally backtrack through closed list
  - Choice:
    - Backup as far as possible
      - $O(N^2)$ cost or unbounded savings
    - Backup only *k* steps O(kN) cost

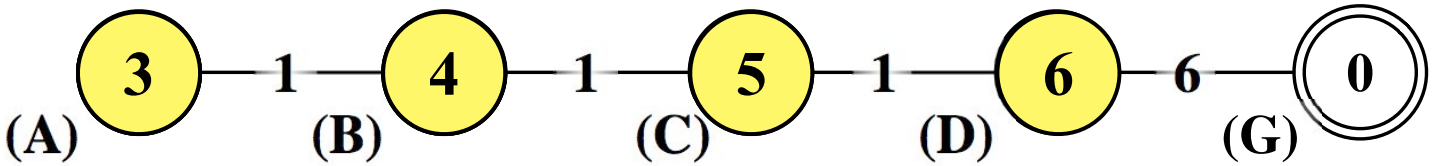
UNIVERSITY OF DENVER

BPMX Best Case



UNIVERSITY OF DENVER

BPMX Worst Case

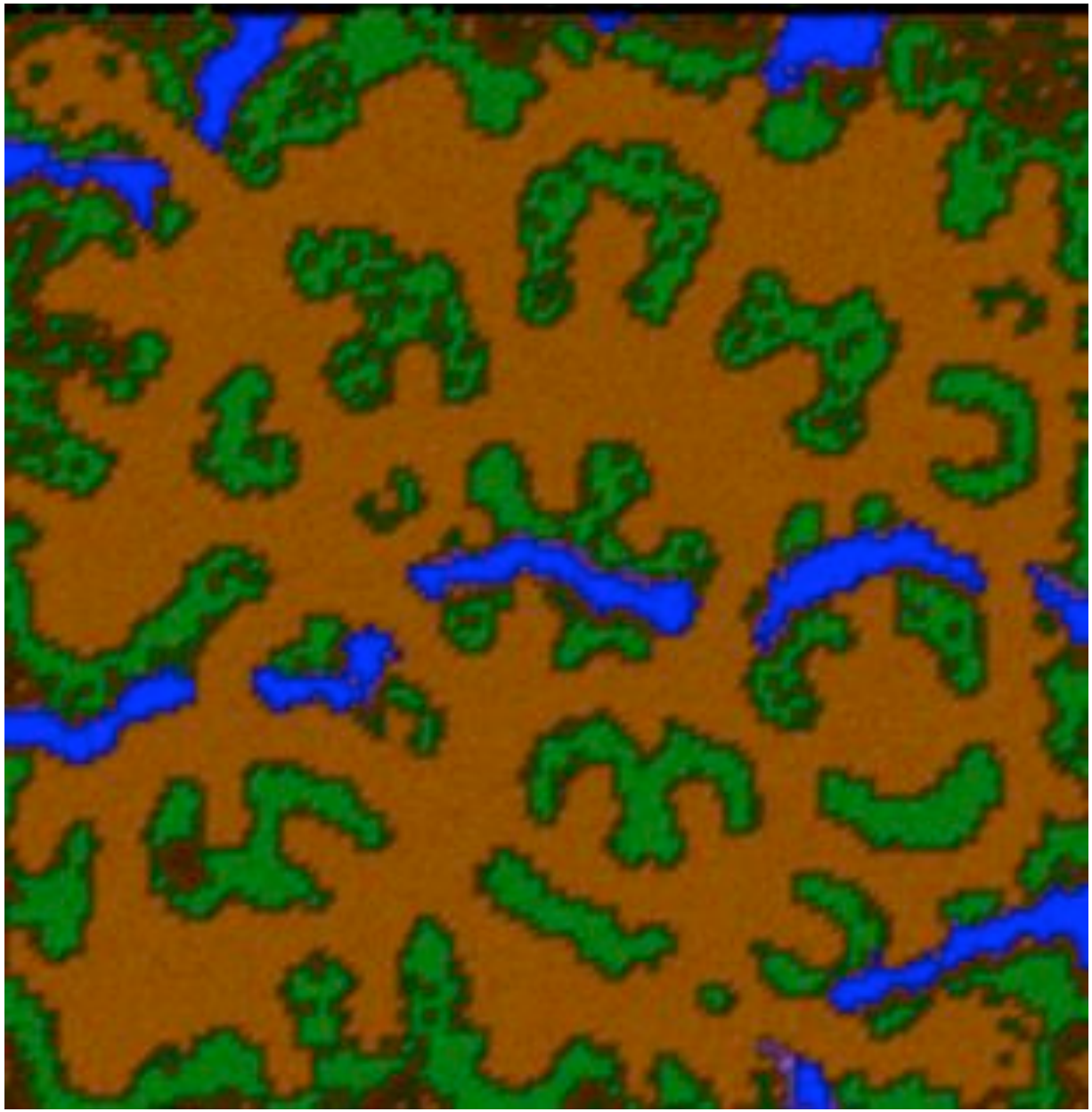


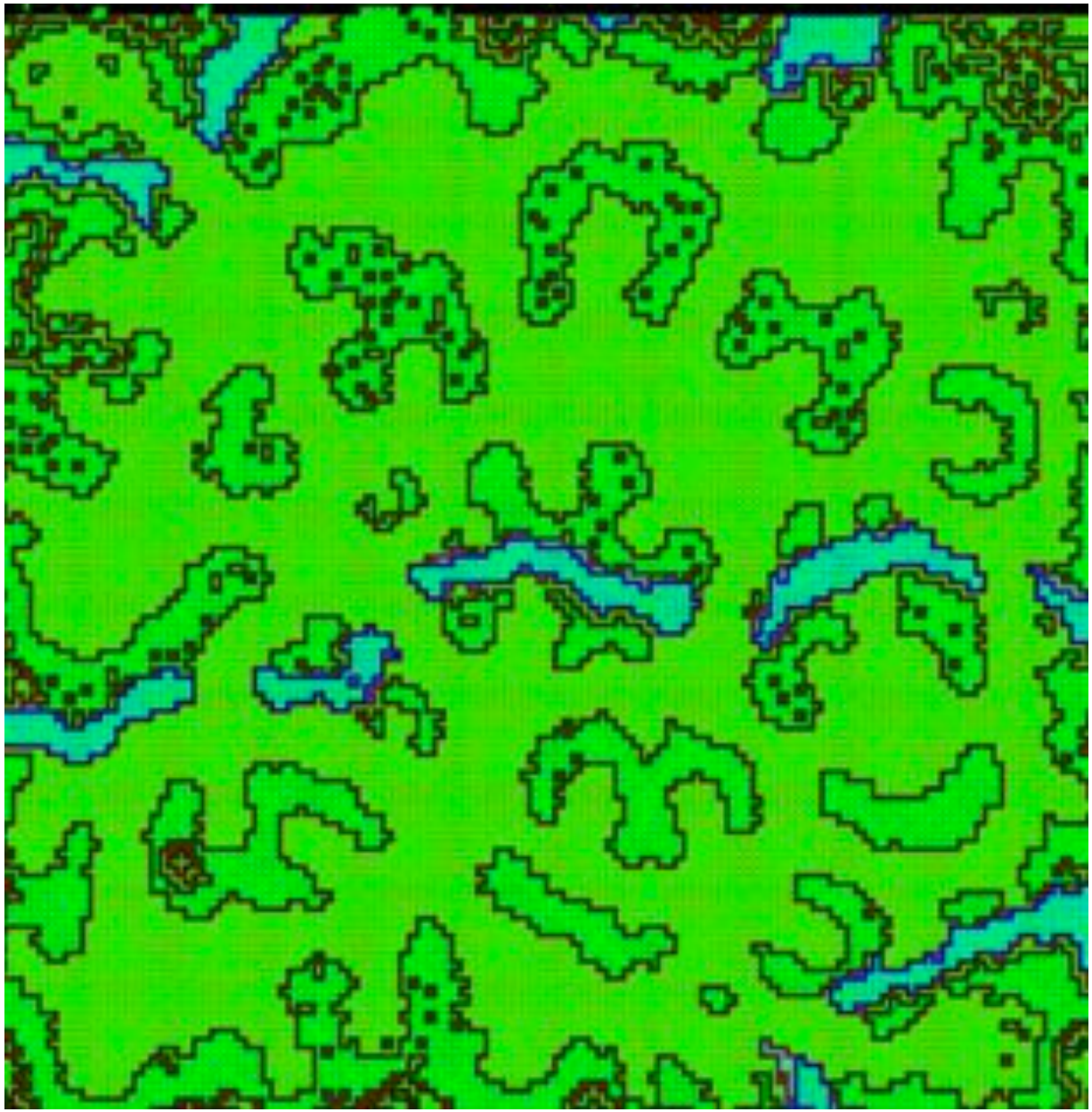
Abstraction

UNIVERSITY OF DENVER

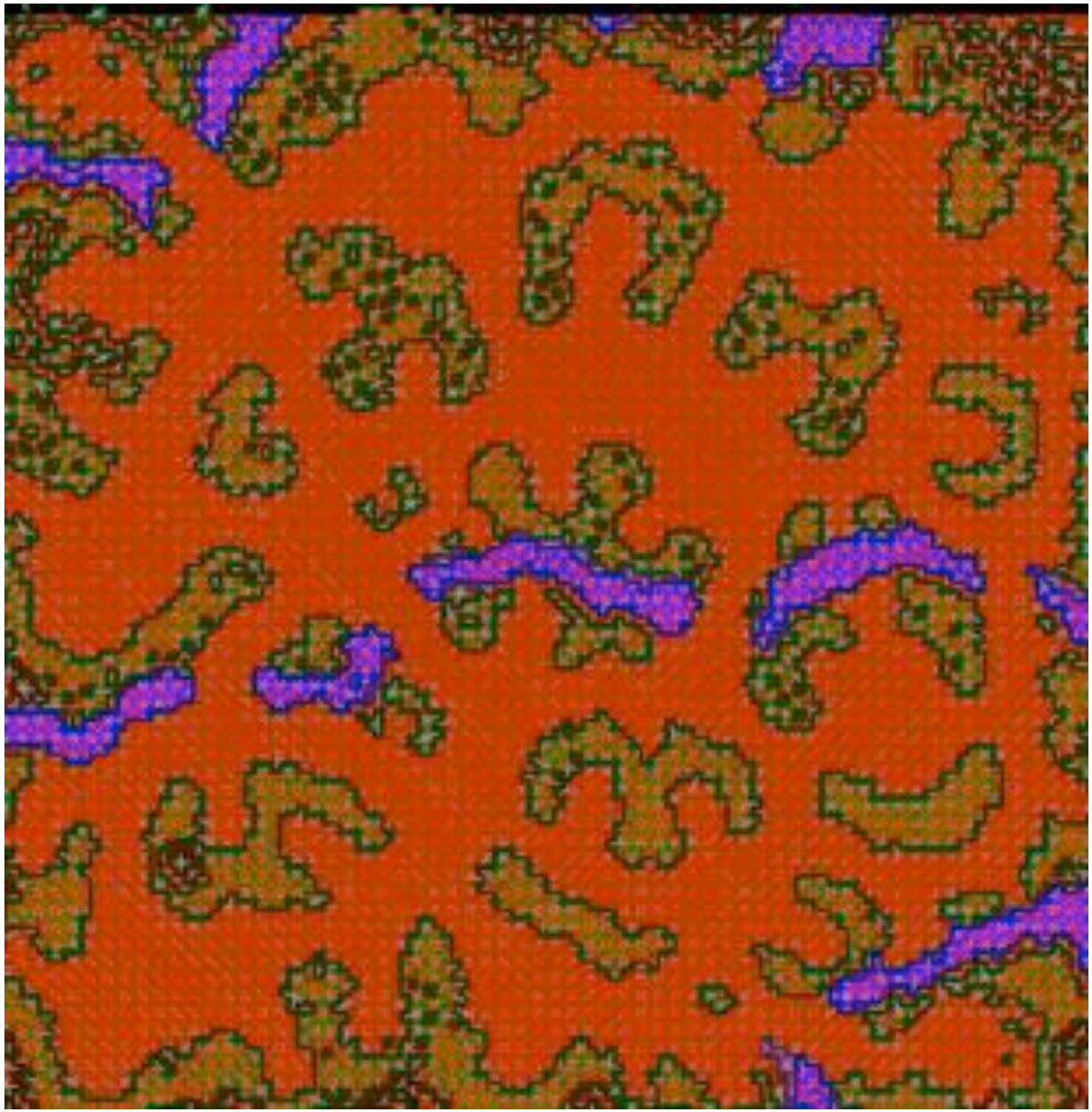
Abstraction and Refinement

- Build a new representation of the state space
  - Want abstraction to be homomorphic (Holte)

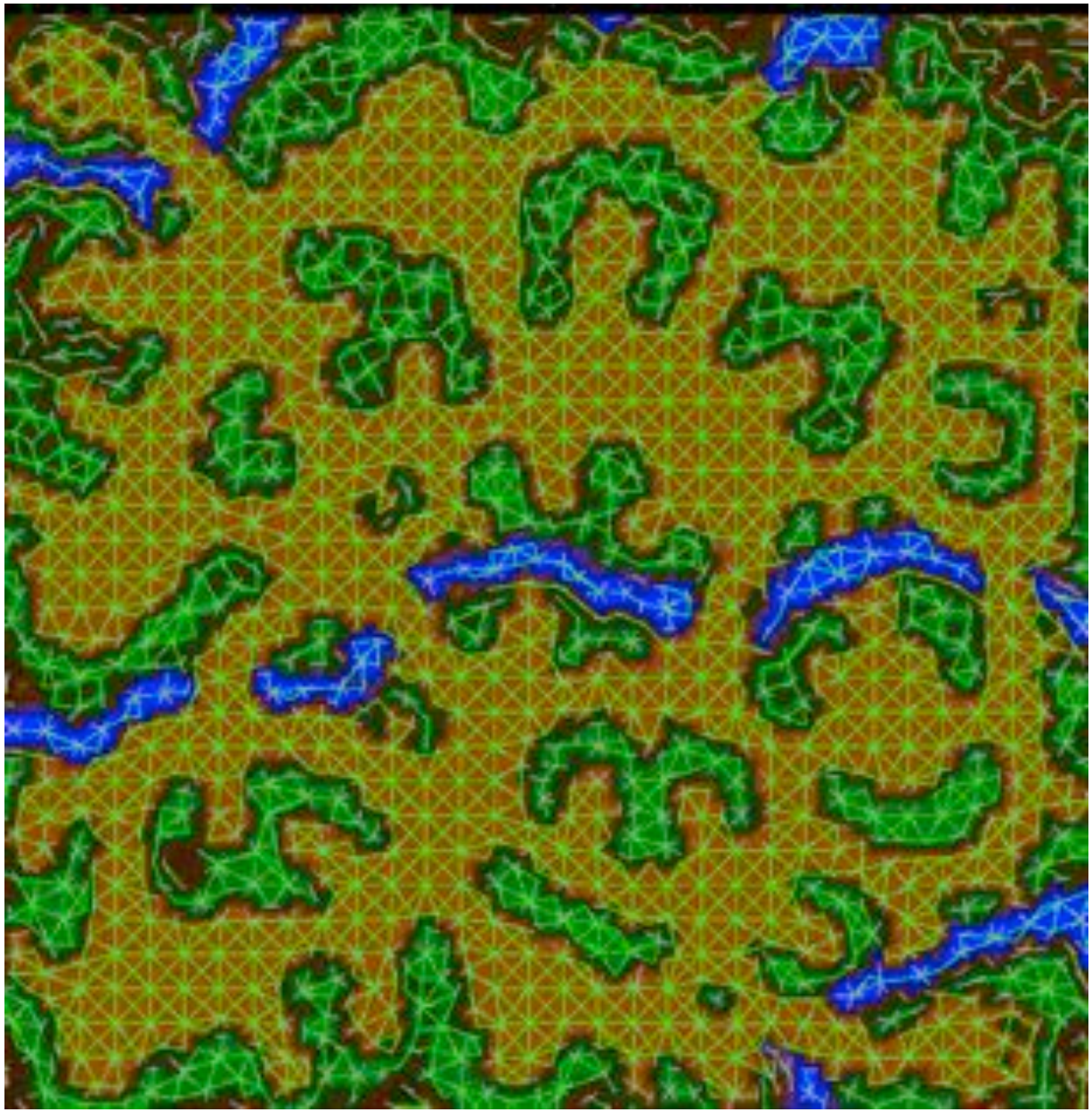




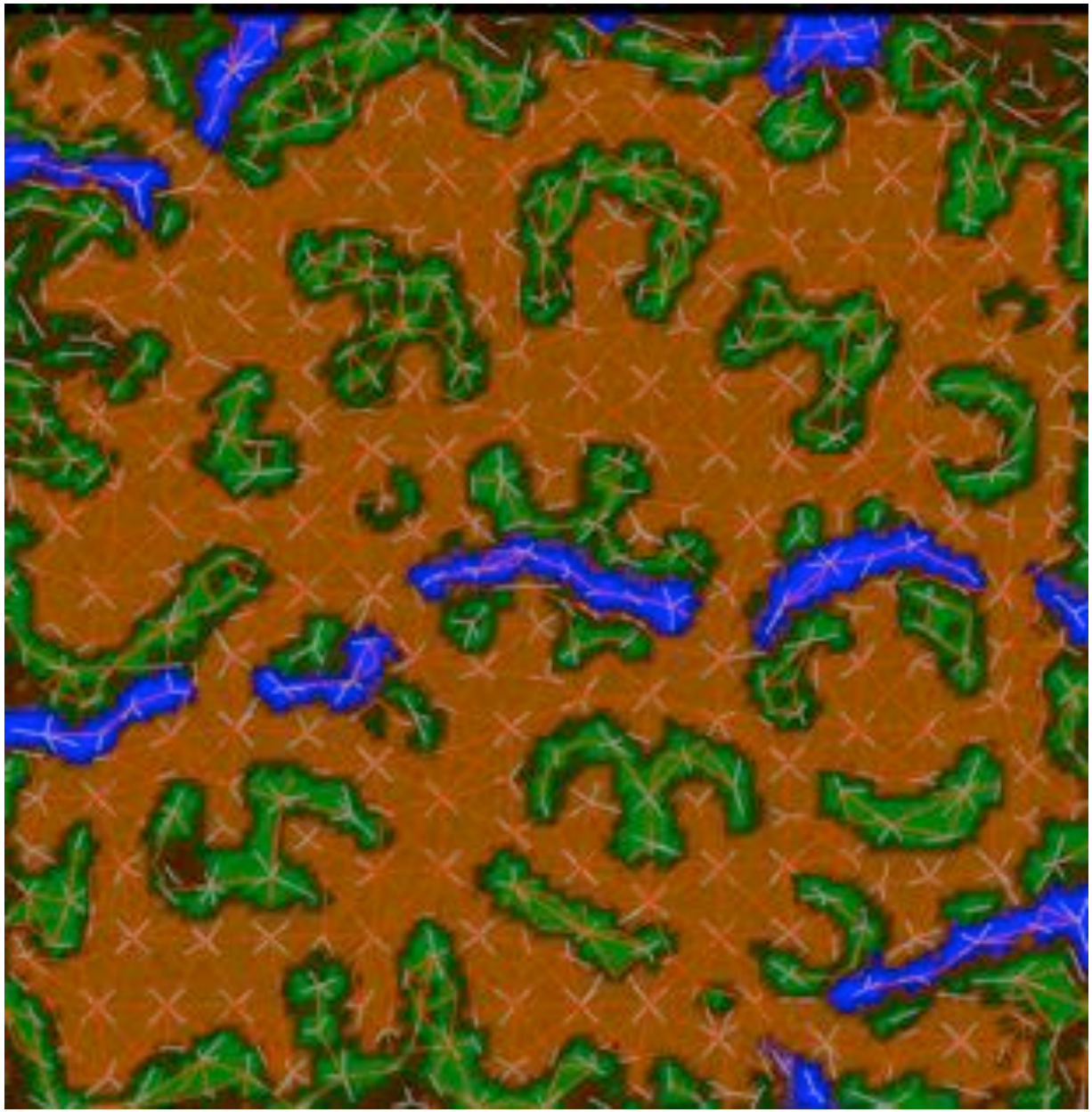
16807 Nodes



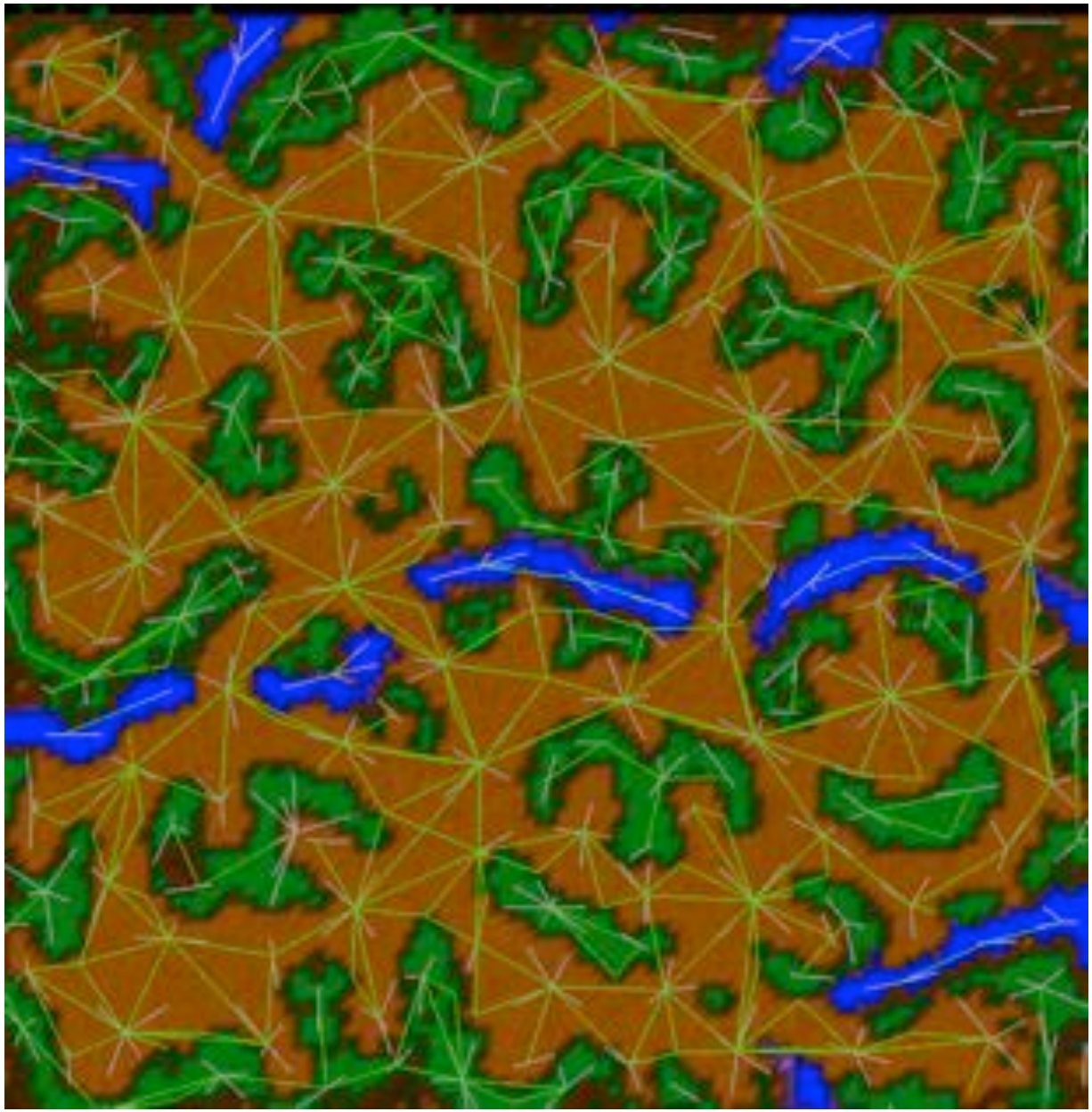
5212 Nodes



1919 Nodes



771 Nodes



316 Nodes

UNIVERSITY OF DENVER

Algorithm

- Choose group of nodes to abstract together
  - Nodes must be connected
  - Nodes cannot already be abstracted
  - Repeat until all nodes abstracted
- Add an edge between abstract groups if there exists an edge between any two nodes abstracted by each group

