# Beyond A*:

## Speeding up pathfinding through hierarchical abstraction

Daniel Harabor
NICTA & The Australian National University

# Motivation

- Myth: Pathfinding is a solved problem.

# Motivation

- Myth: Pathfinding is a solved problem.
  - Reality: Pathfinding is not solved!

# Motivation

- Myth: Pathfinding is a solved problem.
  - Reality: Pathfinding is not solved!
    - A* underlies many pathfinding systems but...
    - Many open problems persist!

# Problem: Efficient pathfinding in very large environments

Image: World of Warcraft (Blizzard Entertainment)

# Problem: Dynamic terrain

Image: Company of Heroes (Relic Entertainment)

# Problem: Different unit classes

Image: Red Alert 3 (Electronic Arts)

# Problem: Cooperative Pathfinding

Image: Settlers: Heritage of Kings (Blue Byte Software)

# Rest of this talk

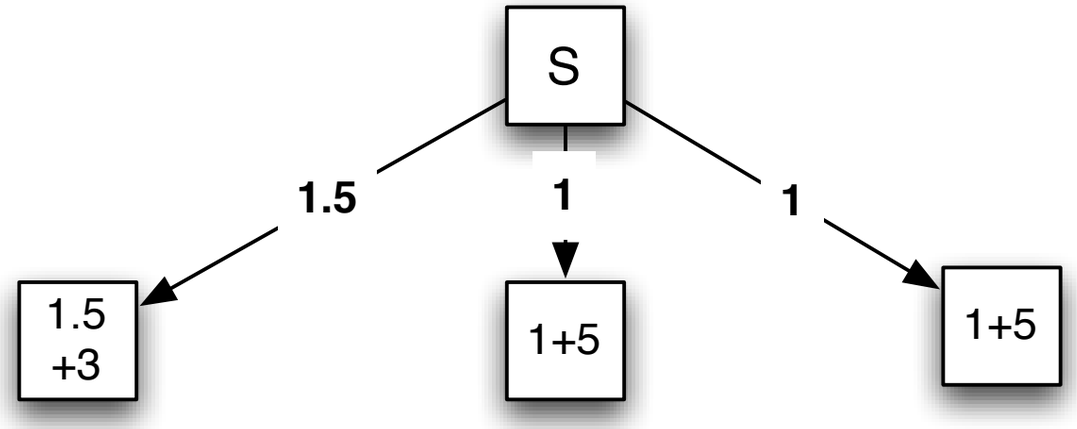
- Some background on A*
- Then, a story in 3 parts:
  - Part 1: Hierarchical pathfinding
  - Part 2: Dealing with diversity
  - Part 3: Probabilistic roadmaps

# Background

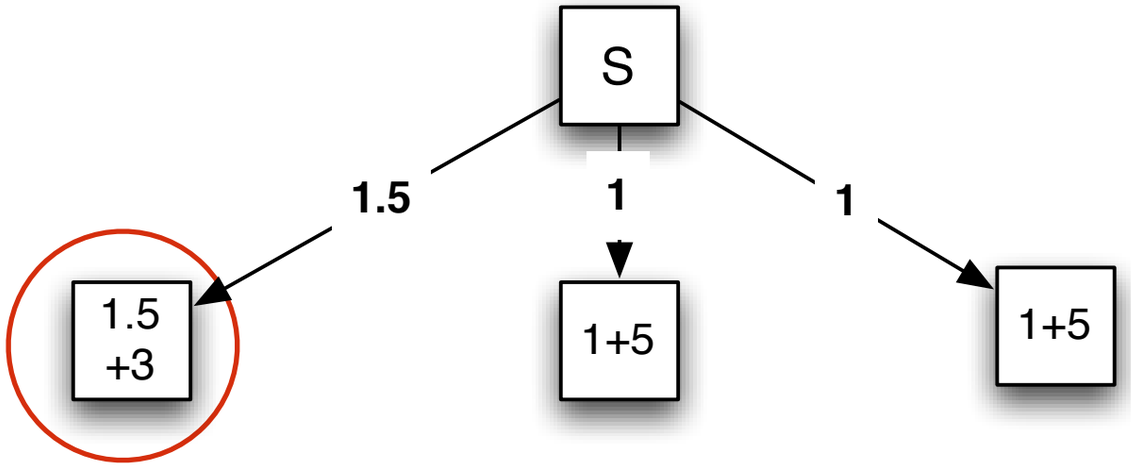
# A* In Brief

- [Hart, Nilsson & Raphael, 1968]
- Heuristic search
- **f(n) = g(n) + h(n),** where:
  **g(n)** = distance so far
  **h(n)** = estimate remaining distance
- Complete + Optimal

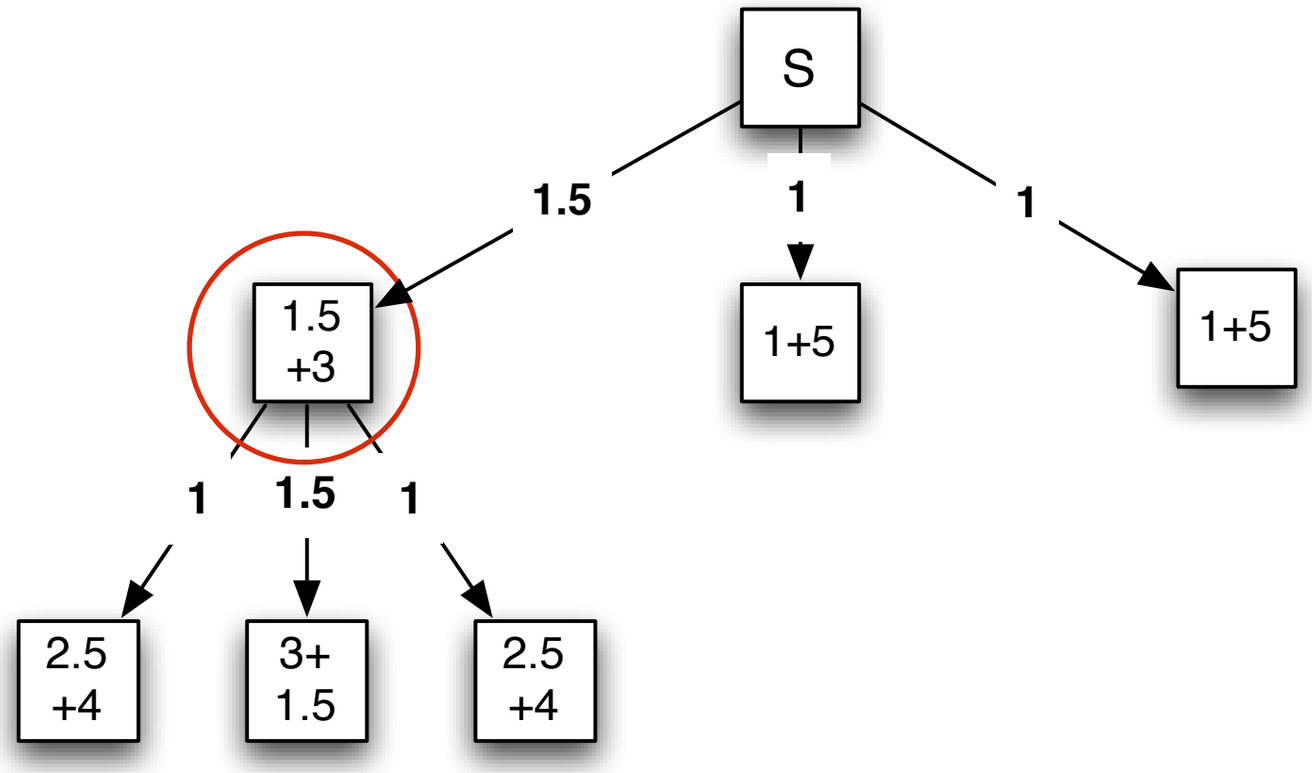
# A* example



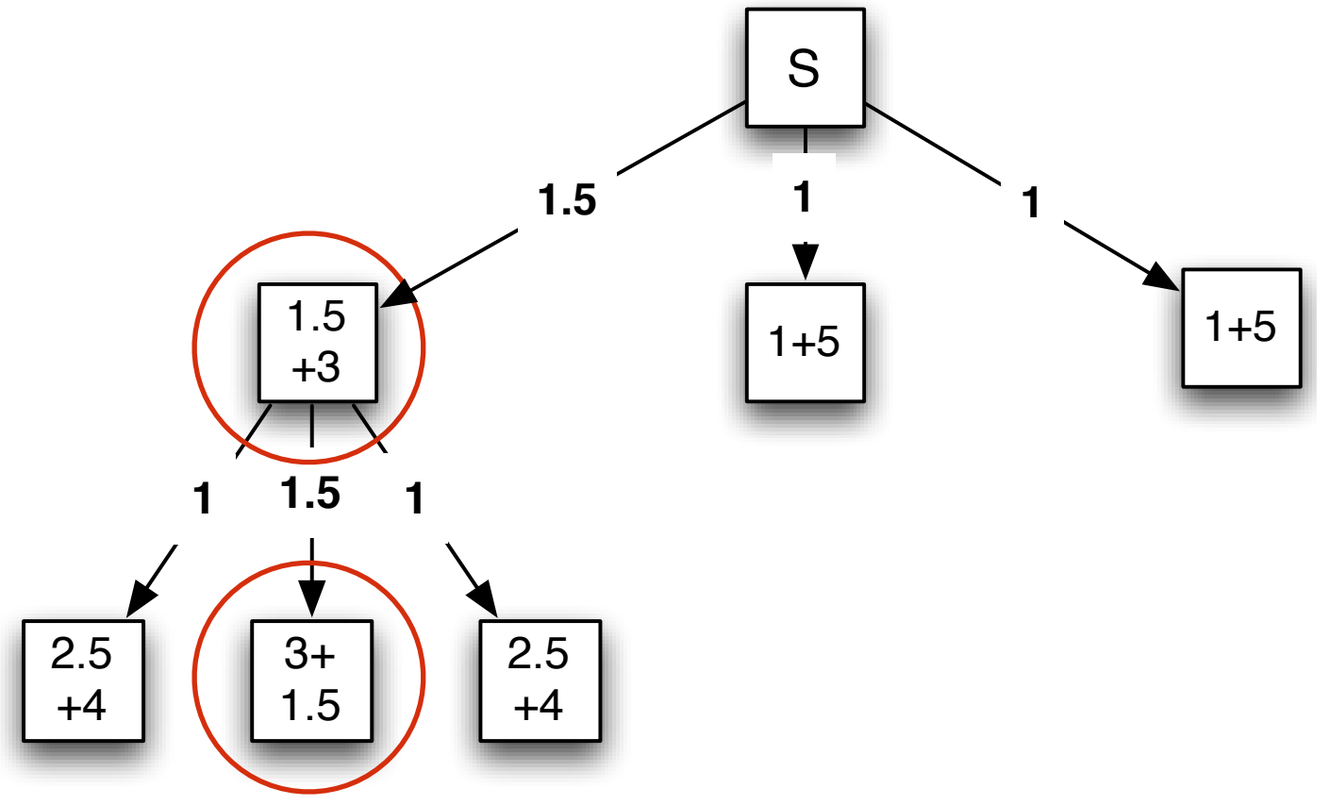
# A* example



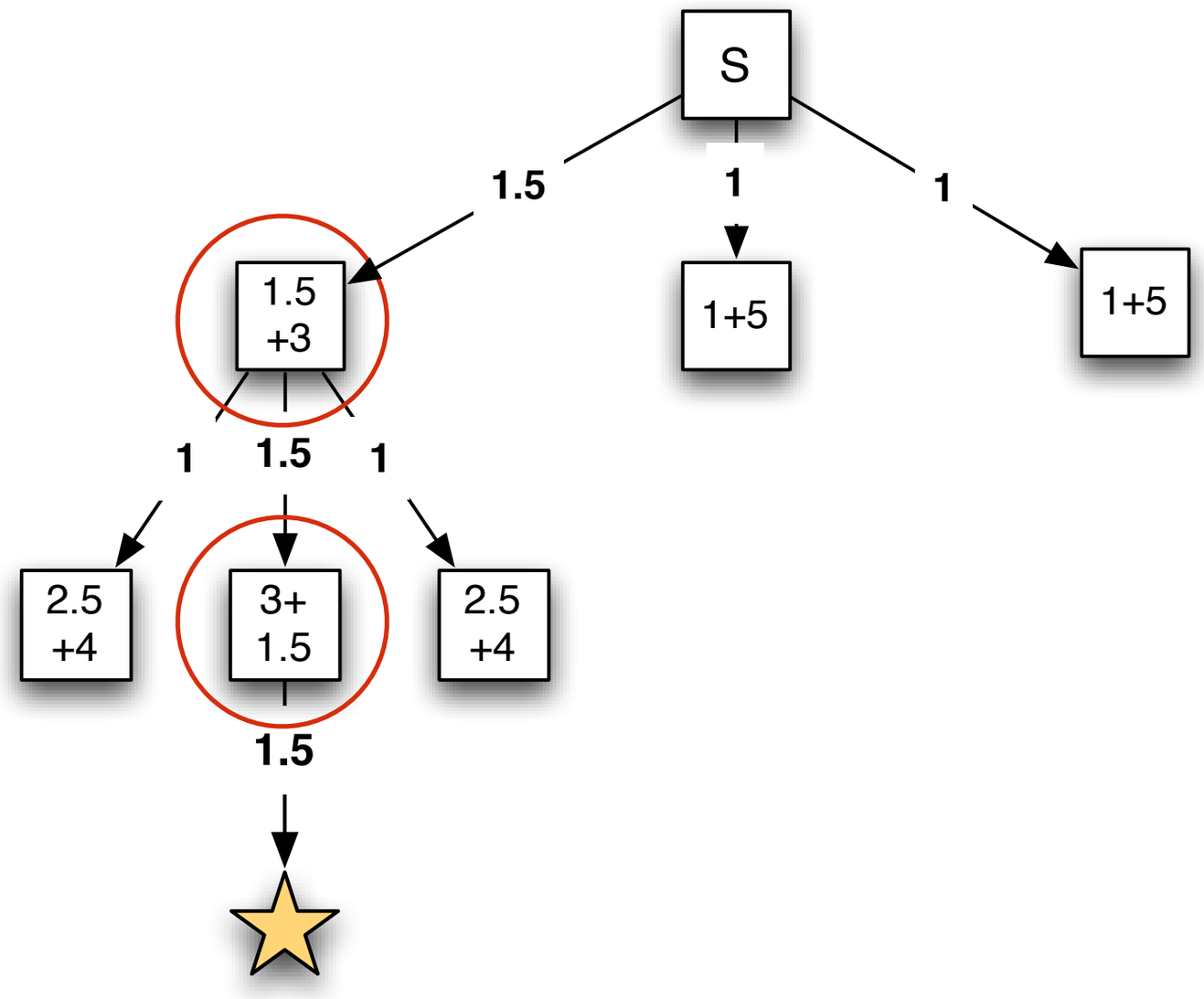
# A* example



# A* example



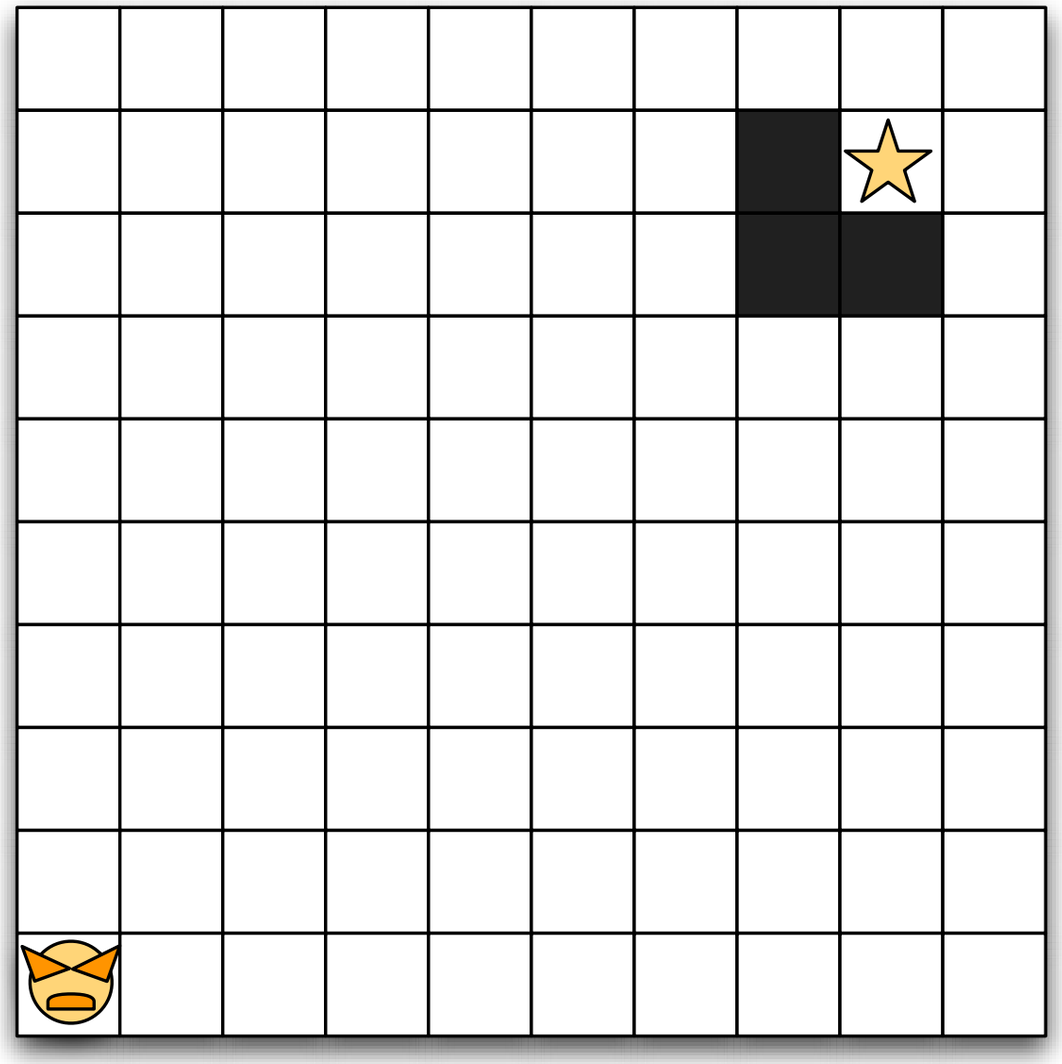
# A* example



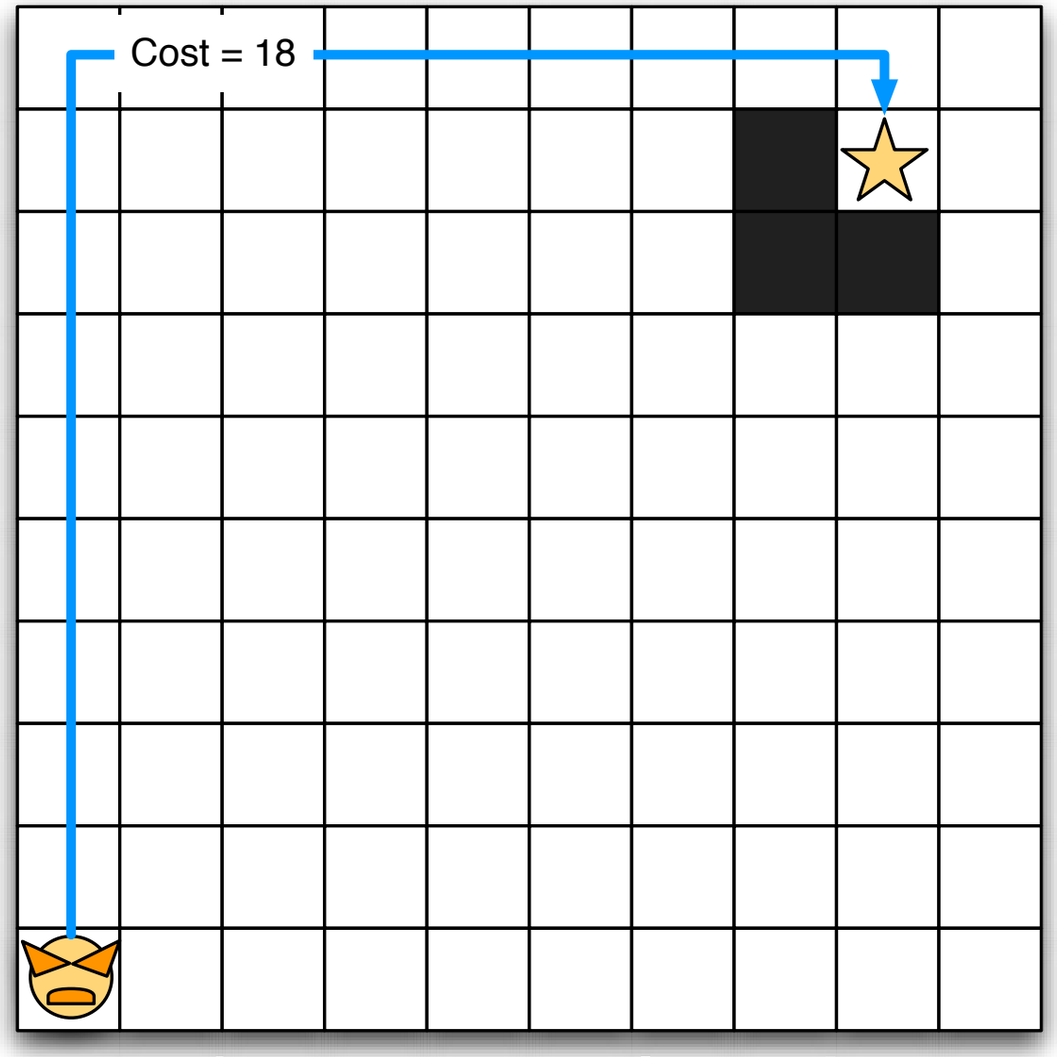
# The problem with A*

- Time complexity (nodes expanded)
- Memory overhead (stuff to remember)
- Good heuristics not enough.

# A* Example

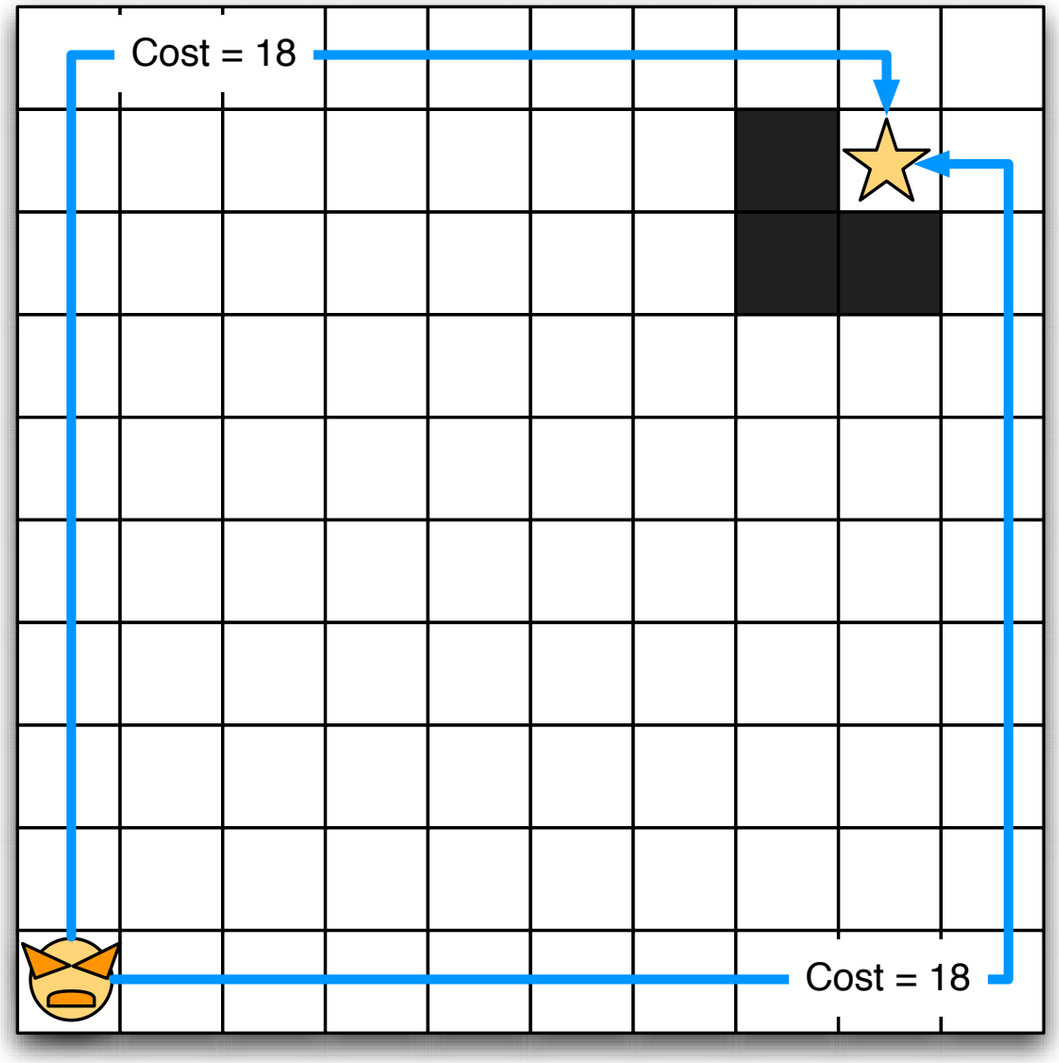


# A* Example



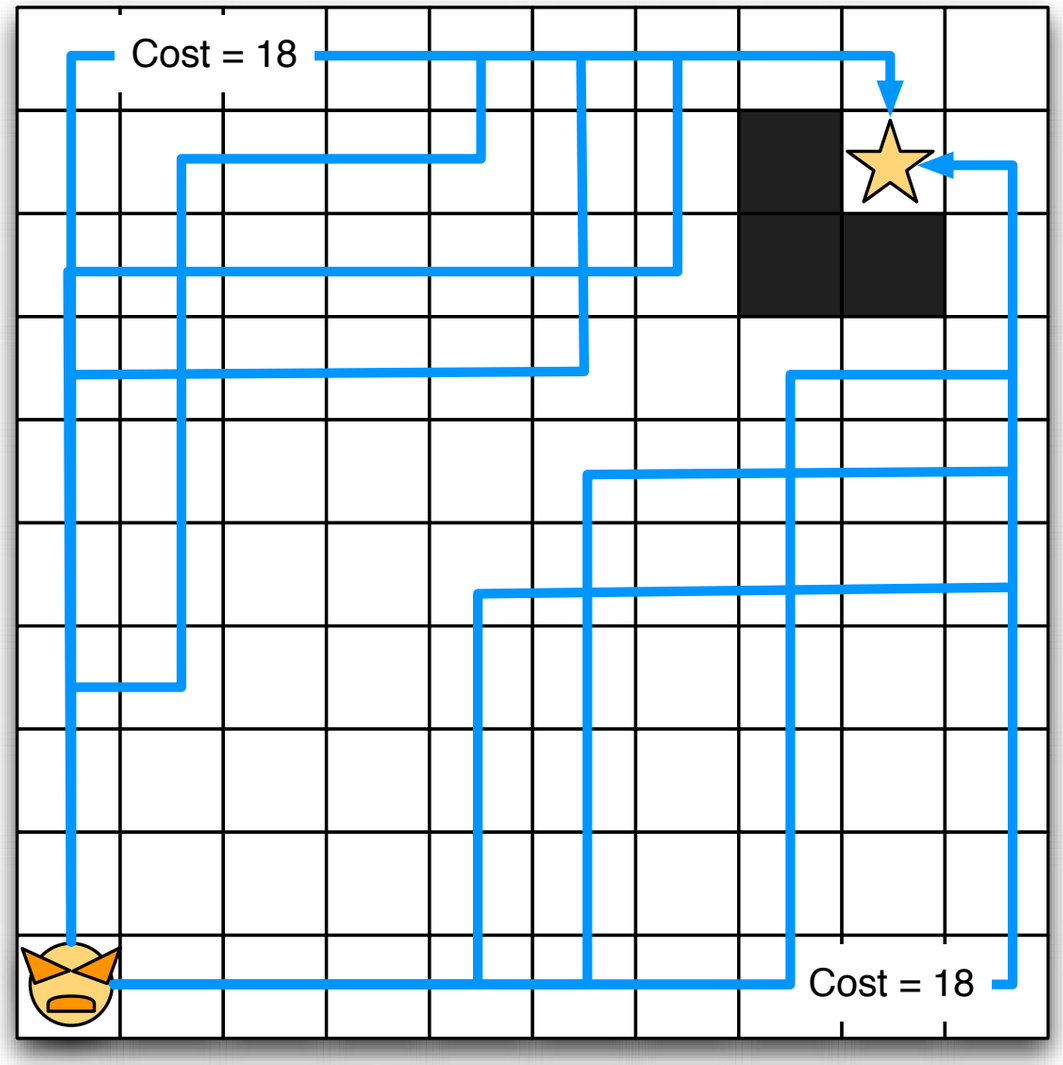

An optimal path
(straight-line movement only)

# A* Example




Another optimal path

# A* Example




Lots of optimal paths
(and many more exist!)