UNIVERSITY OF DENVER

Properties

- Homomorphic abstractions are *refineable*:
  - If any abstract path can be directly refined into a path into the original graph
    - All nodes on refined path abstract into abstract path
  - Every path in the original graph has an abstract counterpart

UNIVERSITY OF DENVER

Refinement uses

- Used in road networks
- Used in model checking
- Used in heuristic search
- Used in robotics

UNIVERSITY OF DENVER

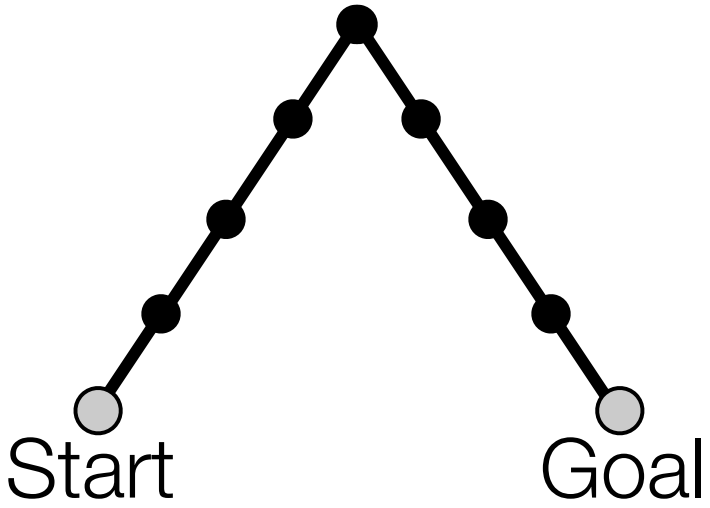
Approaches to Refinement-Style Search

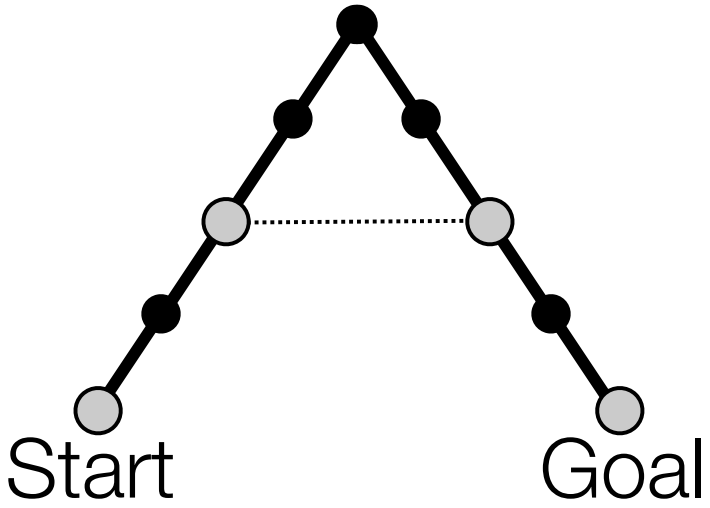
- Find abstract path
- Partially refine path
- Refinement corridor

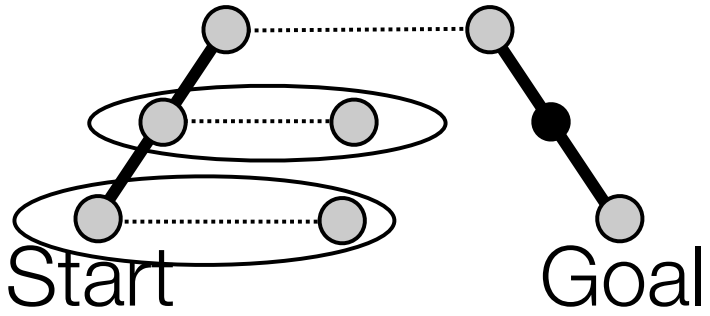
UNIVERSITY OF DENVER

PRA*(k) - Partial-Refinement A*

- Find a complete abstract path
- Until a partial path is available
  - Take first *k* steps of abstract path and refine
- Follow path & refine more steps when needed


Start
Goal


Start
Goal


Start
Goal

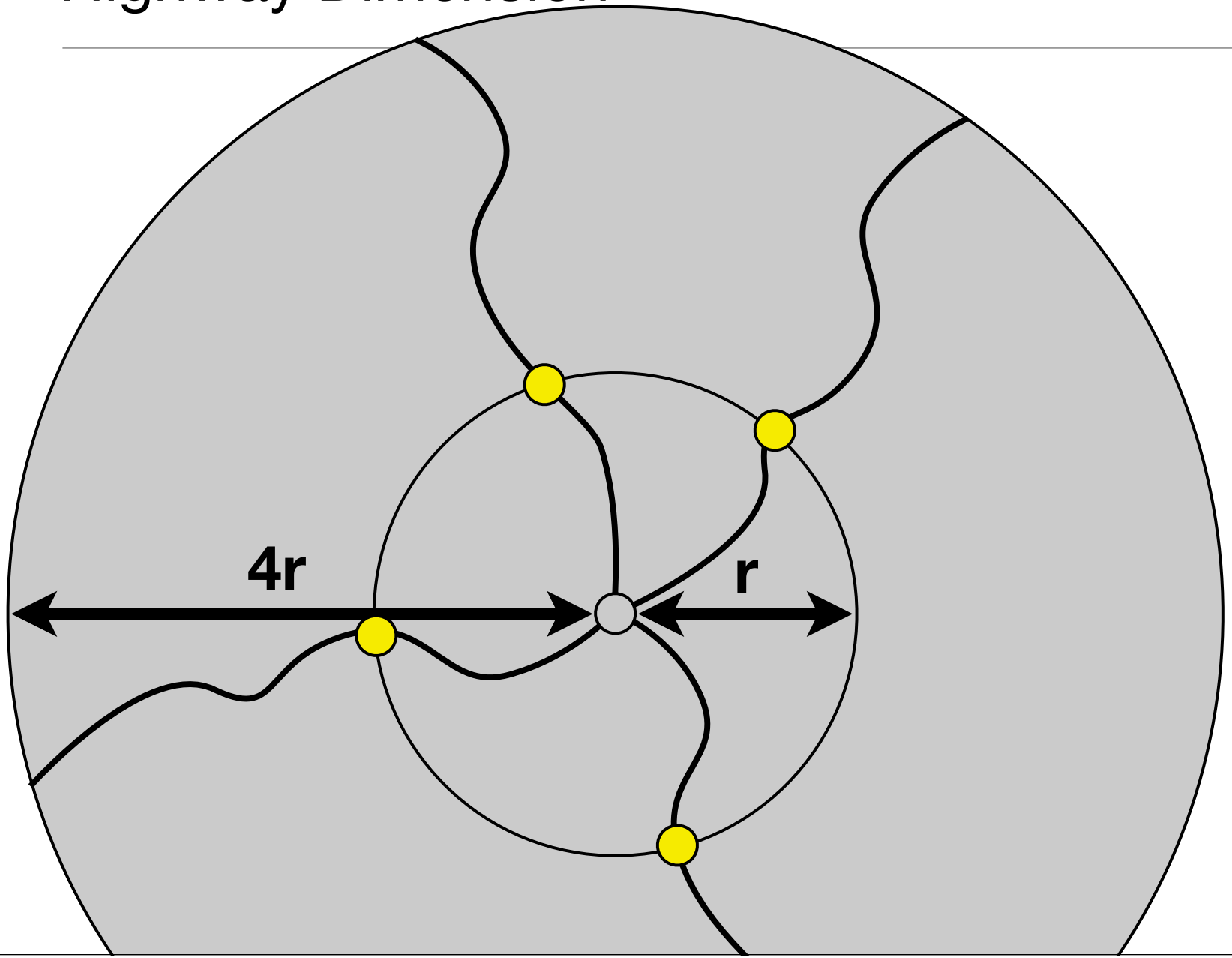
UNIVERSITY OF DENVER

Abstraction in Road Networks

- Road networks happen to have special properties

UNIVERSITY OF DENVER

Highway Dimension



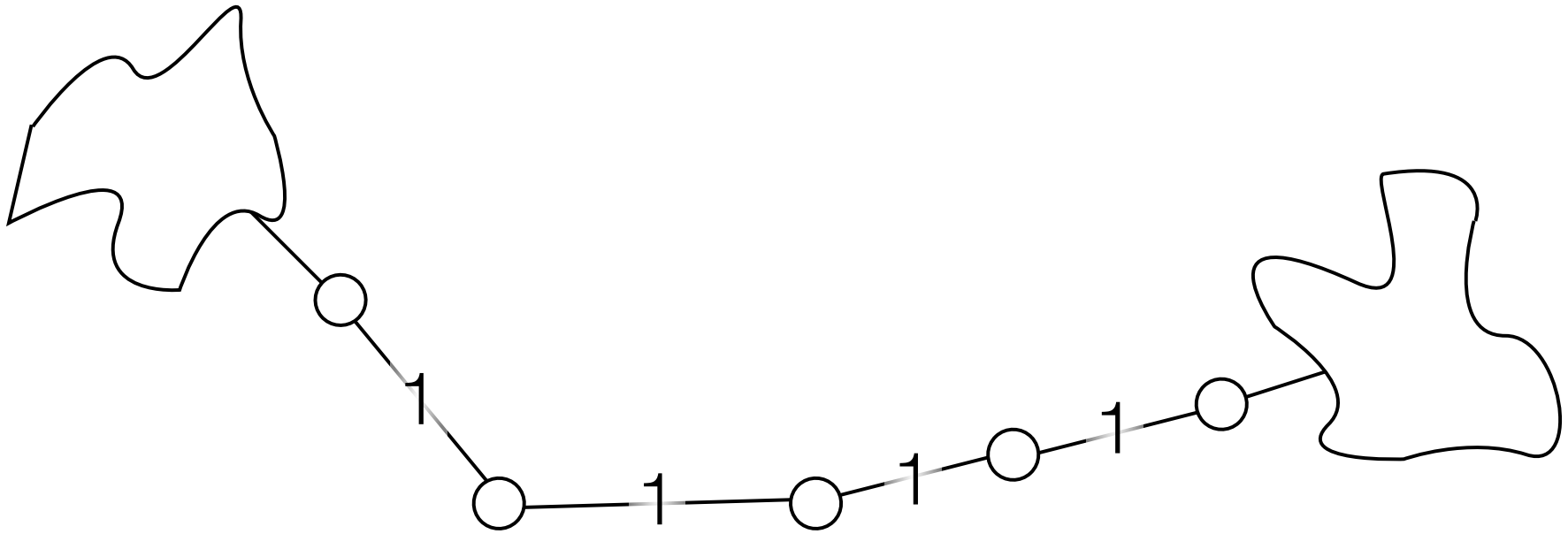
UNIVERSITY OF DENVER

Highway Dimension

- Problems with low highway dimension can be solved quickly
  - Abraham et. al., 2010
- Suite of techniques developed
  - Reach (Goldenberg, et al)
  - Contraction hierarchies (Geisberger, et al)
  - Transit node routing (Bast, et al)

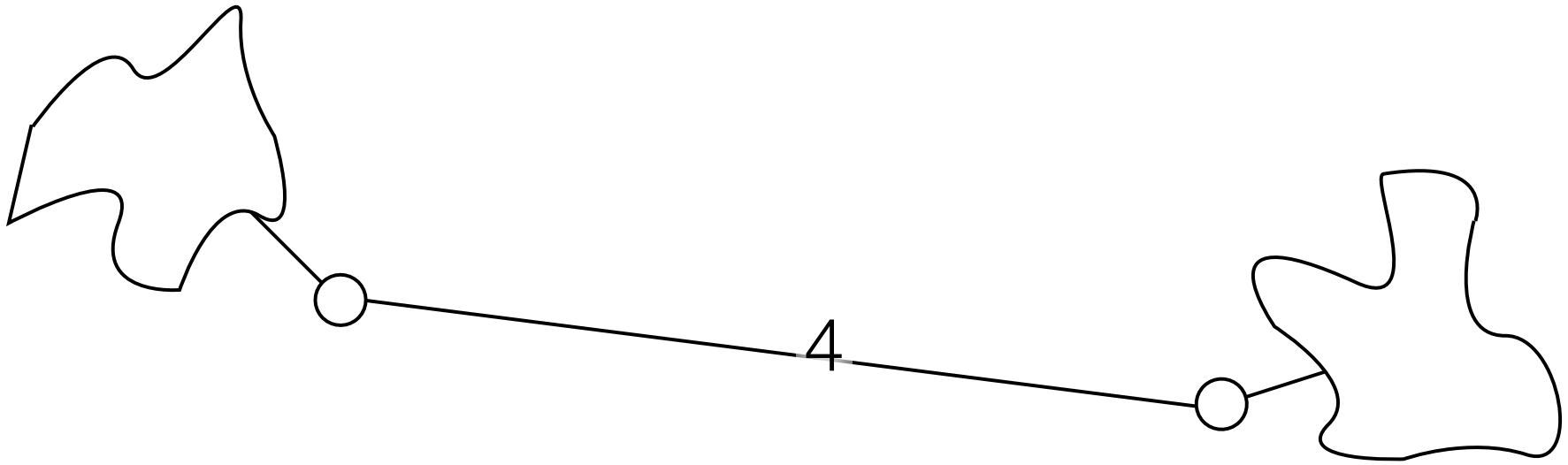
UNIVERSITY OF DENVER

Contraction Hierarchies



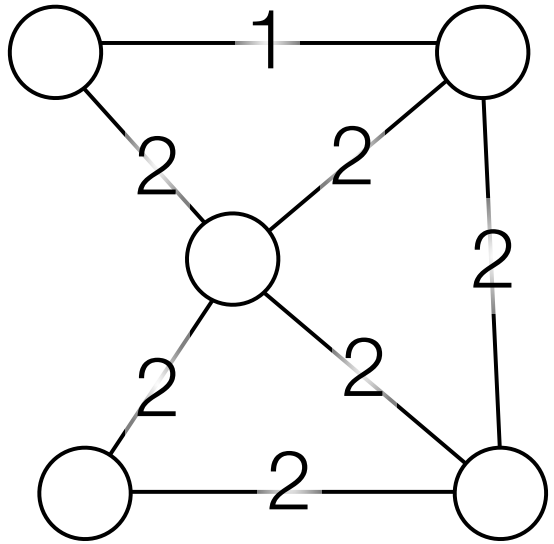
UNIVERSITY OF DENVER

Initial Motivation: Lines



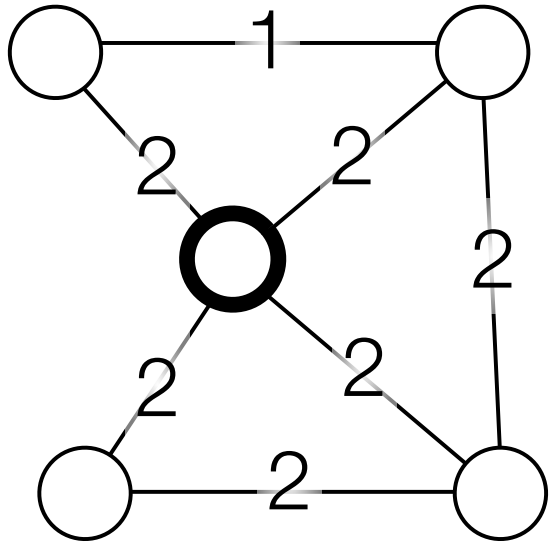
UNIVERSITY OF DENVER

Generalization



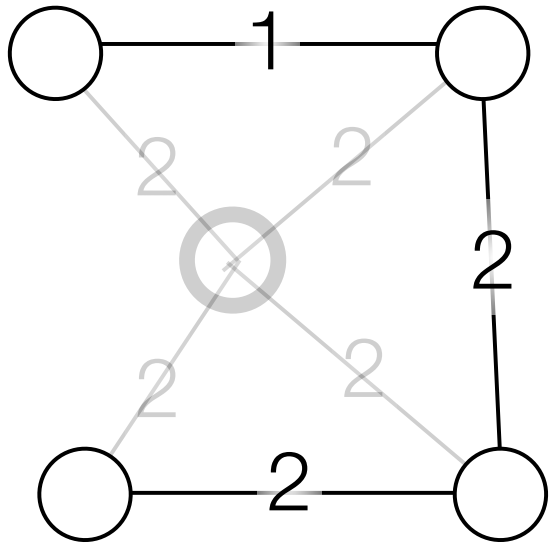
UNIVERSITY OF DENVER

Generalization



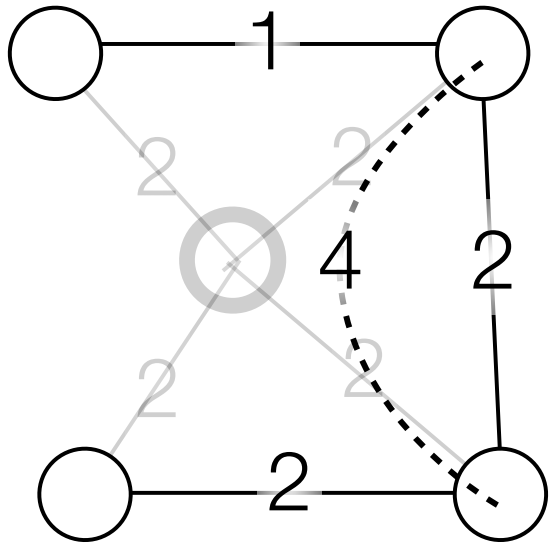
UNIVERSITY OF DENVER

Generalization



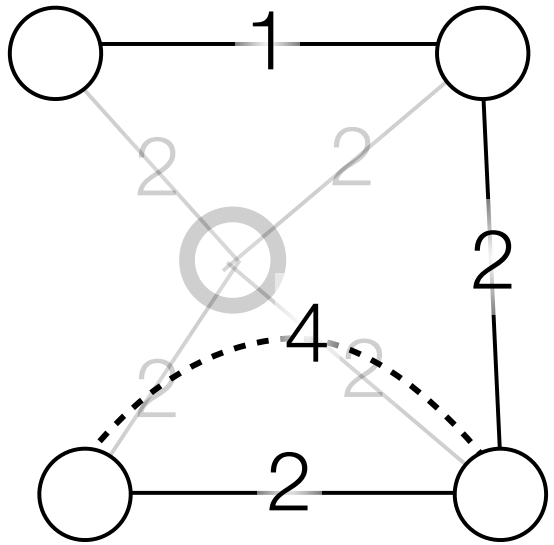
UNIVERSITY OF DENVER

Generalization



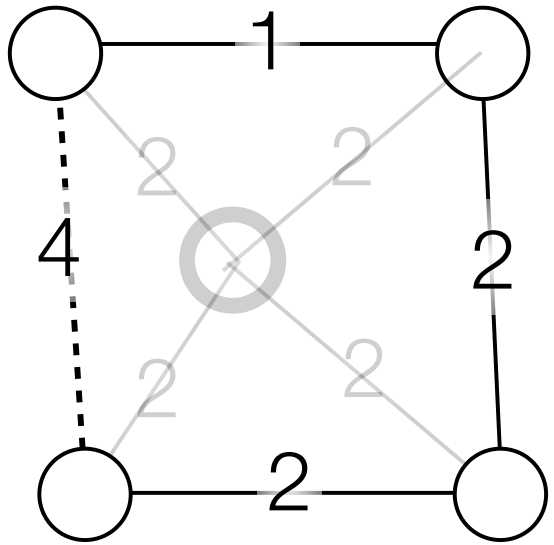
UNIVERSITY OF DENVER

Generalization



UNIVERSITY OF DENVER

Generalization



UNIVERSITY OF DENVER

Building Contraction Hierarchies

- Choose most important node *n*
  - For all pairs of neighbors, check if removing *n* influences the shortest path between neighbors
    - If not, just remove *n*
    - If so, add shortcut edge with the same cost as the shortest path through *n*

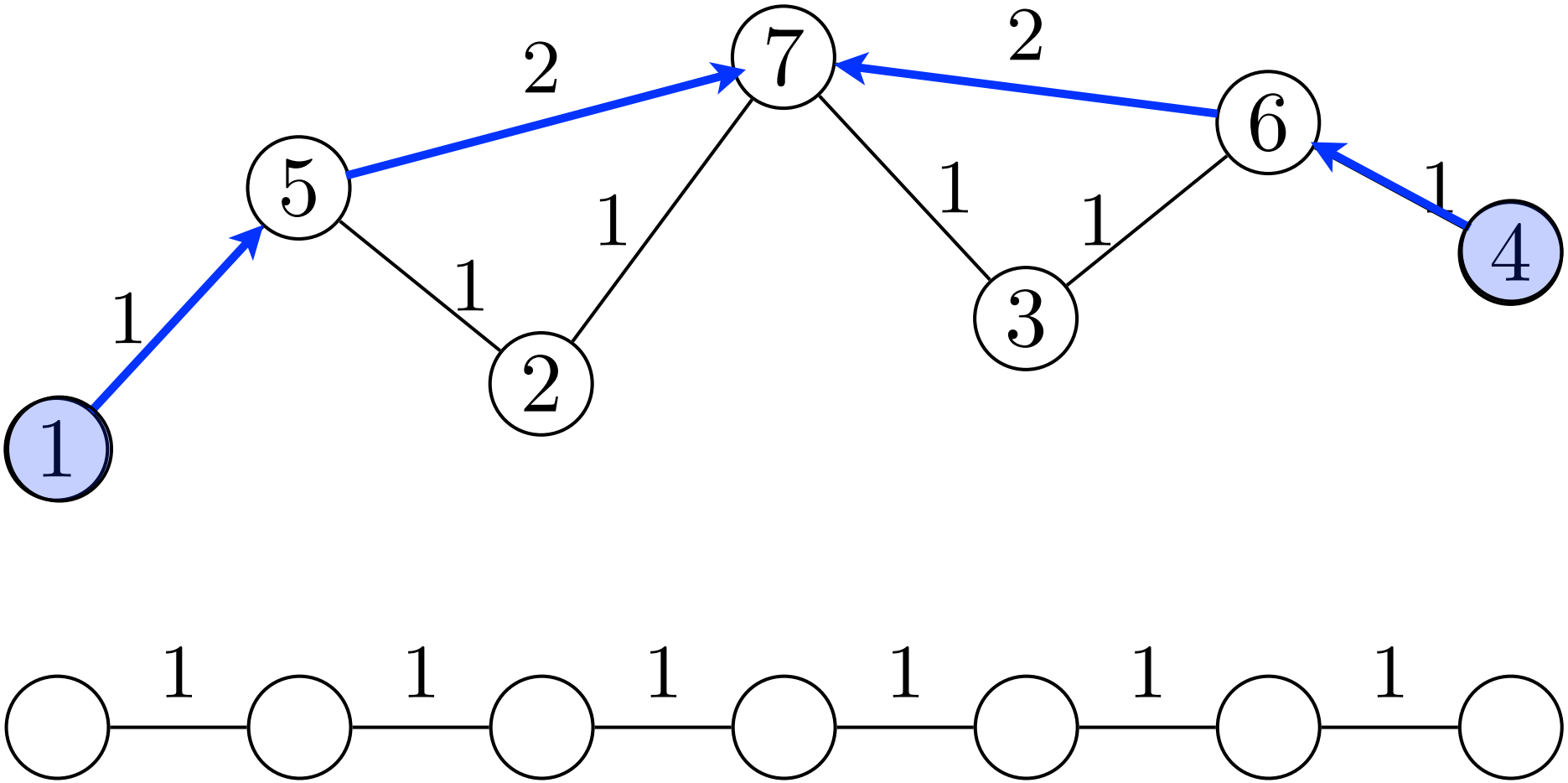
UNIVERSITY OF DENVER

Node Importance

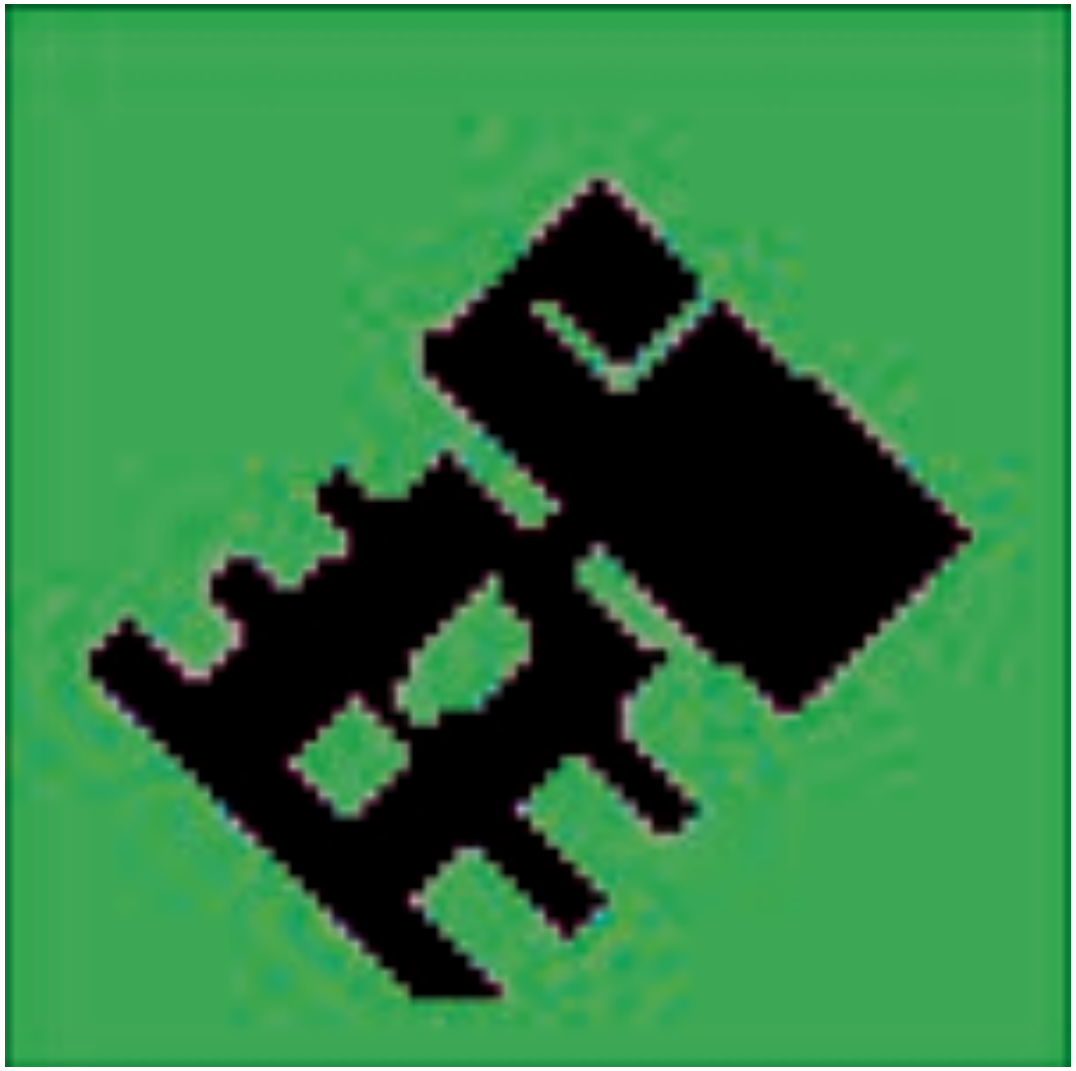
- Ad-hoc ordering for nodes:
  - **Edge difference**: how many edges are removed/ introduced when *n* is contracted
  - **Original edges**: how many edges have already been abstracted below shortcut edges introduced when *n* is contracted
  - **Upward path length**: max of unpacked path length
  - **Contracted neighbors**: how many neighbors are already contracted

UNIVERSITY OF DENVER

Using CHs



Easy for Abstraction
Harder for CH



UNIVERSITY OF DENVER

Easy for CH
Harder for Abstraction

UNIVERSITY OF DENVER

Use in practice

- In practice:
  - Do not contract the whole graph
  - Contract until the graph reaches a particular size
  - Then search with differential heuristics

Deployment:
Dragon Age Origins

UNIVERSITY OF DENVER

Example: Dragon Age (BioWare)

- 2006 - BioWare approached UofA
  - Pathfinding using 100ms (1ms available)
  - No memory budget
- Initial implementation (late 2006 - early 2007)
- Additional enhancements (2008)
- Achieved 100μs average per unit per frame
  - Computation spread across frames

UNIVERSITY OF DENVER

Pathfinding Architecture

High-level planning	Sparse graph
Medium-level planning	Sparse graph
Low-level planning	Grid
Executed path	Arbitrary line segments

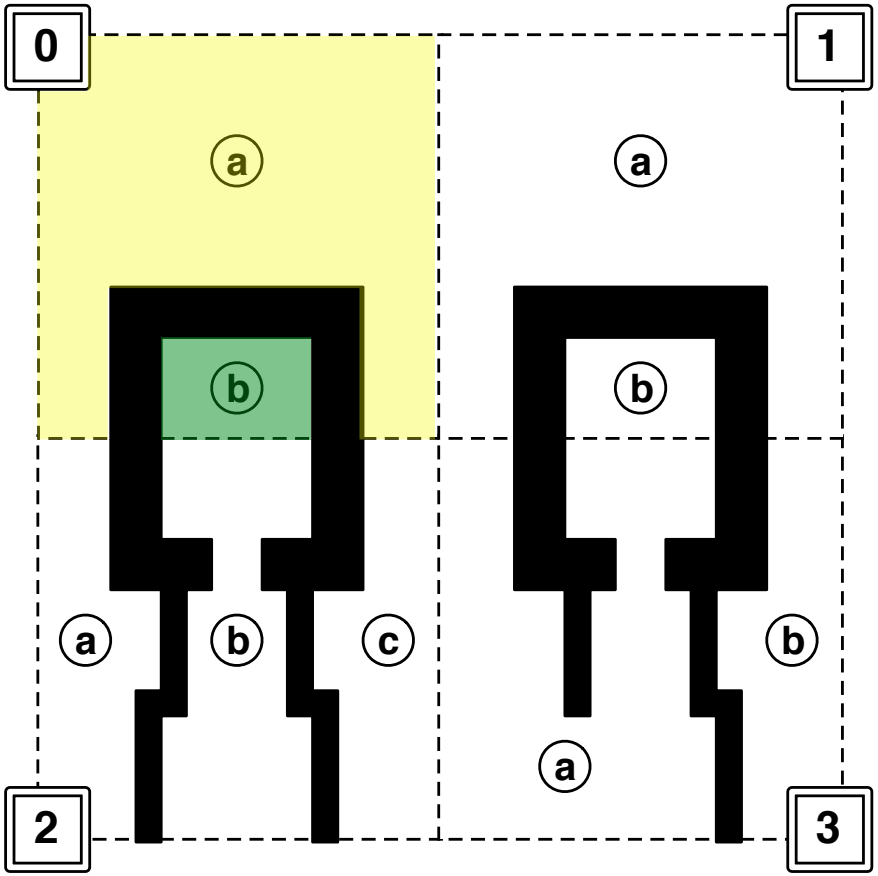
UNIVERSITY OF DENVER

Dragon Age - Additional Enhancements

- Many new features over previous games:
  - Map-wide pathfinding
  - Trap/area effect avoidance
  - Improved static obstacle/creature avoidance
  - Improved path-following animation
  - Many other small performance tweaks

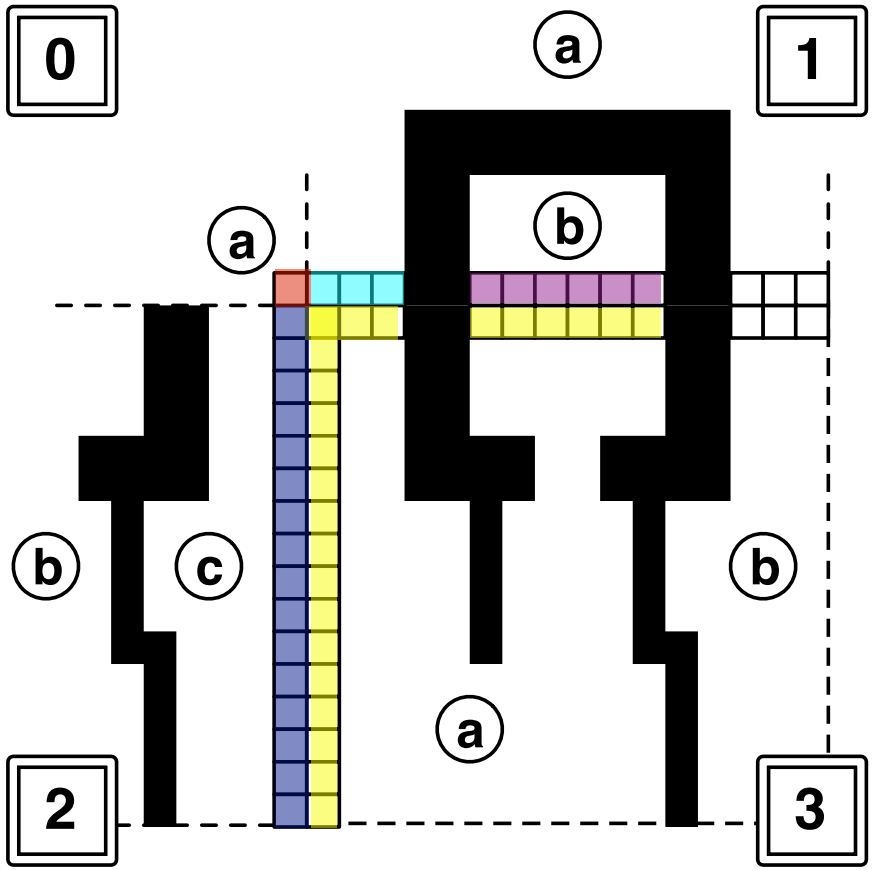
Sectors / Regions

- Divide world into large sectors
  - Fixed size
  - Index implicitly
- Divide sectors into regions
  - Regions entirely connected
  - Regions have a *center* point



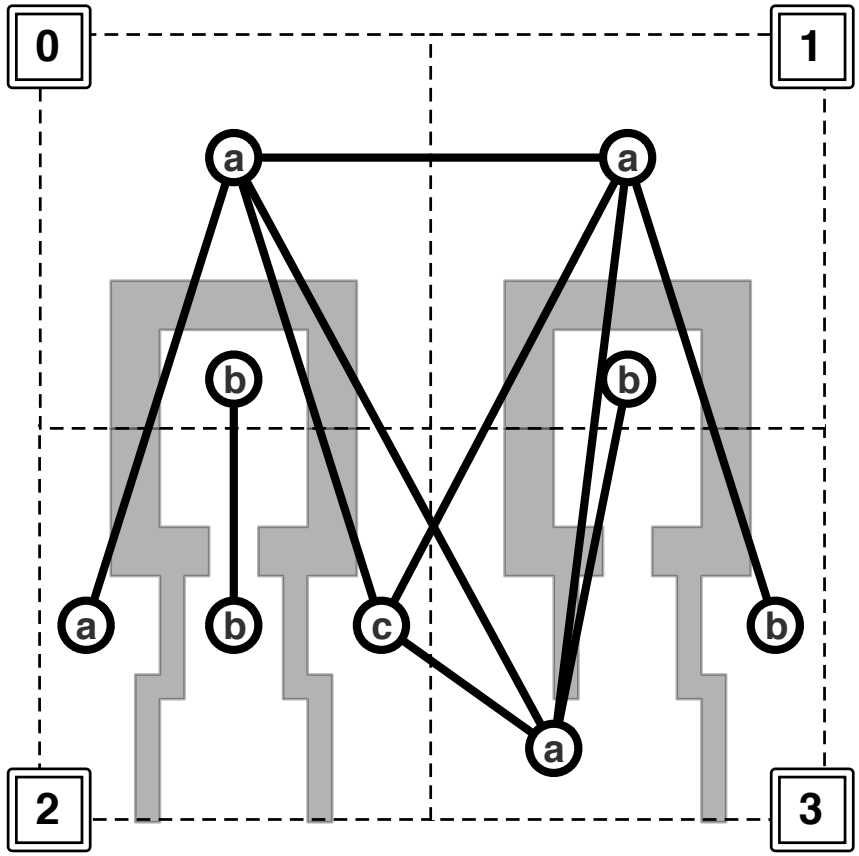
Edges

- Look at borders of regions to determine edges



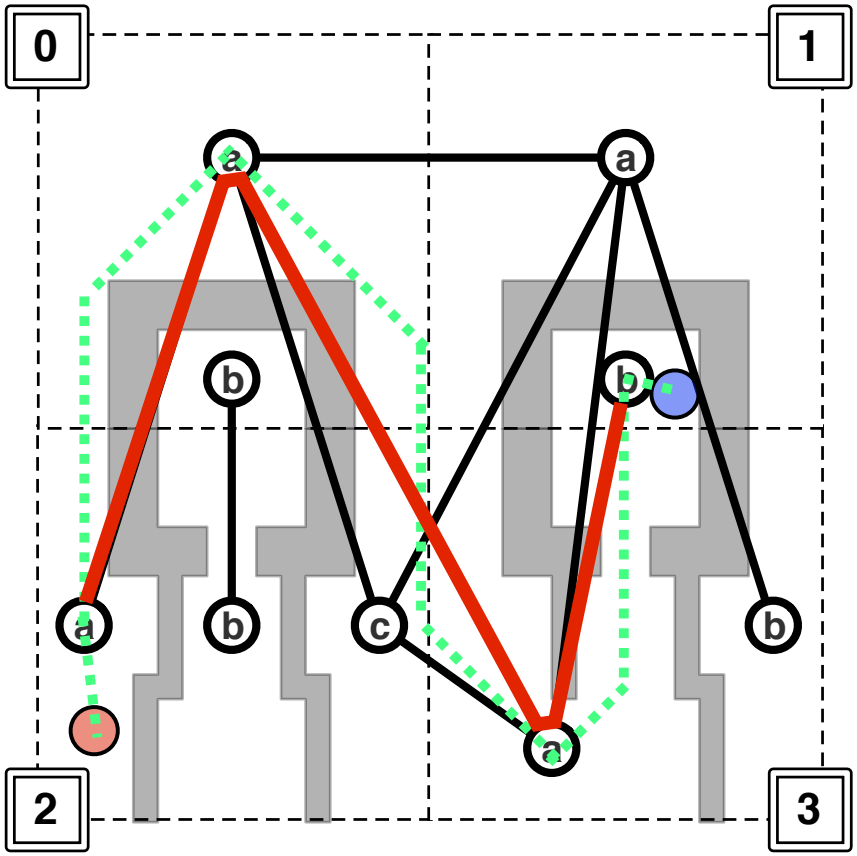
Abstract Graph

- Original Map:
  - 32x32 = 1024 cells
- Abstract Graph:
  - 9 nodes
  - 10 edges