# A* Example

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | |
| 16 | 16 | 16 | 16 | 16 | 16 | 16 | | ★ | 18 |
| 16 | | | | | | 16 | | | 18 |
| 16 | | | | | | 16 | 16 | 16 | 18 |
| 16 | | | | | | | | 16 | 18 |
| 16 | | | | | | | | 16 | 18 |
| 16 | | | | | | | | 16 | 18 |
| 16 | | | | | | | | 16 | 18 |
| 16 | | | | | | | | 16 | 18 |
| | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 18 |

A* expands: all green tiles + some white

# Part 1:
# Hierarchical pathfinding

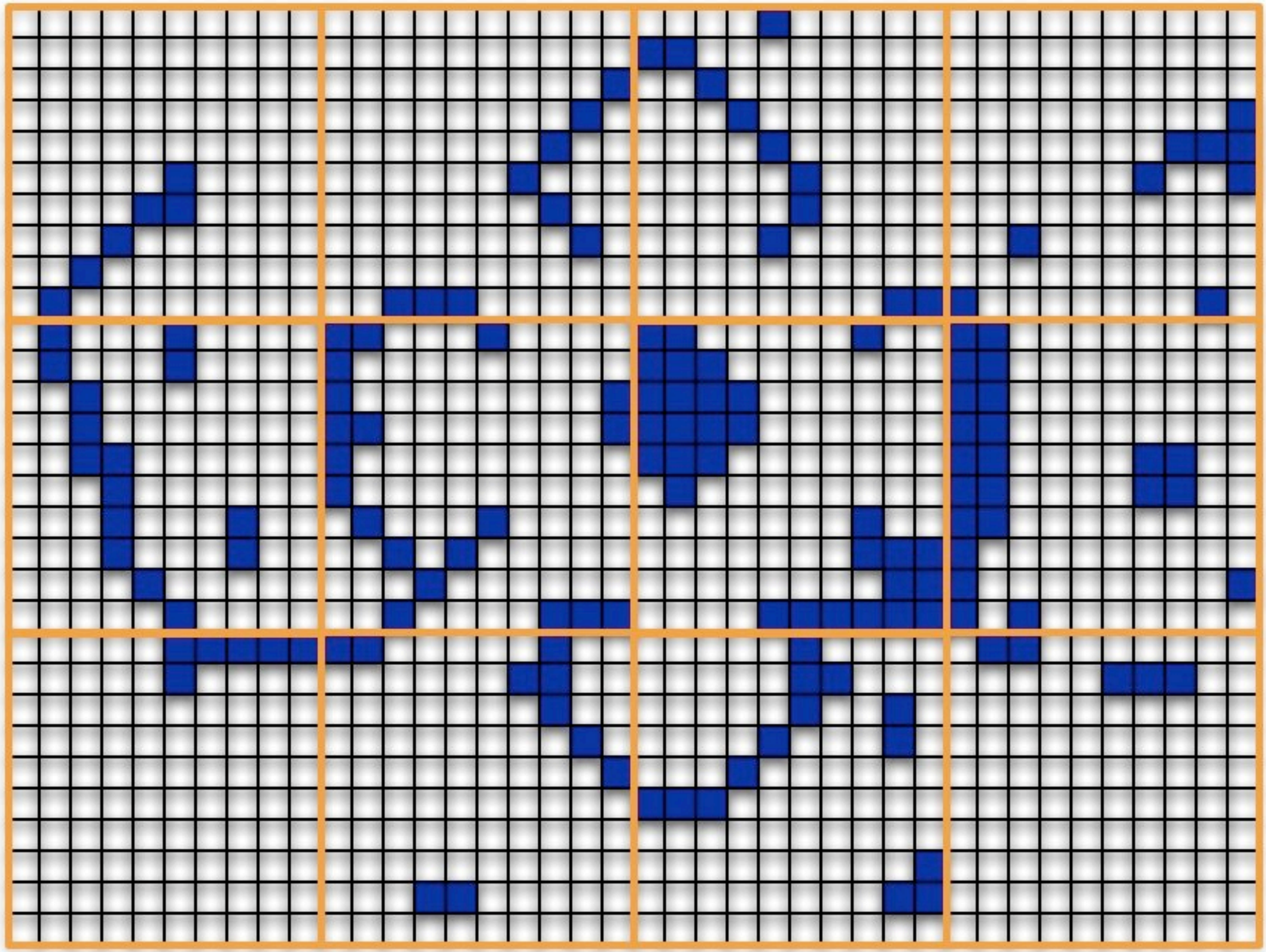
# Hierarchical pathfinding

- Build a smaller, approximate search space.
- Area-based vs. Tile-based pathfinding.
- Factored problems are easy!
- Faster + lower memory requirements.

# HPA* Overview

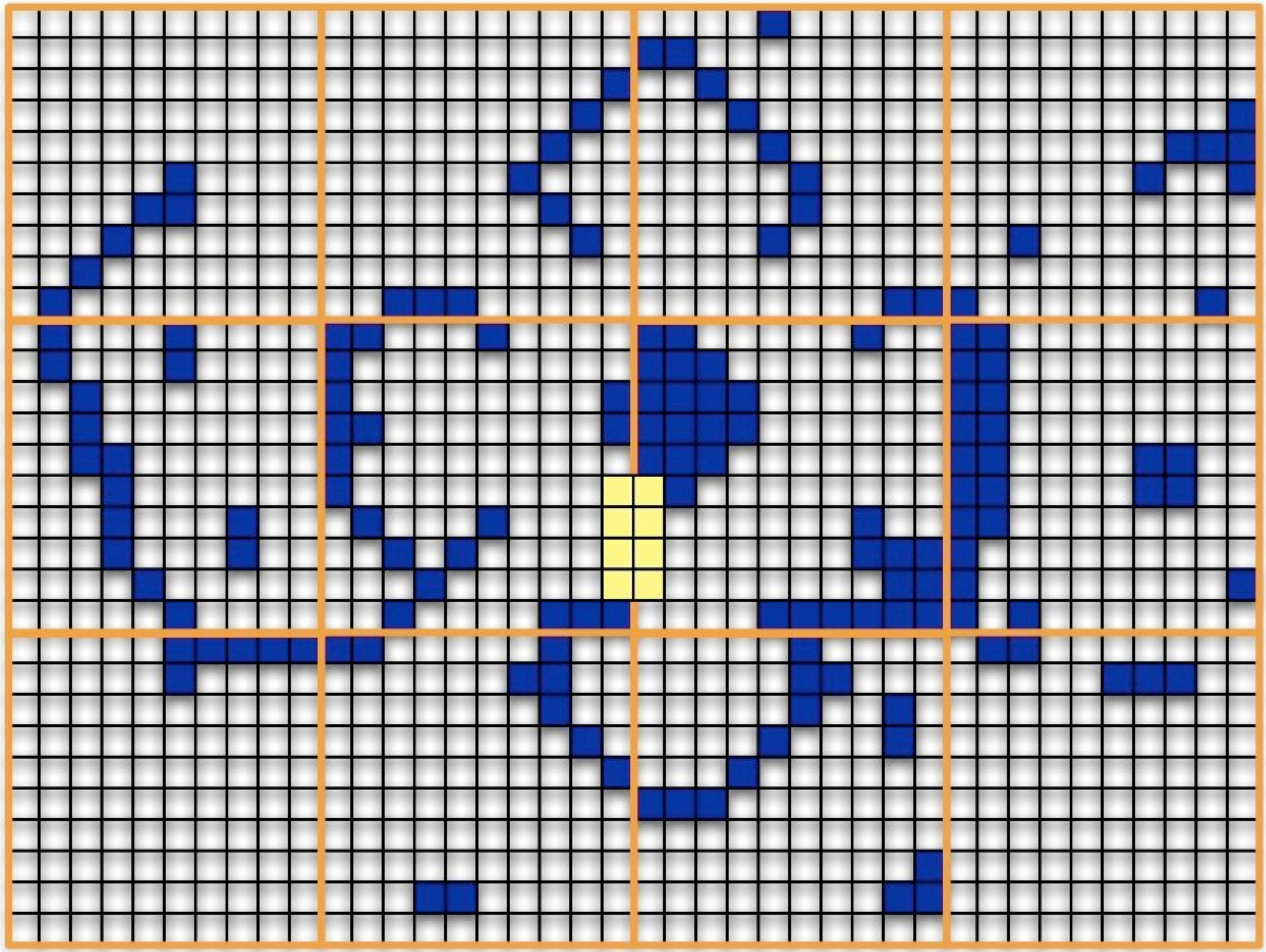
- [Botea, Müller & Schaeffer, 2004]
- Pre-processing:
  - Build abstract graph (Clusters and Entrances)
- Run-time processing:
  - Insert start and goal into abstract graph
  - Find abstract solution
  - Refine if necessary (or use cached info)

Map: Temple Prime (Red Alert 3)



Split map into square clusters (e.g. 10x10)

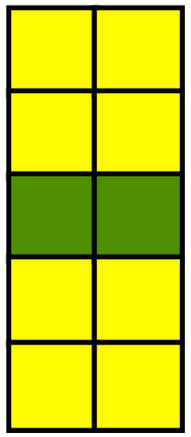
Image: Adi Botea



# Identify entrances between adjacent clusters

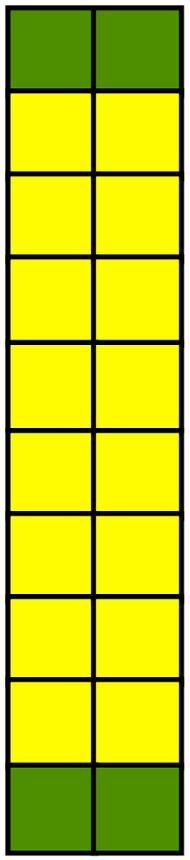
Image: Adi Botea

Small entrance (length < 6)



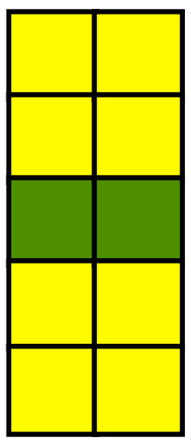
1 central transition point

Large entrance (length >= 6)

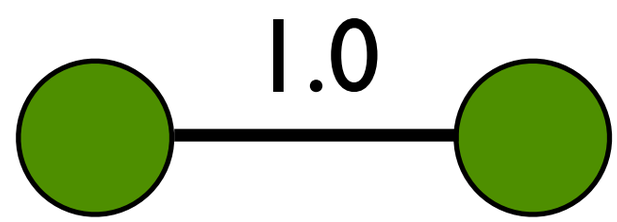


2 perimeter transition points

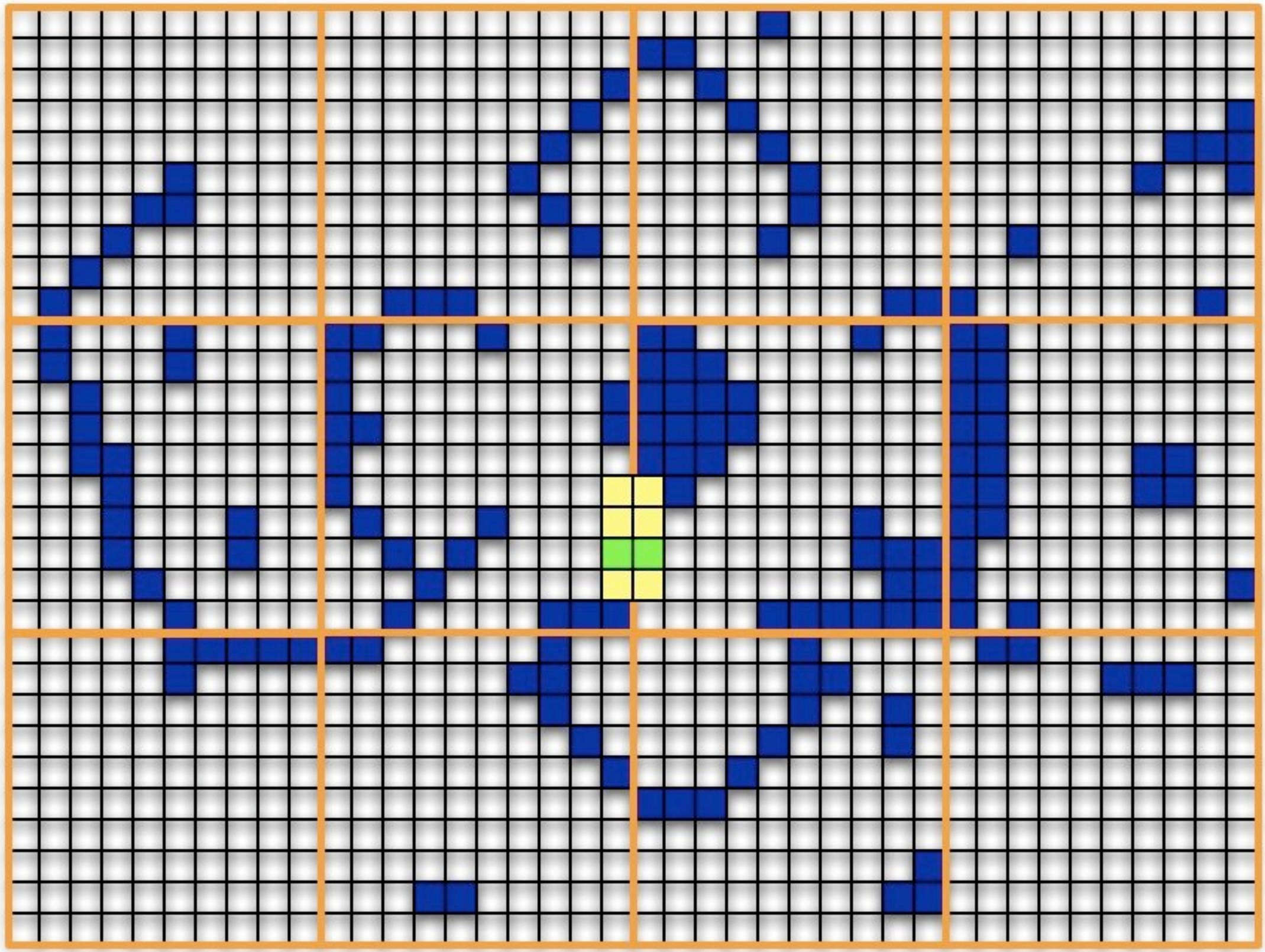
Transition point =
2 abstract nodes + 1 abstract (inter) edge



Entrance

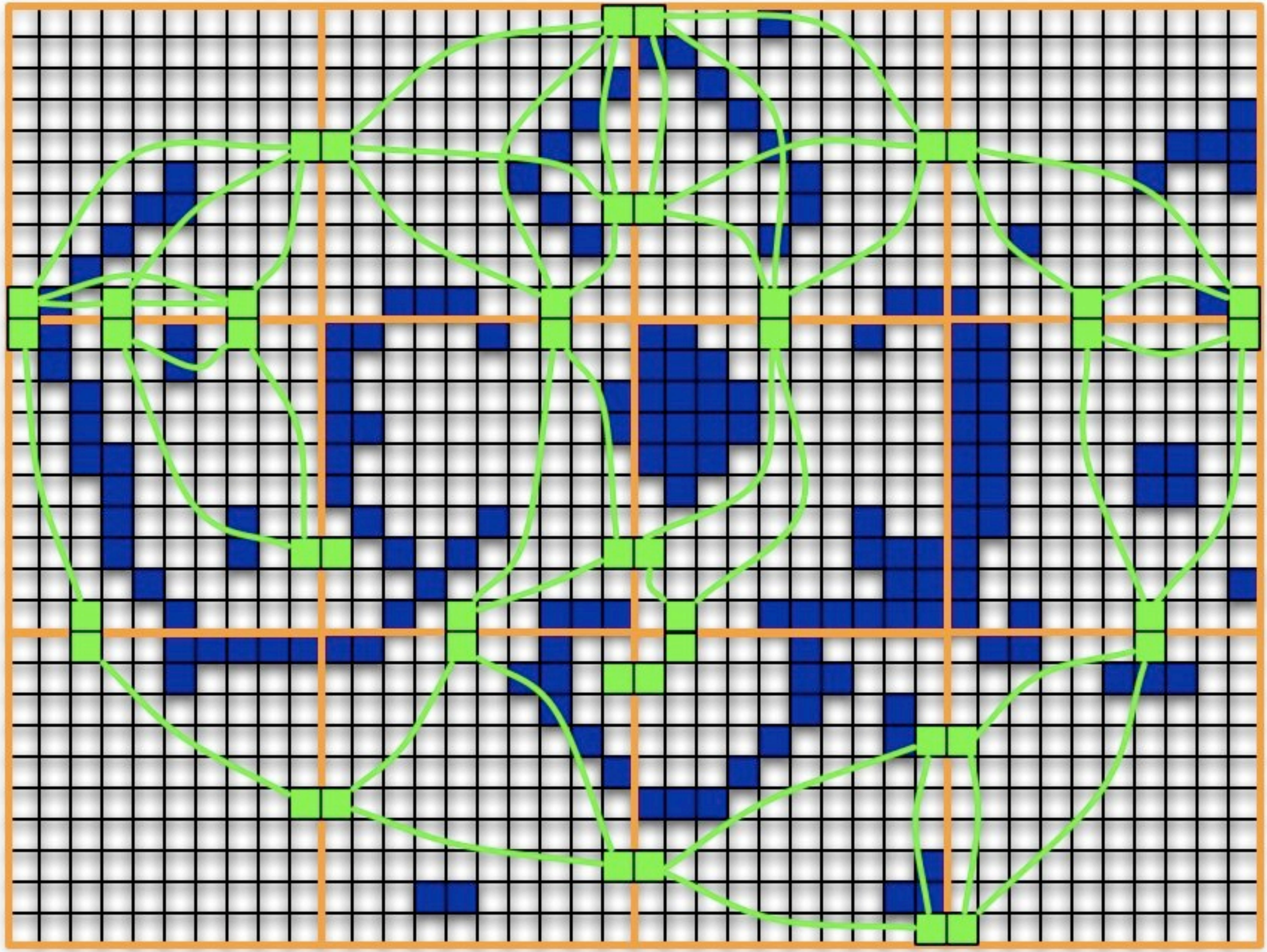



Abstract graph



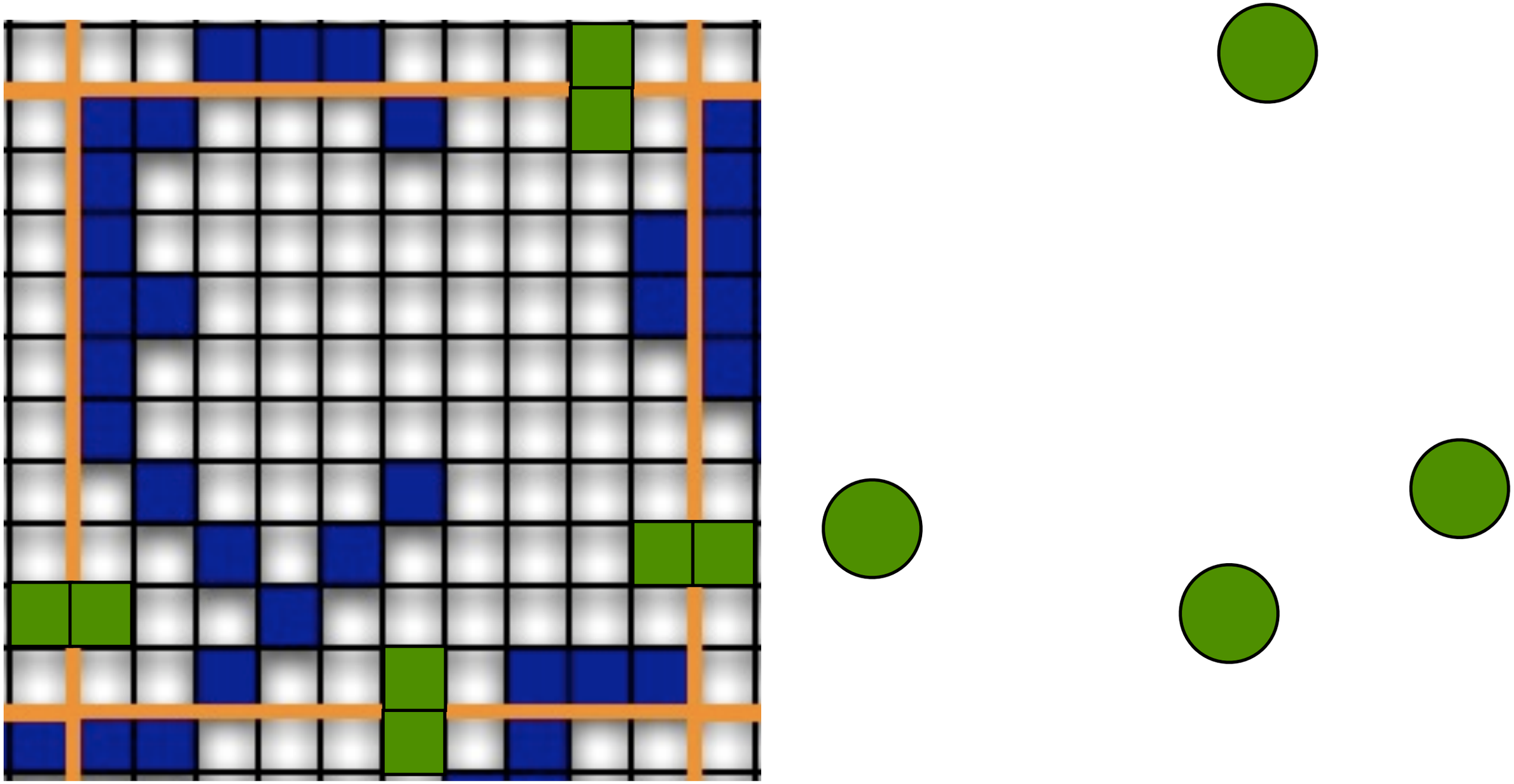
# Select transition points for each entrance

Image: Adi Botea

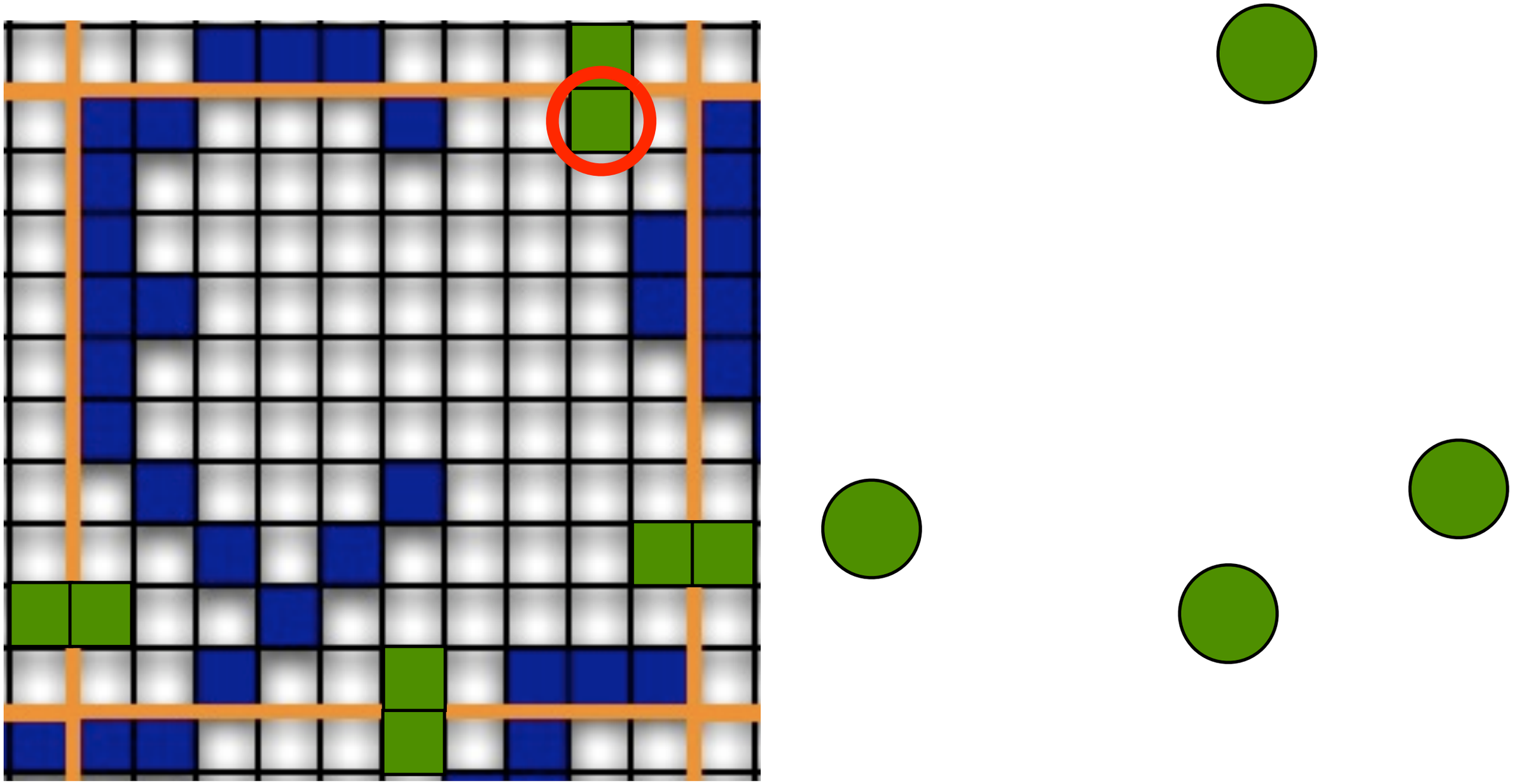


# Connect abstract nodes inside the same cluster

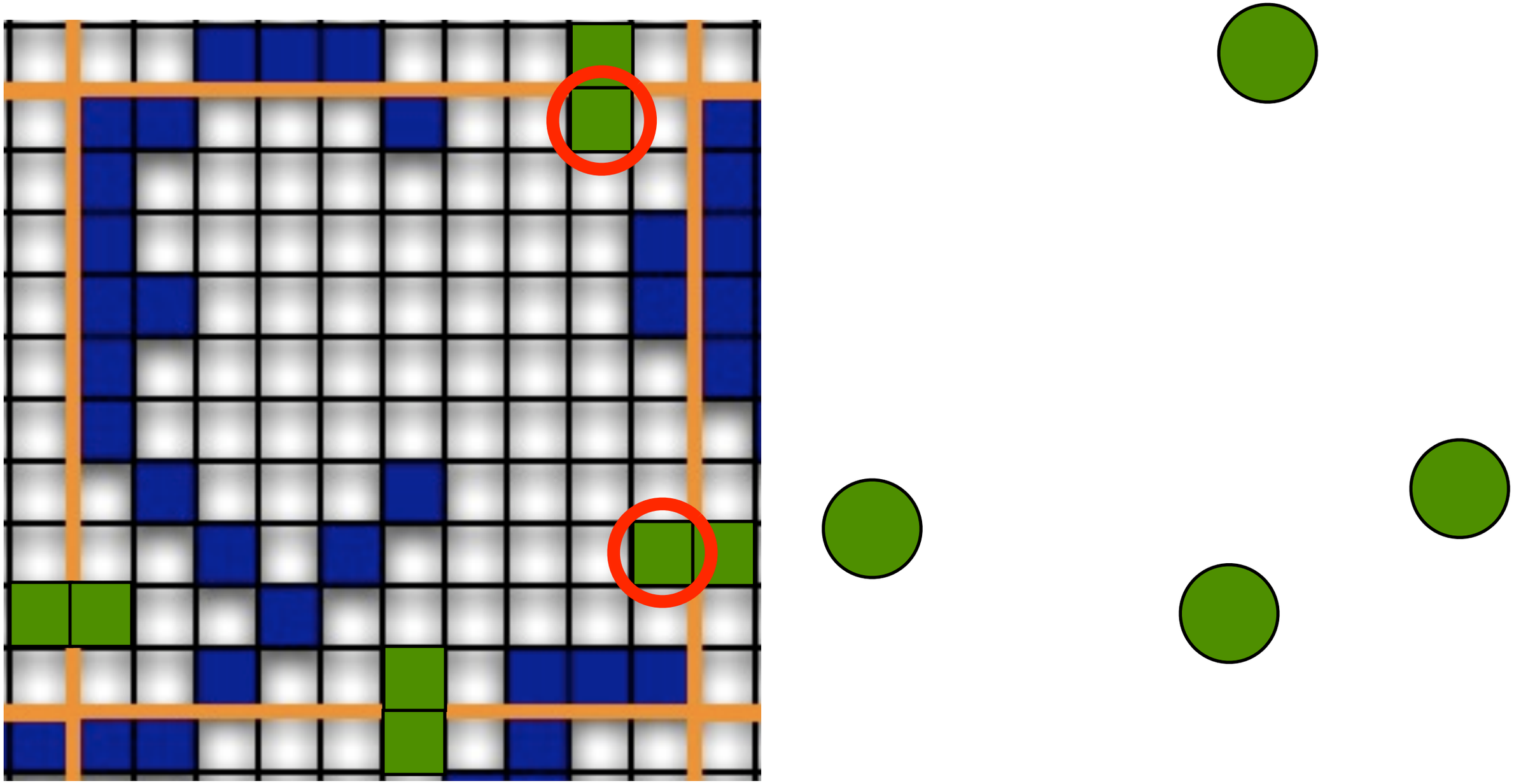
Image: Adi Botea



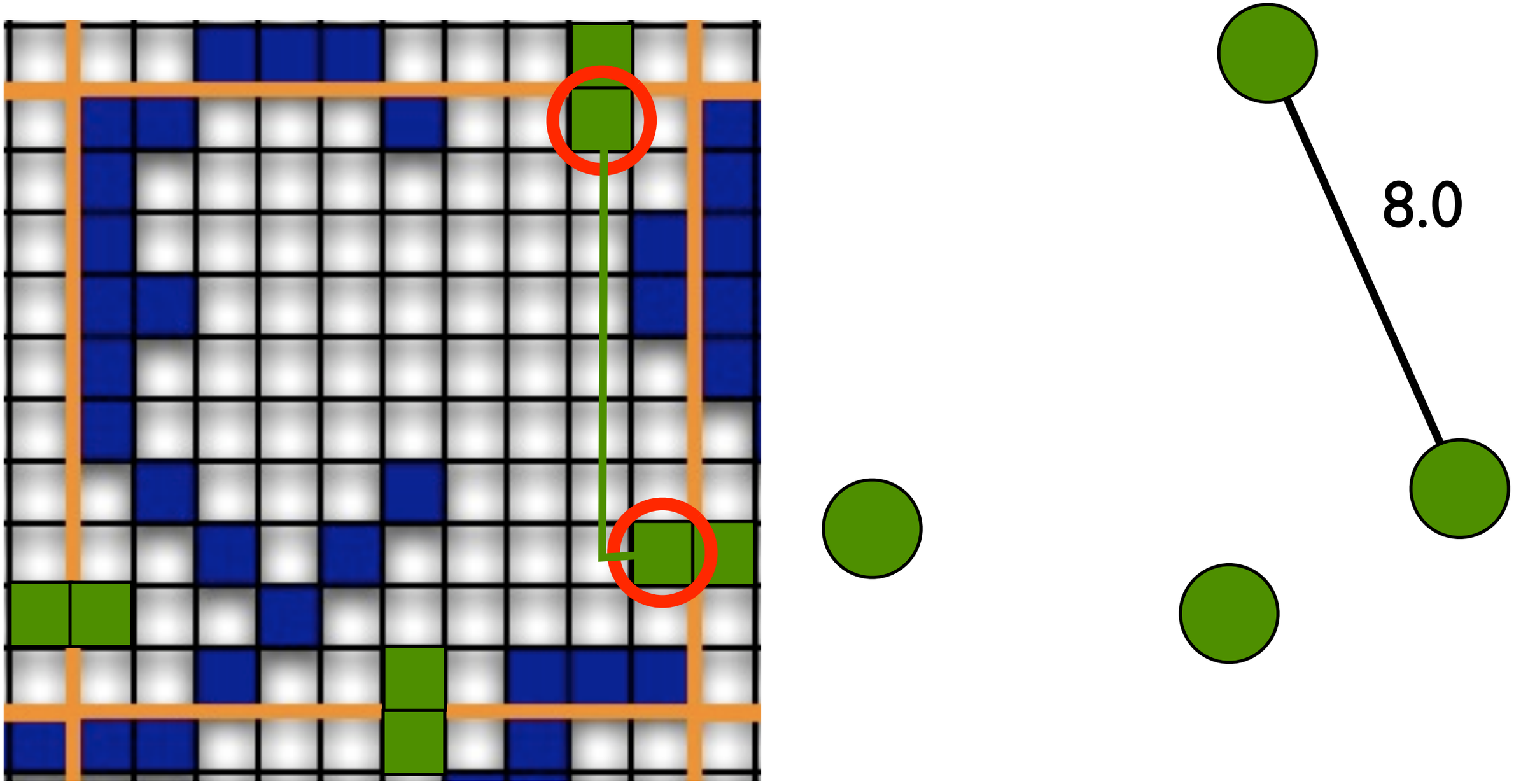
Intra-edges (cost = path length) added to abstract graph