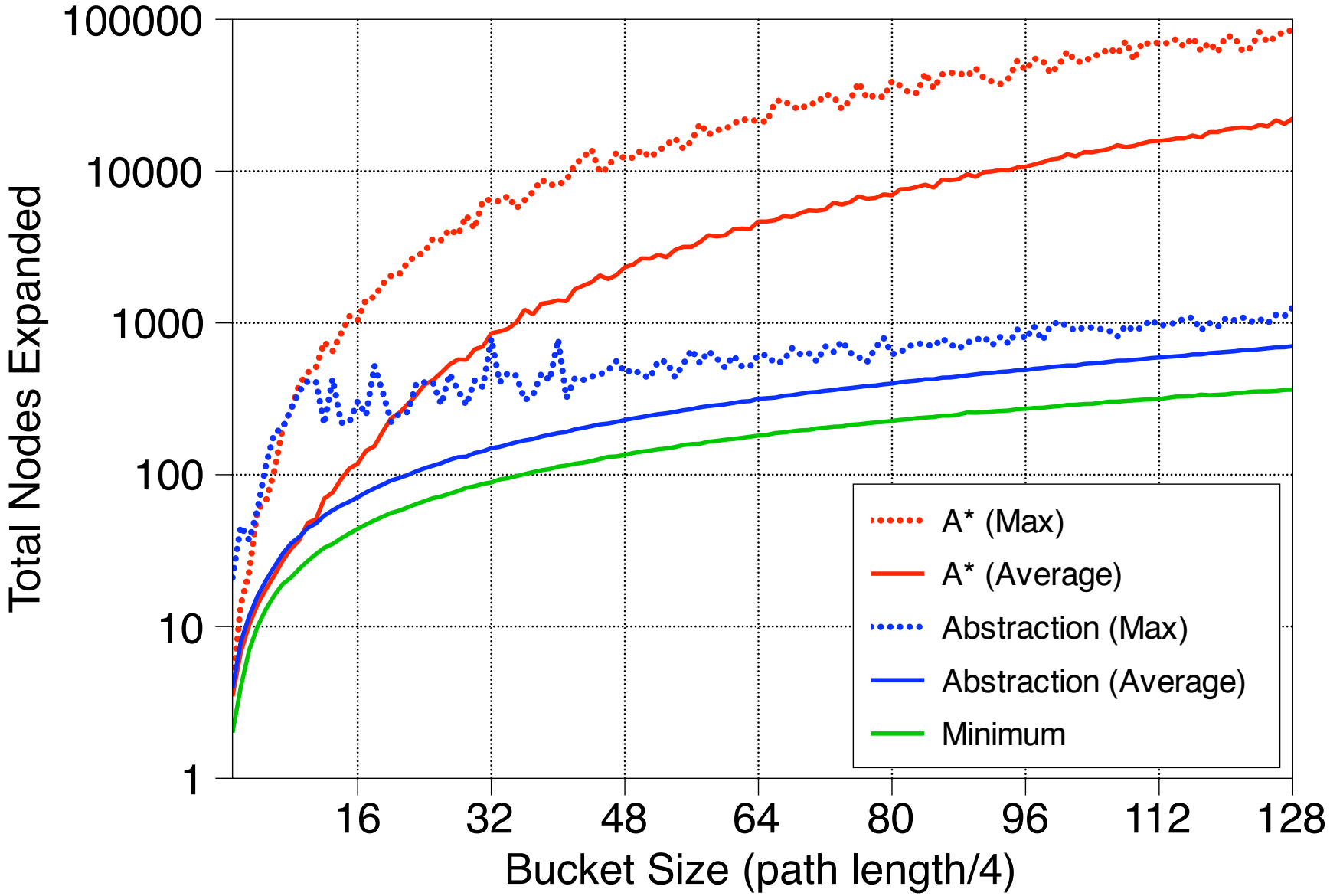
Usage Example

- Find abstract parents
- Find abstract path
- Find real path



UNIVERSITY OF DENVER

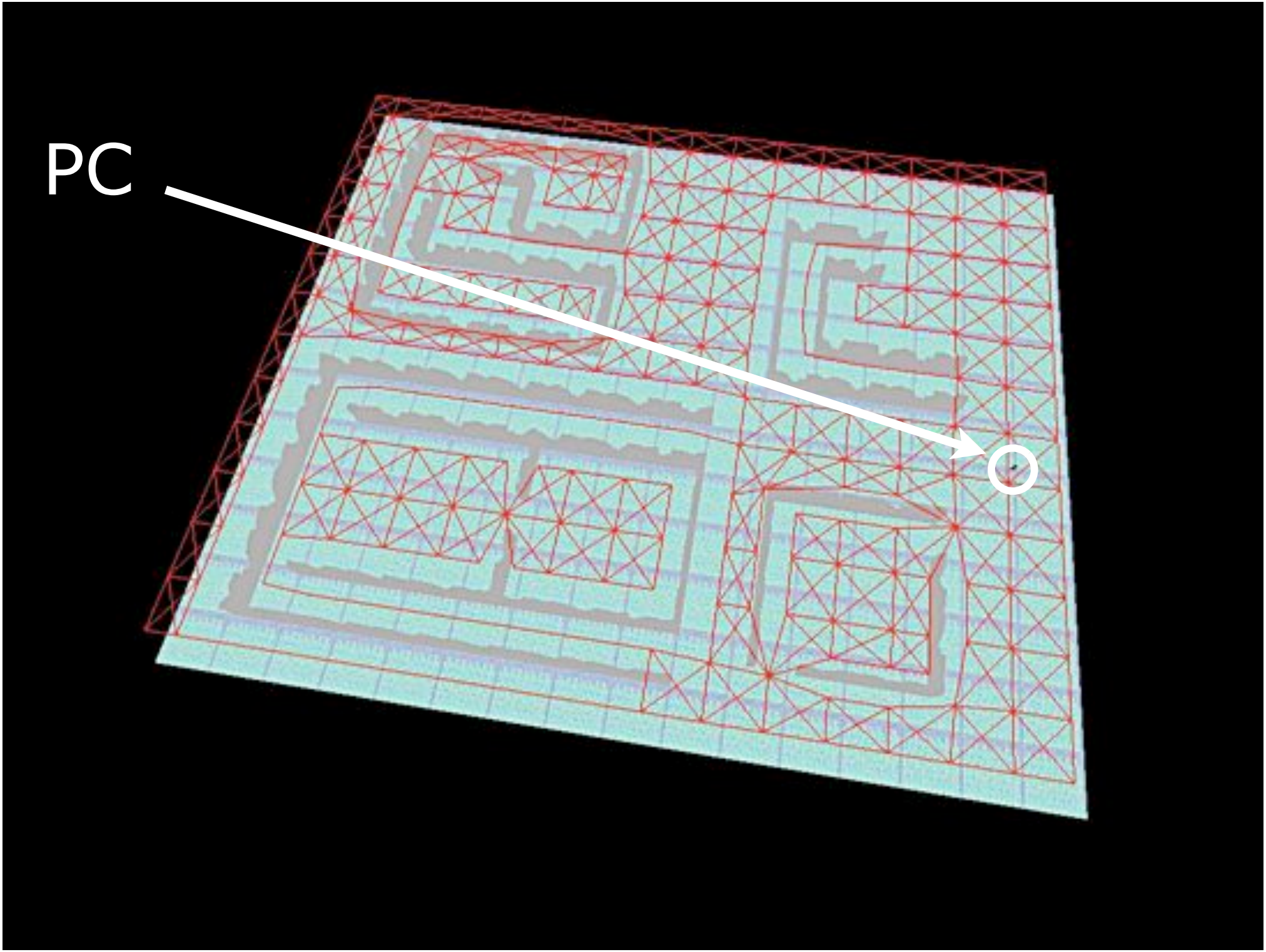
Total Work Compared to A*

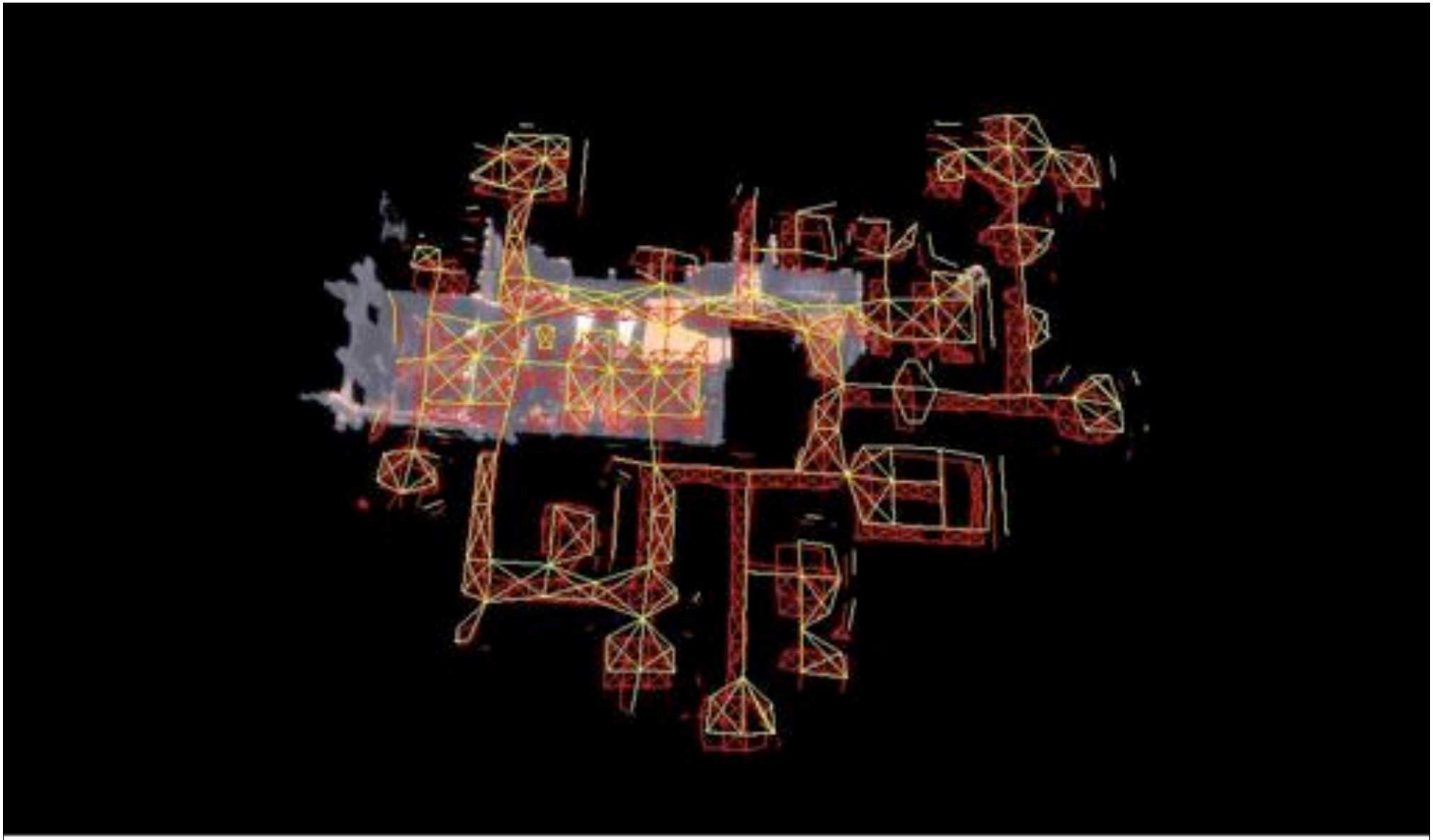


UNIVERSITY OF DENVER

Dragon Age: Origins

Engine Features


PC

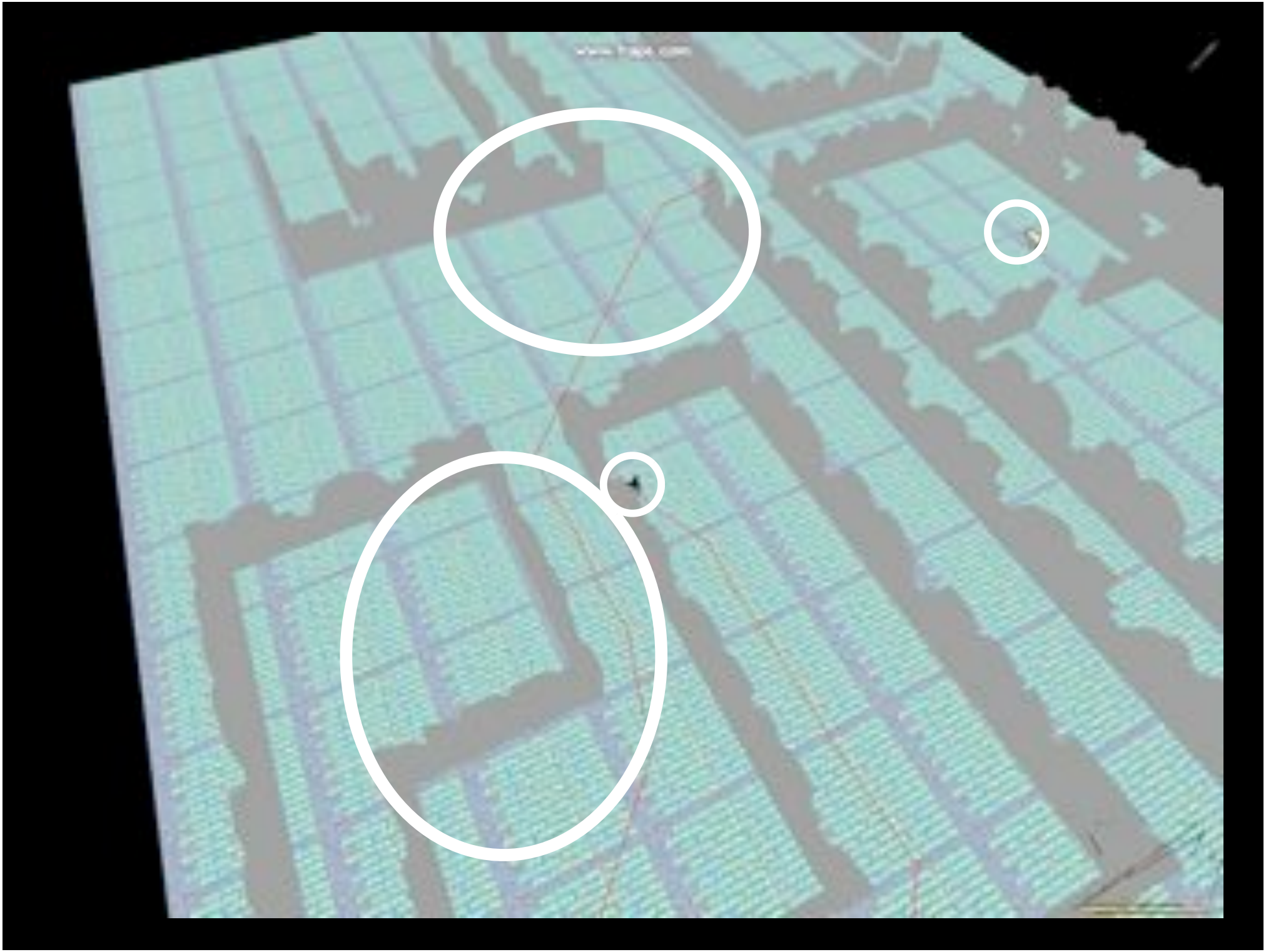


Dragon Age Pathfinding

Top Two Layers of Abstraction

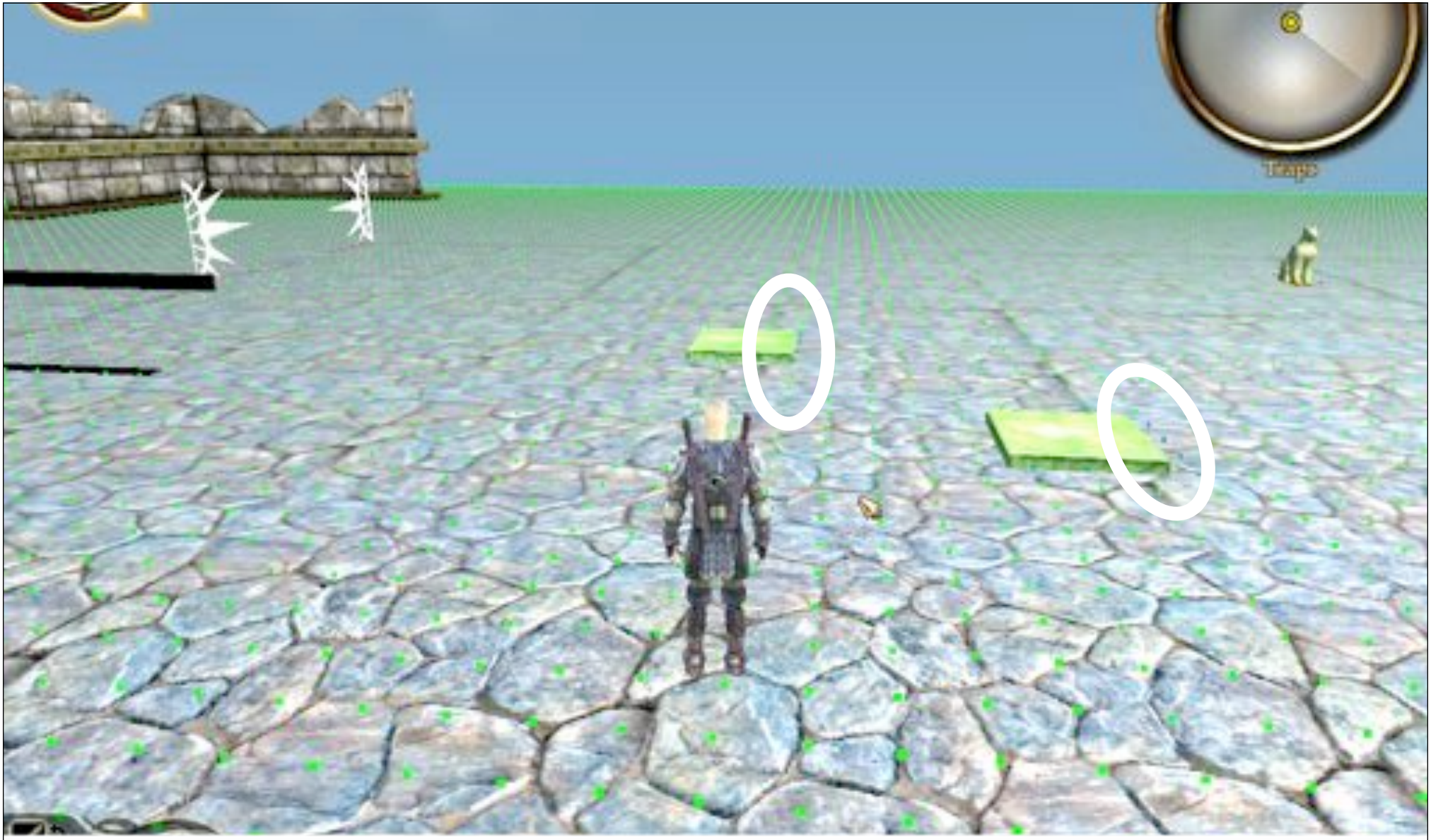
Dragon Age Pathfinding

Outdoor Map




Save:6573
www.fraps.com
Ostagar
DalishElf
[fps] 20.1273


www.fraps.com



Avoiding a trap


Camp

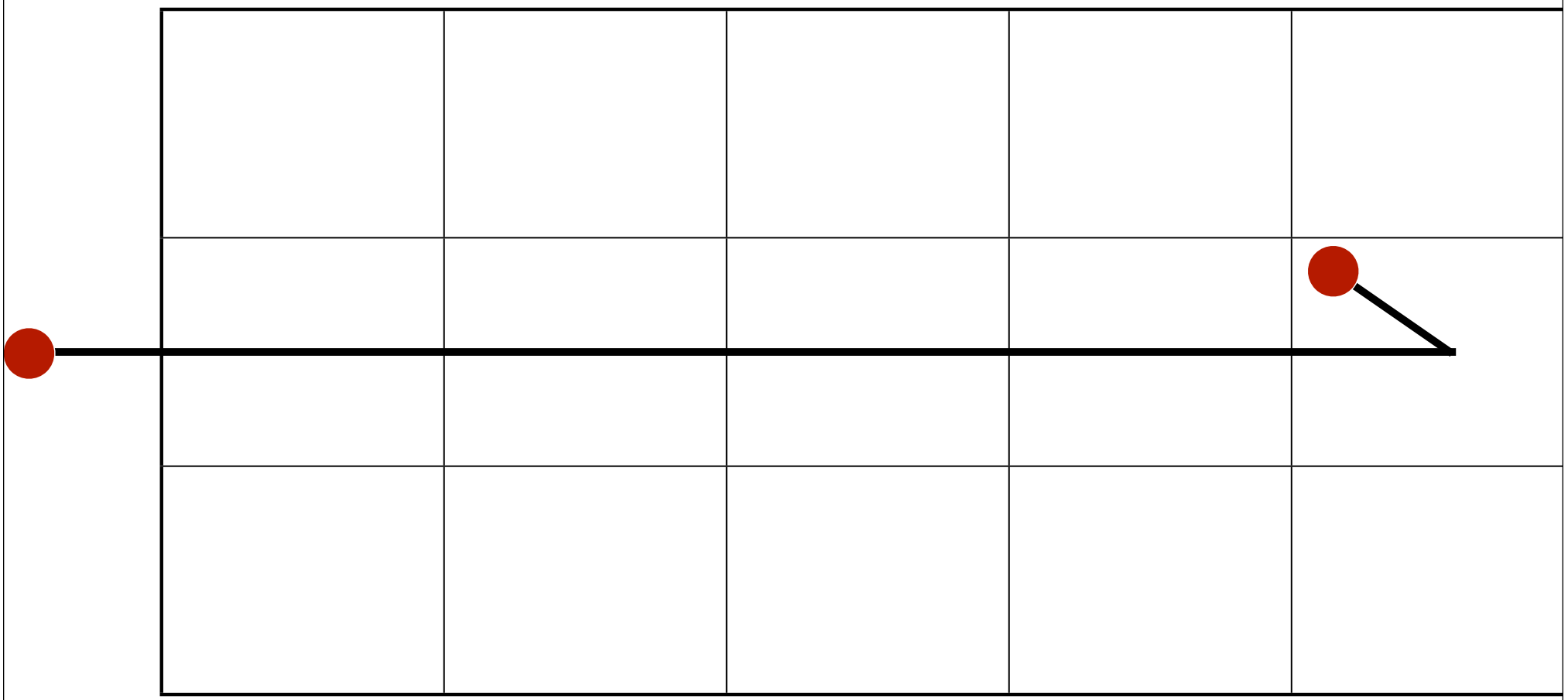
Artifacts of Abstraction

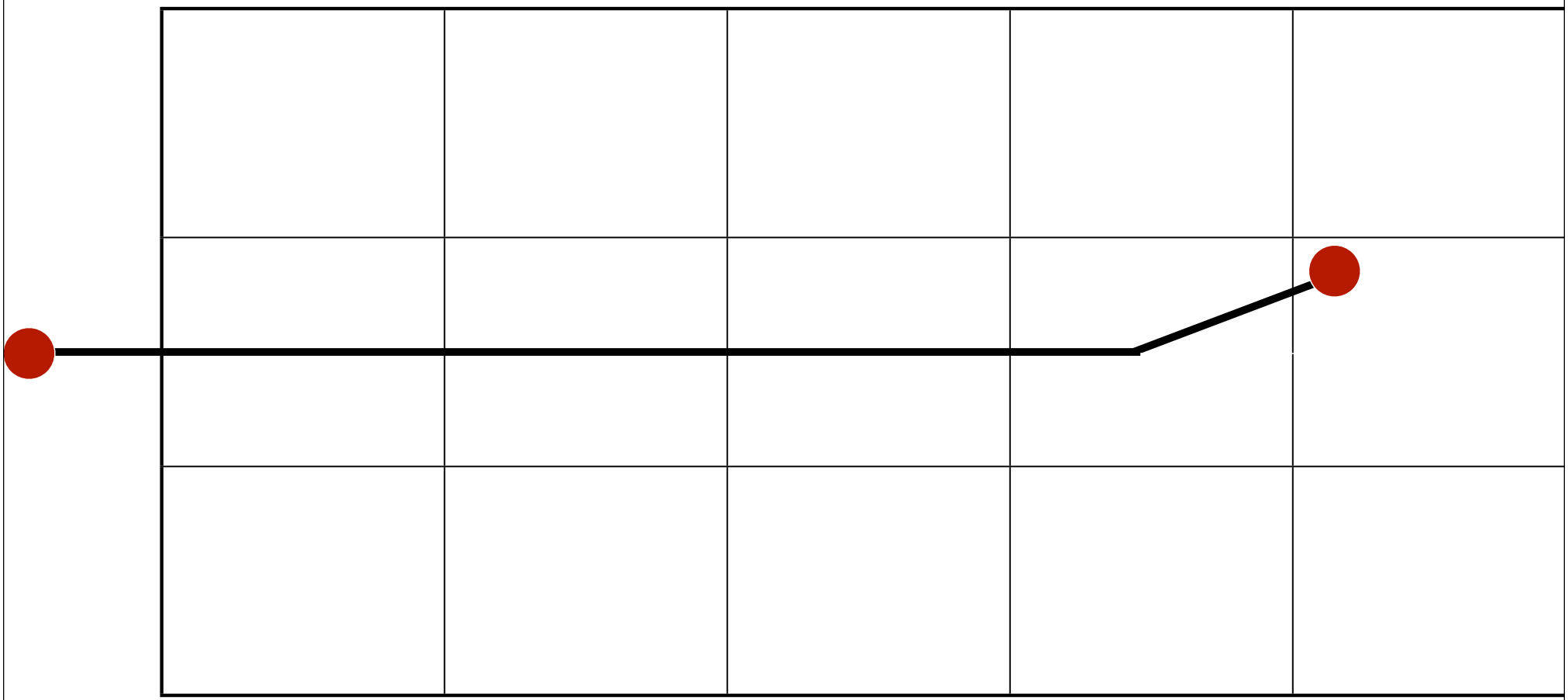
UNIVERSITY OF DENVER

Errors at the grid level

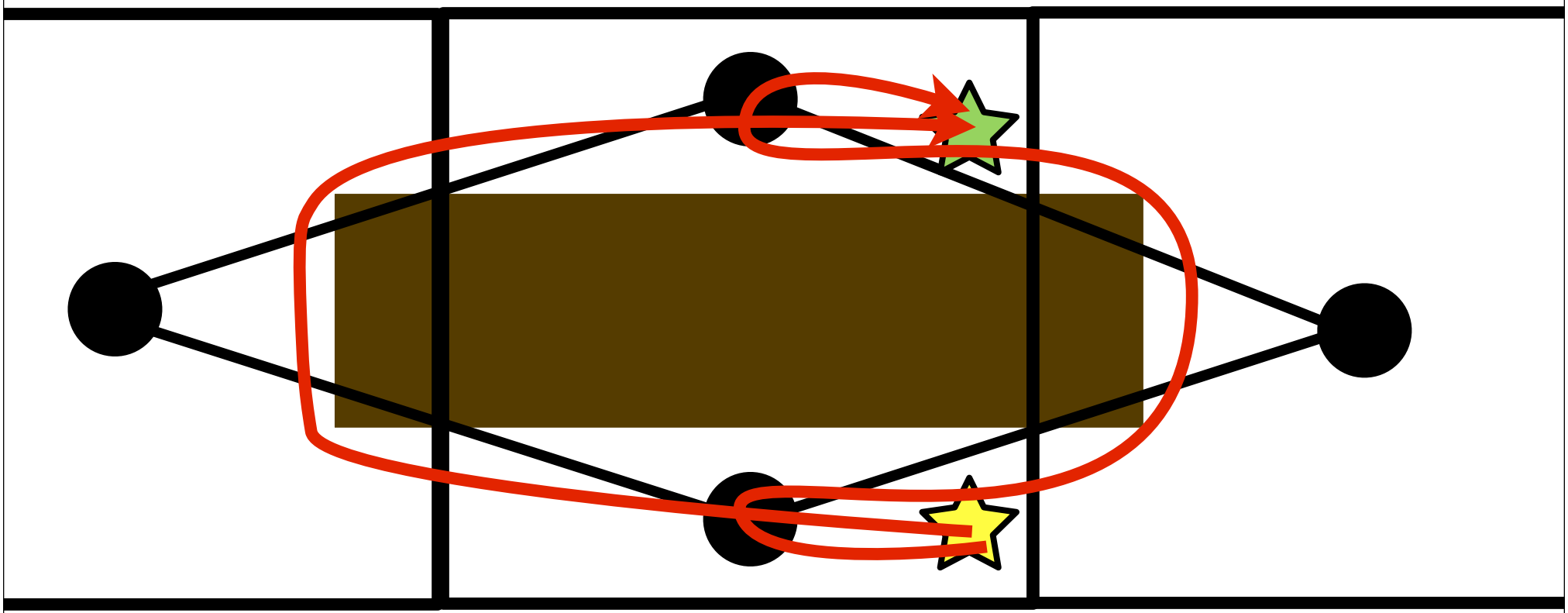
- Grid versus real-valued locations
  - PC/NPC location is a real-valued location
  - Mouse clicks are real-value location
  - Low-level planning is on a grid

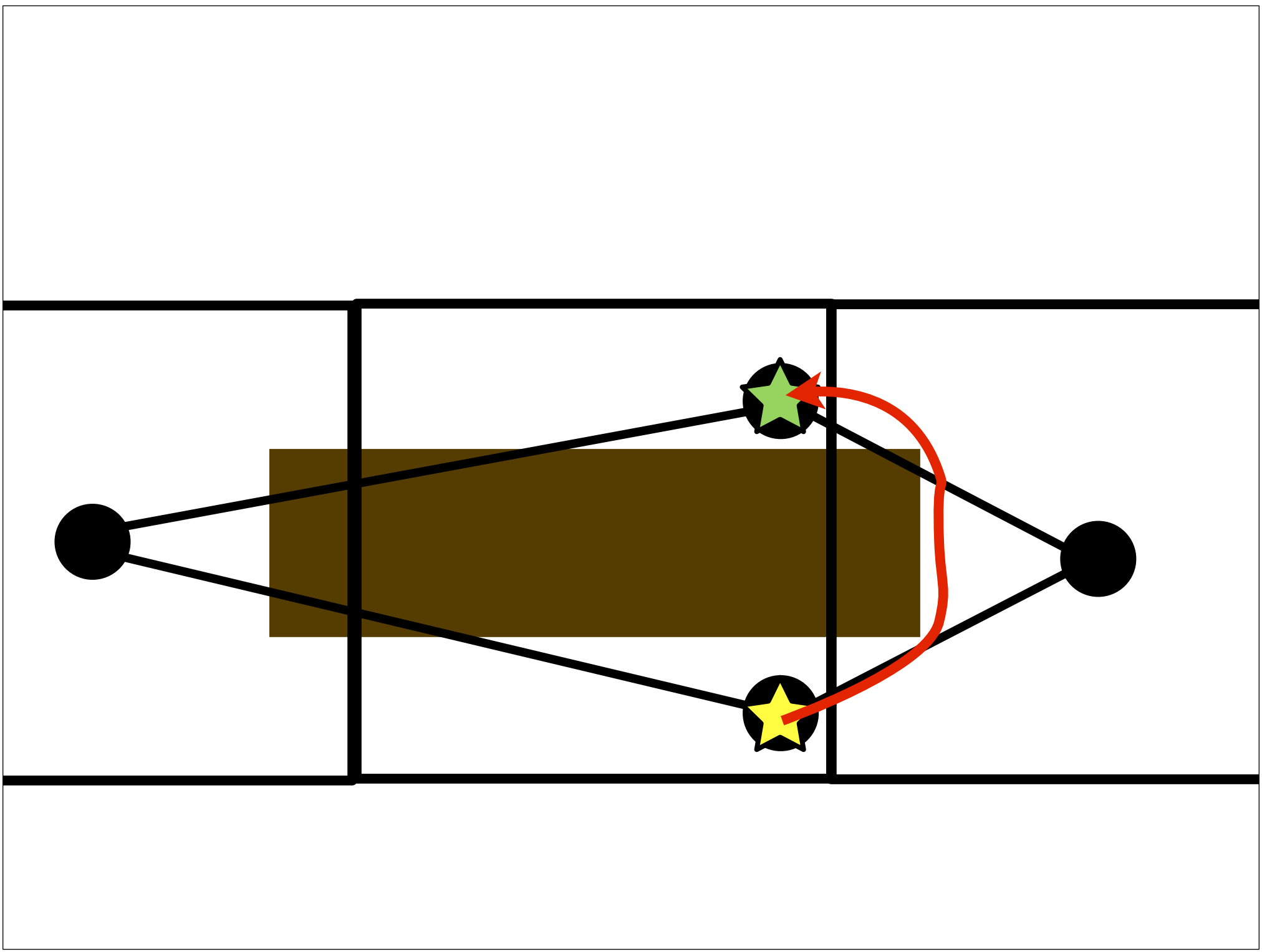

www.fraps.com
Orgrimmar Hall of

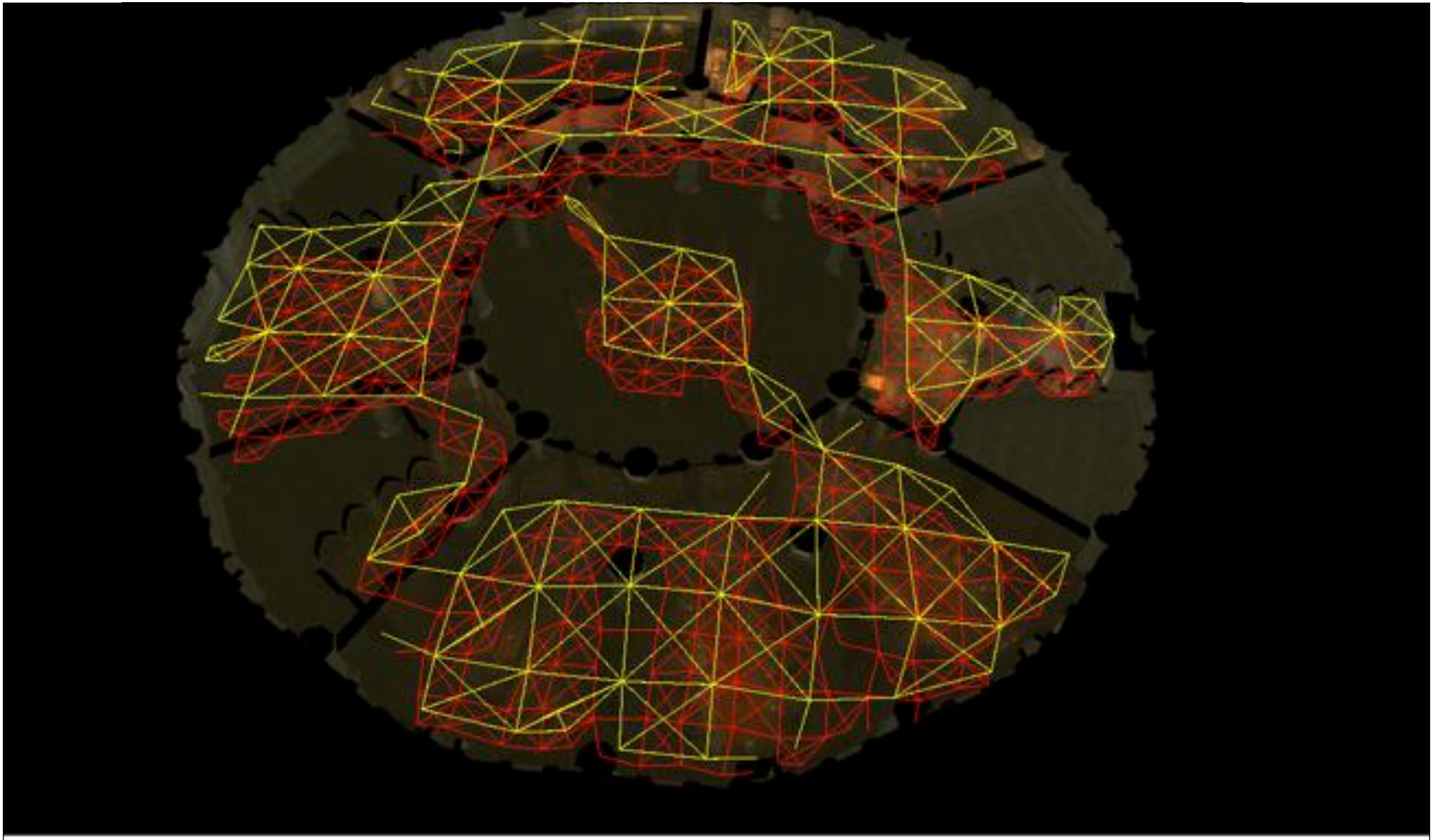




Solution -- end one step early and then go to goal

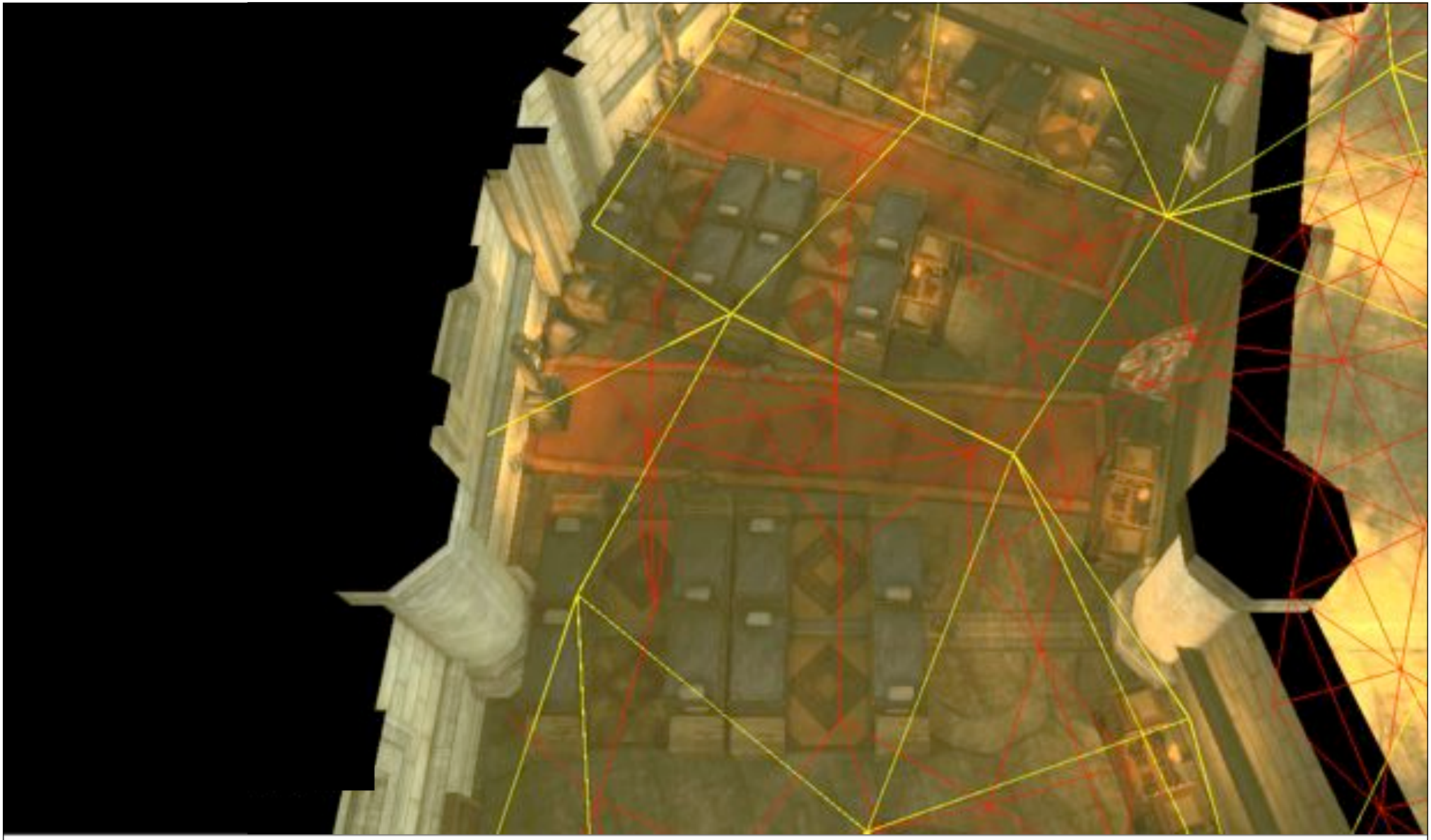






Dragon Age Pathfinding

Indoor Map



Dragon Age Pathfinding

Indoor Map: Detail

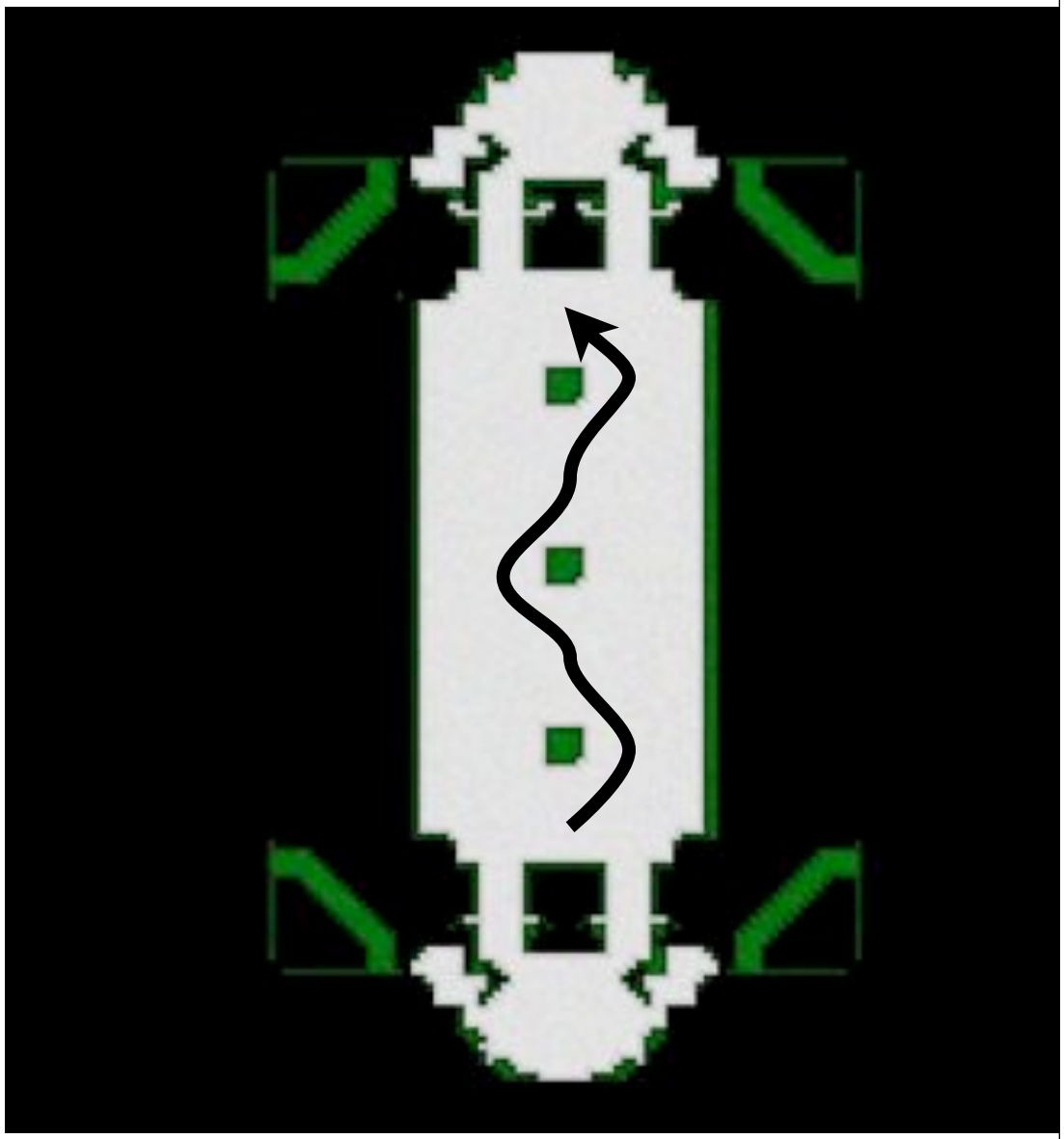

www.fraps.com
Apprentice Quarters

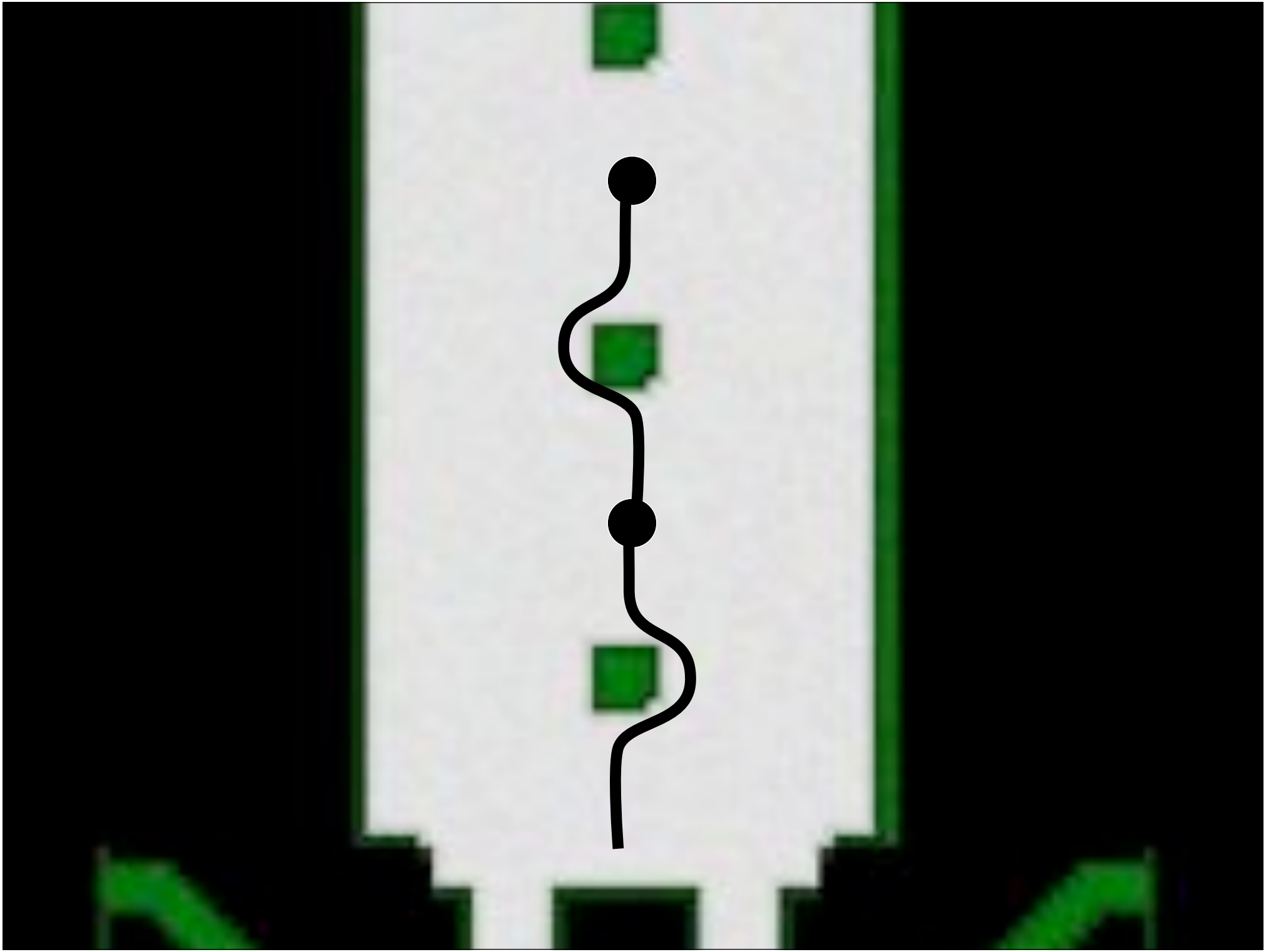

Orgrimmar Hall

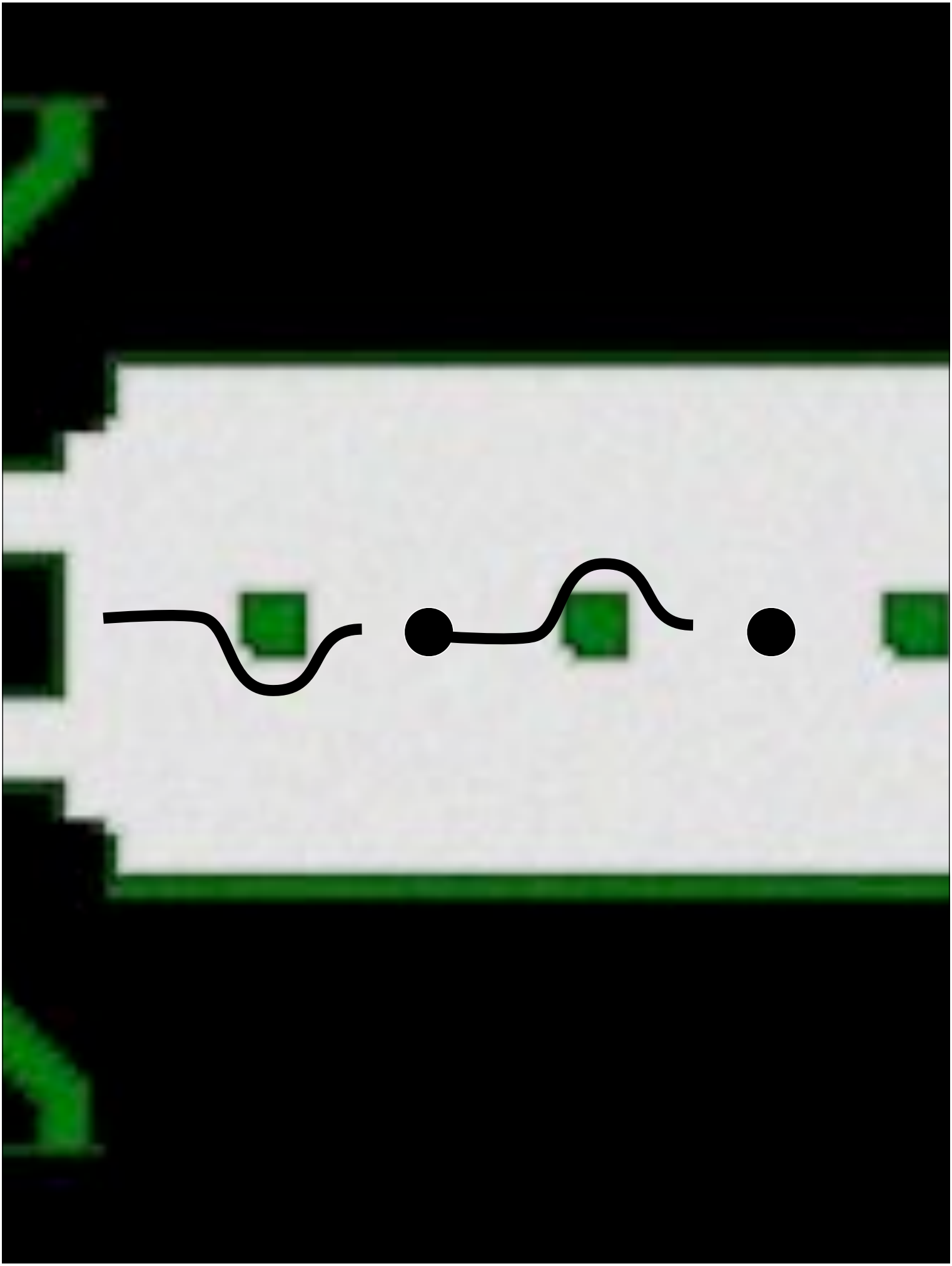
UNIVERSITY OF DENVER

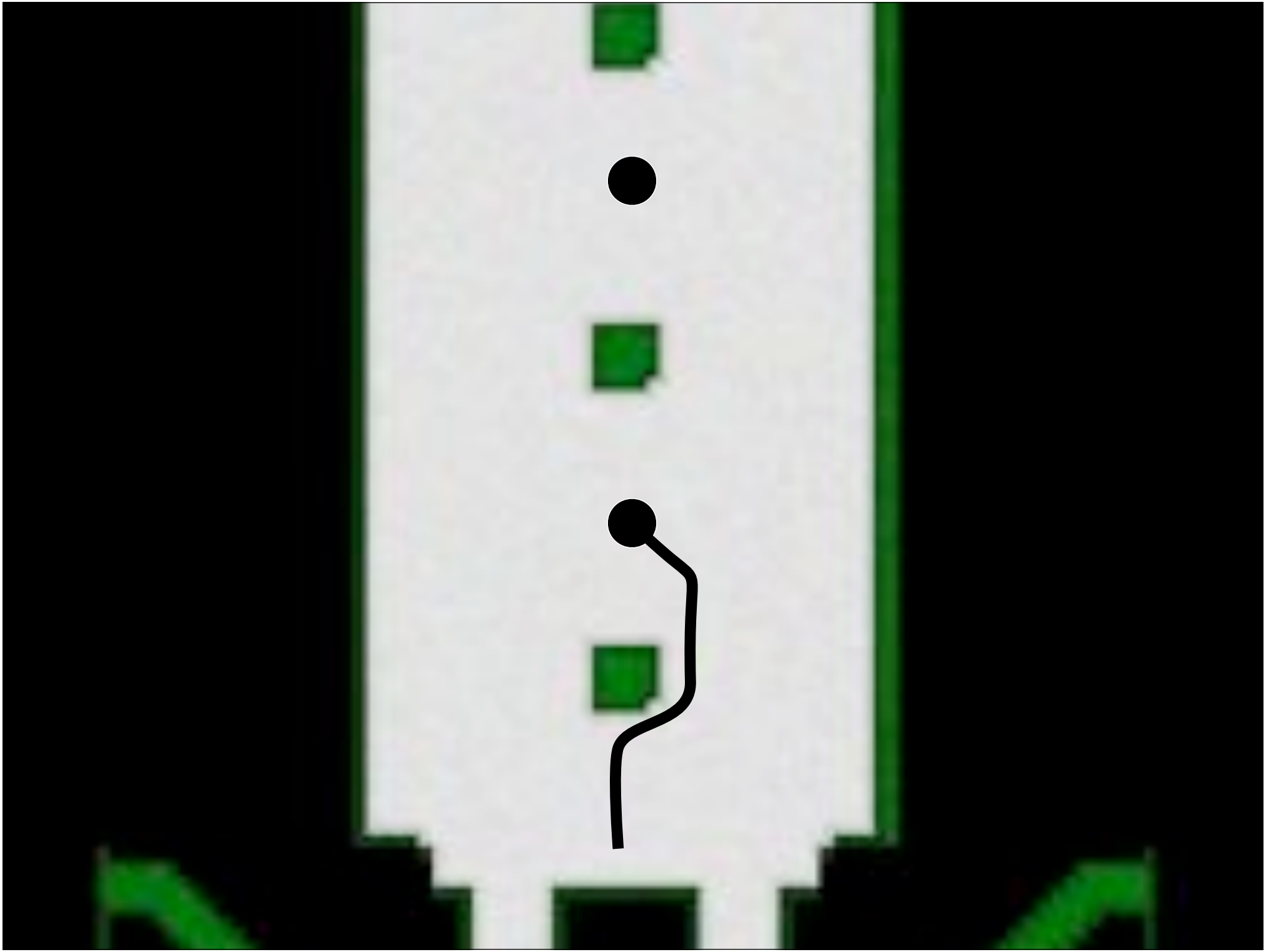
Path Continuity

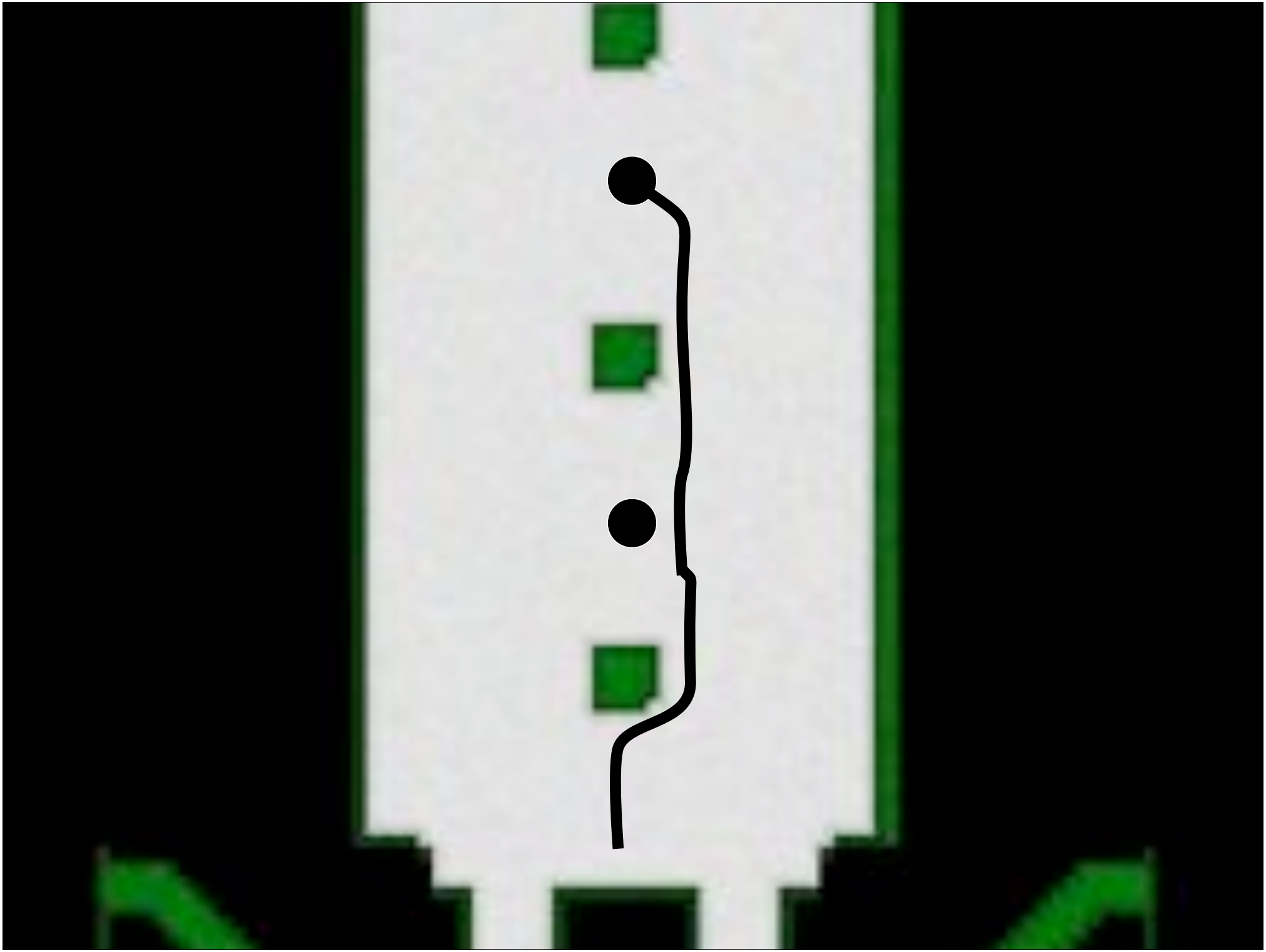