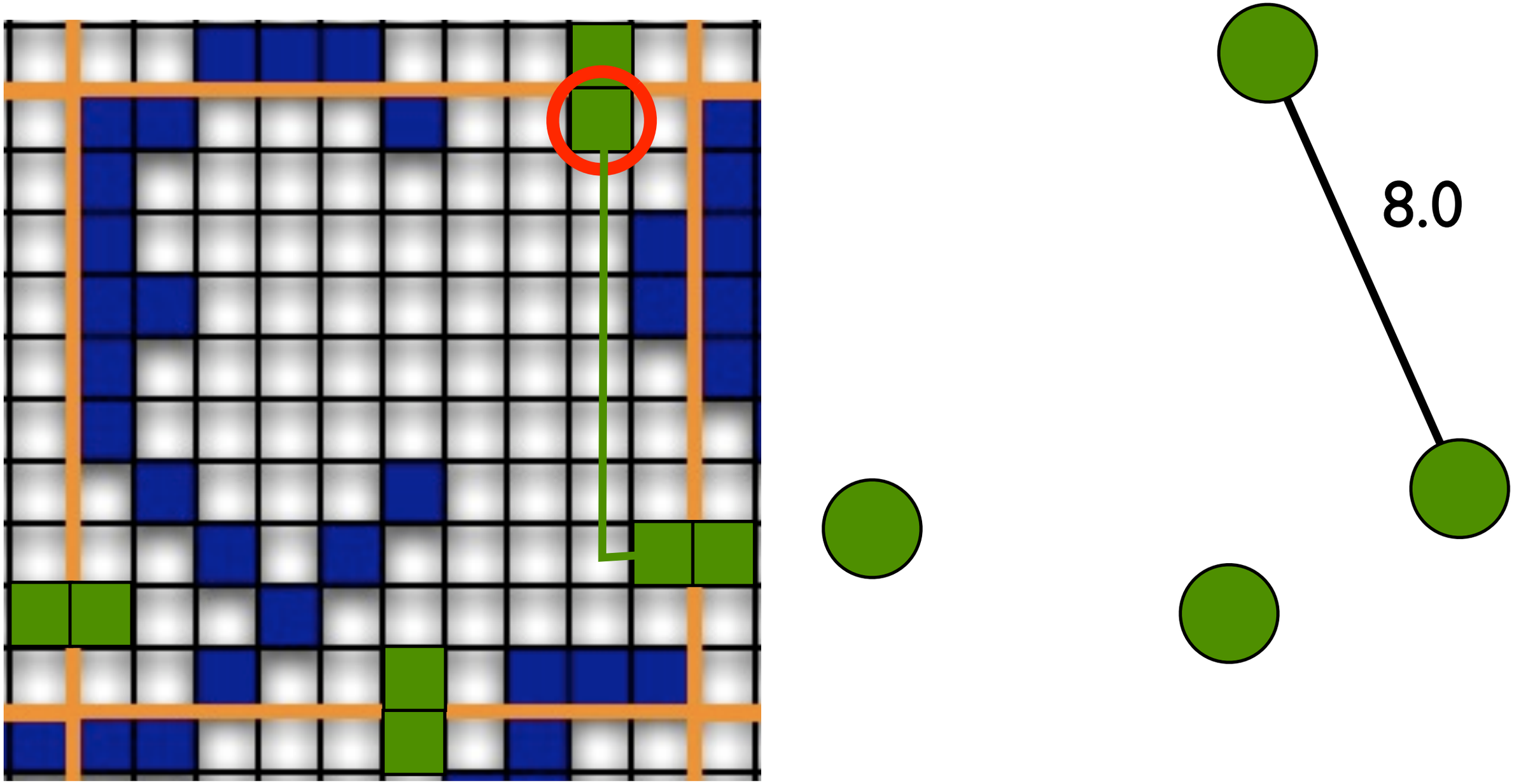
Intra-edges (cost = path length) added to abstract graph



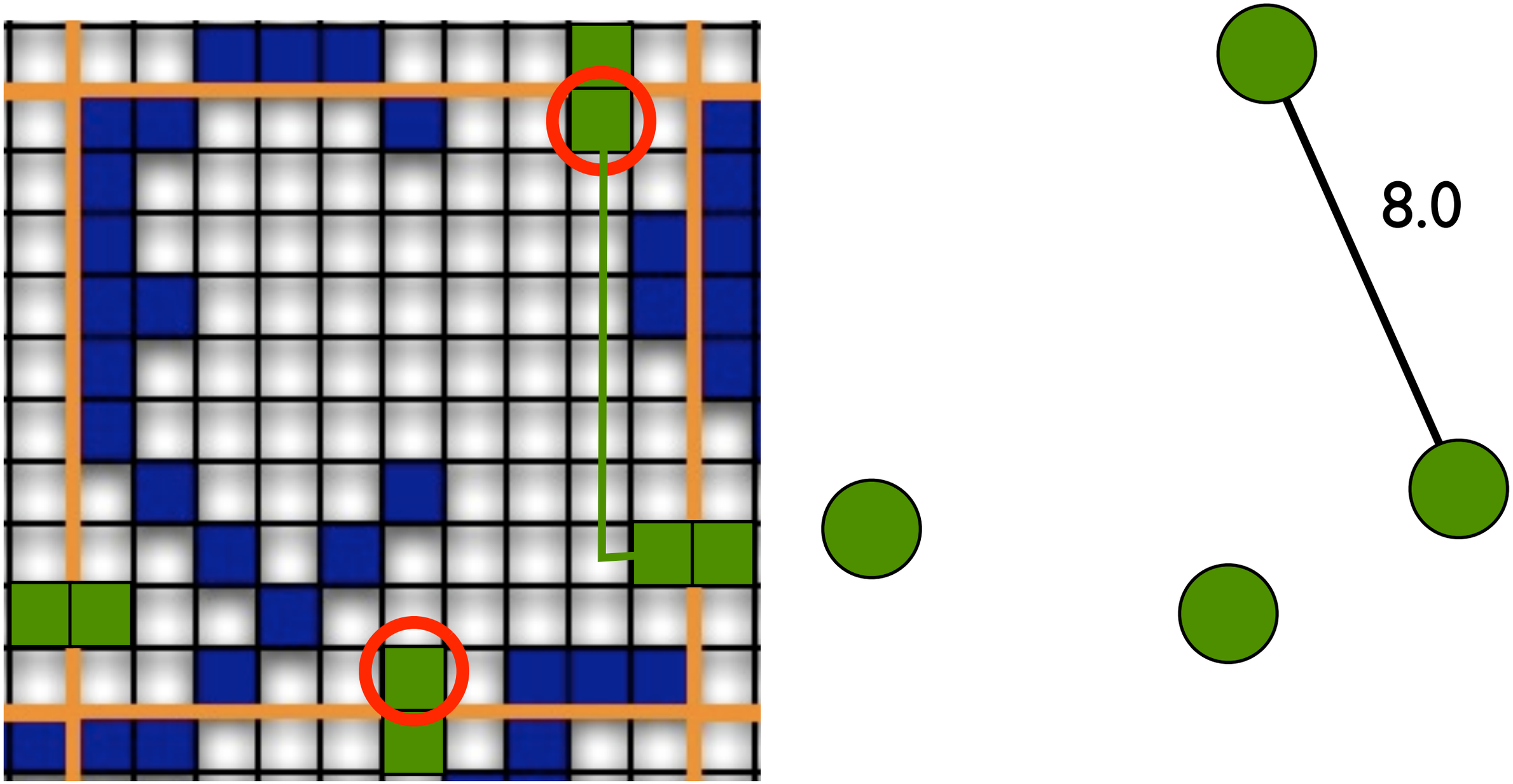
Intra-edges (cost = path length) added to abstract graph