- Orzammar
- NPC doesn’t remember movement history

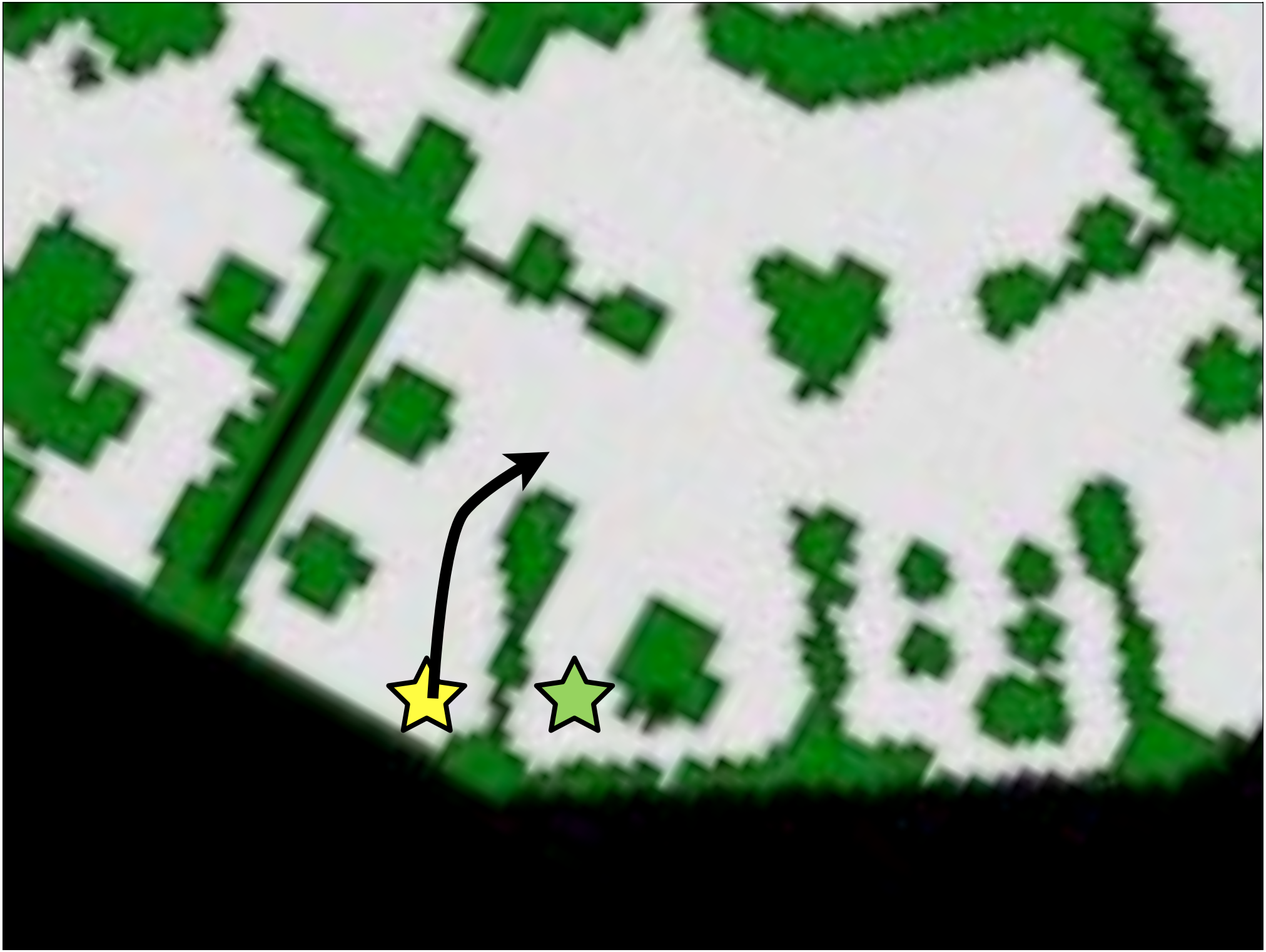


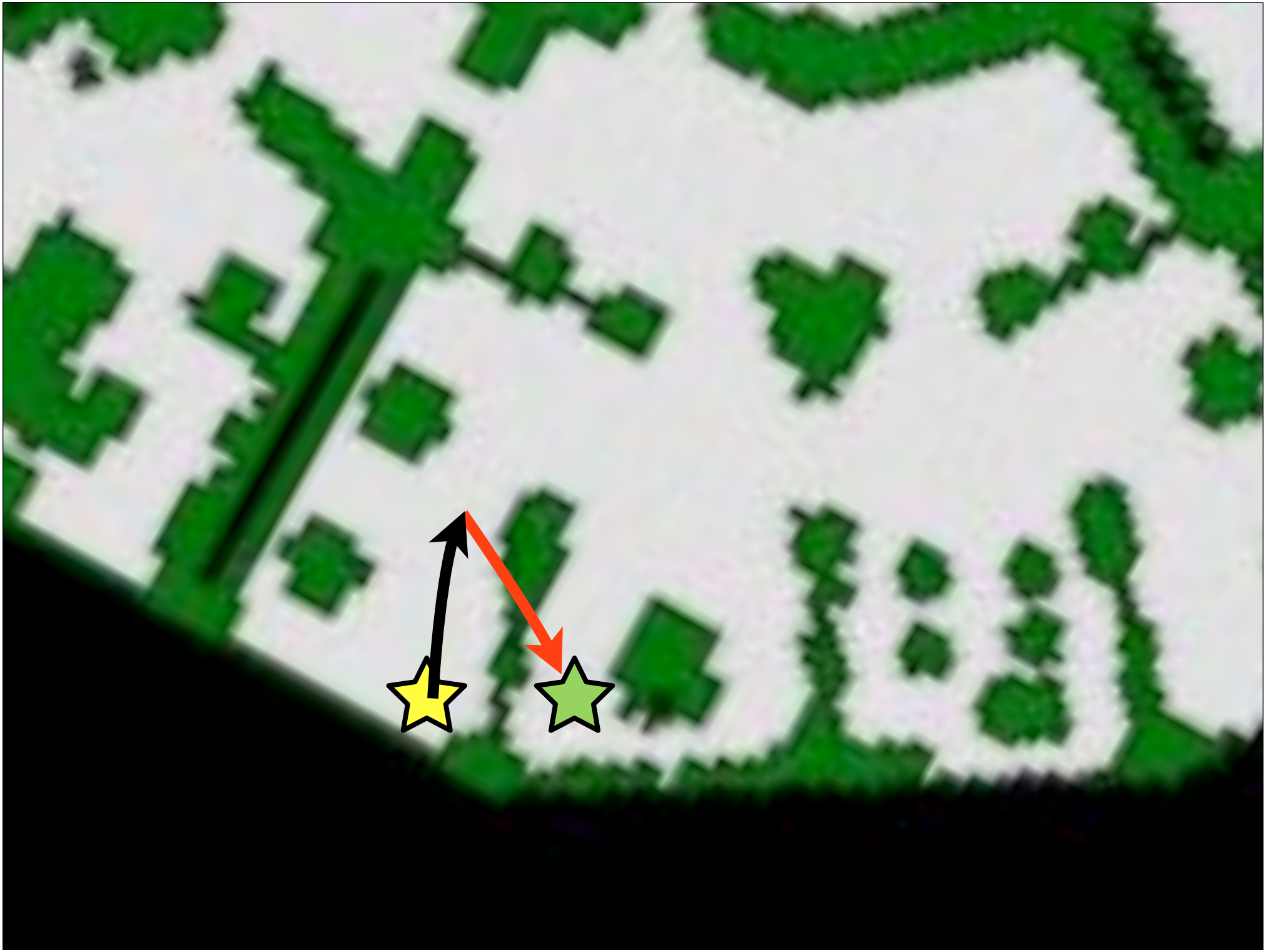












Heuristics

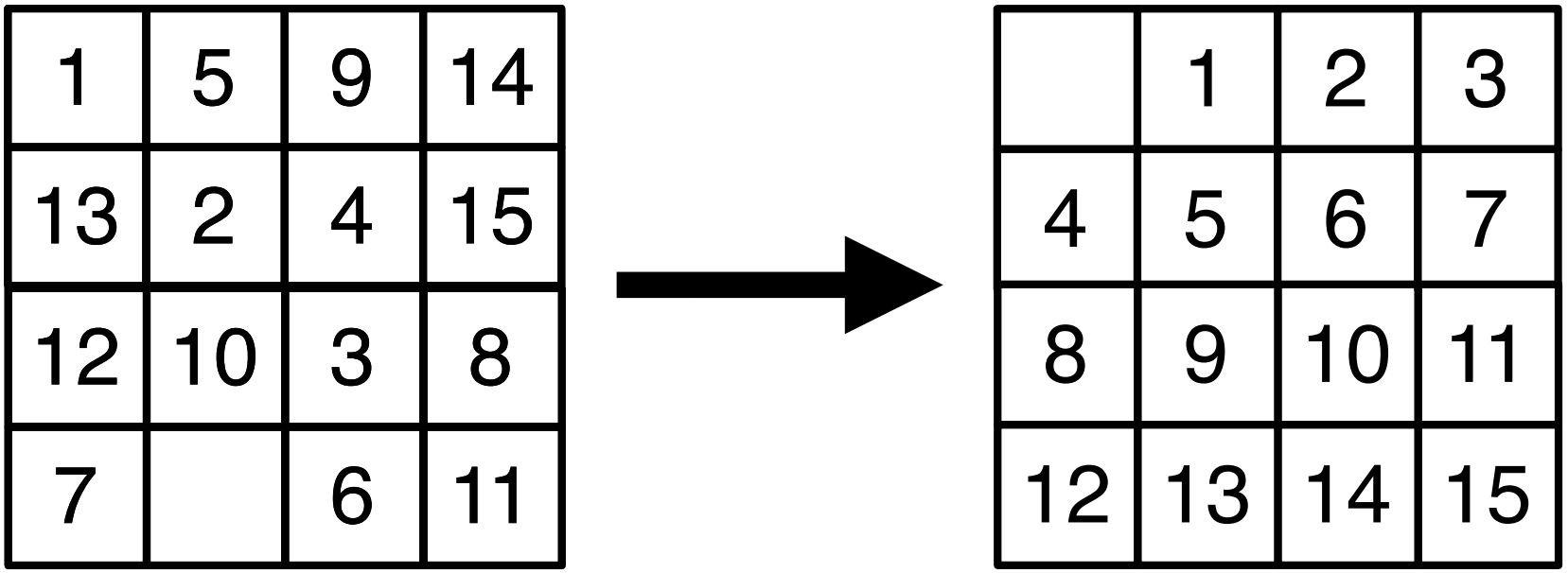
UNIVERSITY OF DENVER

Heuristics

- Background
  - Pattern Databases
- True-Distance Heuristics

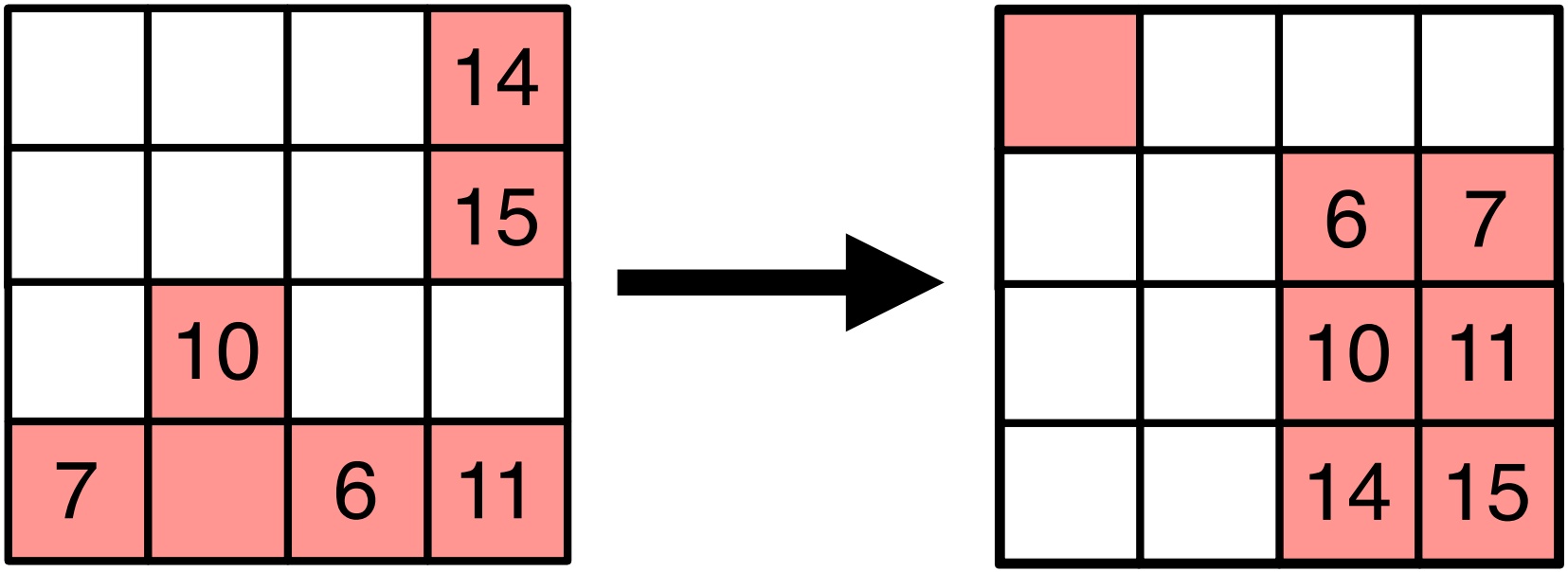
Previous work

- Pattern Databases (PDB)
  - Pre-computed, memory-based heuristic



Previous work - PDB

- Mark some states as “don’t care”
  - Creates abstract state space
  - Solve all states in abstract state space exactly



UNIVERSITY OF DENVER

PDB -- analysis

- PDBs have been applied to state spaces which grow exponentially
  - Branching factor b
  - Search depth d
- State space size $O(b^d)$

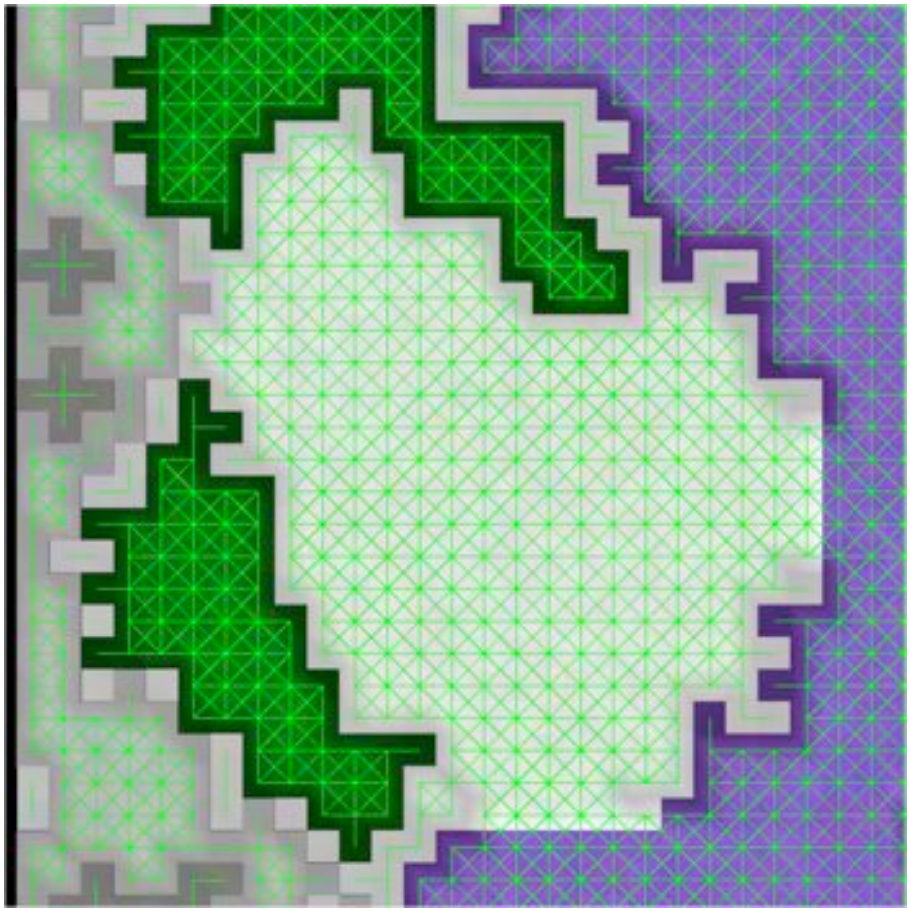
UNIVERSITY OF DENVER

PDB -- analysis

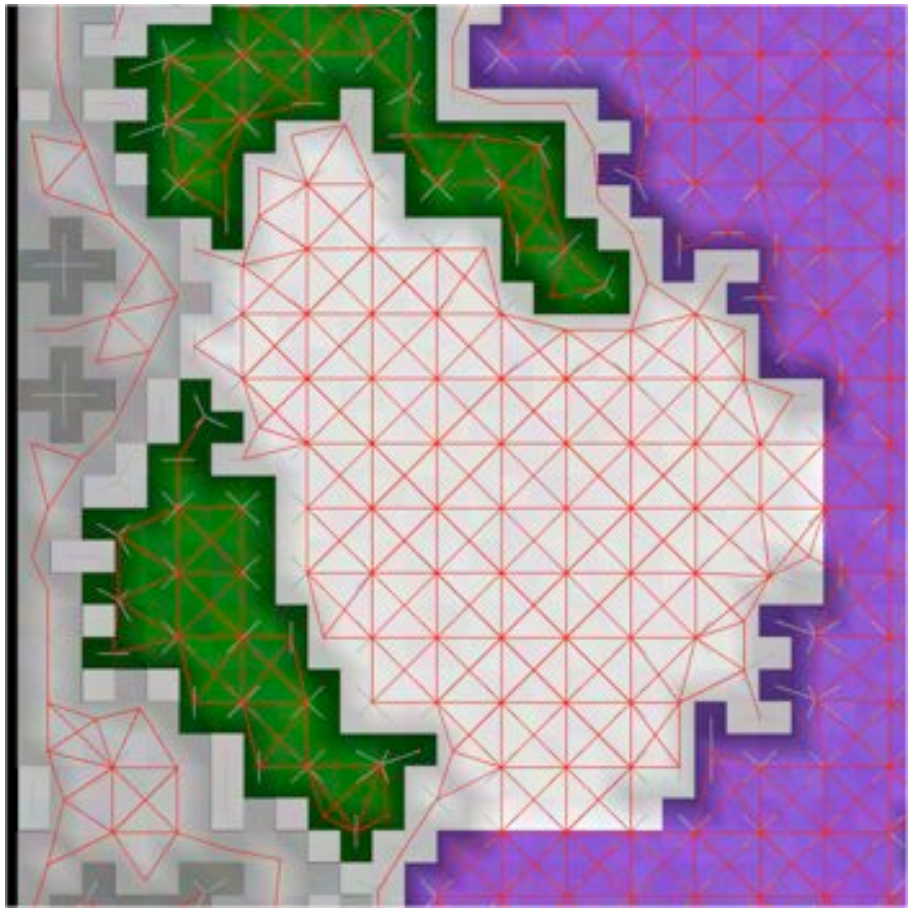
- Assume we can store 1/*f* of the state space
- Assume abstract branching factor is the same
- Let *c* be the max. dist. in the abstract state space
  - $b^c = 1/f \cdot b^d$
  - $c = d - \log_b(f)$
- Heuristics only lose a small amount of accuracy
  - 60MB 15-puzzle PDB has max value of 54
  - 10TB state space has max value of 80

UNIVERSITY OF DENVER

Abstraction in Maps



11,614 states



3,455 states

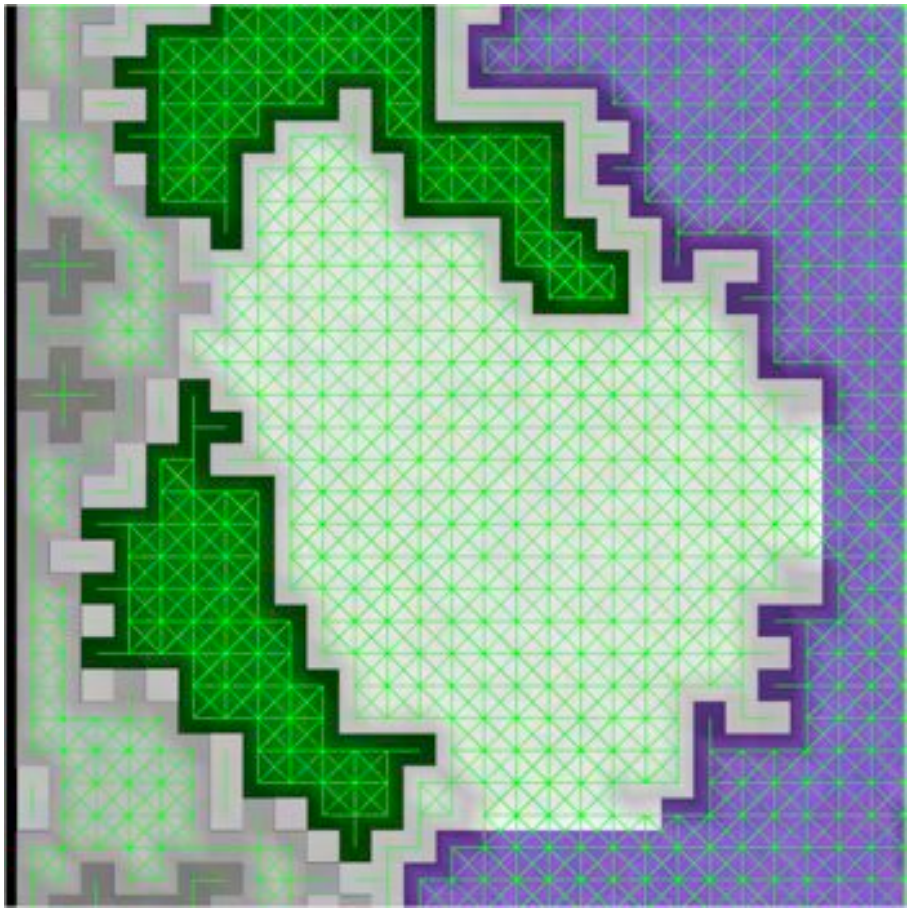
UNIVERSITY OF DENVER

Analysis -- maps

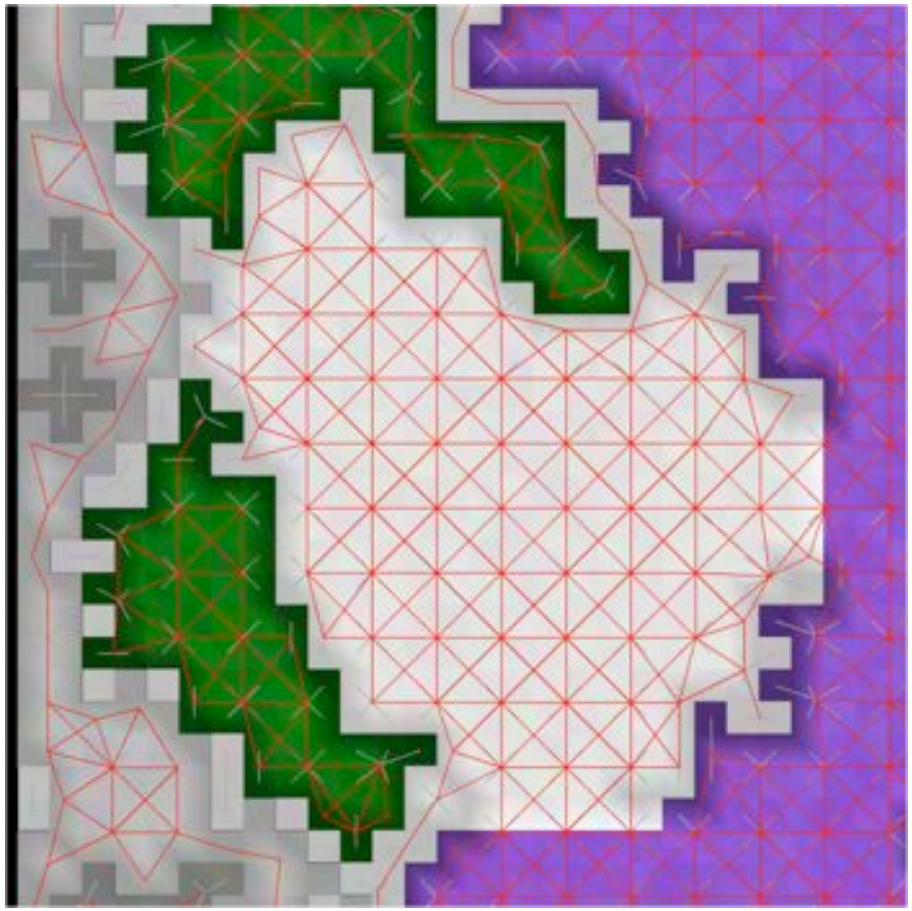
- 2D maps grow with r^2 (*r* is radius of search)
- If we can store $1/f$ of the full state space
  - $c^2 = 1/f \cdot r^2$
  - $c = r/\sqrt{f}$
- Abstract search space by factor of 4
- Maximum heuristic value is reduced by factor of 2
  - Bad for heuristics!

UNIVERSITY OF DENVER

Abstraction in Maps



11,614 states
Width: 157



3,455 states
Width: 80

UNIVERSITY OF DENVER

Ideals

- Want a heuristic that can be used between any two states
- Want to minimize cost of the heuristic
- Heuristic should be based on true-distances in the world instead of abstract distances
  - New class, True-Distance Heuristic

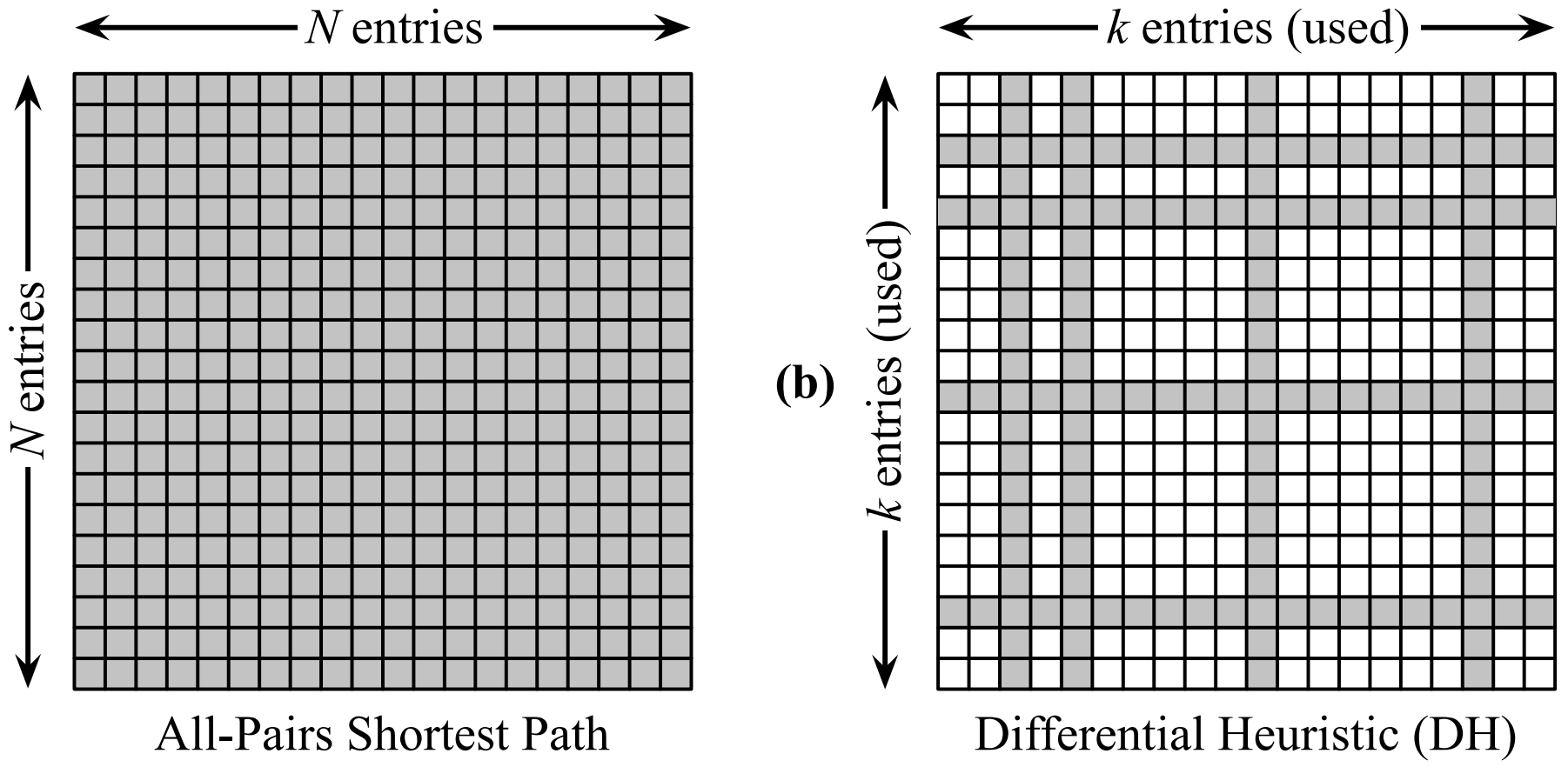
UNIVERSITY OF DENVER

Differential Heuristic (ALT)

- If we have a solution to the single-source shortest-path problem for a state *s*, we can use it to get a heuristic between any two states
  - Assume undirected graph
  - h(a, b) = | d(a, *s*) - d(b, *s*) |
- Invented and re-invented in several different communities

Differential Heuristic

- Need to store multiple heuristics and take the max to get good results

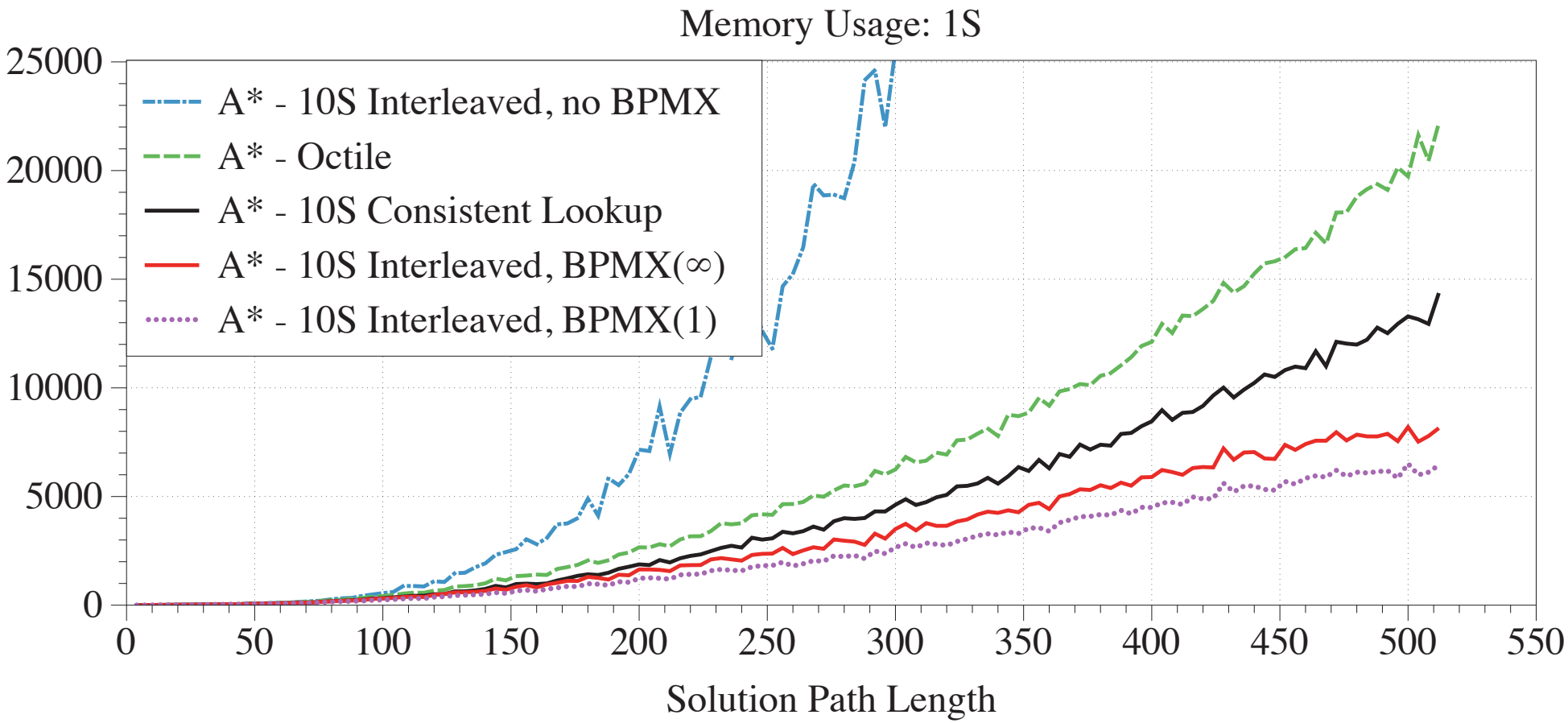


A* - 10 diff. heuristics

UNIVERSITY OF DENVER

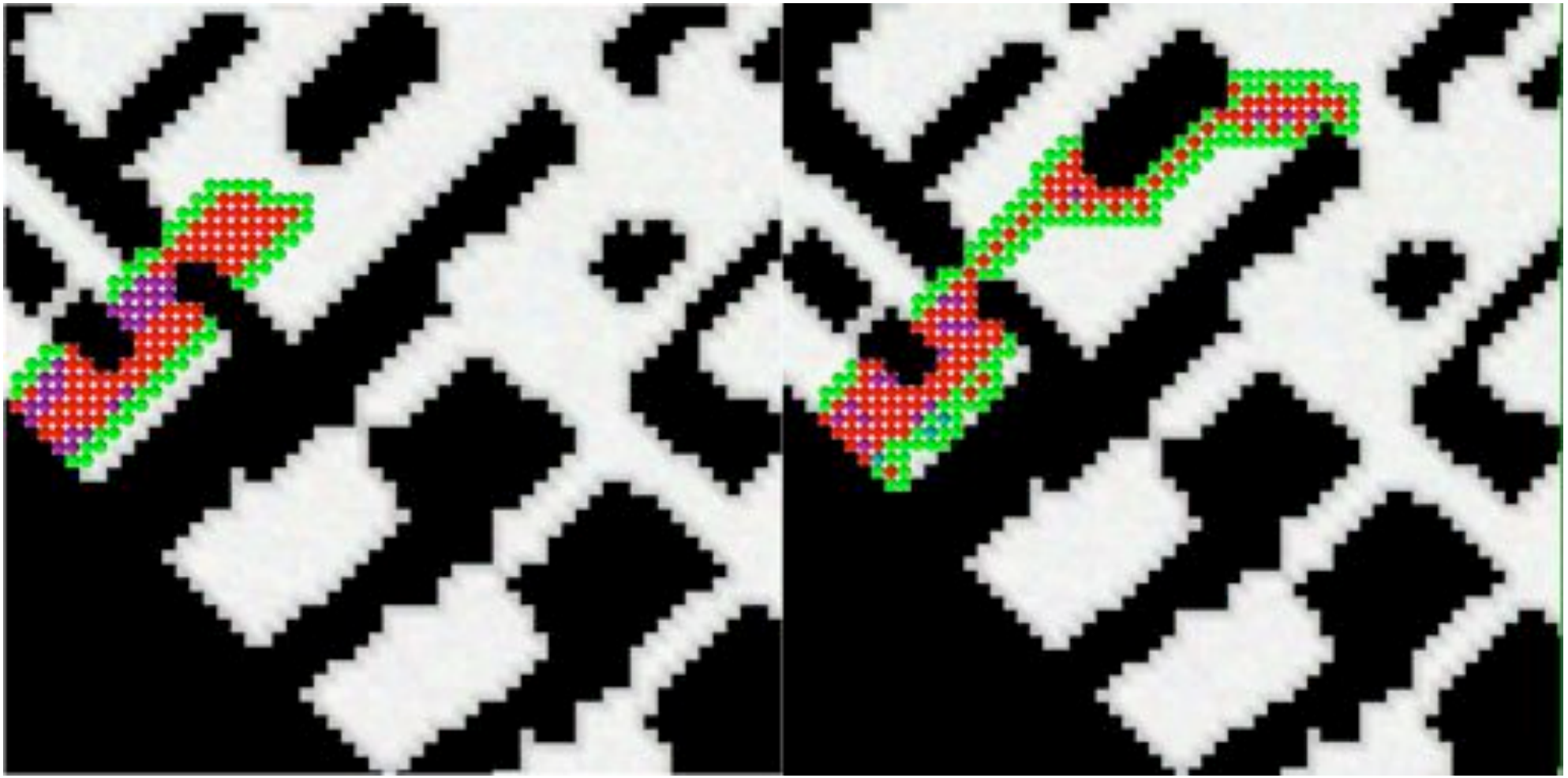
Improving performance

- What if we don't have much memory available?
- Can the heuristic be compressed?
  - Only store heuristics at some nodes
  - Goal is fixed, so find all heuristics around the goal
  - During search use whatever heuristic is available
    - Subtract distance from goal