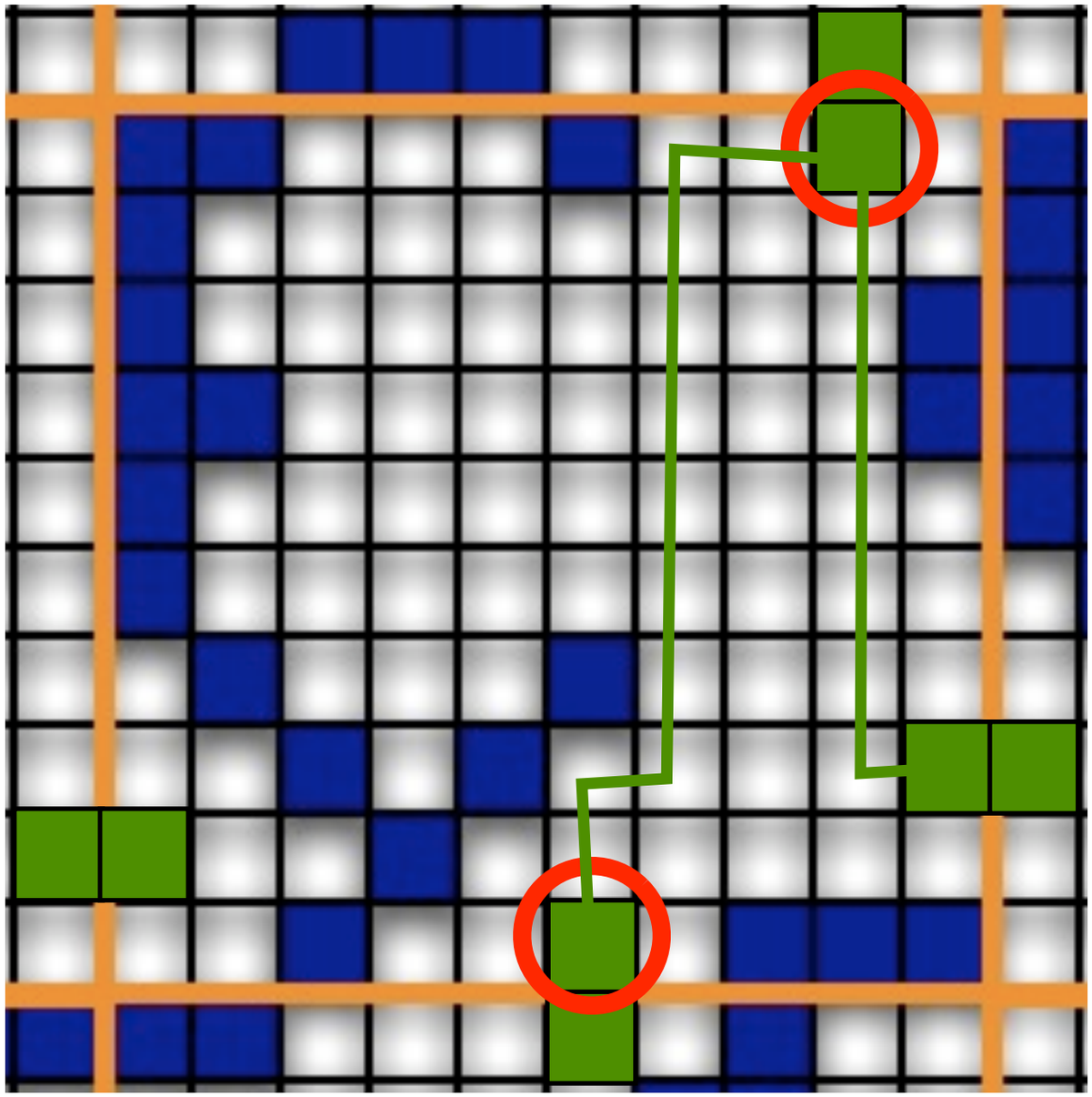

Intra-edges (cost = path length) added to abstract graph

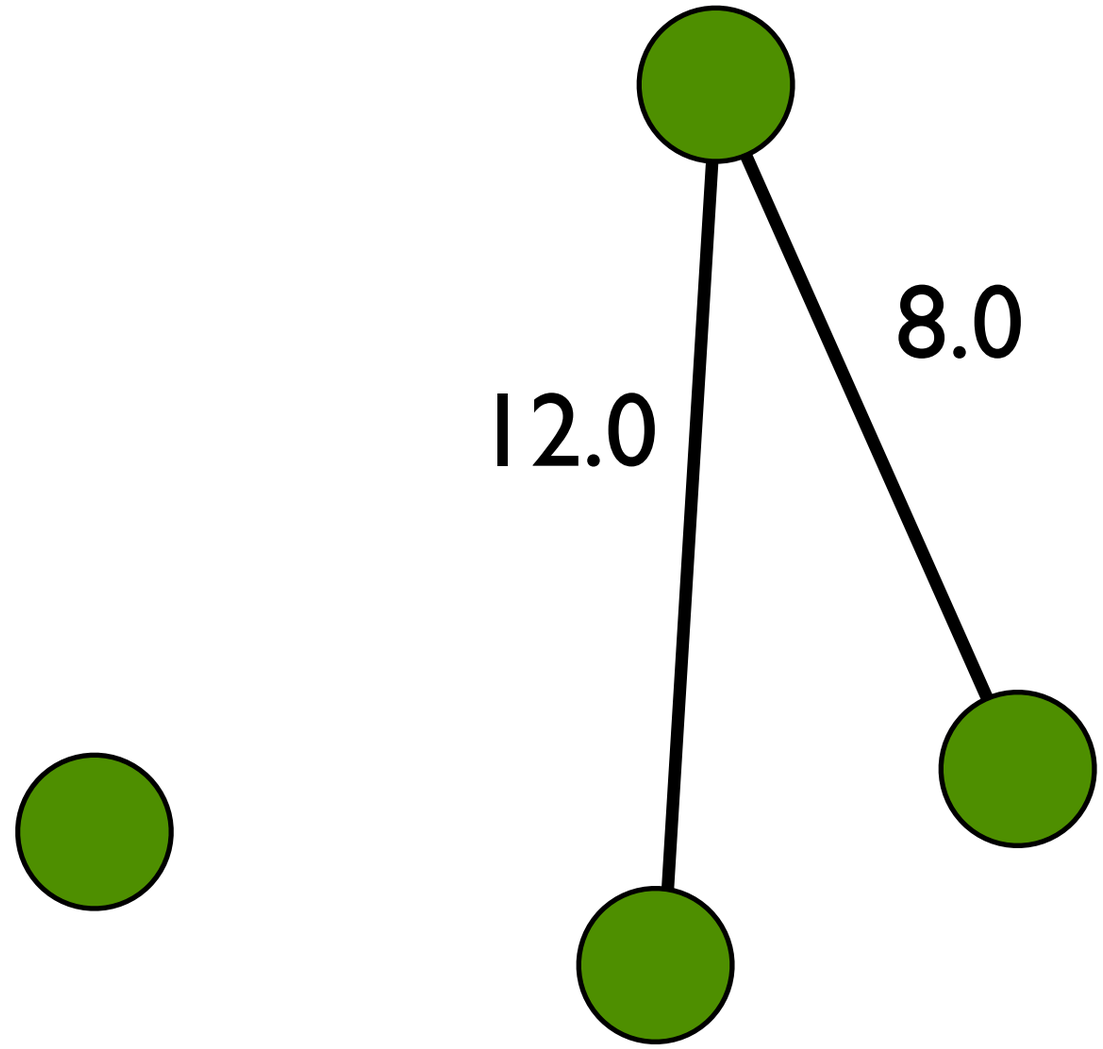



Intra-edges (cost = path length) added to abstract graph

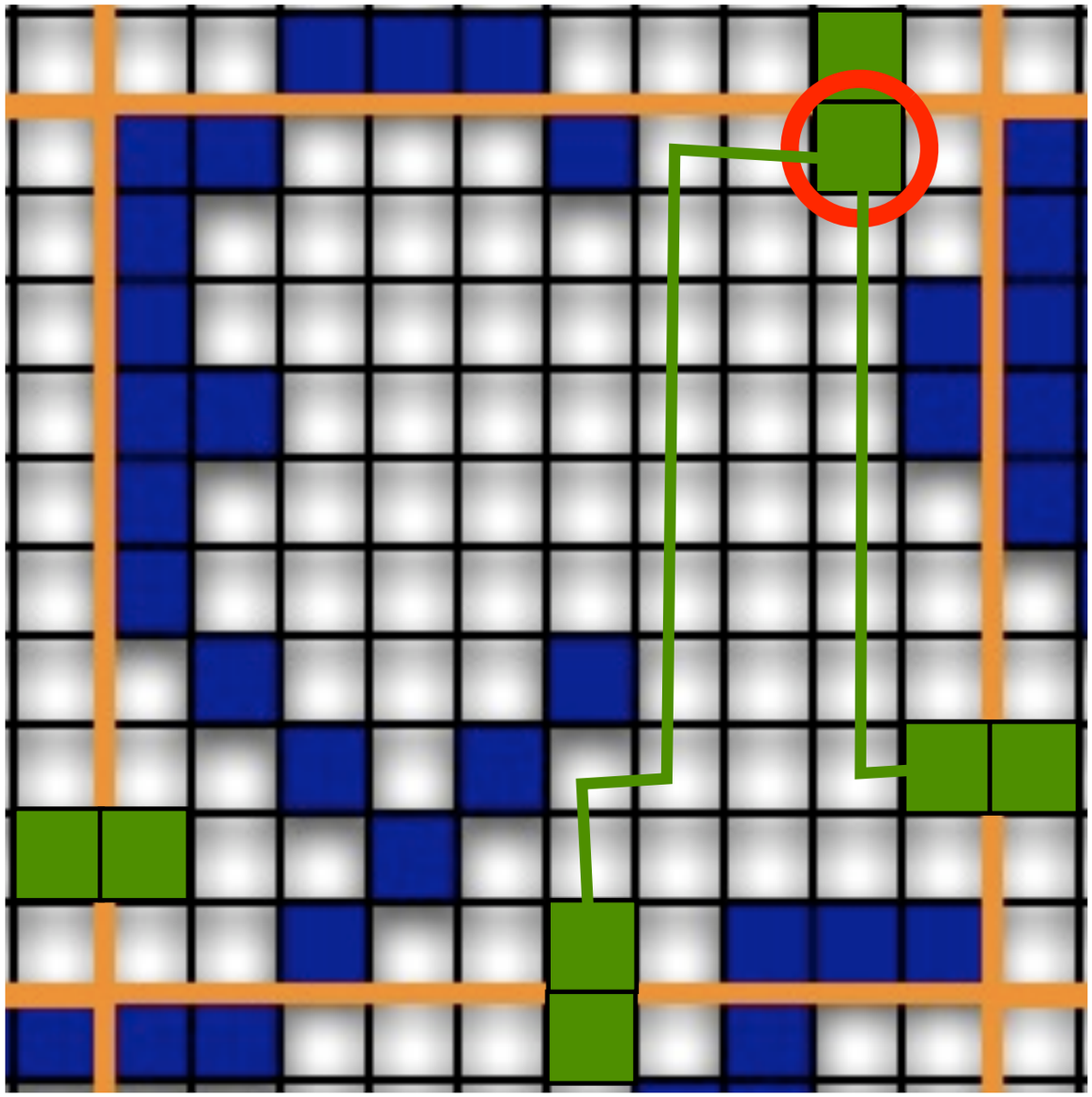


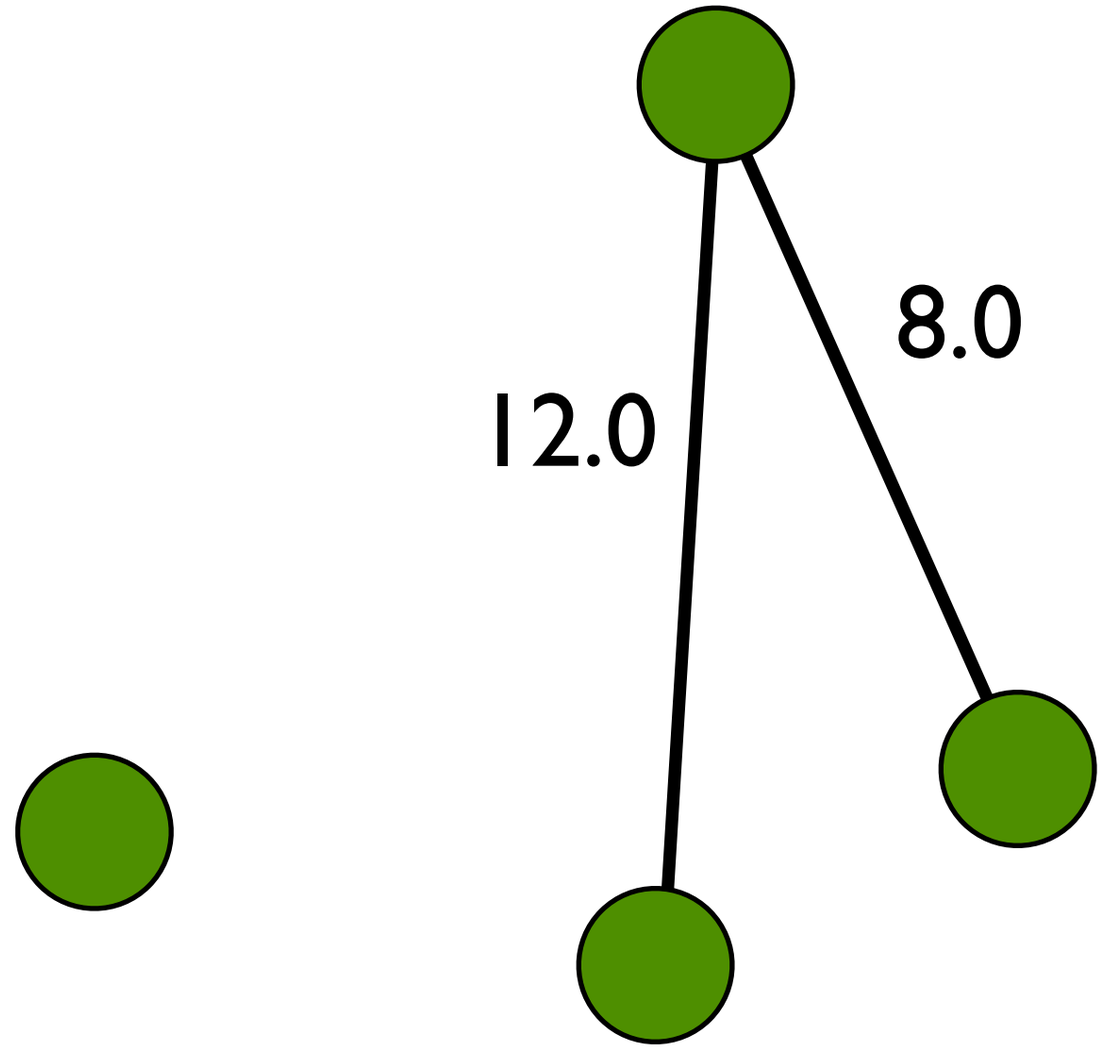

Intra-edges (cost = path length) added to abstract graph



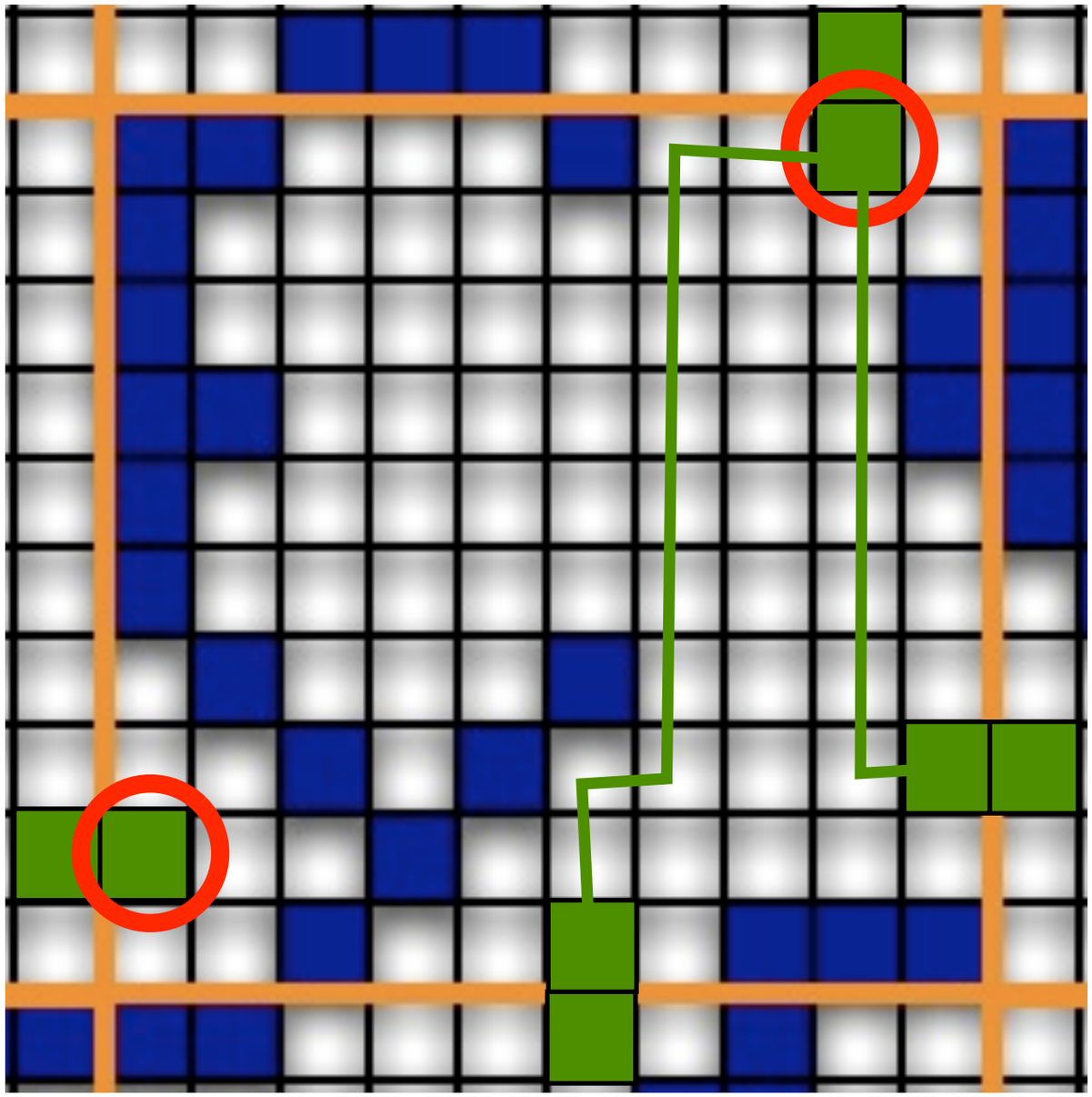


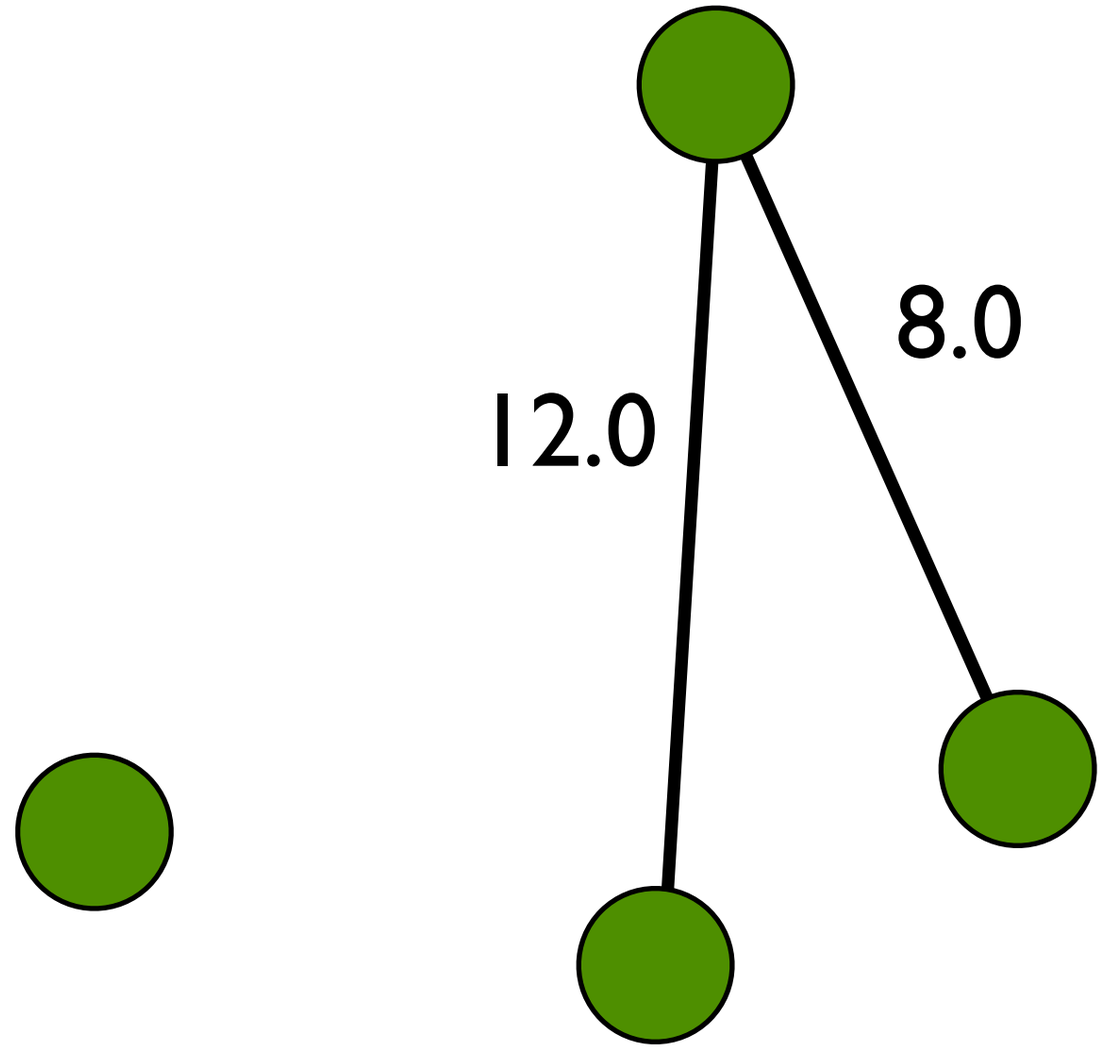

Intra-edges (cost = path length) added to abstract graph



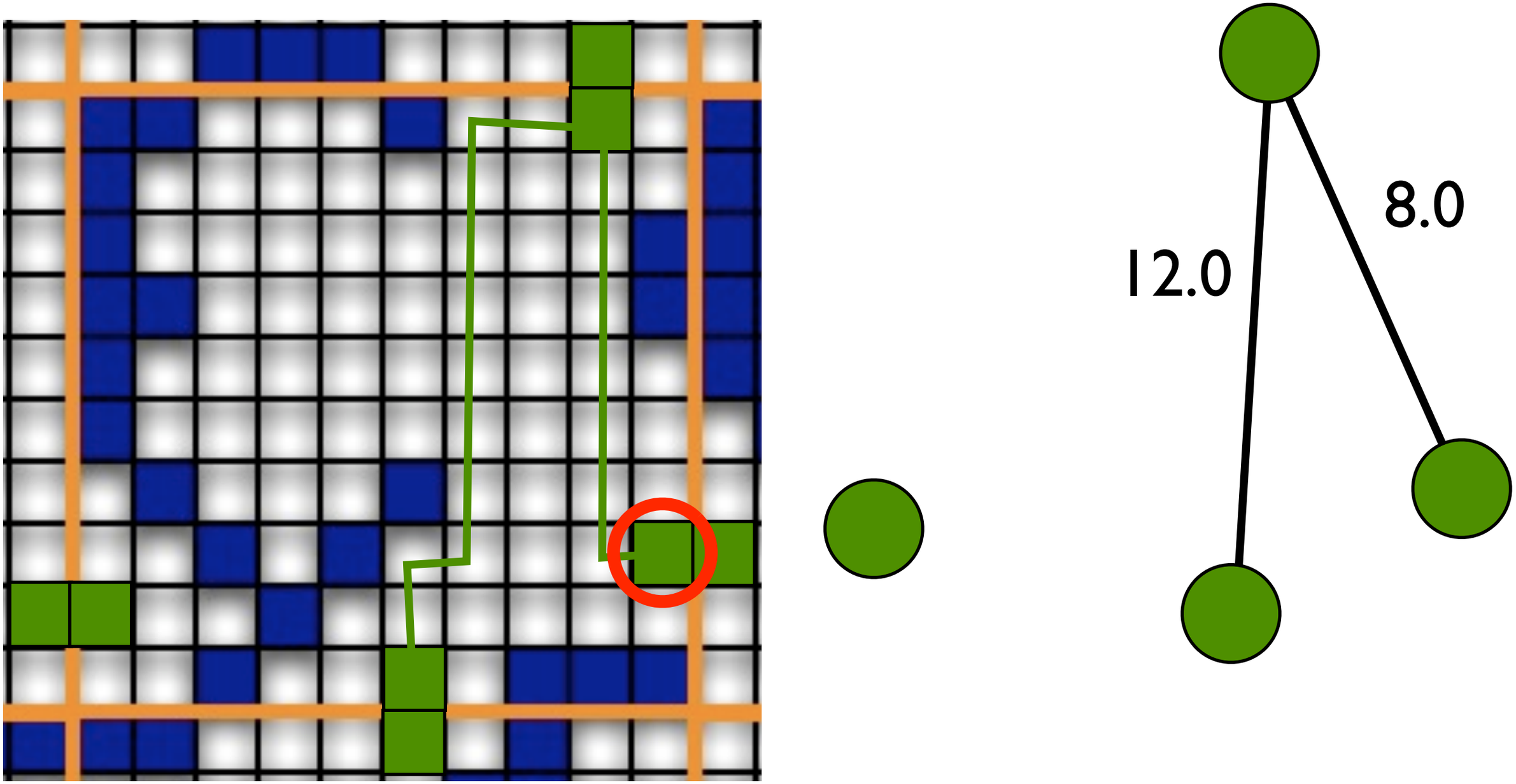



Intra-edges (cost = path length) added to abstract graph

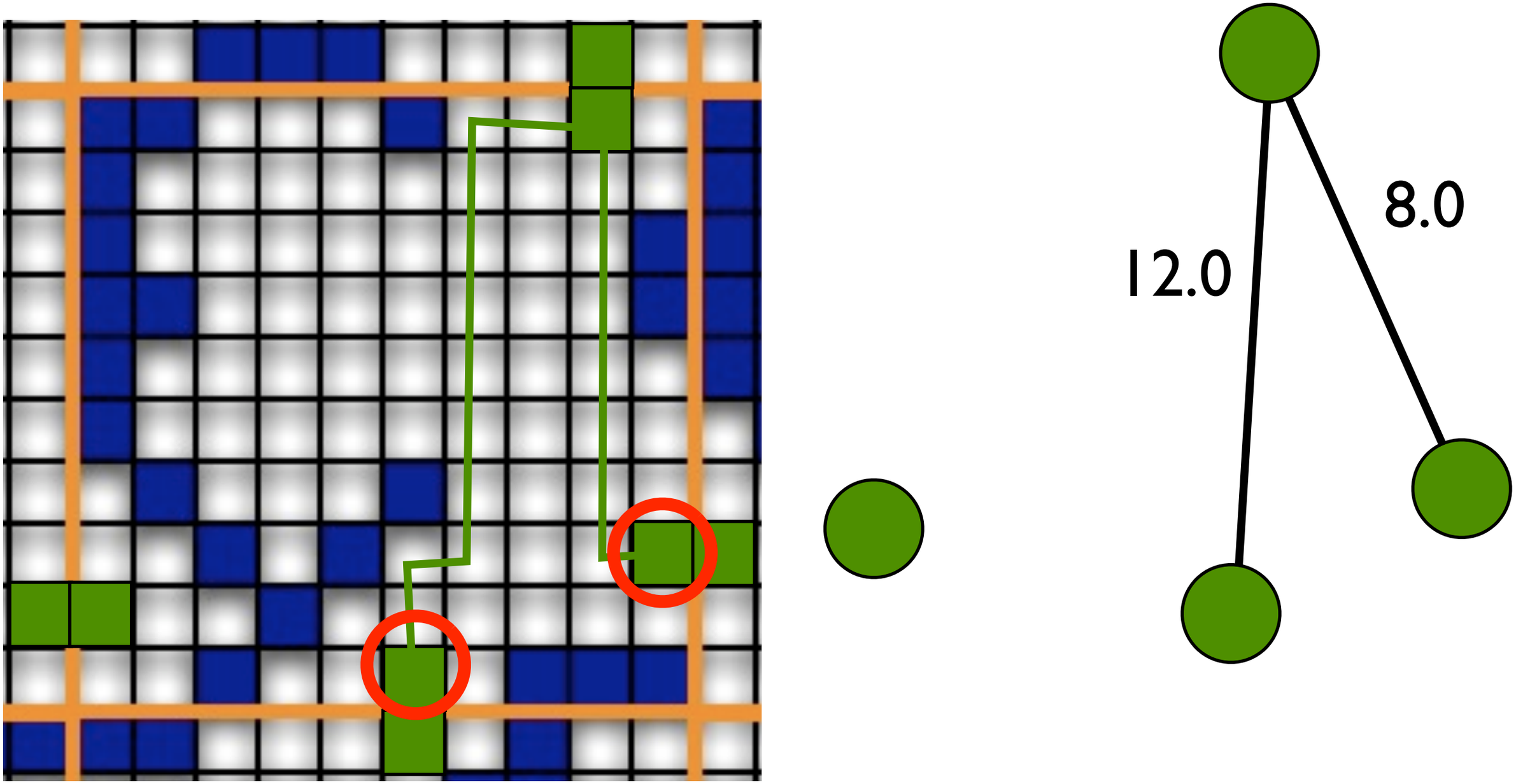




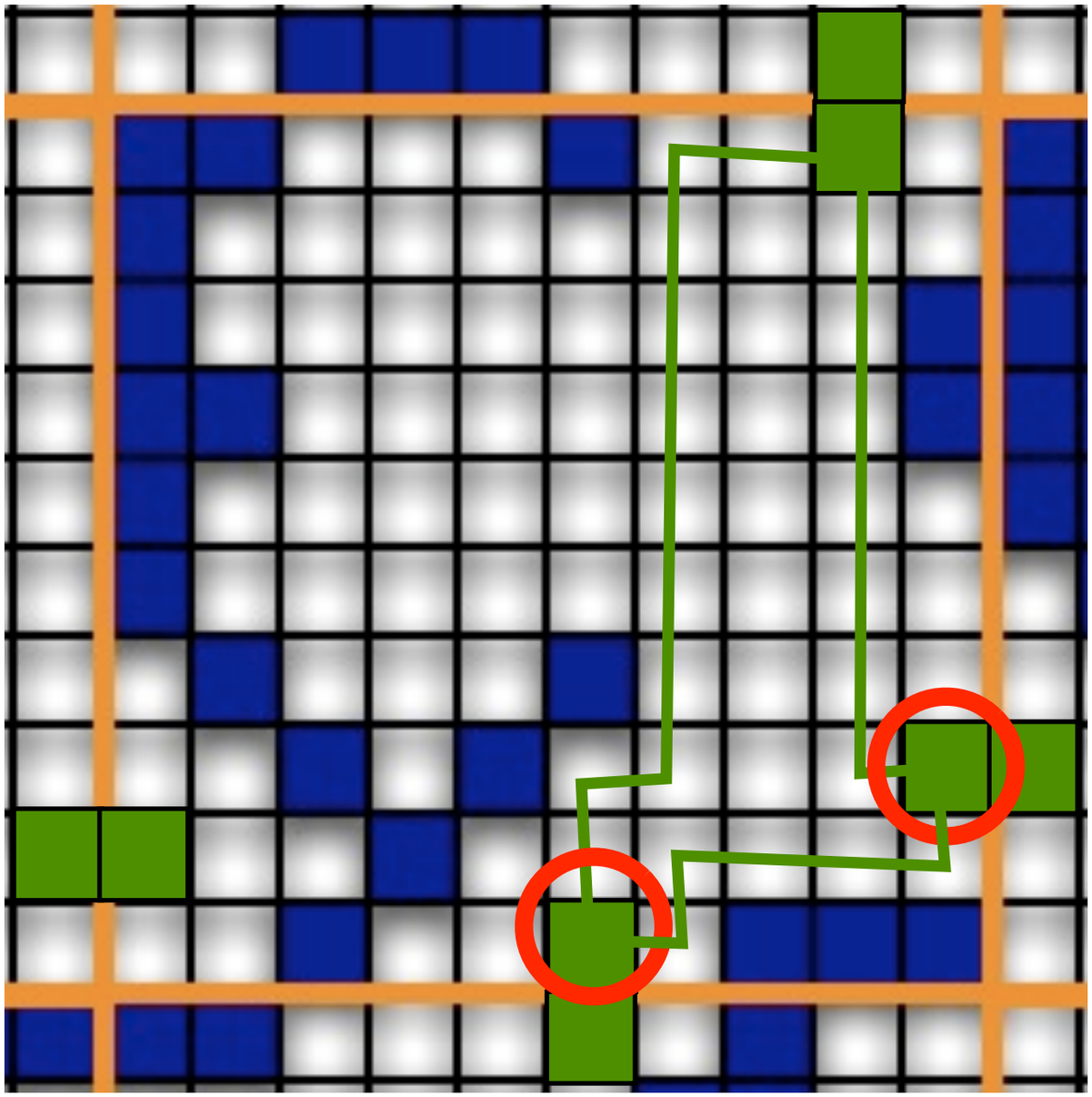

Intra-edges (cost = path length) added to abstract graph

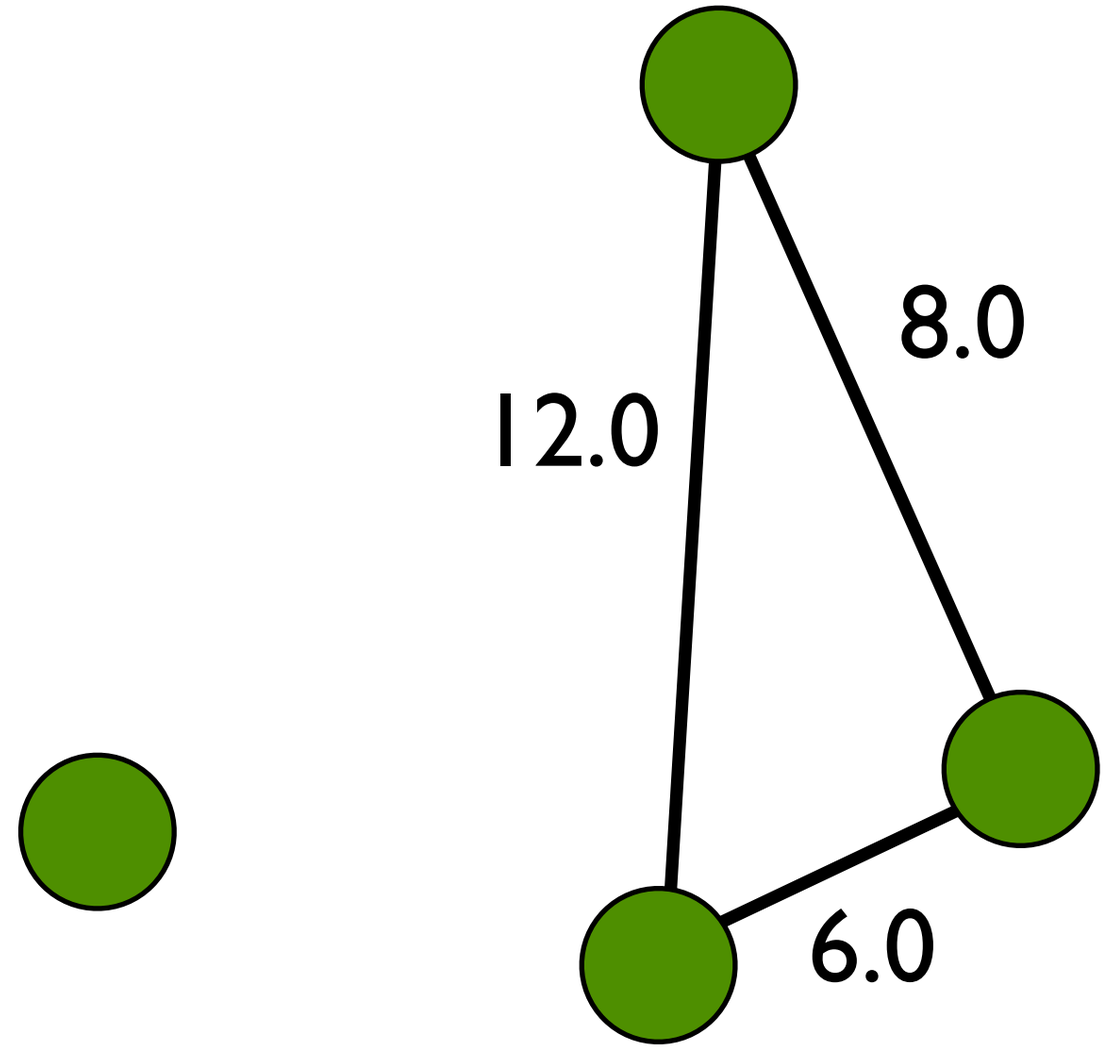



Insert intra-edges (of cost = path length) to abstract graph.

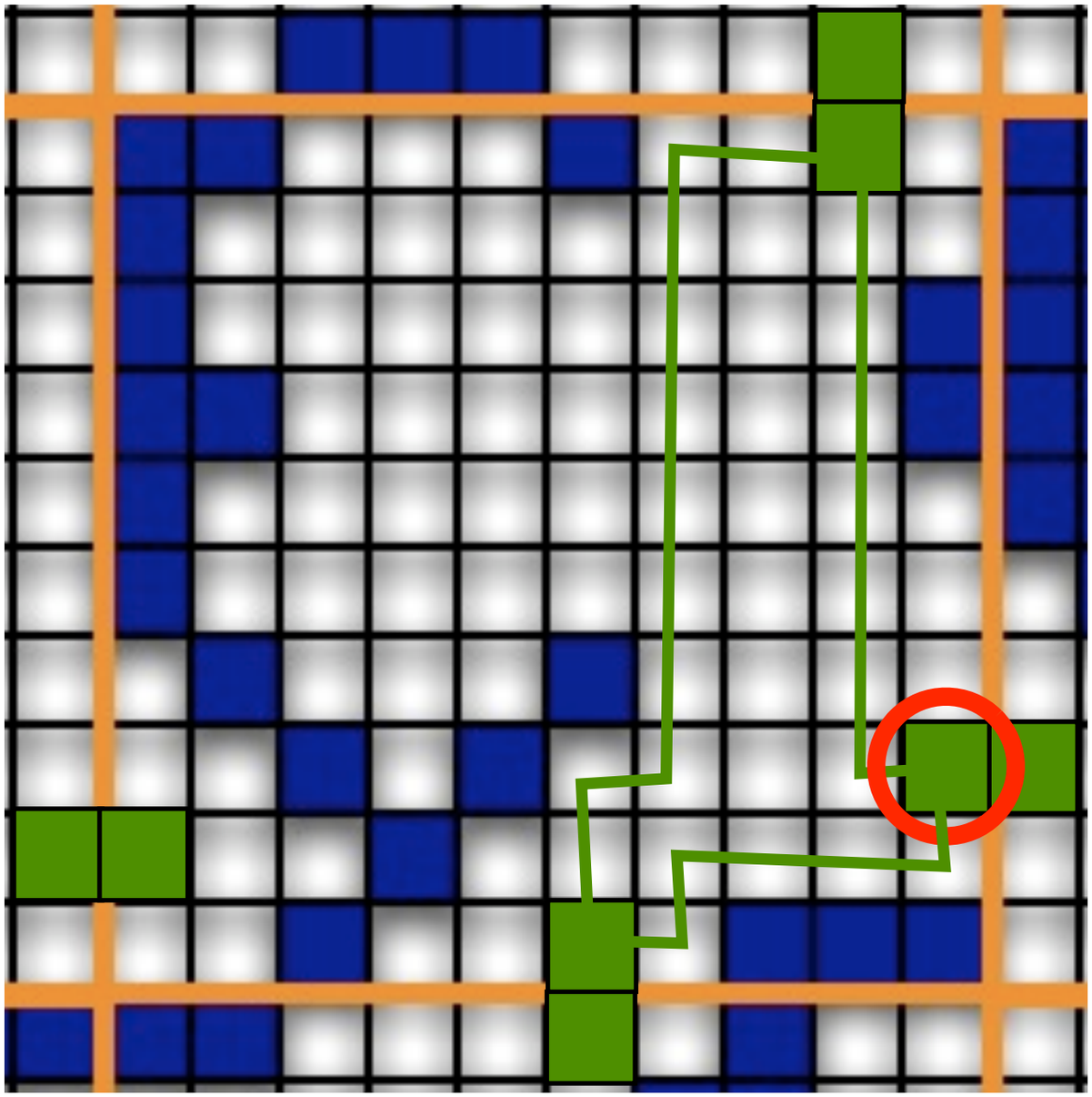


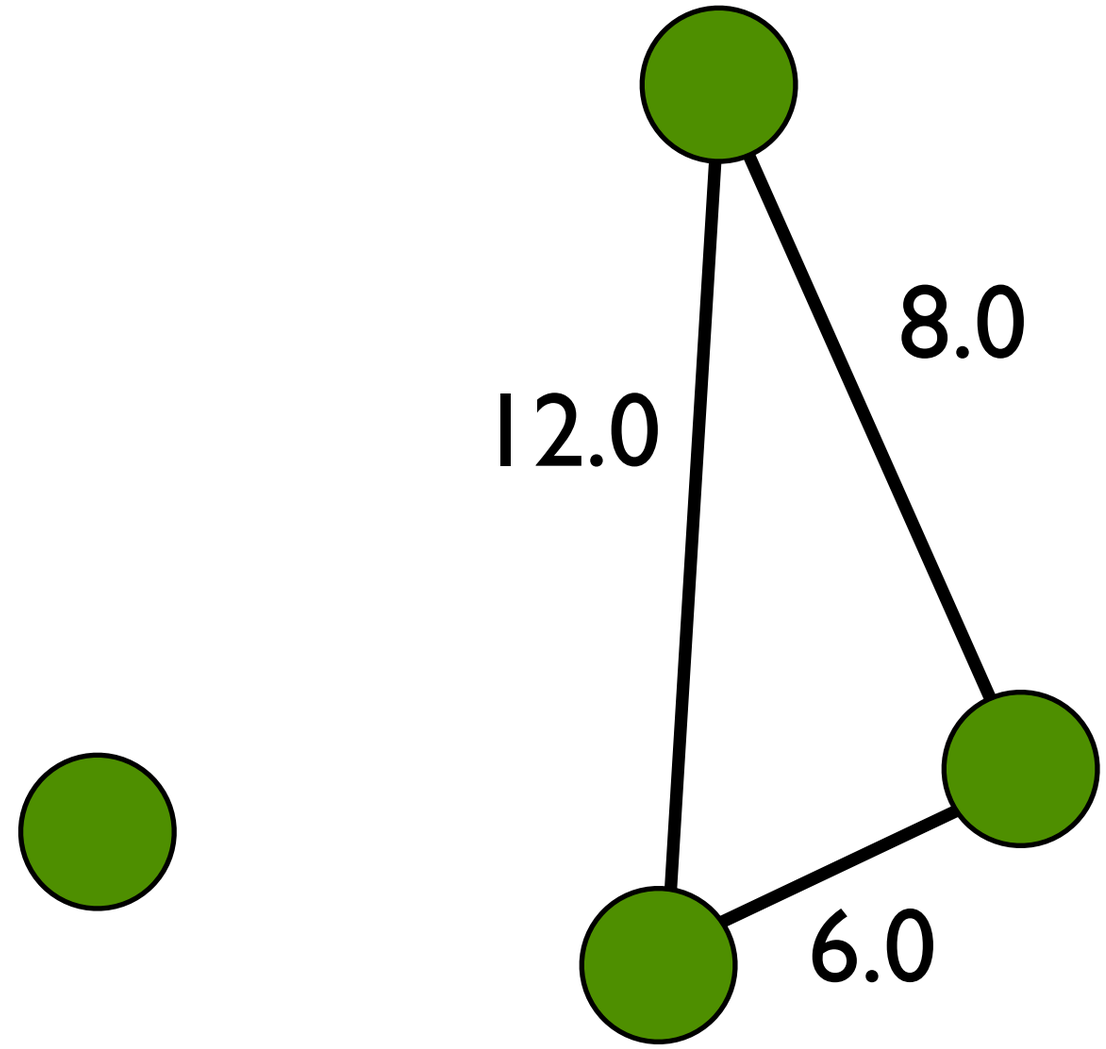

Insert intra-edges (of cost = path length) to abstract graph.



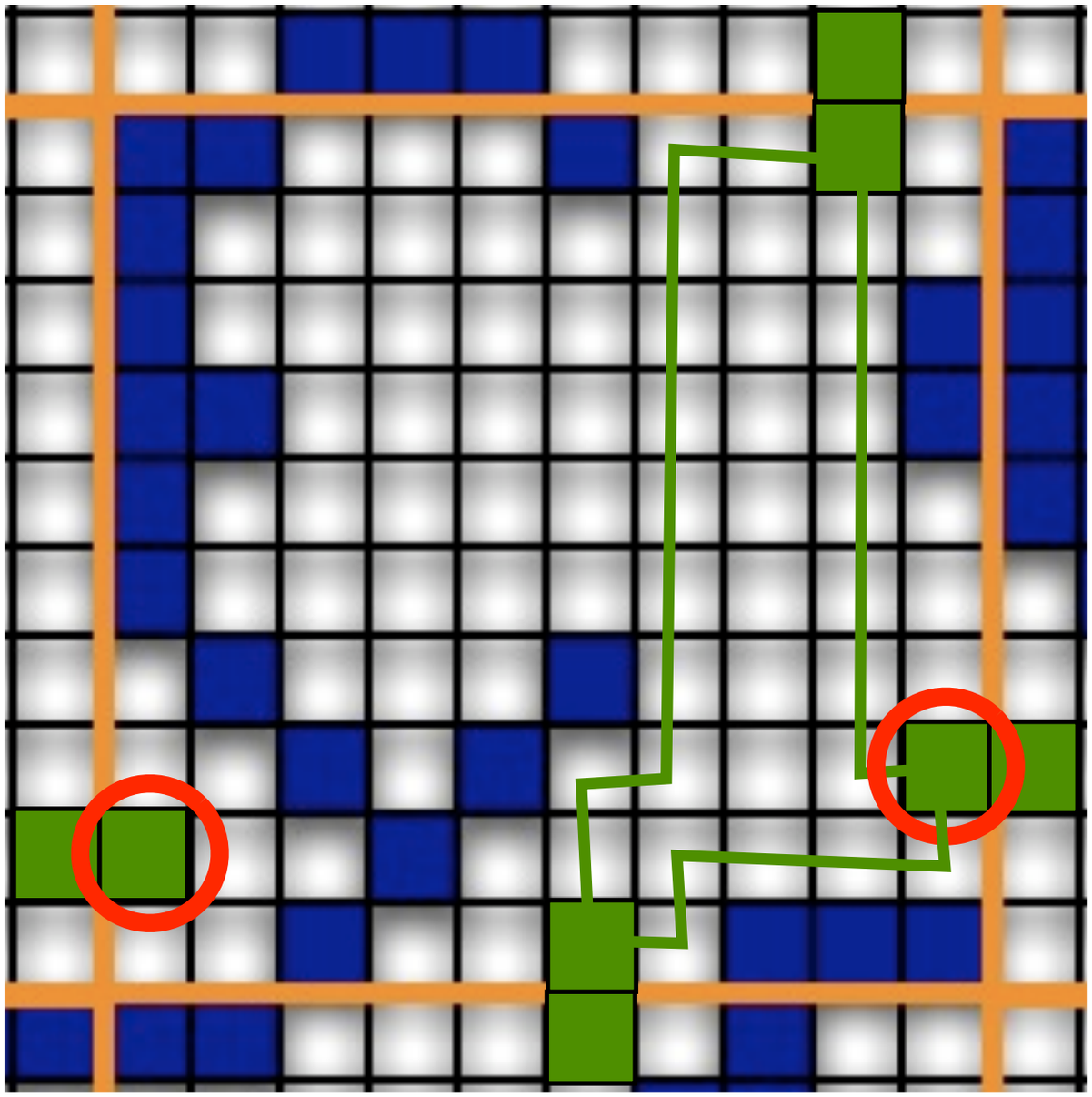


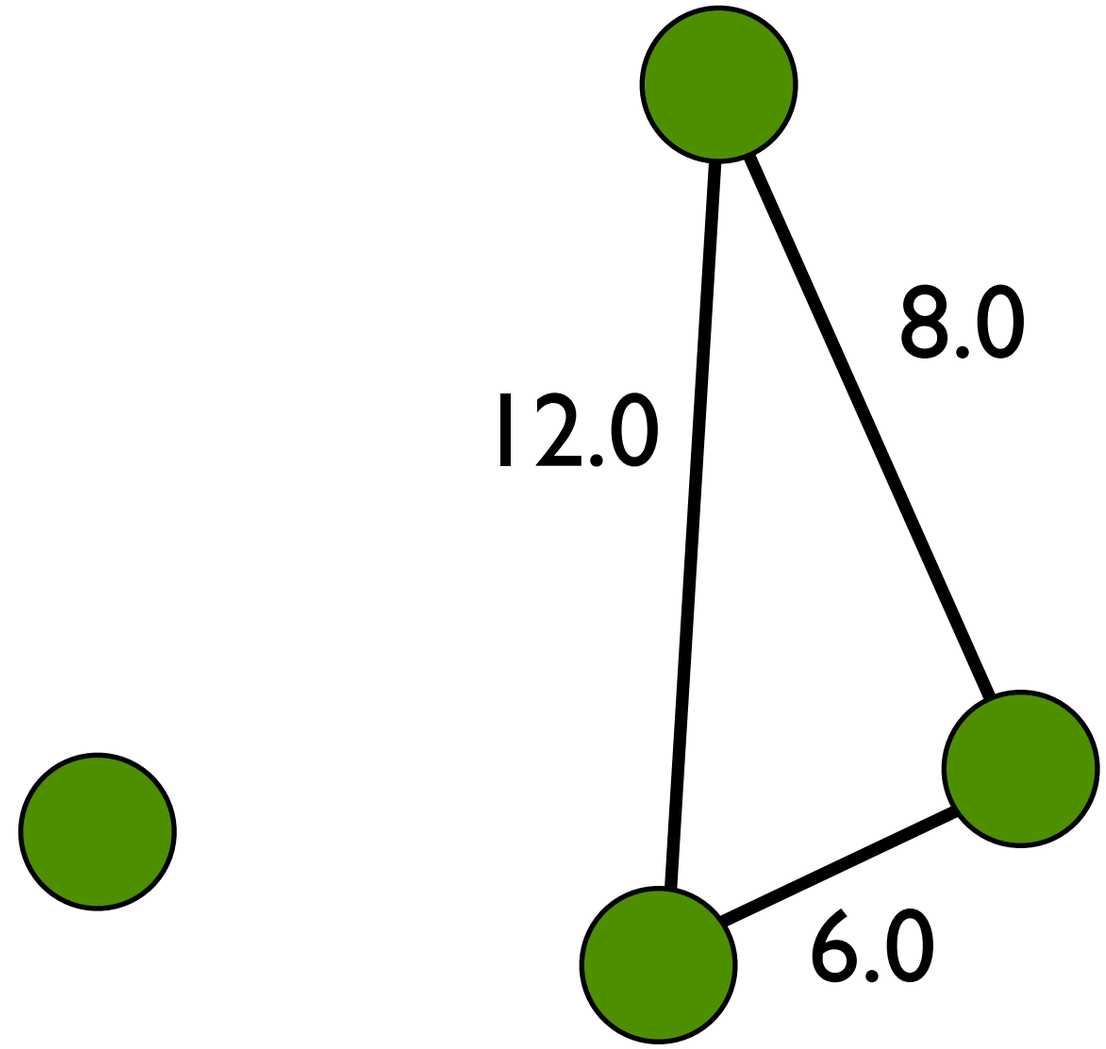

Insert intra-edges (of cost = path length) to abstract graph.



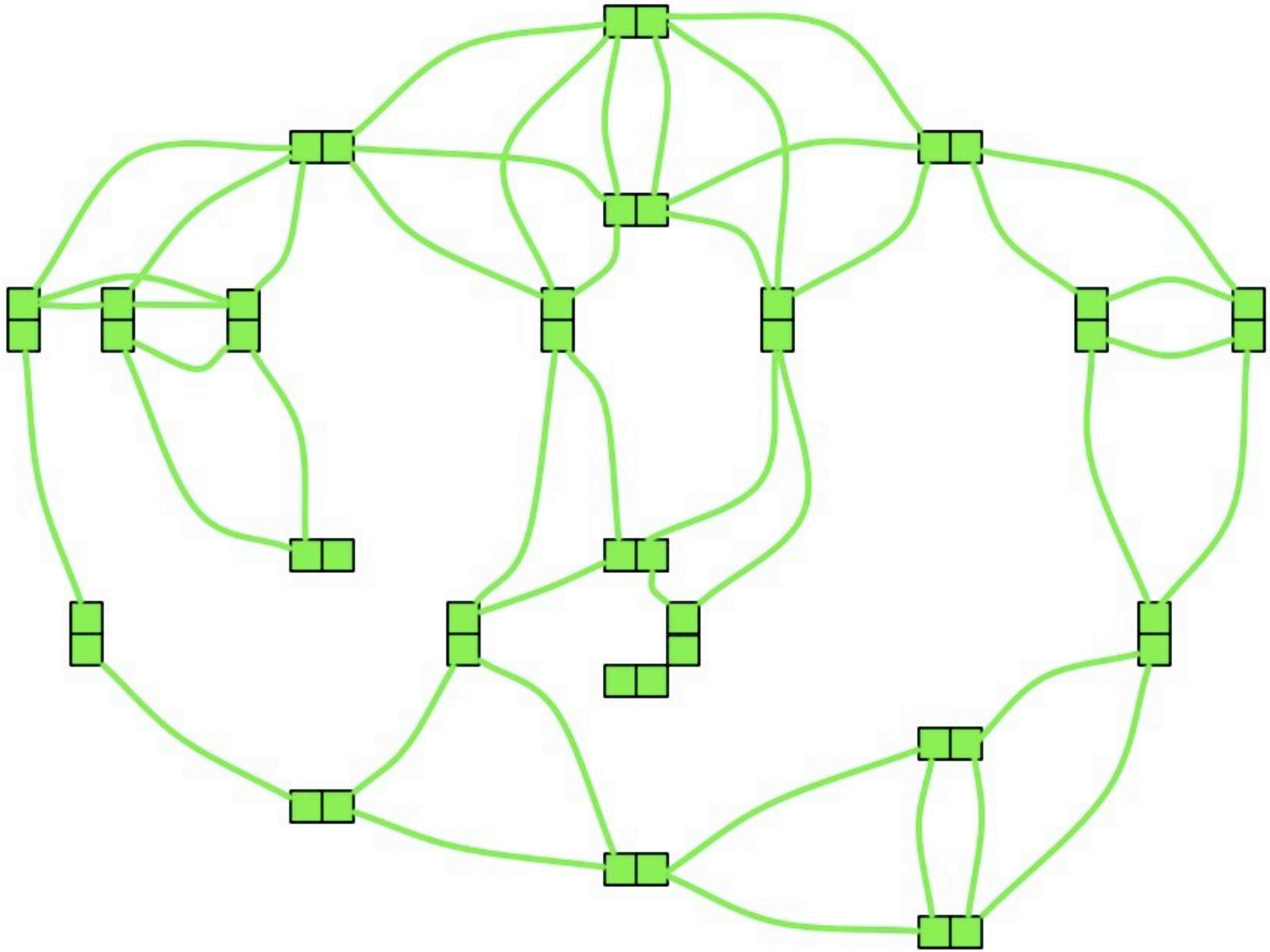



Insert intra-edges (of cost = path length) to abstract graph.






Insert intra-edges (of cost = path length) to abstract graph.

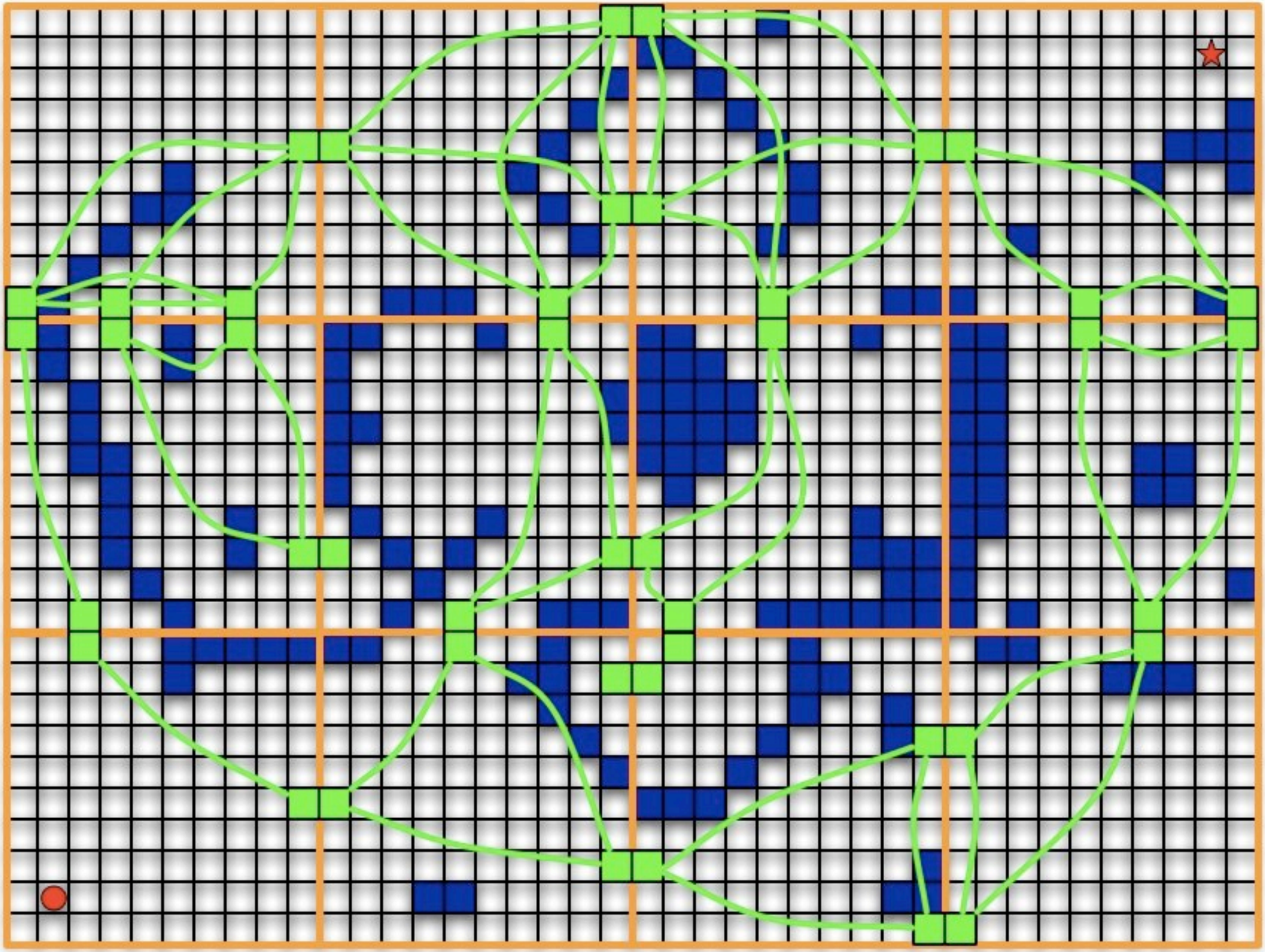


# Final abstract graph

Image: Adi Botea

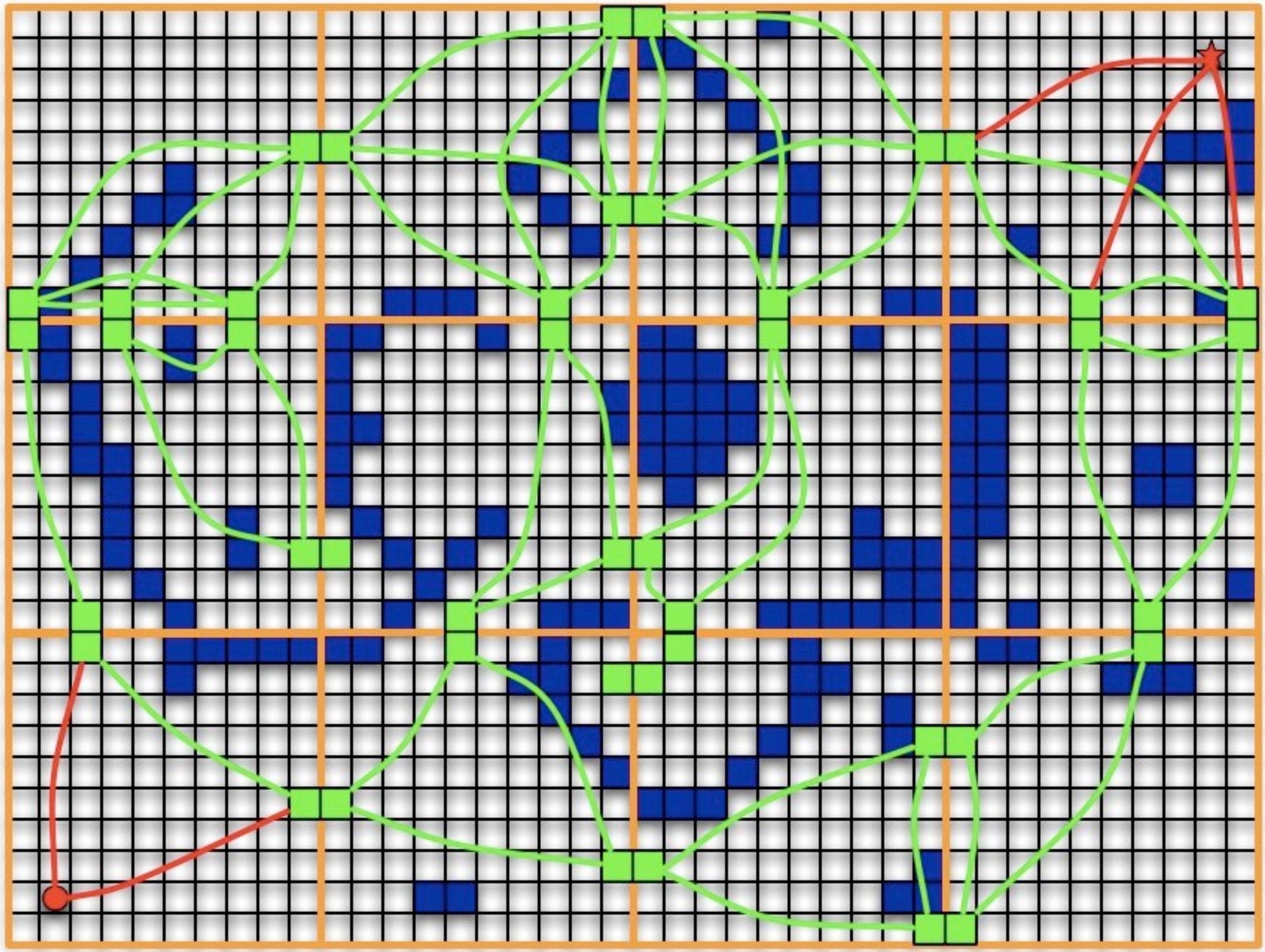
# Runtime processing

- Insert start + goal into abstract graph.
- Find an abstract path connecting them.
- Refine each abstract step using a small A* search to traverse each cluster (free if we cache intra-edge paths).



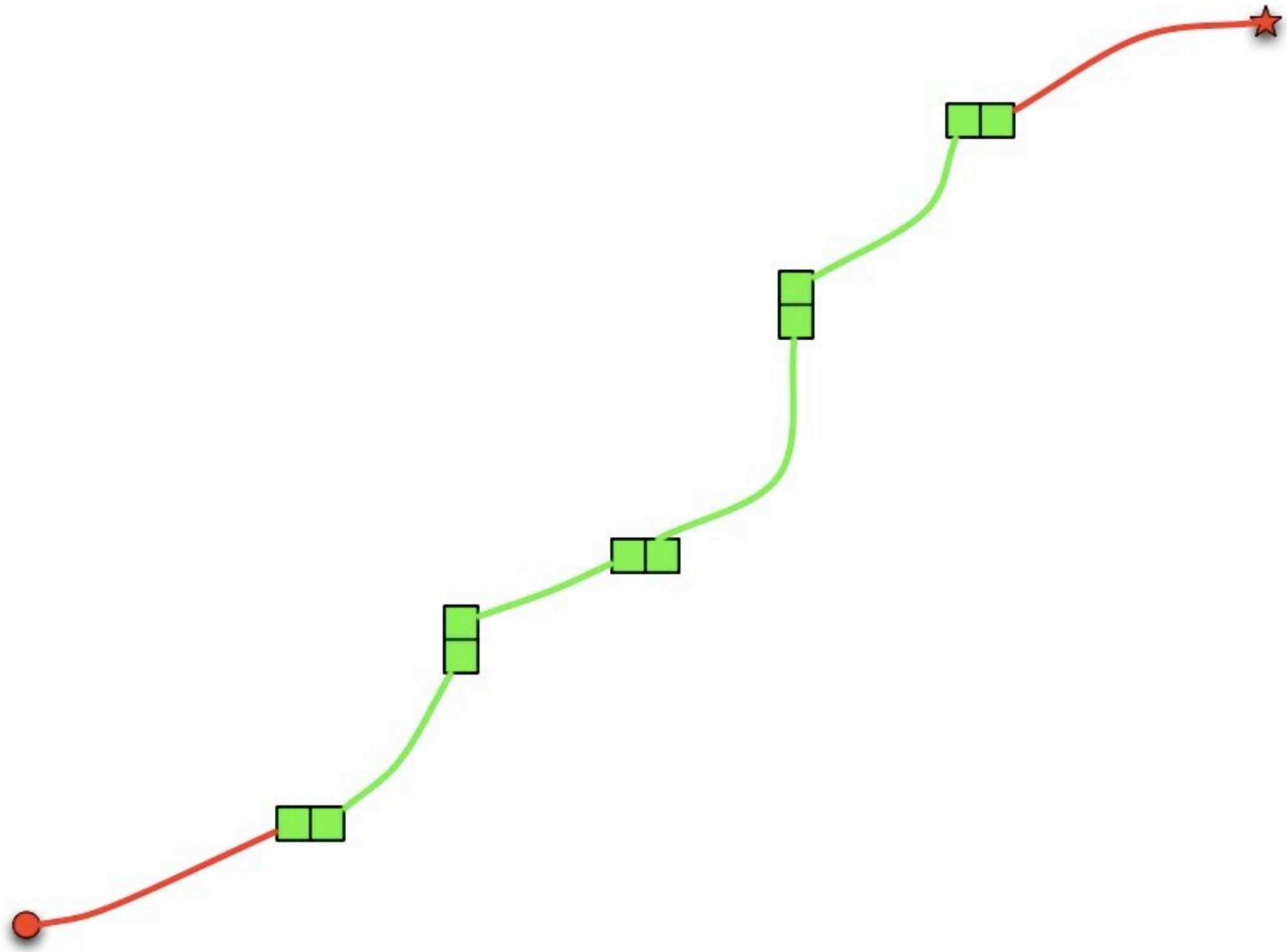
# Add Start + Goal as (temp.) abstract nodes

Image: Adi Botea



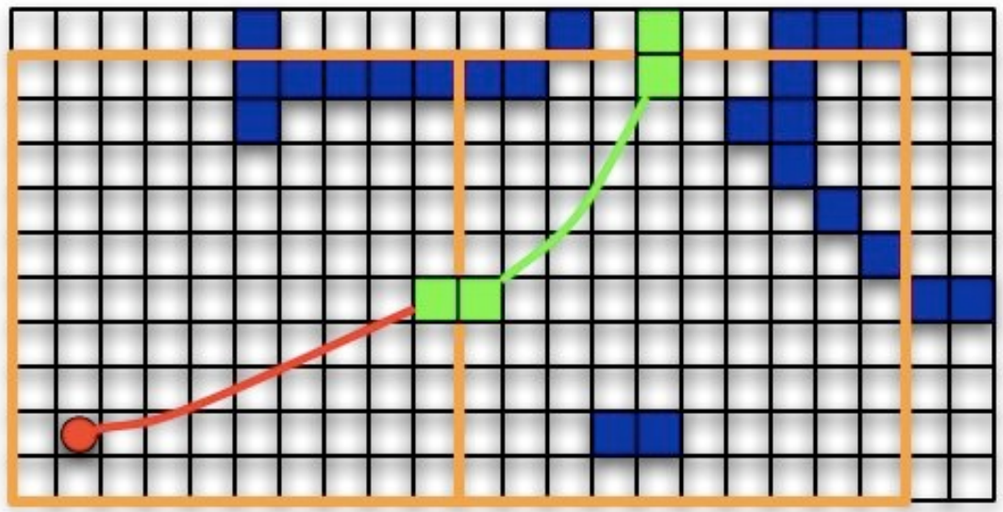
Insert temp. edges to connect Start + Goal

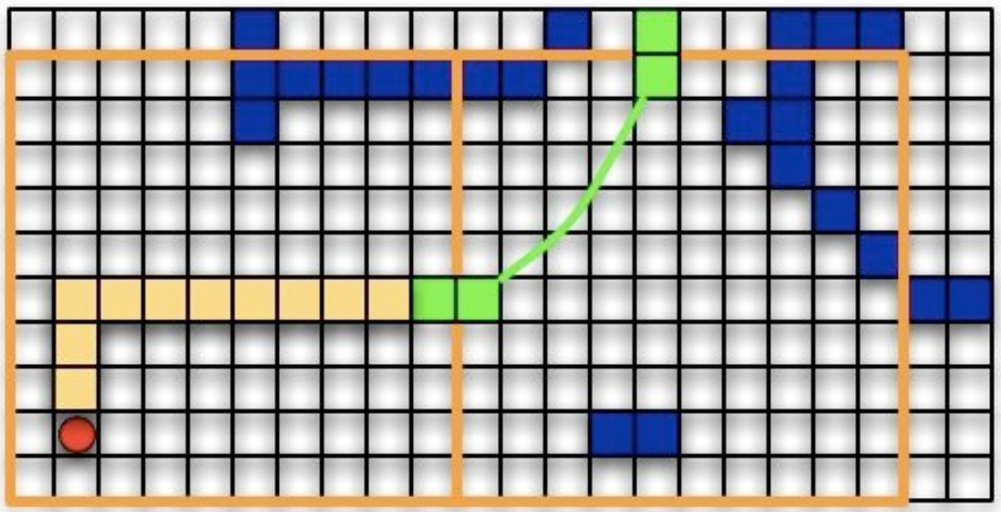
Image: Adi Botea

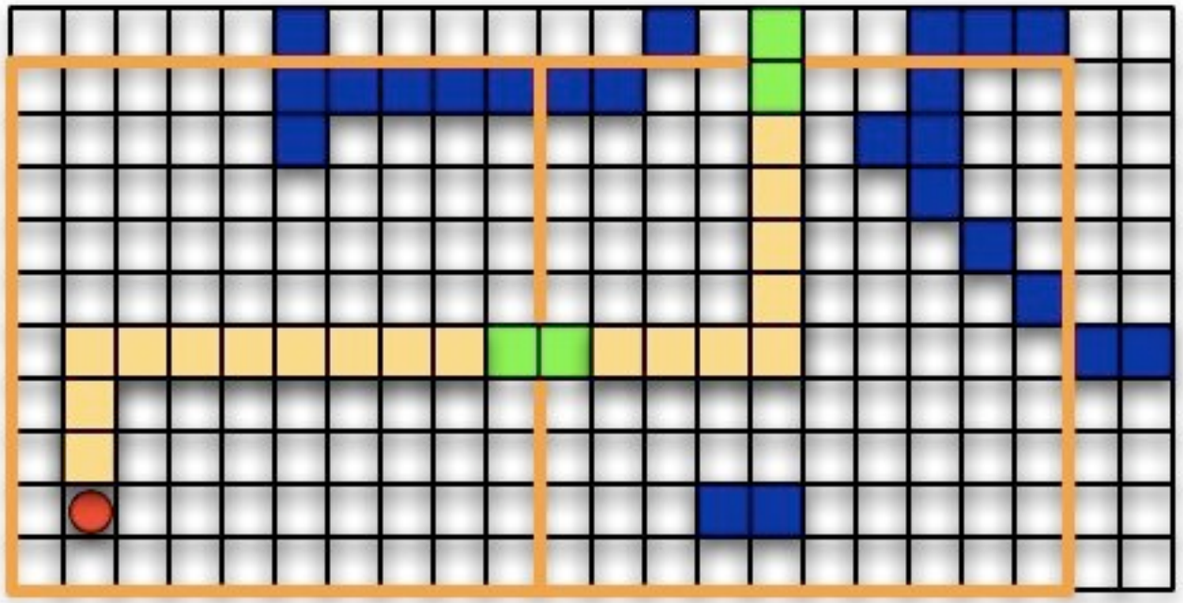


Run A* to find an abstract path

Image: Adi Botea







Refine the abstract path before execution. Each step is independent (and fast!)

Images: Adi Botea

# HPA* Advantages

- Easy to understand + implement
- Very fast (Beats A* by 10x)
- Low memory overhead (vs. A*)
- Complete

# HPA* Disadvantages

- Non-optimal (but very close!).
- Assumes fixed-size 1x1 units.
- Only supports single terrain type.
- Insertion effort.

# Part 2:
# Dealing with diversity

# Assumptions in pathfinding literature

# Assumptions in pathfinding literature

- Assumption #1: all units are the same size.

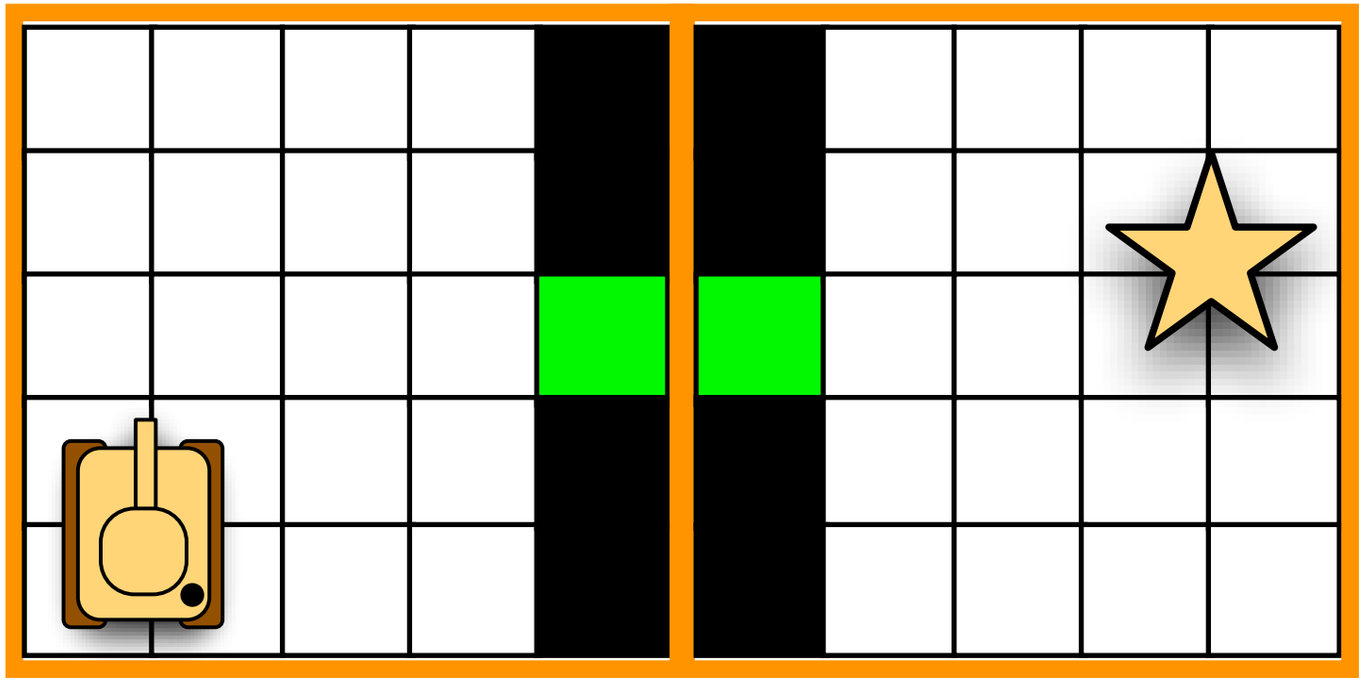
# Assumptions in pathfinding literature

- Assumption #1: all units are the same size.
- Assumption #2: any terrain traversable for one unit is traversable for all.

# Assumptions in pathfinding literature

- Assumption #1: all units are the same size.
- Assumption #2: any terrain traversable for one unit is traversable for all.
- Pathfinding breaks when either assumption is lifted.

# HPA* Tank Fail



HPA* loses information (like sizes of entrances)

# Challenging pathfinding

Unit sizes: Small, Medium, Large
Terrains: Snow, Road, Water, Ground

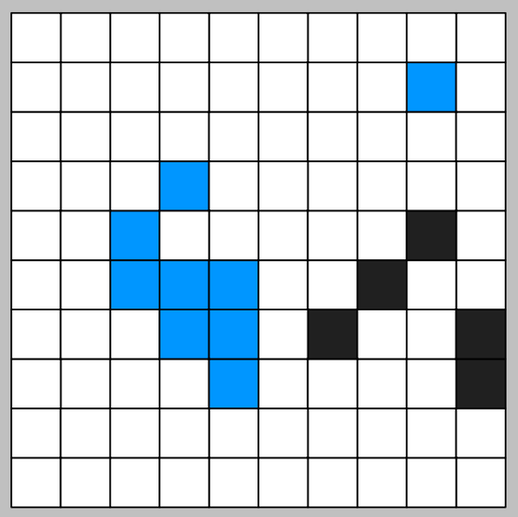
Images: Red Alert 3 (Electronic Arts)

# HAA* Overview

- [Harabor, Botea, 2008]
- HPA*-style abstraction & search.
- Combined with clearance-based pathfinding.
- Caters for different terrains and unit sizes.
- Constructs complete abstractions.

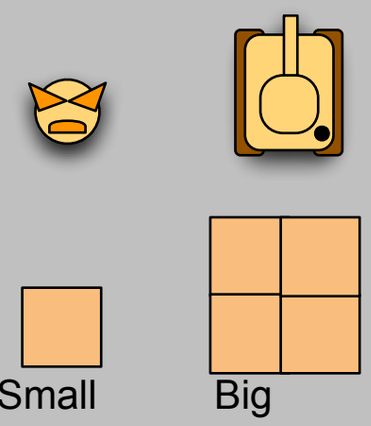
# Modeling diversity

## Example Map

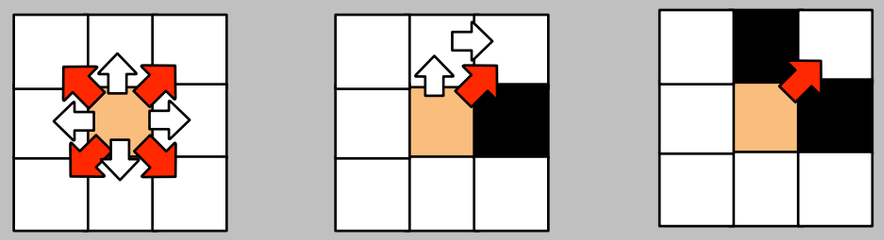


Ground (white tiles)
Water (blue tiles)
Obstacles (black tiles)

## Agent Sizes




## Movement rules




## Terrain traversal Capabilities

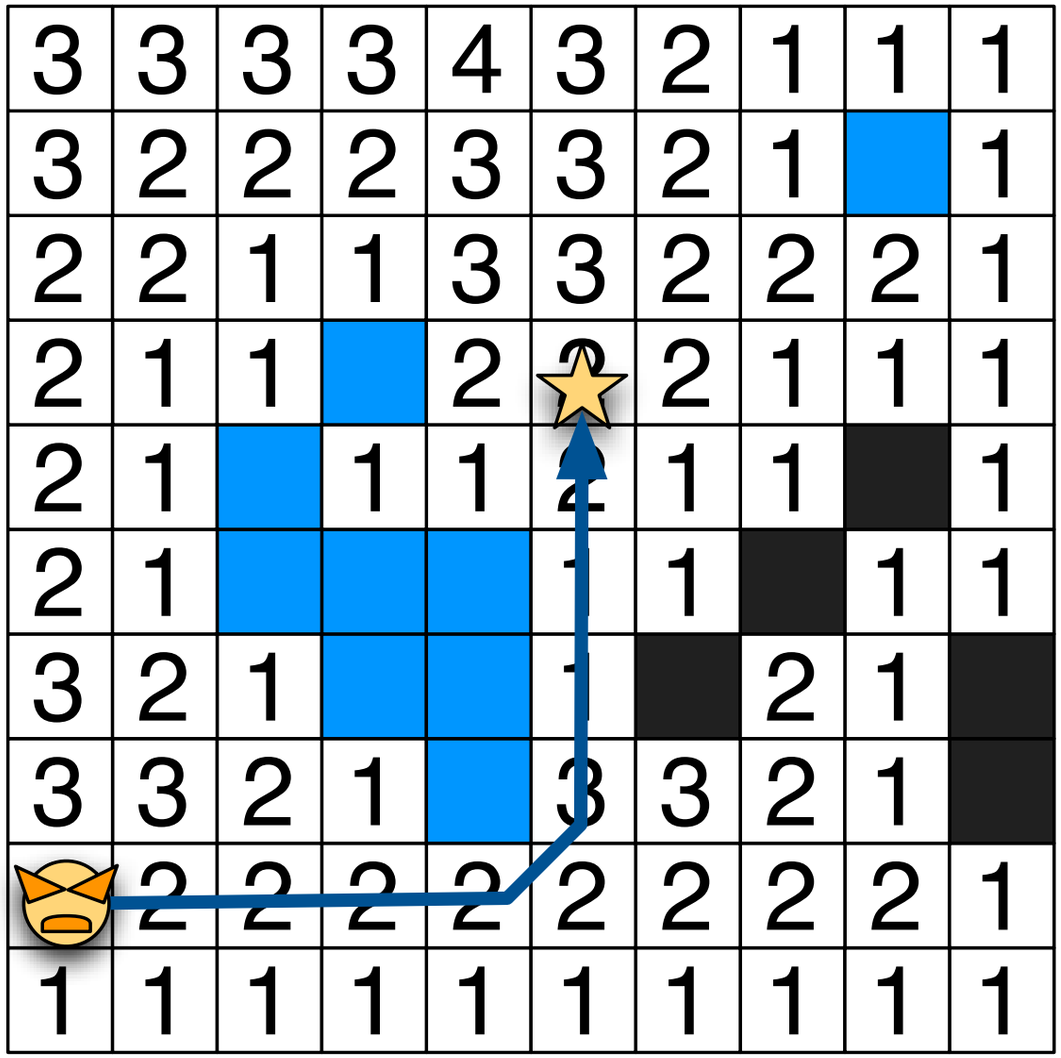
{Ground}
{Water}
{Ground or Water}

# Clearance-based pathfinding

- Intuition:
  - Calculate how much clearance (traversable space) exists at a given tile (t) on the map.
  - A tile (t) is traversable by an agent (a) if:

    1. terrain(t) is in capability(a)
    2. clearance(t, capability(a)) >= size(a)

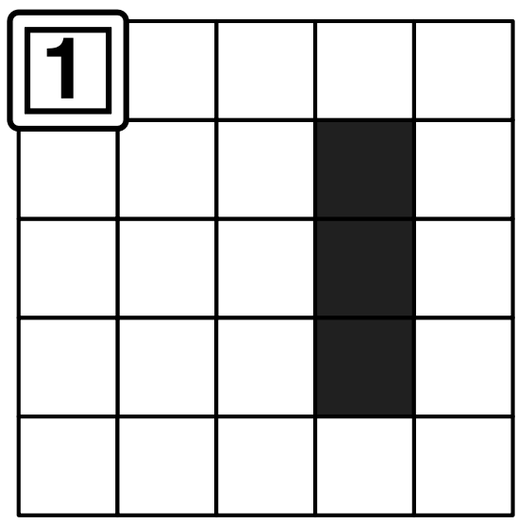
# Example




Agent size = 1
Capability = {Ground}

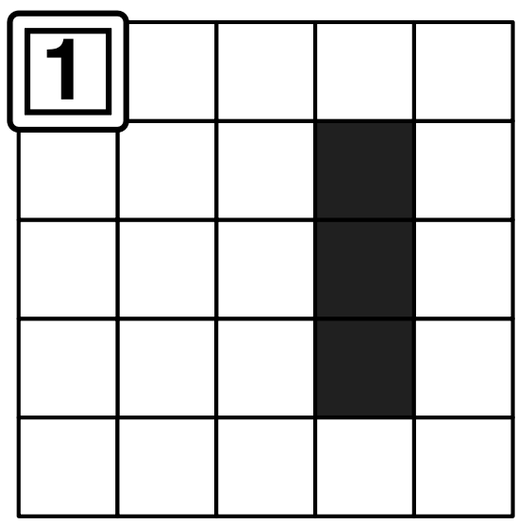
Each tile on the path is traversable for the agent