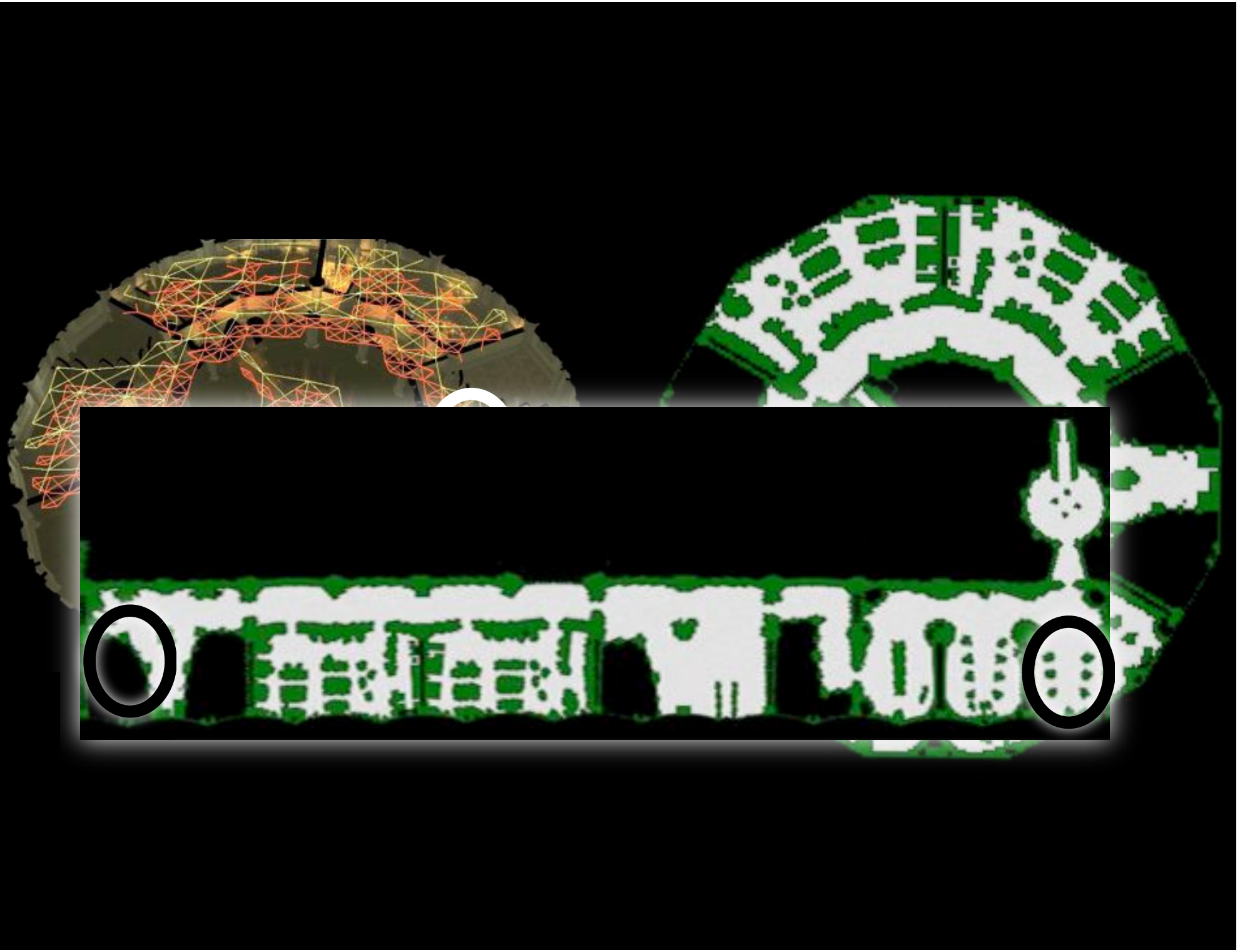
Inconsistent heuristics in theory and practice Ariel Felner, Uzi Zahavi, Robert Holte, Jonathan Schaeffer, Nathan Sturtevant, Zhifu Zhang, Artificial Intelligence

The Compressed Differential Heuristics
Meir Goldenberg, Nathan Sturtevant, Ariel Felner, Jonathan Schaeffer, AAAI 2011



Building Heuristics and Maximum Variance Unfolding

Euclidean Heuristic Optimization, Chris Rayner, Michael Bowling, Nathan Sturtevant, AAAI 2011



UNIVERSITY OF DENVER

Euclidean heuristics

- Euclidean heuristics are heuristic values that can be computed as distances in some Euclidean space of d dimensions
  - $h(i, j) = \|y_i - y_j\|$
- Let Y be the n by d matrix storing the vectors y_i. Y implicitly encodes the heuristic function.
- Can we find the best Euclidean heuristic?

UNIVERSITY OF DENVER

Finding Euclidean Heuristics

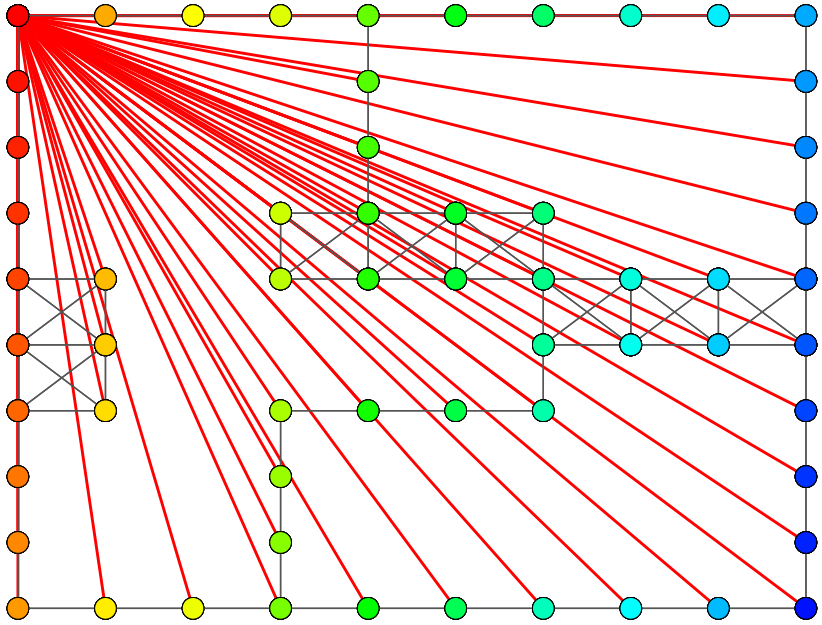
$$\underset{Y}{\textbf{minimize}} \quad \mathcal{L}(Y)$$

$$\textbf{subject to} \quad Y \text{ is admissible and consistent}$$

UNIVERSITY OF DENVER

Expressing admissibility constraints

- Want to avoid pre-computing all paths and adding them as constraints in problem formulation



- Local consistency → global consistency → admissibility

Loss function

- Minimize squared error between true distance and heuristic
  - Weight coordinates *i*, *j* which are more important

$$\mathcal{L}(Y) = \sum_{i,j} W_{ij} \left| d(i,j)^2 - \|y_i - y_j\|^2 \right|$$

- Reformulate as:

$$\textbf{maximize}_{Y} \sum_{i,j} W_{ij} \|y_i - y_j\|^2$$

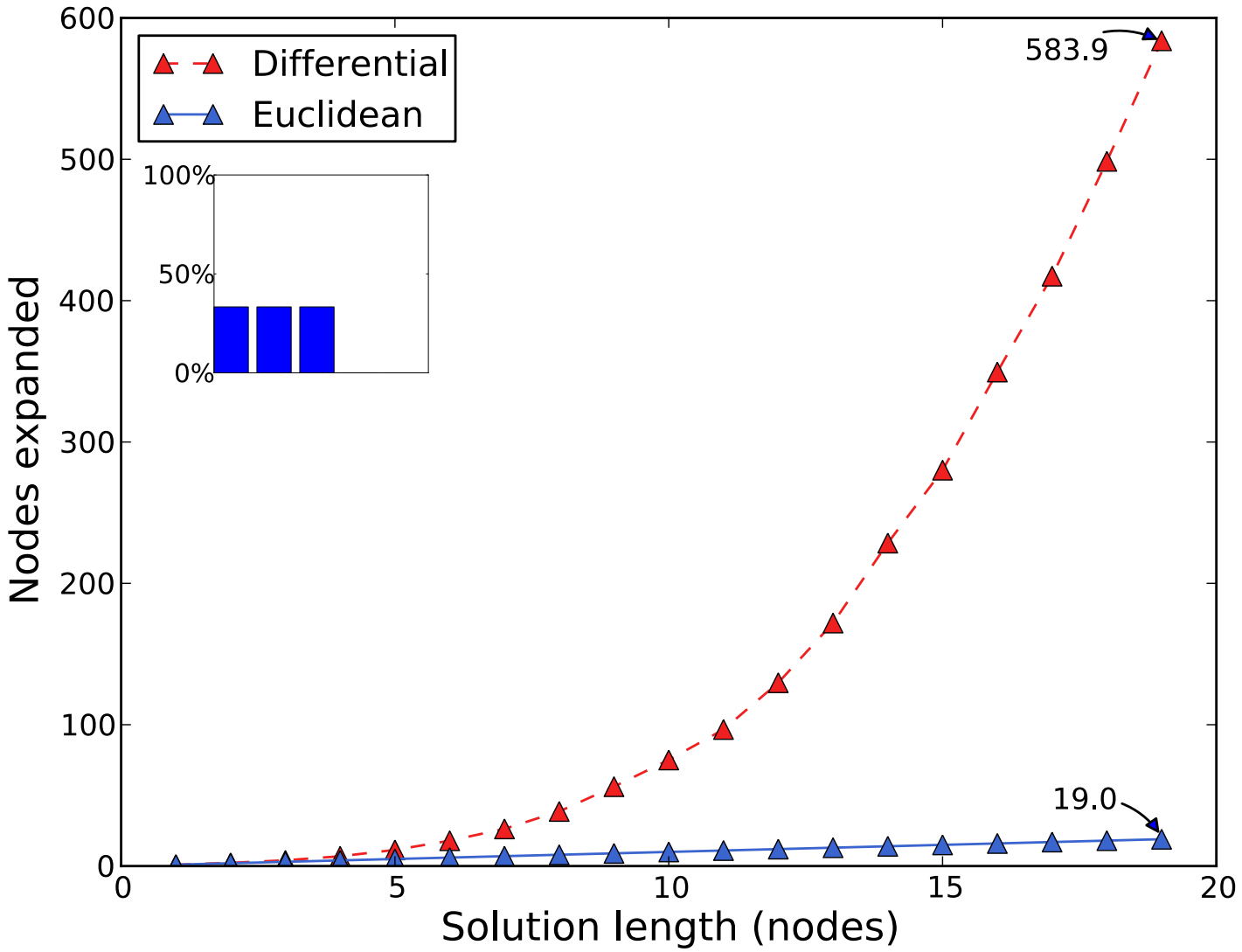
UNIVERSITY OF DENVER

Optimization problem

- Weighted generalization of MVU [Weinberger et al 2006] (for nonlinear dimensionality reduction)
  - Links learning a heuristic to manifold learning
  - Two separate areas which haven't been connected before
- A differential heuristic is a **one-dimensional embedding** in Euclidean space
  - Previously had heuristic methods for choosing where to place heuristics
  - Now have formal optimization

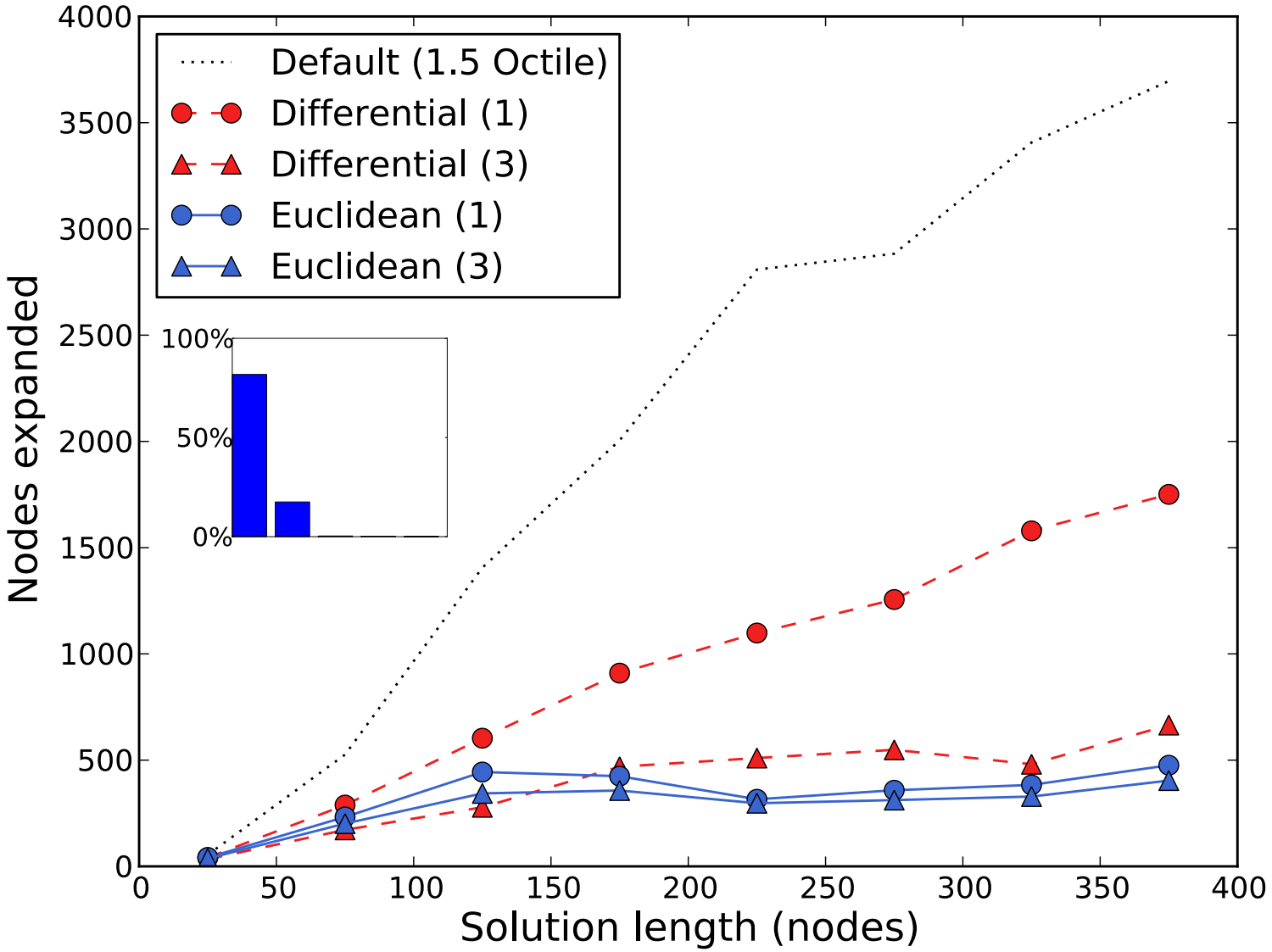
UNIVERSITY OF DENVER

Example usage: 3-dimensional cube



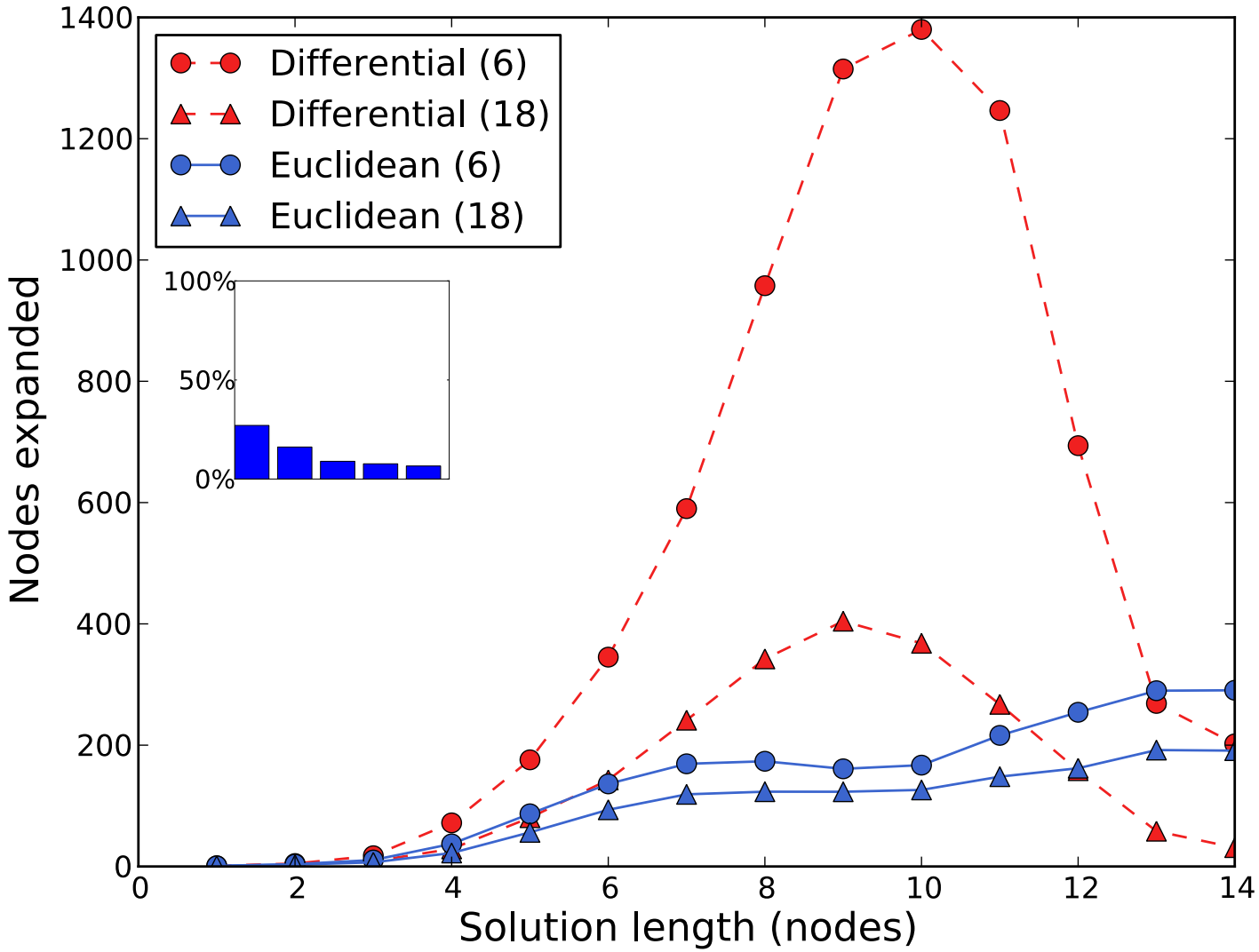
UNIVERSITY OF DENVER

Example: Dragon Age Maps



UNIVERSITY OF DENVER

Example: Word search (edit distance)



FOUR
FOUL
FOIL
FAIL
FALL
FILL
FILE
FIVE

Summary

UNIVERSITY OF DENVER

Summary

- Looking for paths through state space
- Abstraction and heuristic techniques used to speed search
  - Availability of technique depends on underlying problem characteristics
- Large class of interesting problems which fit in memory

UNIVERSITY OF DENVER

Many other areas of search

- Planning
  - Building better heuristics from logic description of problem
- Randomizing search
  - Multiple OPEN lists
  - Monte-Carlo variations
- Suboptimal search