# Computing clearance

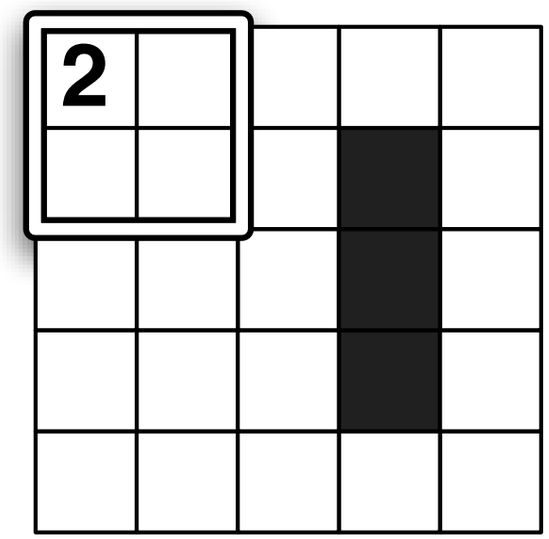



Initial Clearance.
Capability = {Ground}
(white tiles)

# Computing clearance

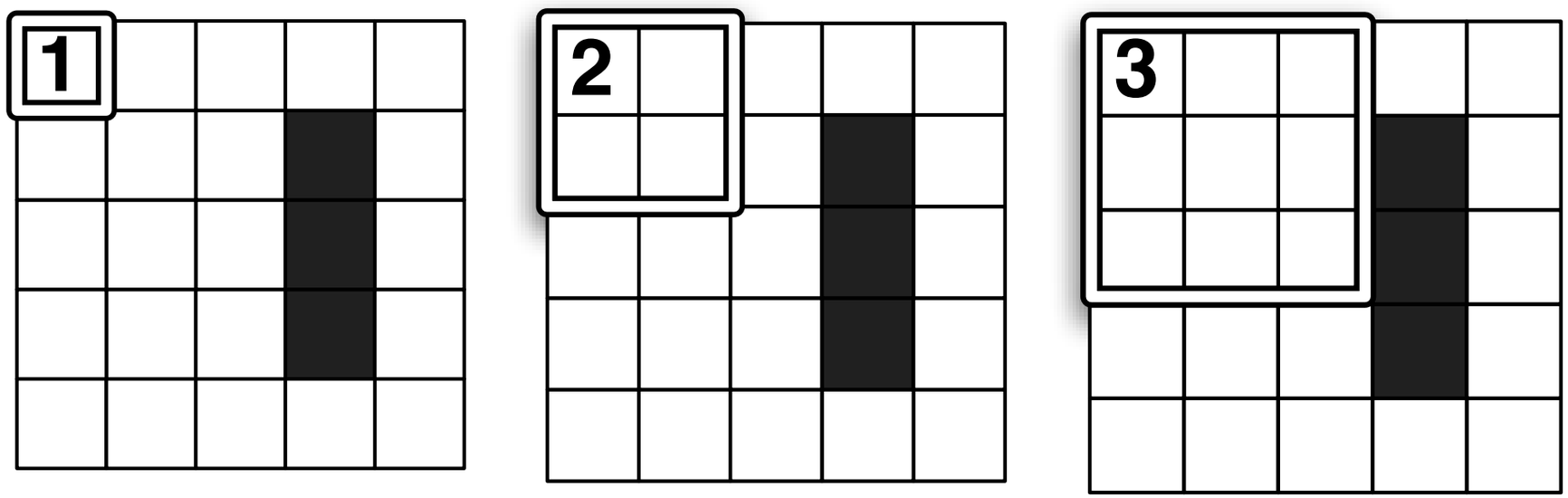



Initial Clearance.
Capability = {Ground}
(white tiles)




First Expansion
OK

# Computing clearance

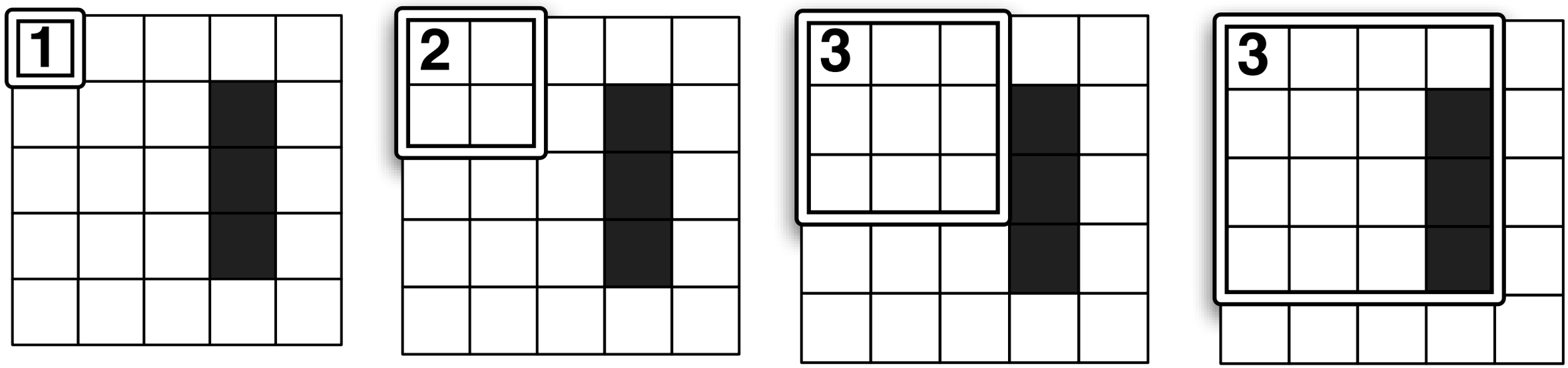



Initial Clearance.
Capability = {Ground}
(white tiles)

First Expansion
OK

Second Expansion
OK

# Computing clearance




Initial Clearance.
Capability = {Ground}
(white tiles)

First Expansion
OK

Second Expansion
OK

Third Expansion
Fail

# Maps with multiple terrain types

- Easy! Each tile has a clearance-value for each possible capability.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 4 | 3 | 2 | 1 | 1 | 1 |
| 3 | 2 | 2 | 2 | 3 | 3 | 2 | 1 | | 1 |
| 2 | 2 | 1 | 1 | 3 | 3 | 2 | 2 | 2 | 1 |
| 2 | 1 | 1 | | 2 | 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | | 1 | 1 | 2 | 1 | 1 | | 1 |
| 2 | 1 | | | | 1 | 1 | | 1 | 1 |
| 3 | 2 | 1 | | | 1 | | 2 | 1 | |
| 3 | 3 | 2 | 1 | | 3 | 3 | 2 | 1 | |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Clearances for {Ground} Capability

# Maps with multiple terrain types

- Easy! Each tile has a clearance-value for each possible capability.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 4 | 3 | 2 | 1 | 1 | 1 |
| 3 | 2 | 2 | 2 | 3 | 3 | 2 | 1 | | 1 |
| 2 | 2 | 1 | 1 | 3 | 3 | 2 | 2 | 2 | 1 |
| 2 | 1 | 1 | | 2 | 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | | 1 | 1 | 2 | 1 | 1 | | 1 |
| 2 | 1 | | | | 1 | 1 | | 1 | 1 |
| 3 | 2 | 1 | | | 1 | | 2 | 1 | |
| 3 | 3 | 2 | 1 | | 3 | 3 | 2 | 1 | |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Clearances for {Ground} Capability

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | | | | | | | 1 | |
| | | | | | | | | | |
| | | | 1 | | | | | | |
| | | 1 | | | | | | | |
| | | 1 | 2 | 1 | | | | | |
| | | | 1 | 1 | | | | | |
| | | | | 1 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

Clearances for {Water} Capability

# Maps with multiple terrain types

- Easy! Each tile has a clearance-value for each possible capability.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 4 | 3 | 2 | 1 | 1 | 1 |
| 3 | 2 | 2 | 2 | 3 | 3 | 2 | 1 | | 1 |
| 2 | 2 | 1 | 1 | 3 | 3 | 2 | 2 | 2 | 1 |
| 2 | 1 | 1 | | 2 | 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | | 1 | 1 | 2 | 1 | 1 | | 1 |
| 2 | 1 | | | | 1 | 1 | | 1 | 1 |
| 3 | 2 | 1 | | | 1 | | 2 | 1 | |
| 3 | 3 | 2 | 1 | | 3 | 3 | 2 | 1 | |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Clearances for {Ground} Capability

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | | | | | | | 1 | |
| | | | | | | | | | |
| | | | 1 | | | | | | |
| | | 1 | | | | | | | |
| | | 1 | 2 | 1 | | | | | |
| | | | 1 | 1 | | | | | |
| | | | | 1 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

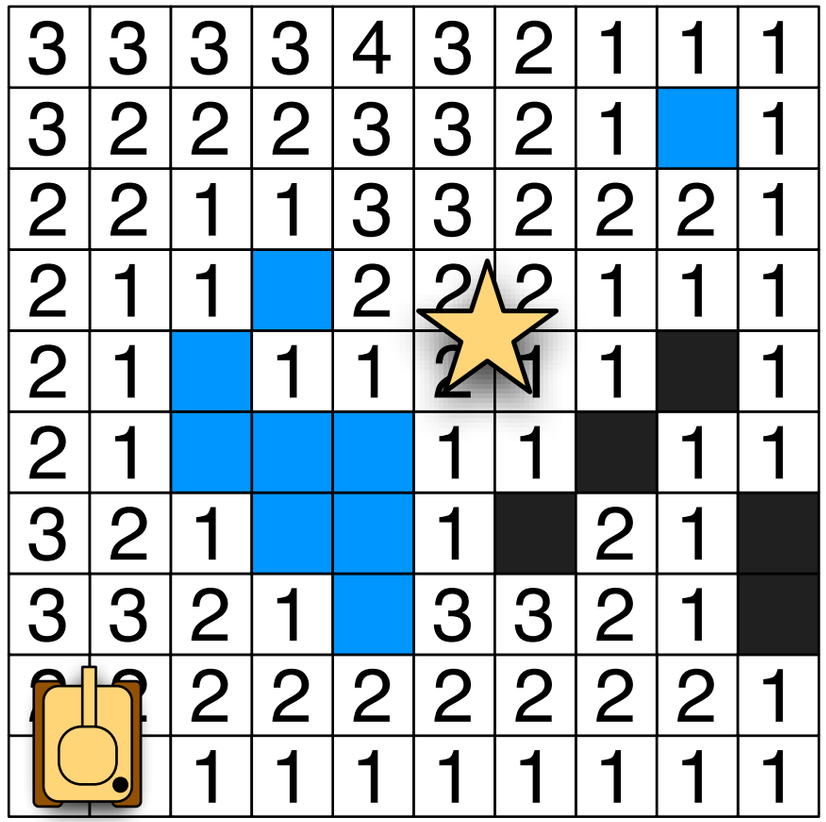
Clearances for {Water} Capability

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 6 | 5 | 5 | 4 | 4 | 4 | 3 | 2 | 1 |
| 6 | 5 | 5 | 4 | 4 | 3 | 3 | 3 | 2 | 1 |
| 6 | 5 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 1 |
| 6 | 5 | 4 | 3 | 3 | 2 | 2 | 1 | 1 | 1 |
| 6 | 5 | 4 | 3 | 2 | 2 | 1 | 1 | | 1 |
| 5 | 5 | 4 | 3 | 2 | 1 | 1 | | 1 | 1 |
| 4 | 4 | 4 | 3 | 2 | 1 | | 2 | 1 | |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 1 | |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Clearances for {Ground, Water} Capability

# Dealing with large agents

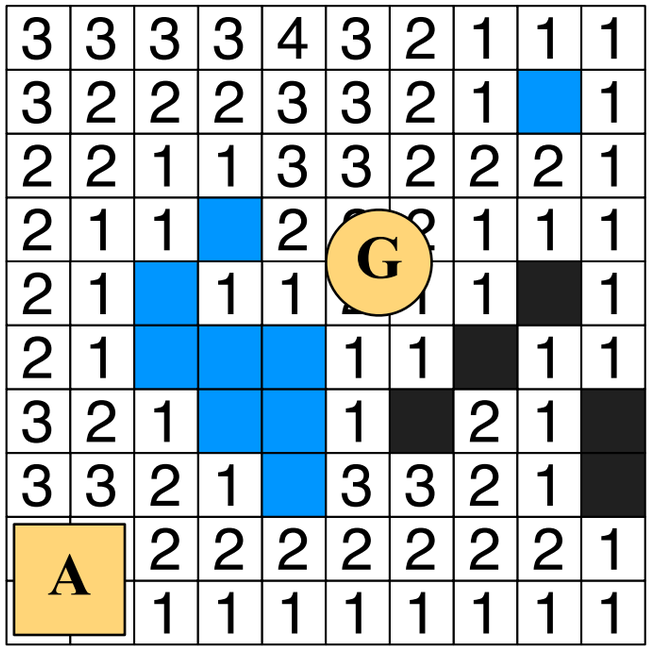
Each 2x2 agent occupies 4 tiles.
Which clearance value do we consider?




Agent size = 2, Capability = {Ground}

# Problem reduction

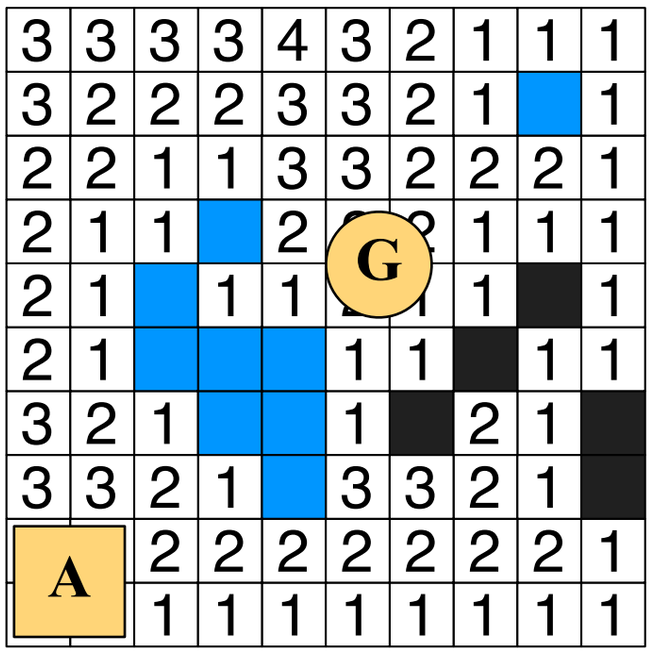
Theorem: Any pathfinding problem can be reduced into a canonical problem (agent size = 1, capability = 1 terrain)



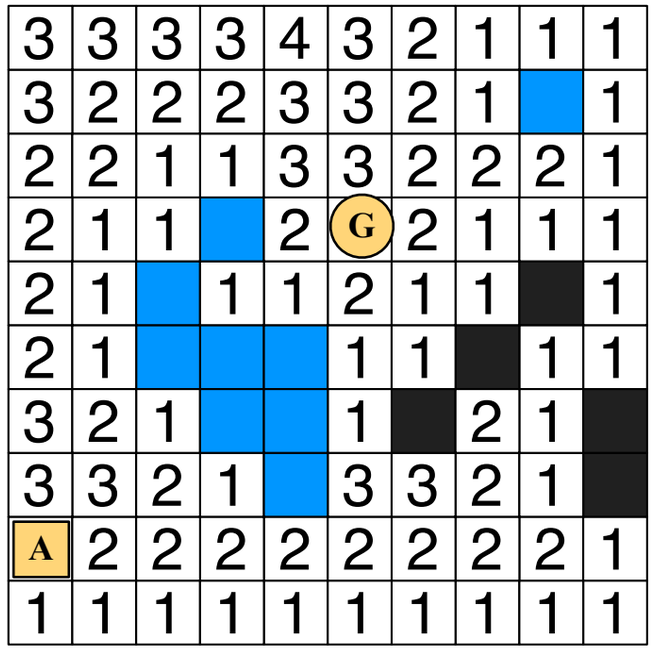

Initial problem

# Problem reduction

Theorem: Any pathfinding problem can be reduced into a canonical problem (agent size = 1, capability = 1 terrain)



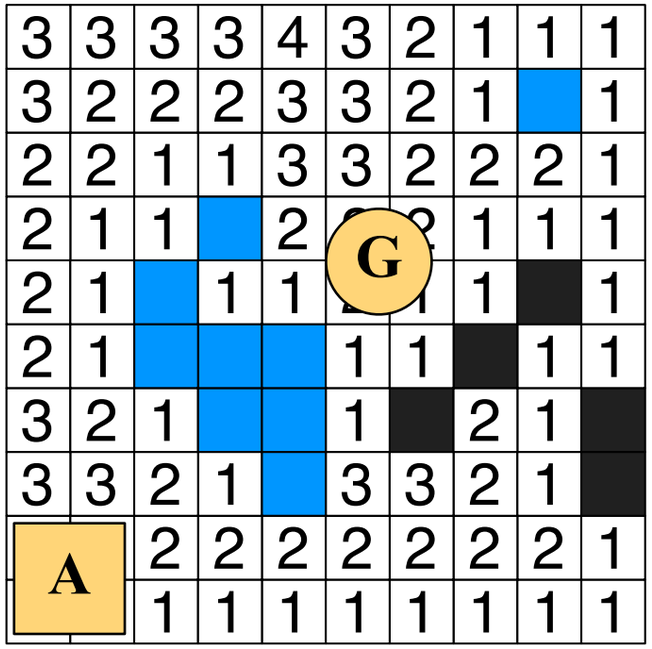

Initial problem



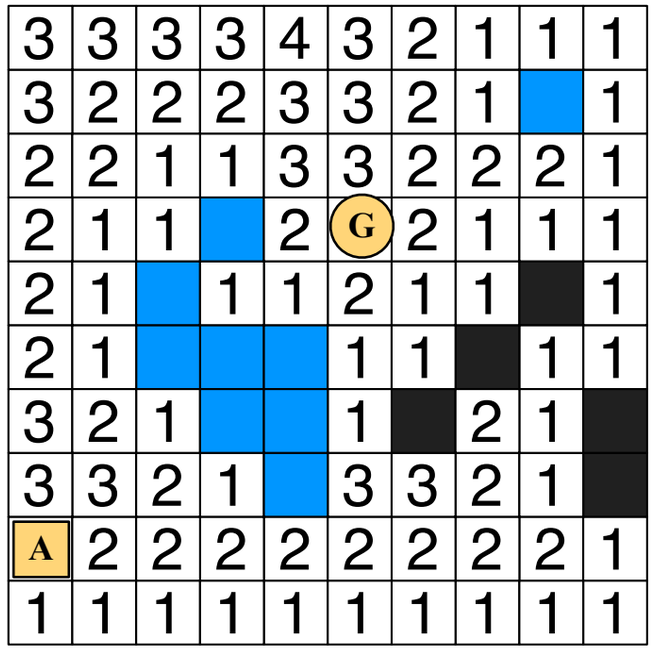

Reduced size

# Problem reduction

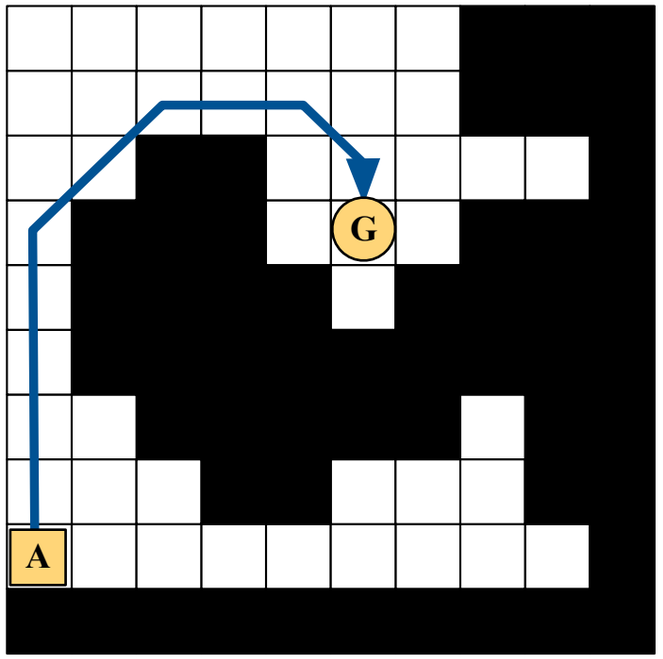
Theorem: Any pathfinding problem can be reduced into a canonical problem (agent size = 1, capability = 1 terrain)




Initial problem




Reduced size




Reduced size and capability

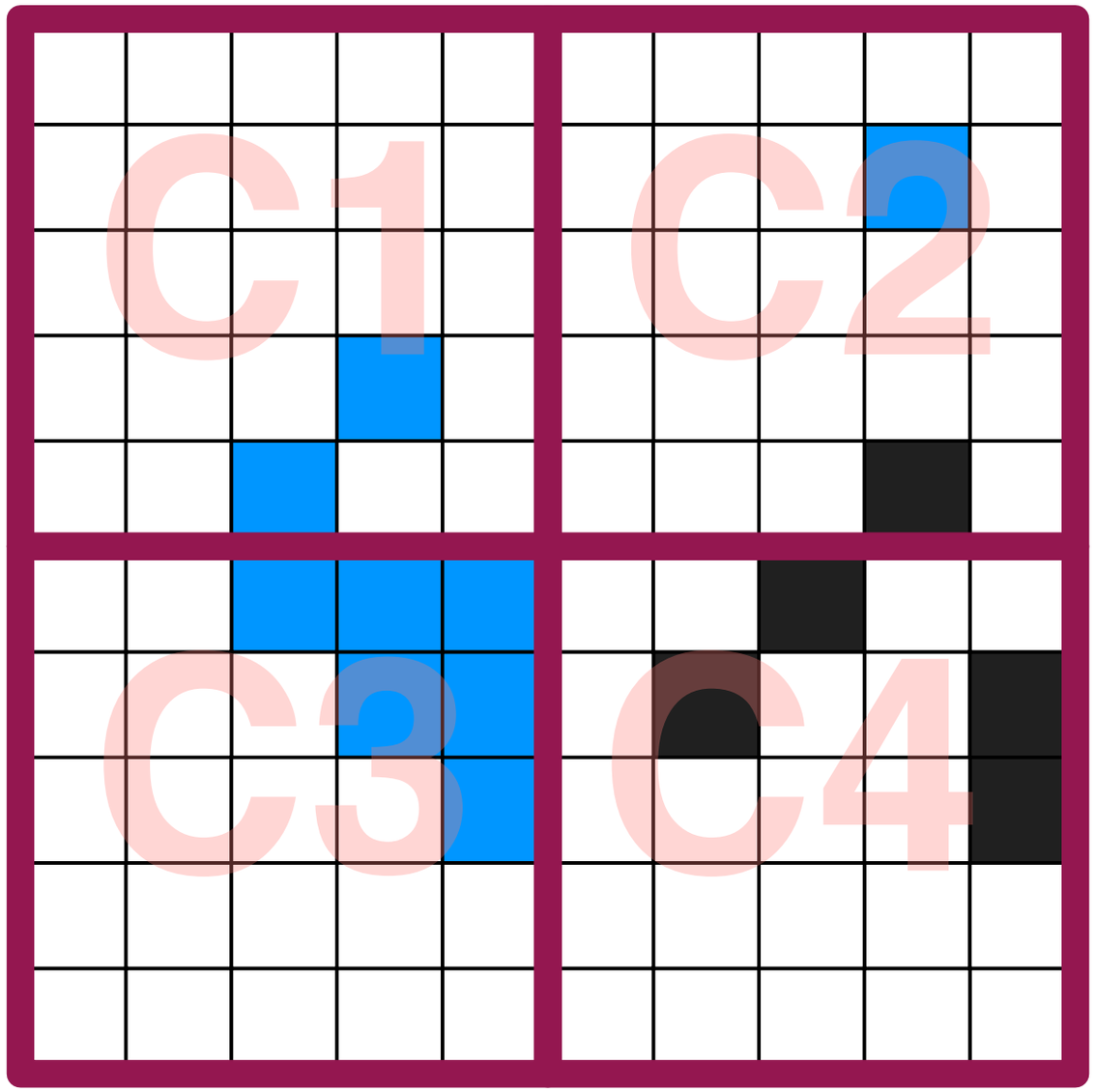
# Annotated A*

- Search process:
  - Similar to A*.
  - Extra parameters: Agent's size and capability.
  - Only expand nodes with clearance > agent size.
- Works great!
- For small problem sizes...

# Abstraction Process

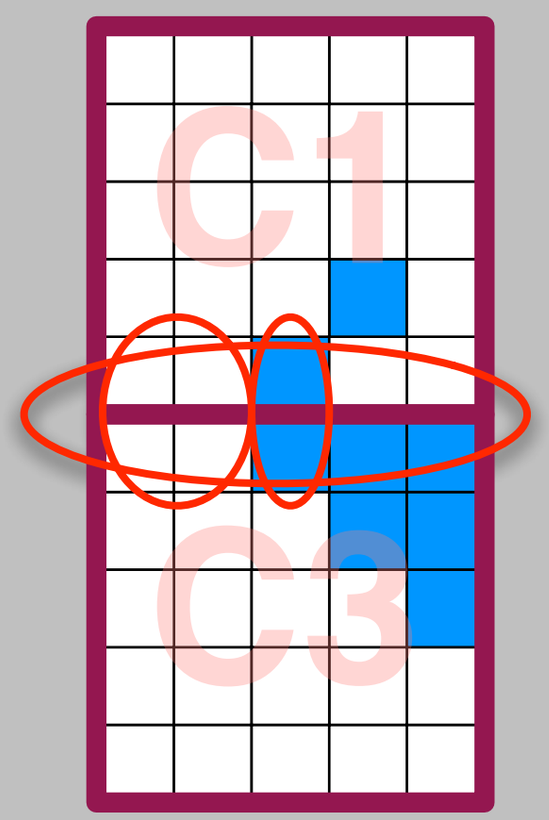
- Similar to HPA*:
  - Divide map into NxN clusters.
  - Identify entrances for all capabilities.
  - Using AA*, find intra-edges for all agent sizes and capabilities
  - Reduce the graph using dominance

# Clusters

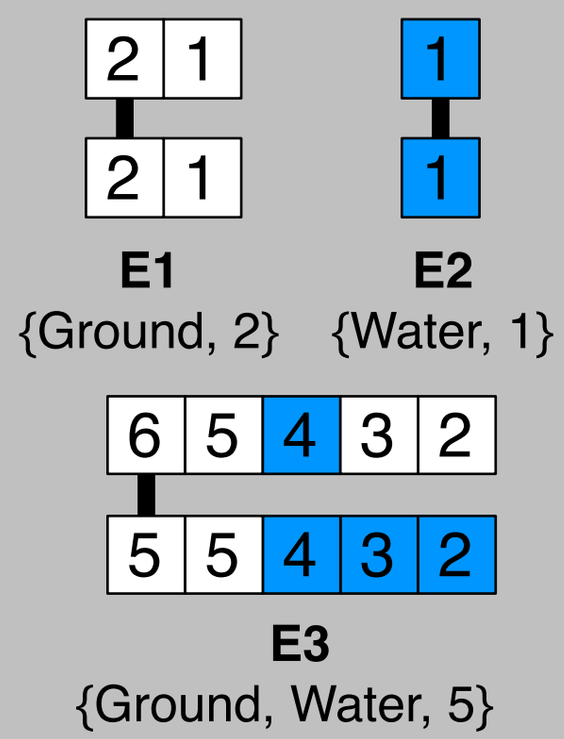


# Entrances

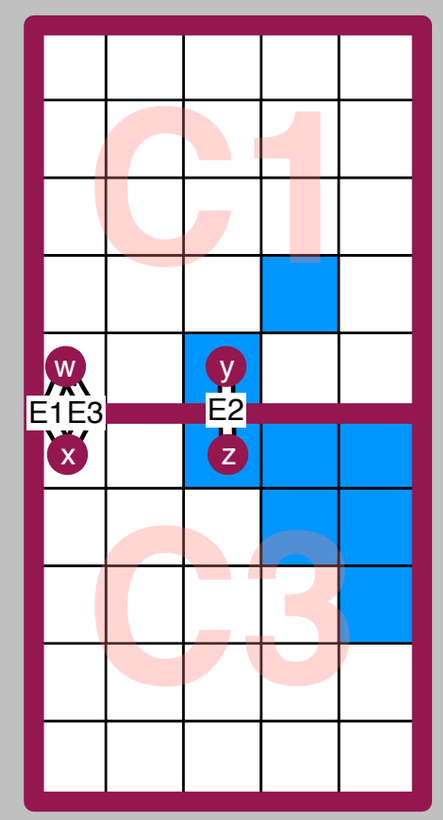
## Identifying entrances




## Transition points

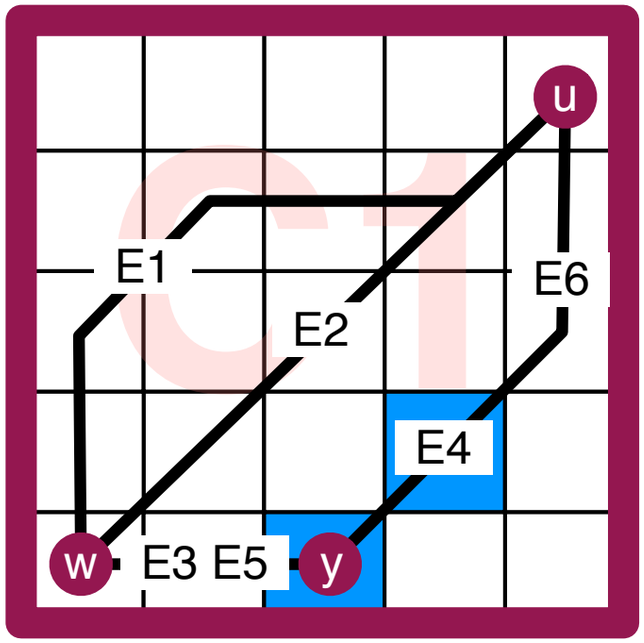



## Abstract graph



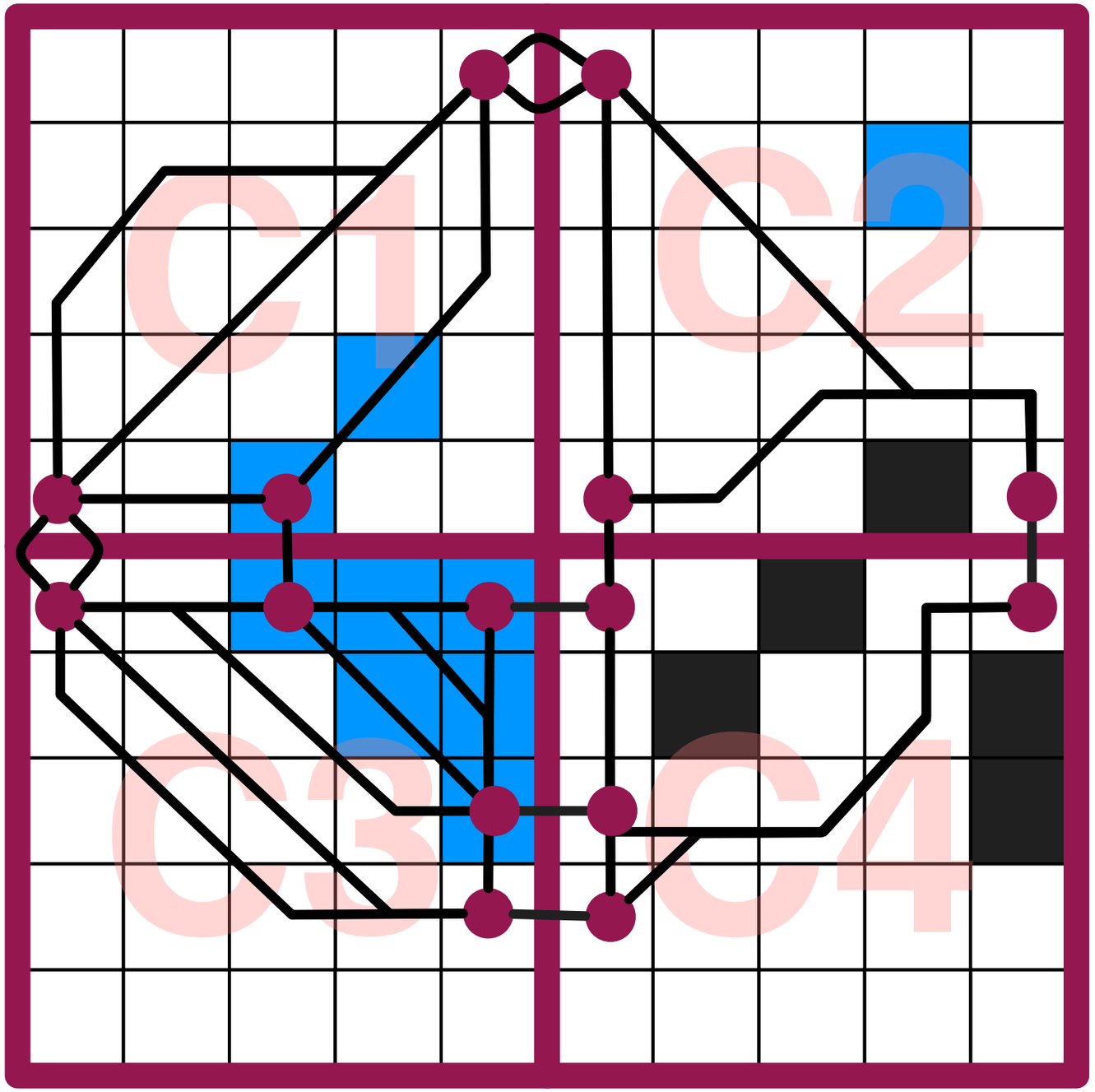
# Intra-edges

- Use AA* to find all possible ways (sizes/capabilities) to traverse across each cluster.
- For each path found, add a new abstract edge.




Edge Annotations
{Terrain, Clearance}:
E1 = {Ground, 2}
E2 = {Ground, 1}
E3 = {Ground, Water, 2}
E4 = {Ground, Water, 2}
E3 = {Ground, Water, 1}
E4 = {Ground, Water, 1}

# Initial abstraction

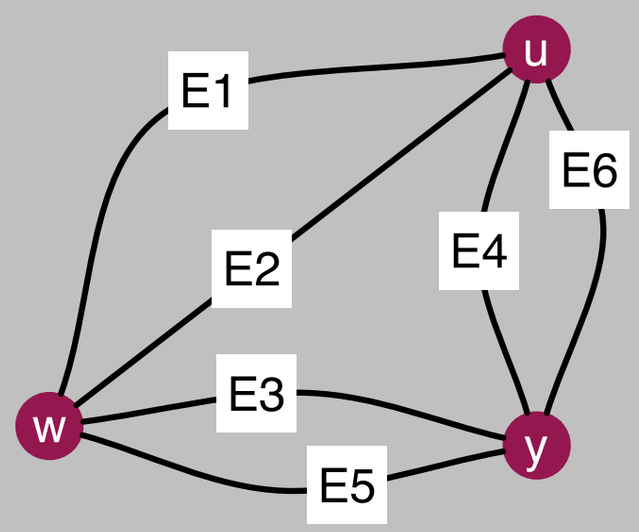


# Compacting the abstract graph

- Method produces a representationally complete graph but can get rather large.
- Solutions:
  - Strong dominance
  - Weak dominance

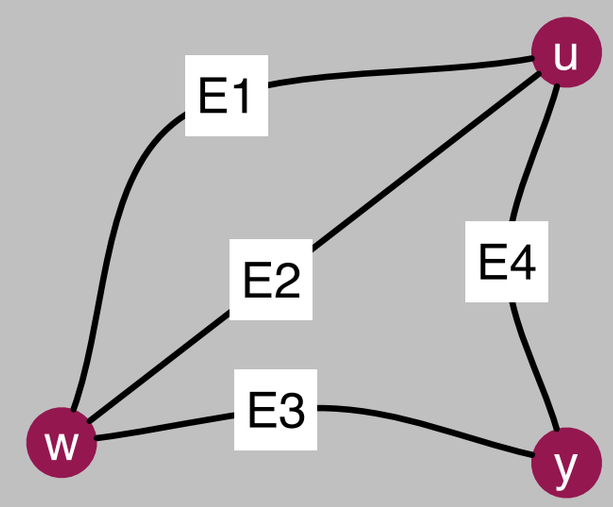
# Strong dominance

## Initial graph




Edge Annotations
[Capability, Clearance, Length]:
E1 = [{Ground}, 2, 7.5]
E2 = [{Ground}, 1, 4.5]
E3 = [{Ground, Water}, 2, 3.0]
E4 = [{Ground, Water}, 2, 3.0]
E5 = [{Ground, Water}, 1, 3.0]
E6 = [{Ground, Water}, 1, 3.0]

## High Quality graph

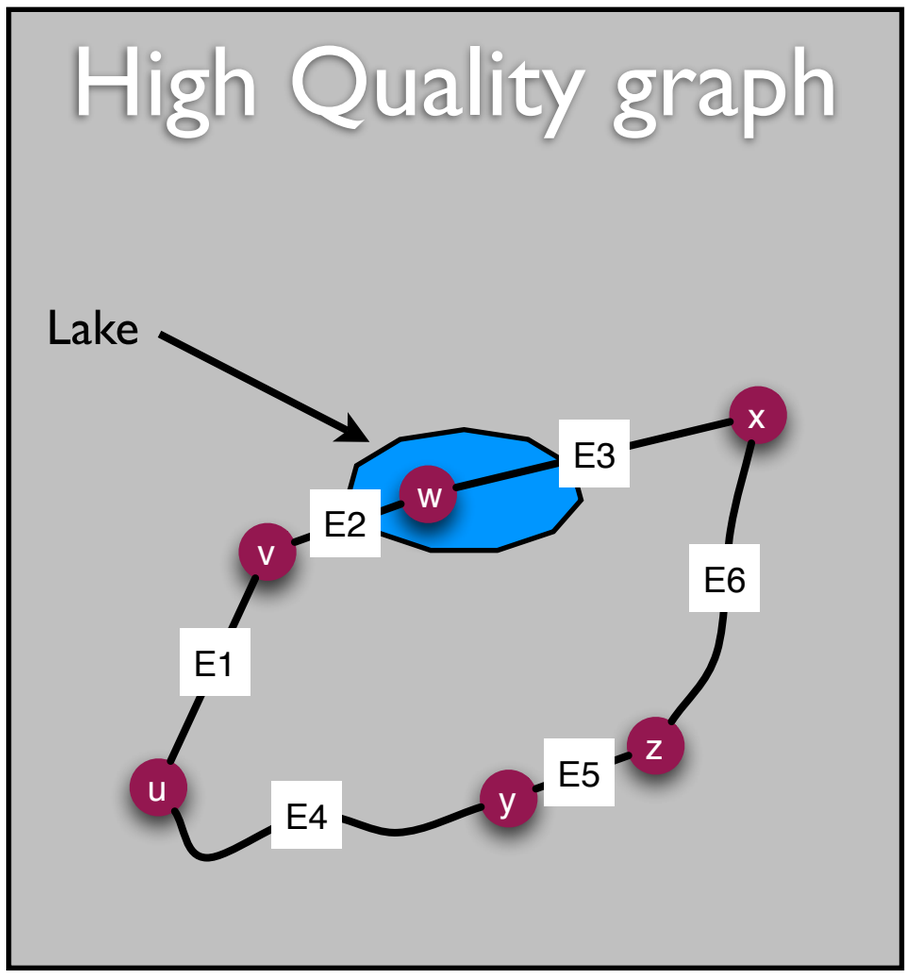


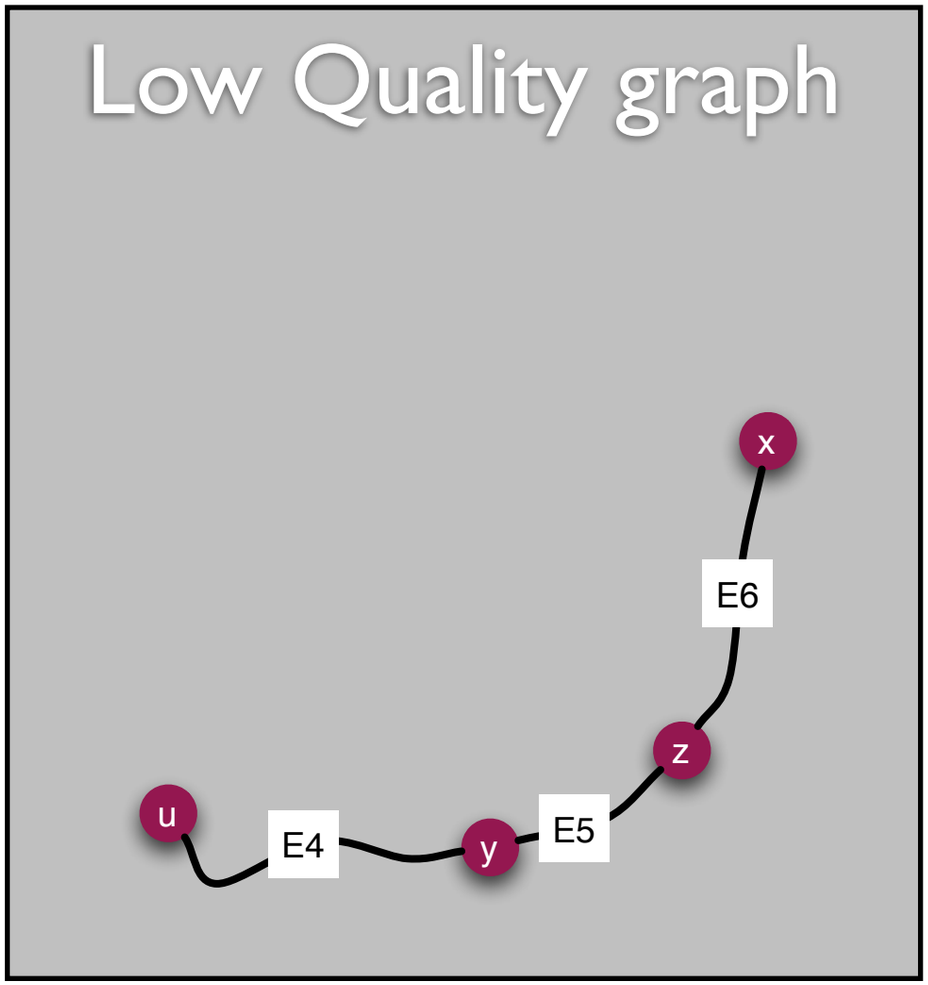

Edge Annotations
[Capability, Clearance, Length]:
E1 = [{Ground}, 2, 7.5]
E2 = [{Ground}, 1, 4.5]
E3 = [{Ground, Water}, 2, 3.0]
E4 = [{Ground, Water}, 2, 3.0]

Remove edges with smaller clearance (all else being identical)

# Weak dominance

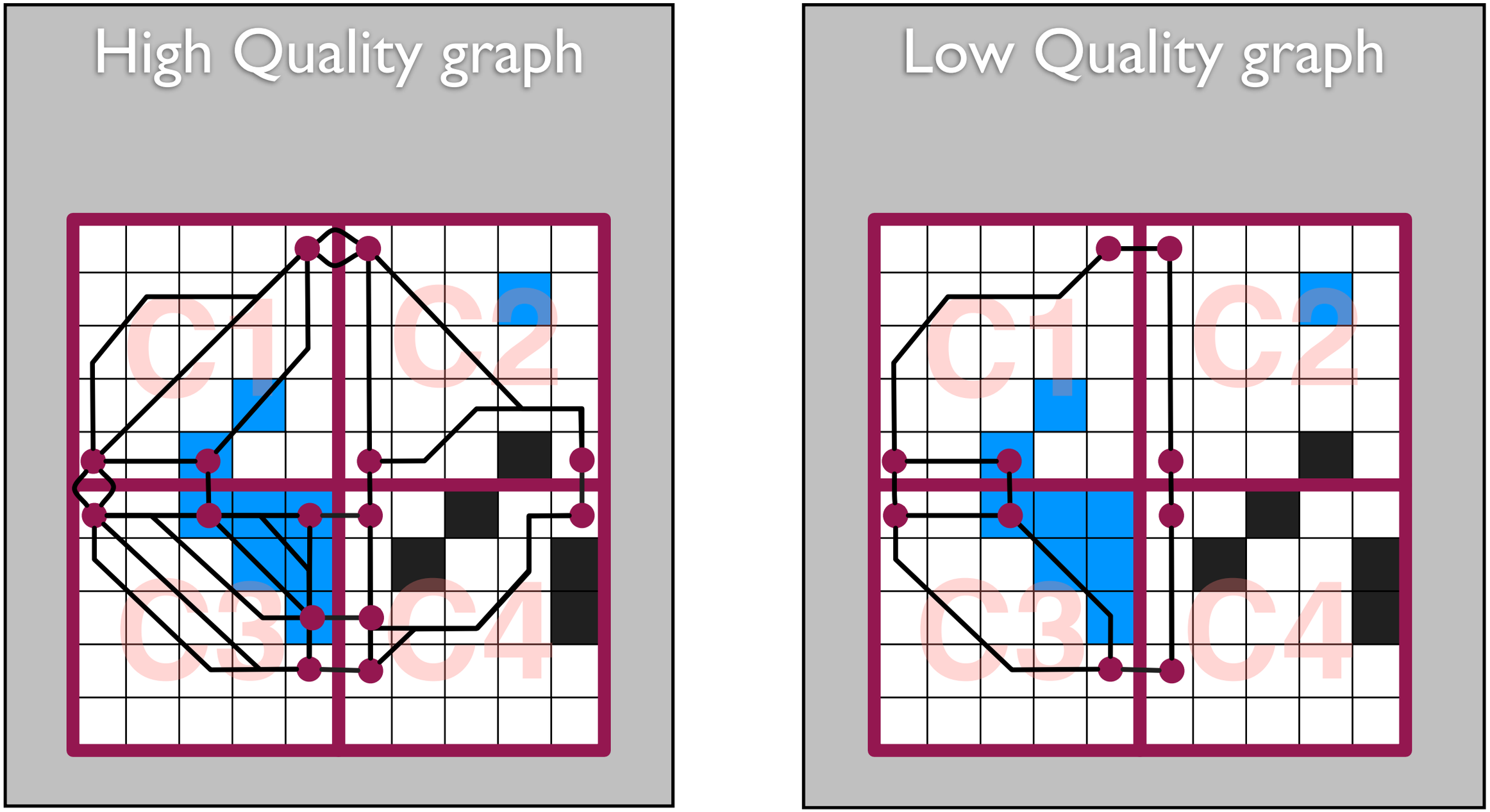
Retain only edges traversable by largest number of agents (aka. prefer freeways where possible)




Two places of interest: U and X.
Path p = {E1, E2, E3}. Length: 15, Max Size = 1, Capability: {Ground, Water}
Path q = {E4, E5, E6}. Length: 17, Max Size = 2, Capability: {Ground}

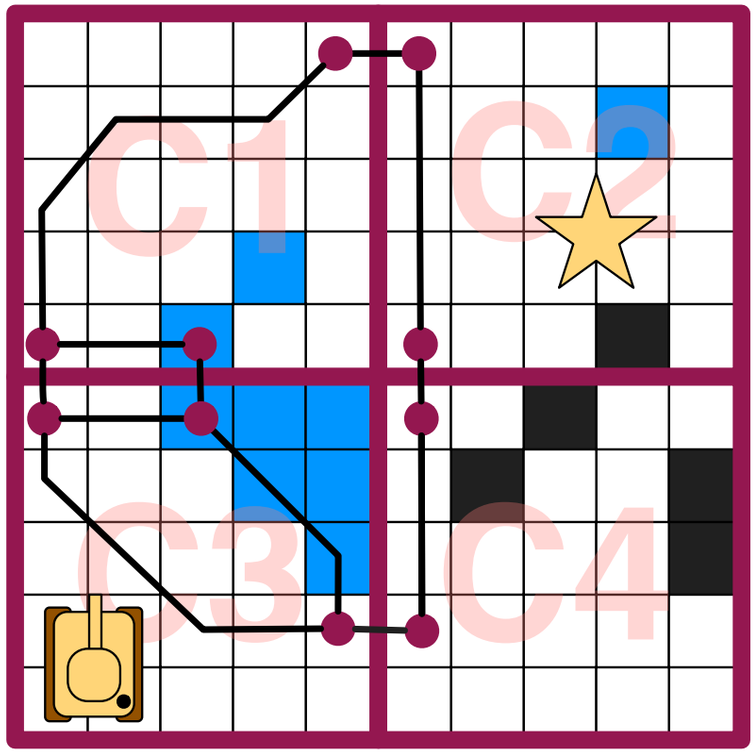
# Weak dominance



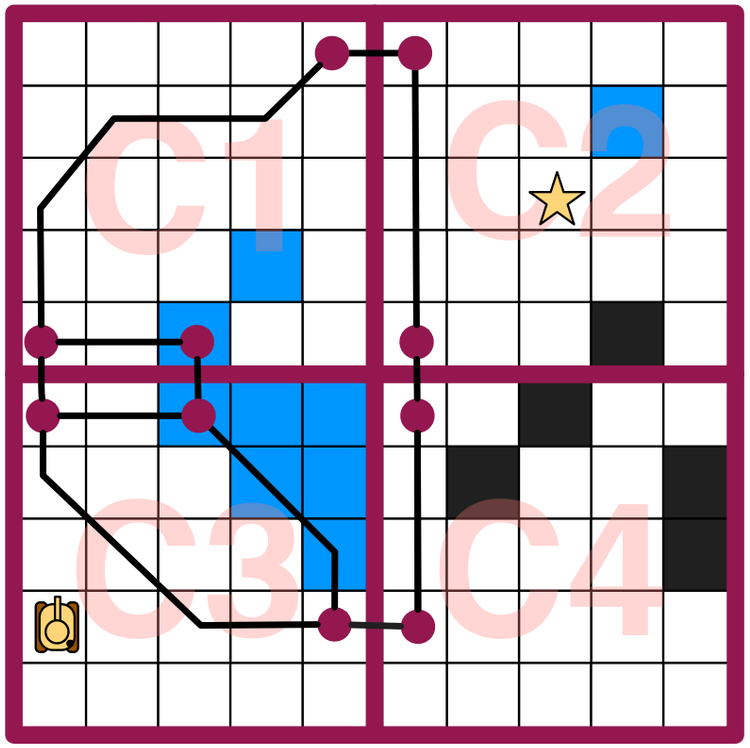

Weak dominance is applied to transitions between adjacent clusters (inter-edges).

# HAA*

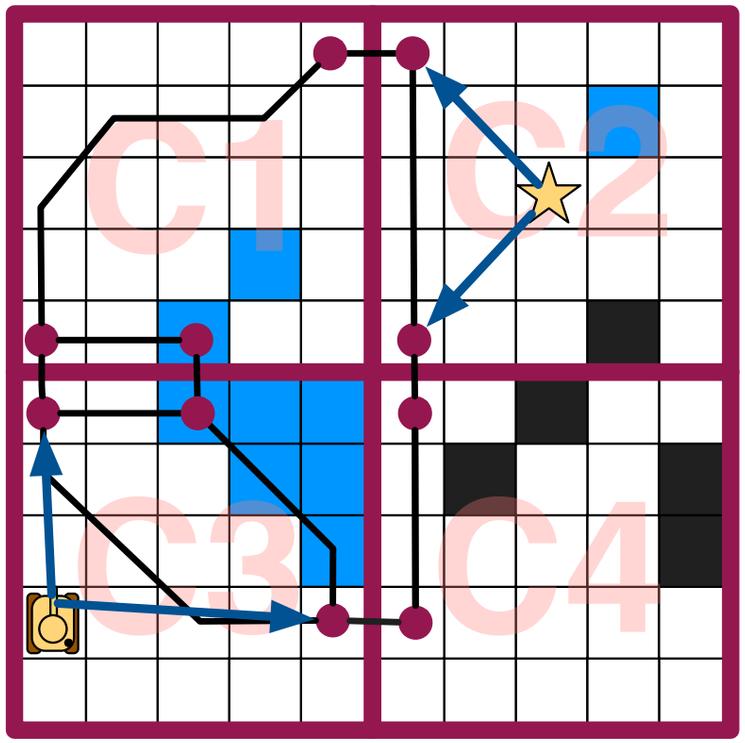
- Similar search process to HPA*
  - Insert Start + Goal
  - Find + refine abstract path
- Differences:
  - Add a node to open list only if reachable according to edge annotations.



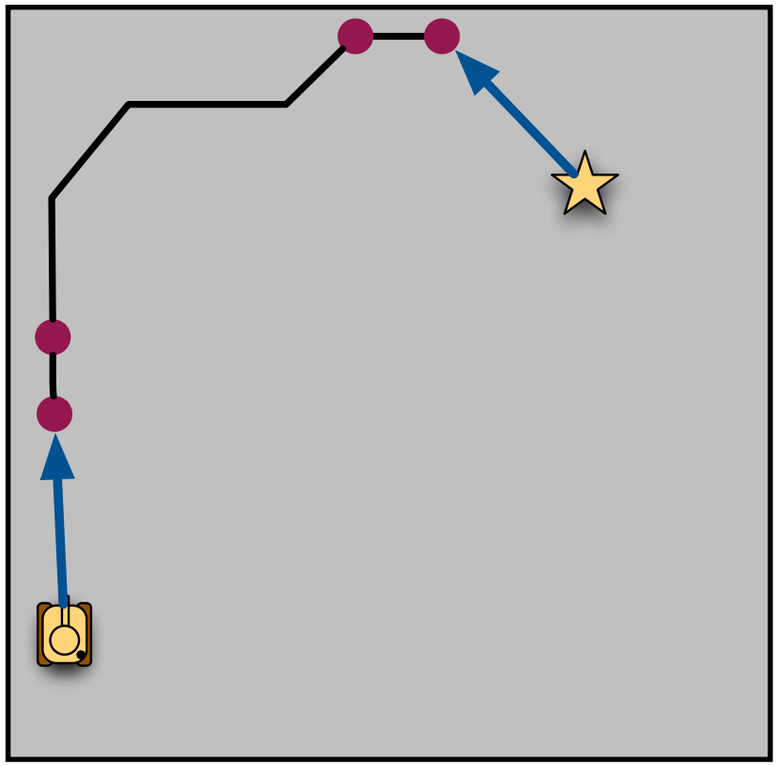

Initial problem
Size = 2, Capability = {Ground}




Reducing the problem




Inserting Start + Goal



Solution

# HAA* Advantages

- Works for units with different sizes and capabilities.
- Complete and near-optimal (within 4-8% from optimal on average)
- Fast (similar performance to HPA*)
- Low overhead (storing abstract graph requires little memory in practice)

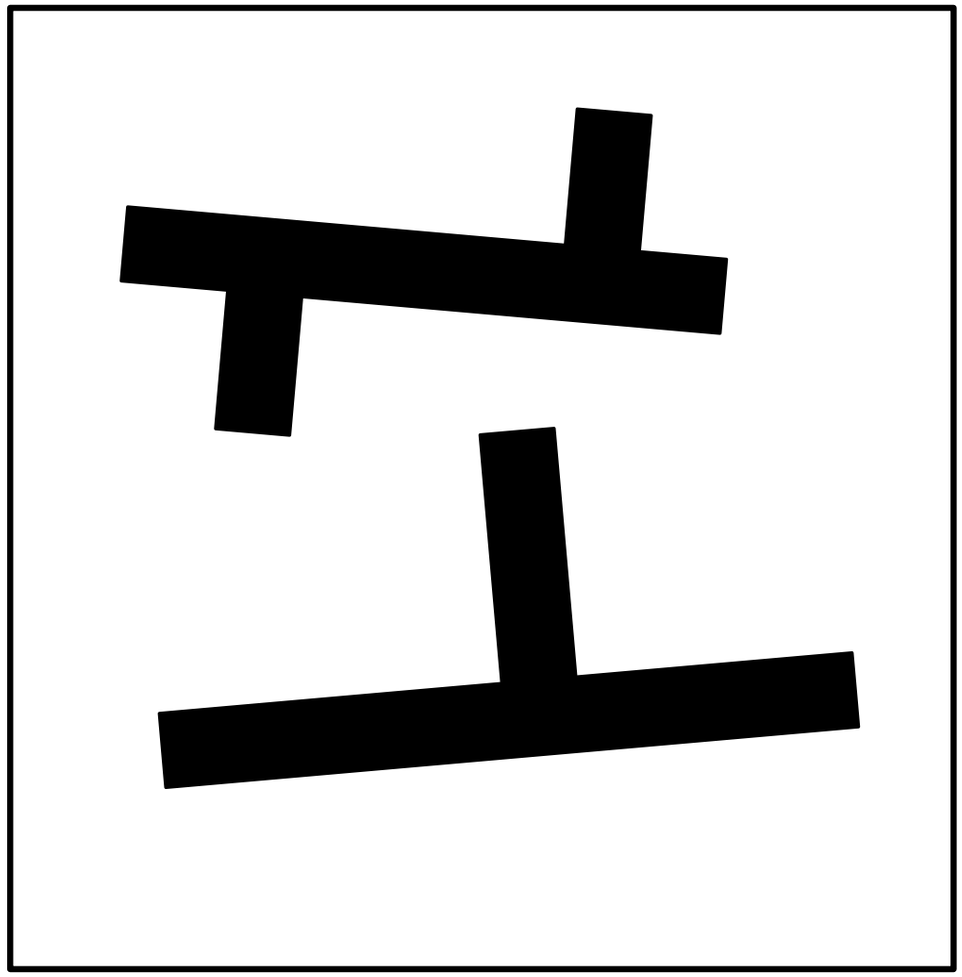
# HAA* Disadvantages

- Only works on grids.
- Need to model units using square bounding volumes.
- Insertion effort.

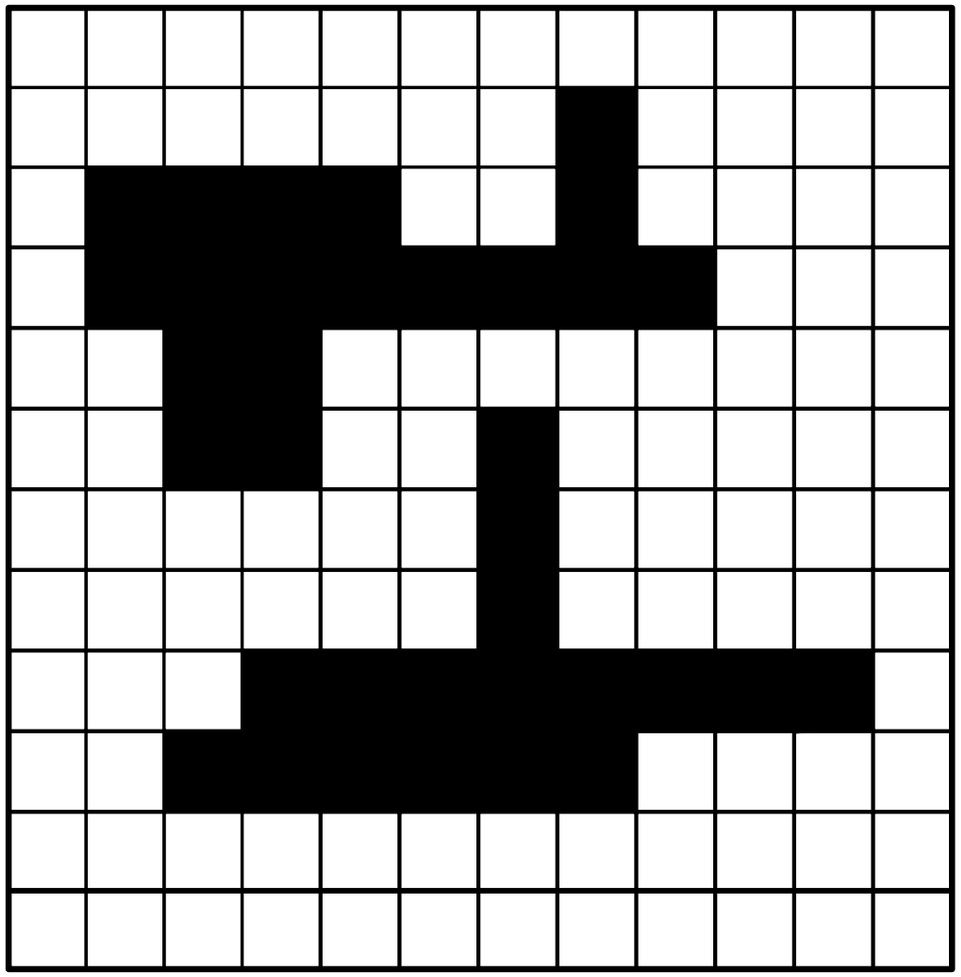
# Part 3:
# Probabilistic Roadmaps

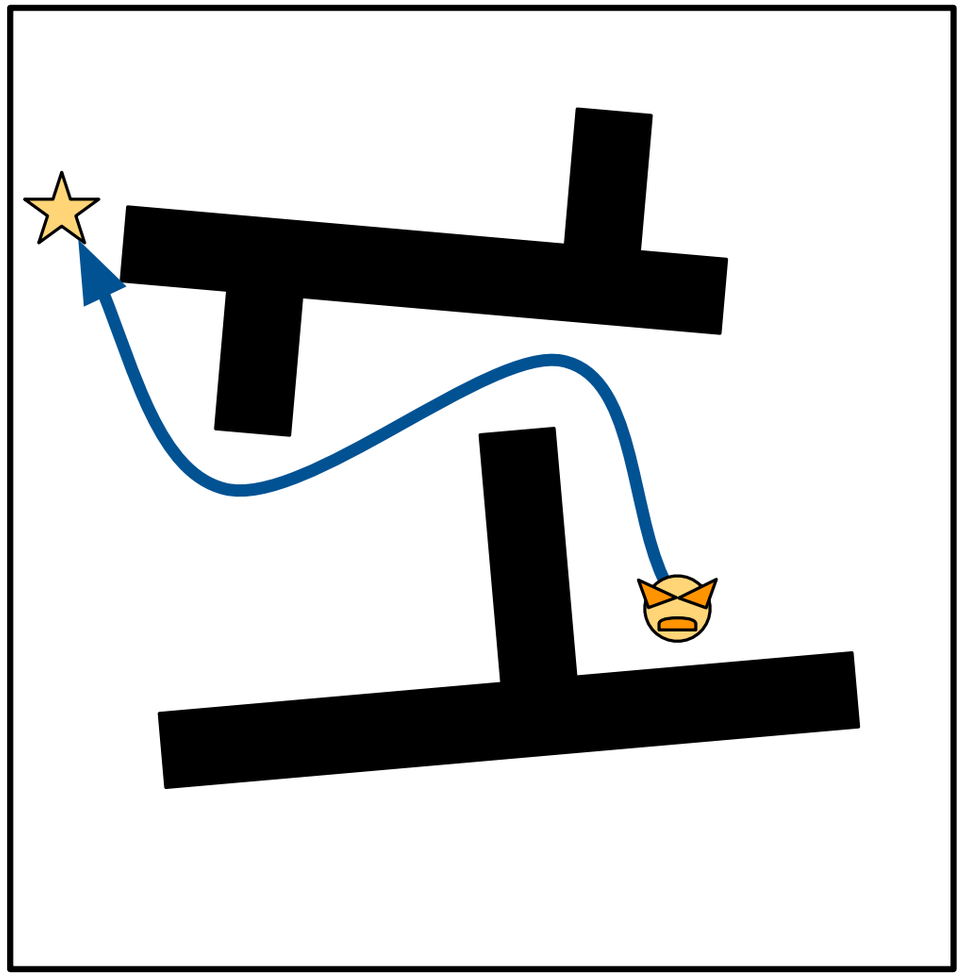
# Limitations of grid methods

- Coarse coverage of underlying terrain.
- Restricted movement (fixed angles)
- Paths don’t look realistic
- Often need to apply smoothing
- Not very useful in 3D spaces

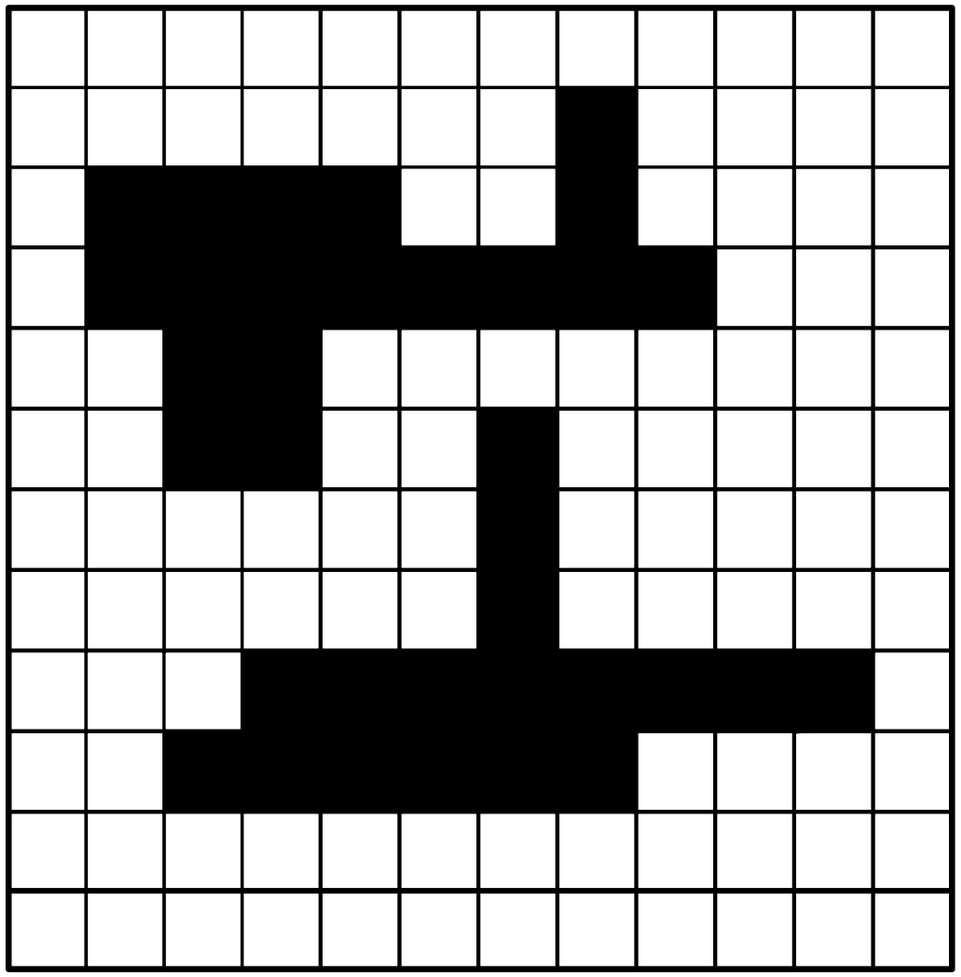


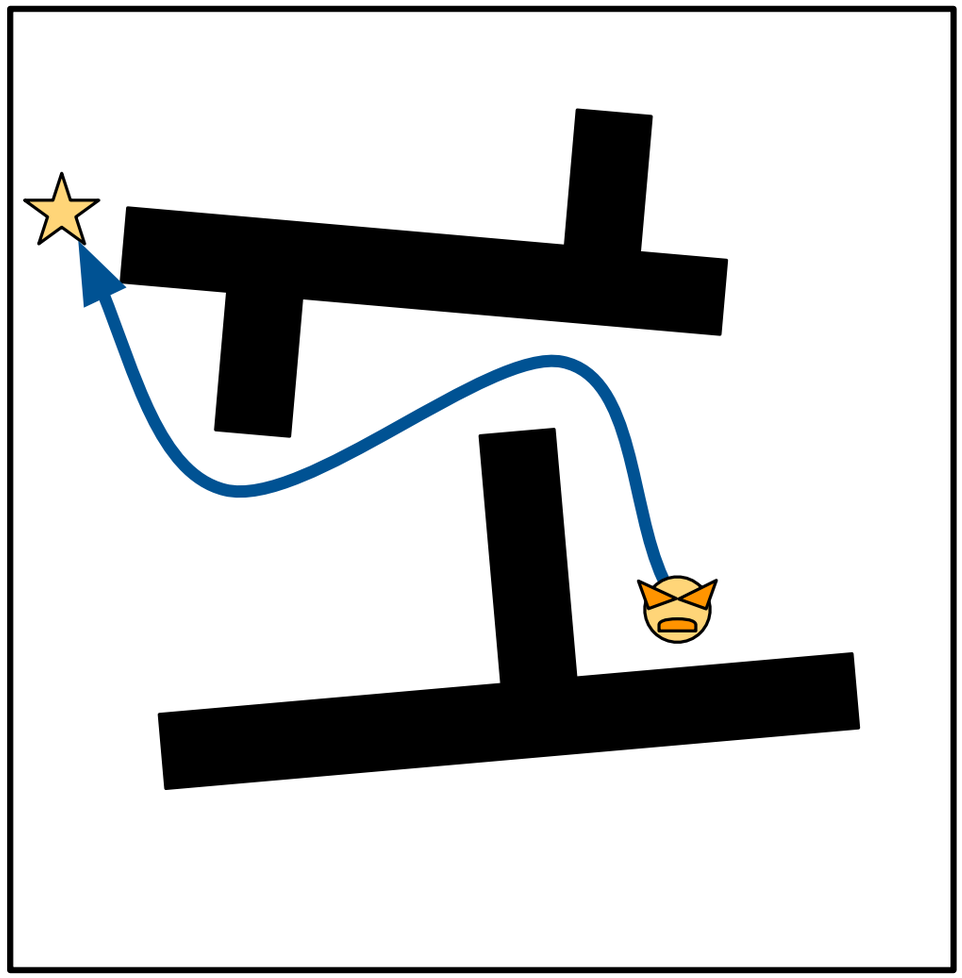
vs.



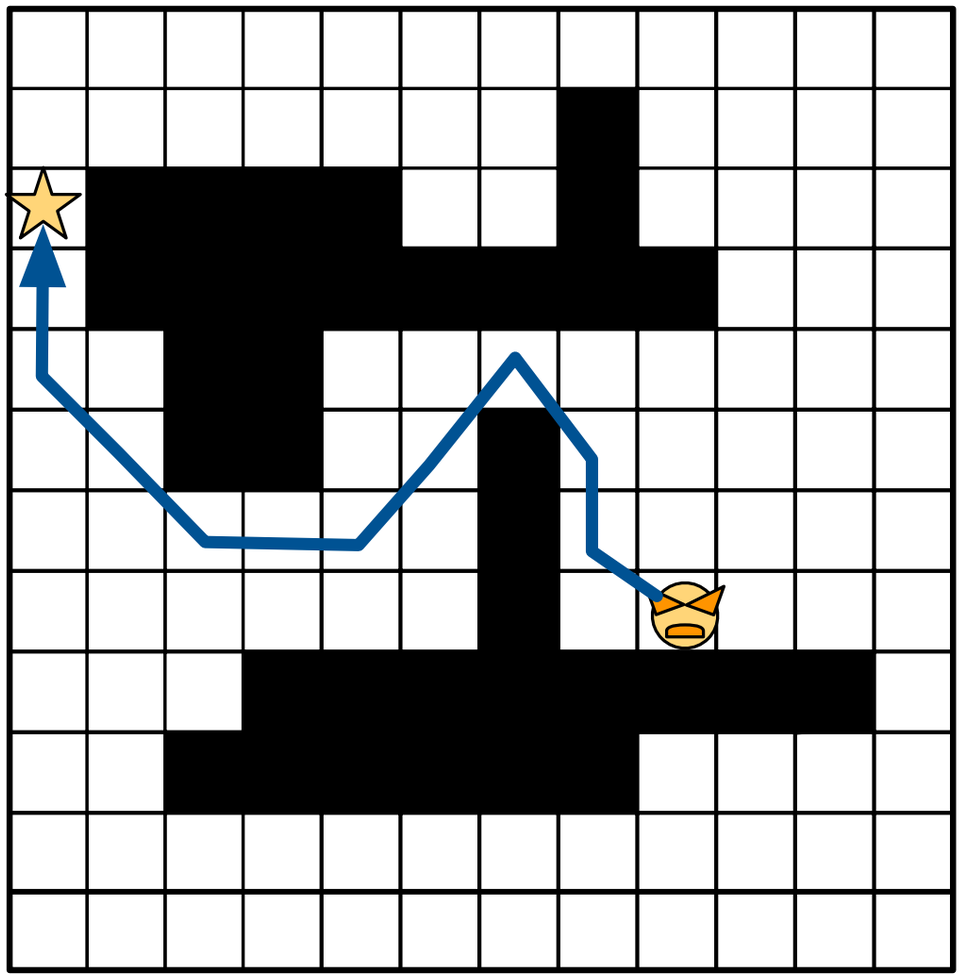


vs.



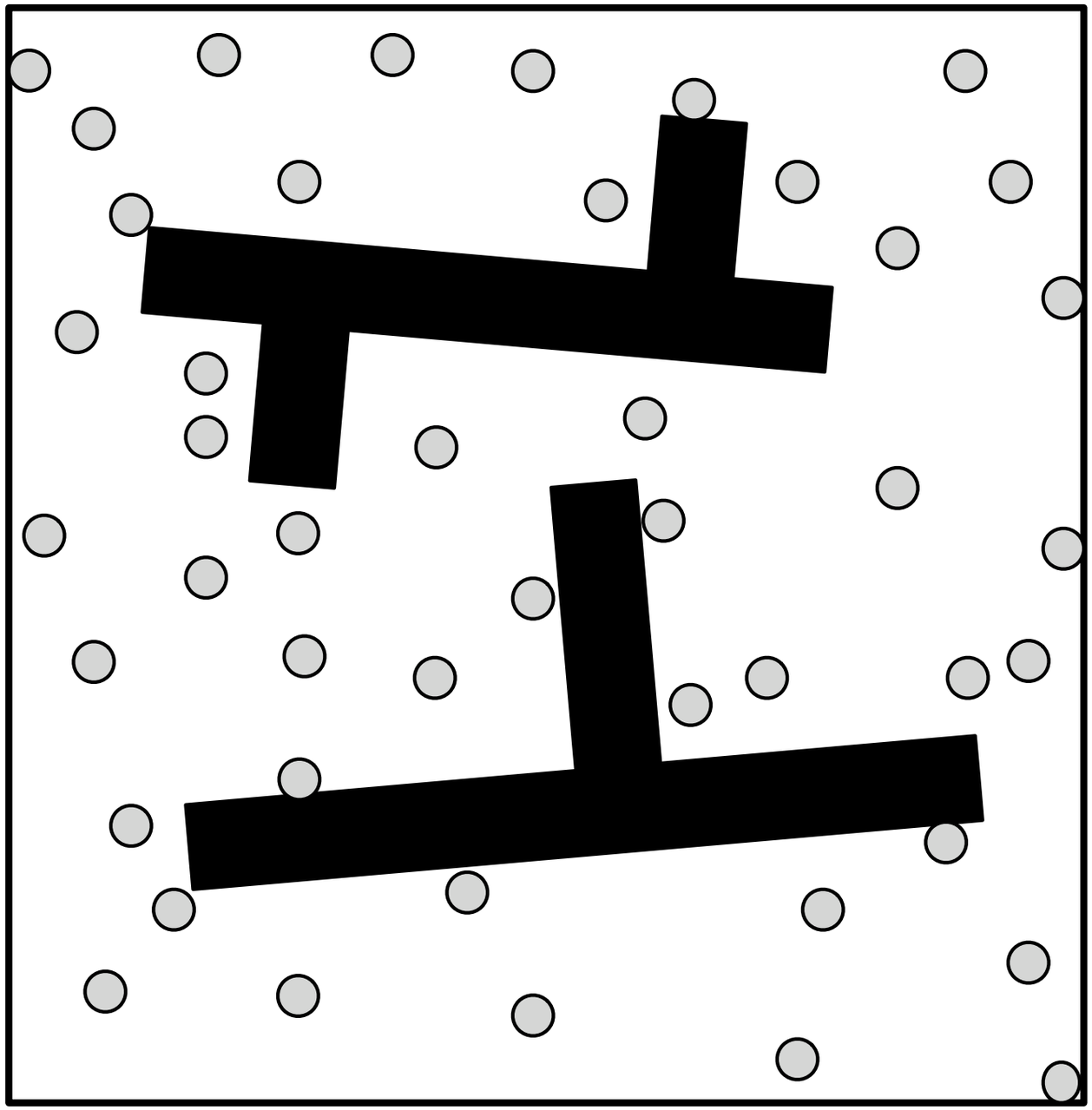


vs.

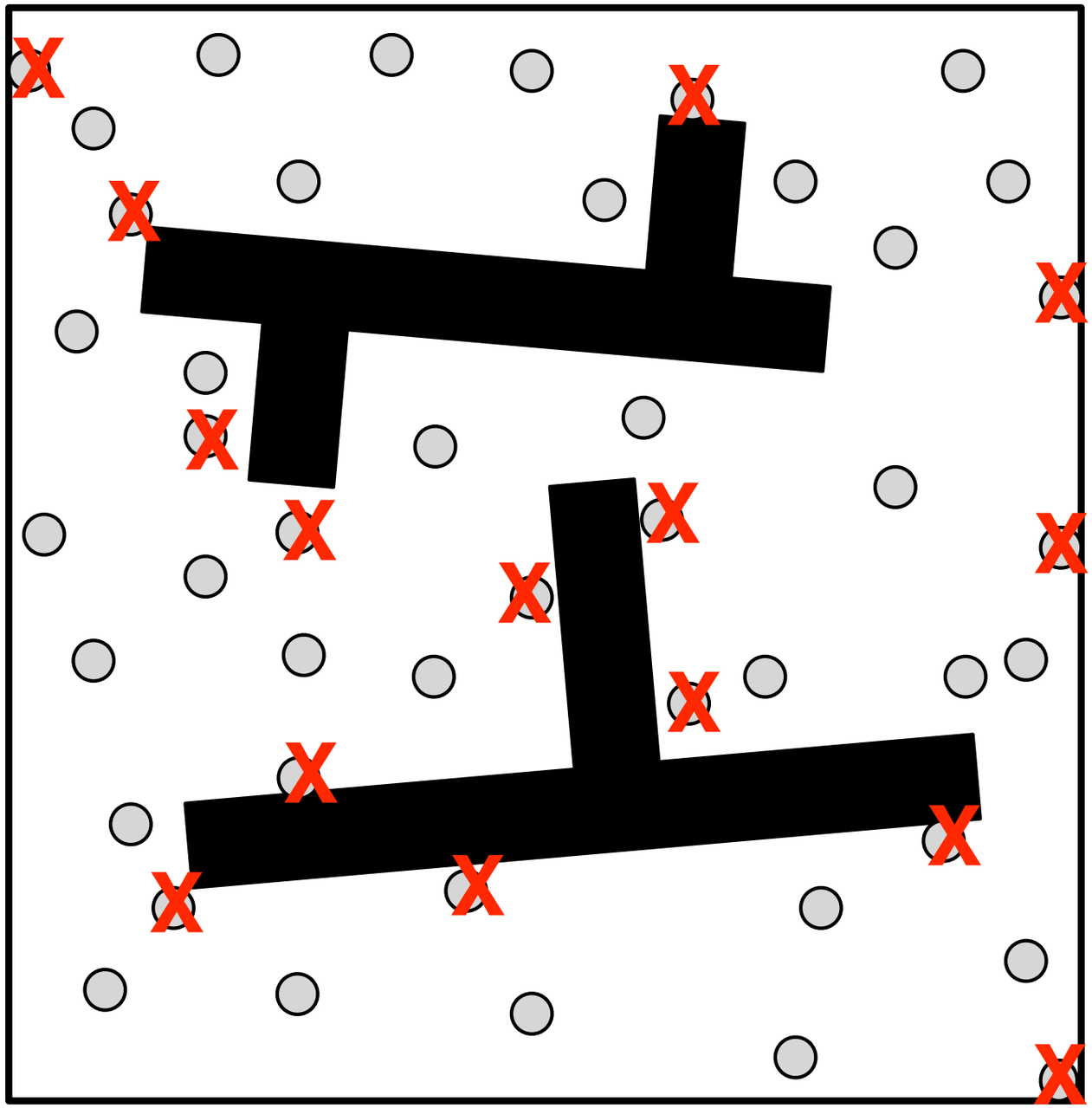


# Probabilistic Road Maps

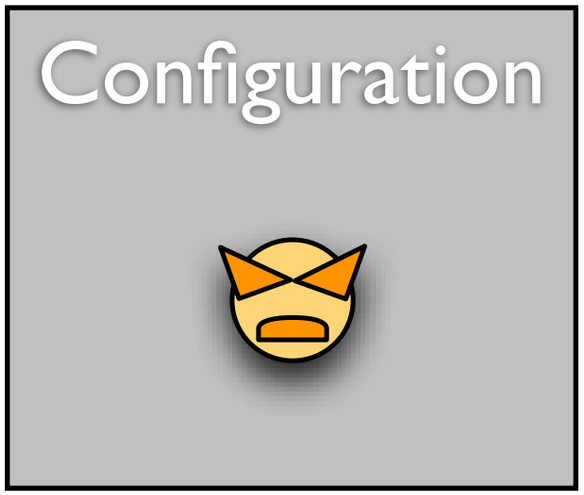
- PRM: Robotics inspired method from mid 90s.
  e.g. see [Choset et al, 2004].
- Pick a robot configuration.
- Build a graph by taking lots of sample points.

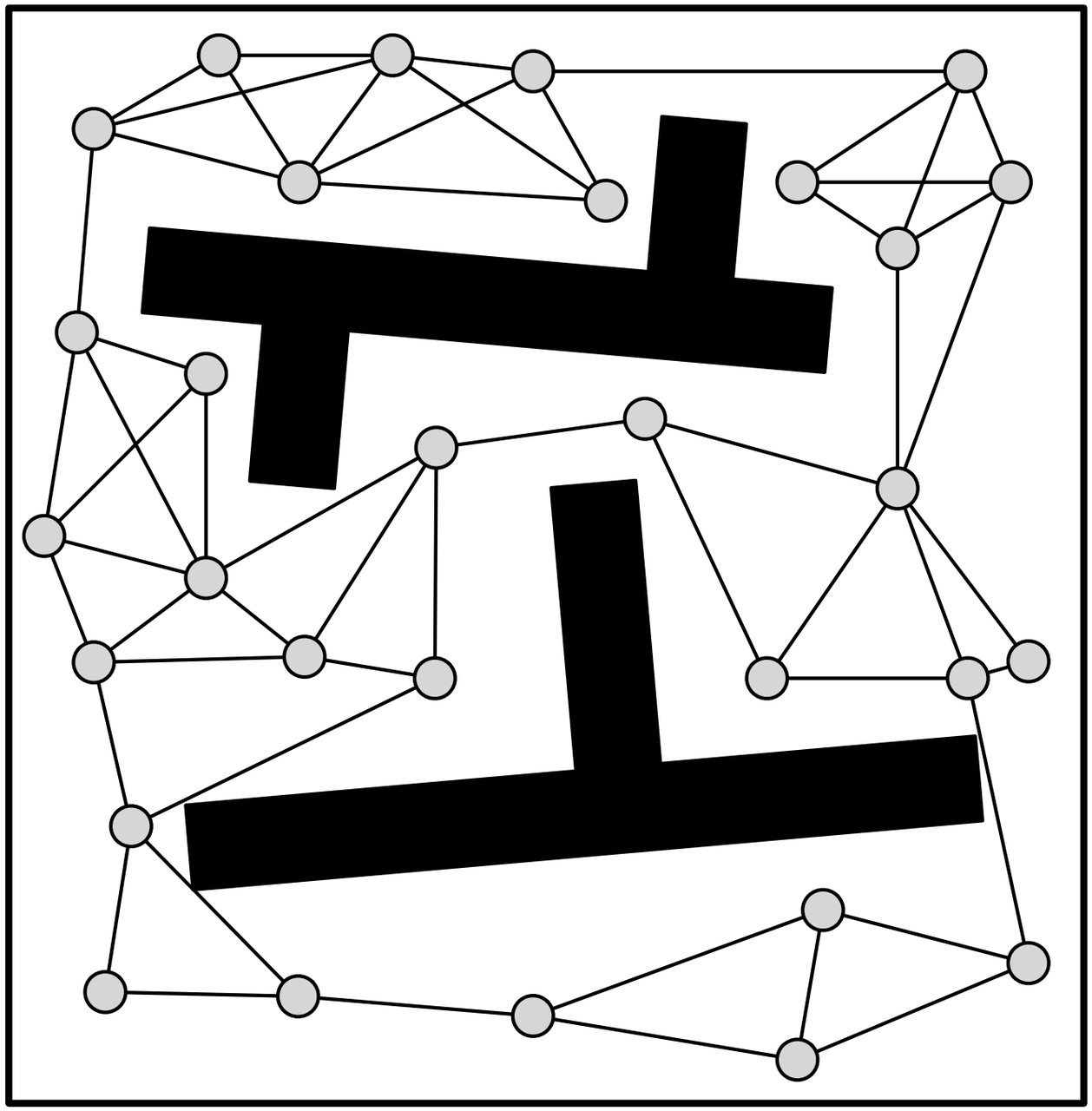


Select non-obstacle sample points from the map

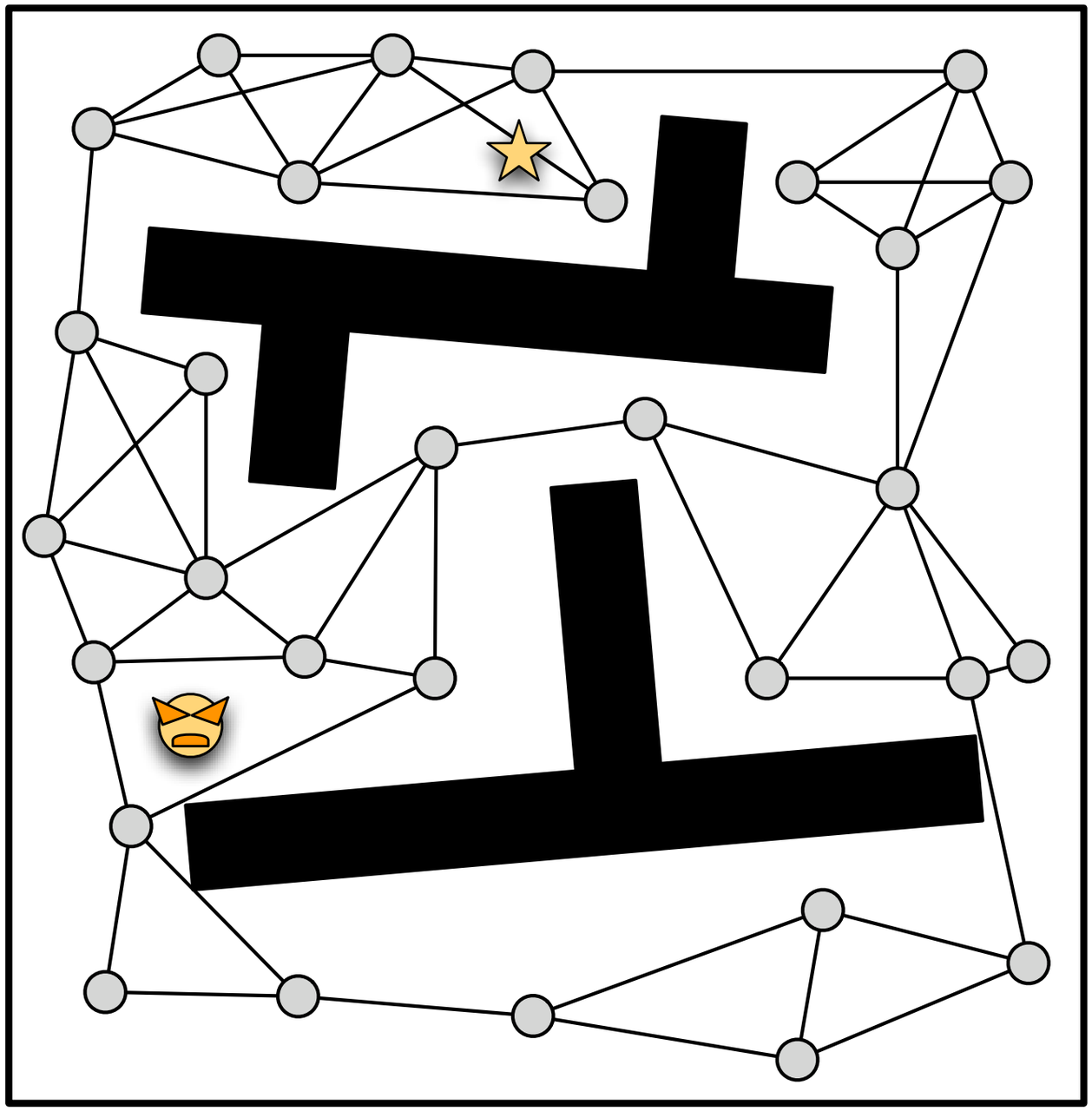


Retain collision-free points for some configuration of an agent

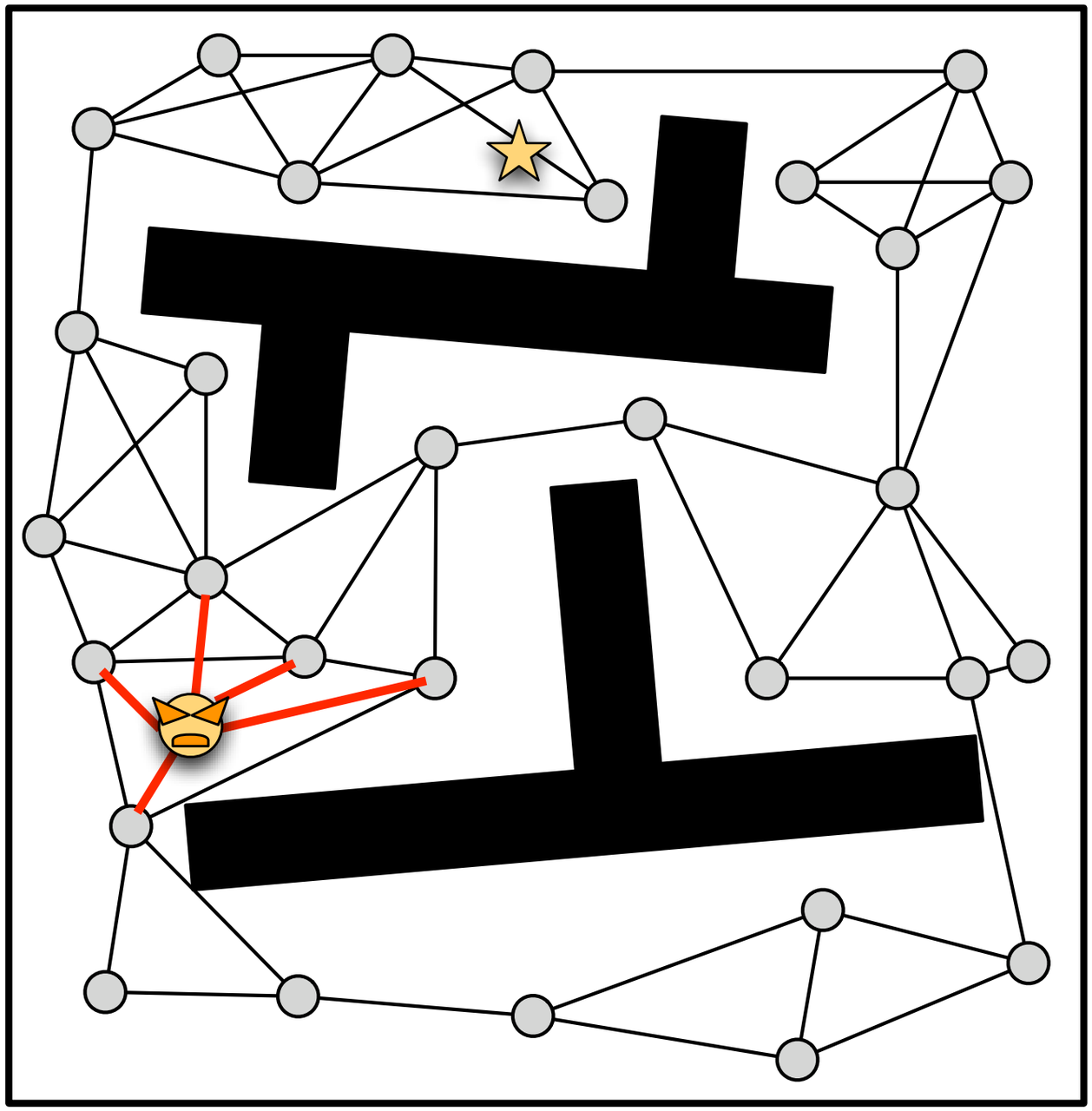




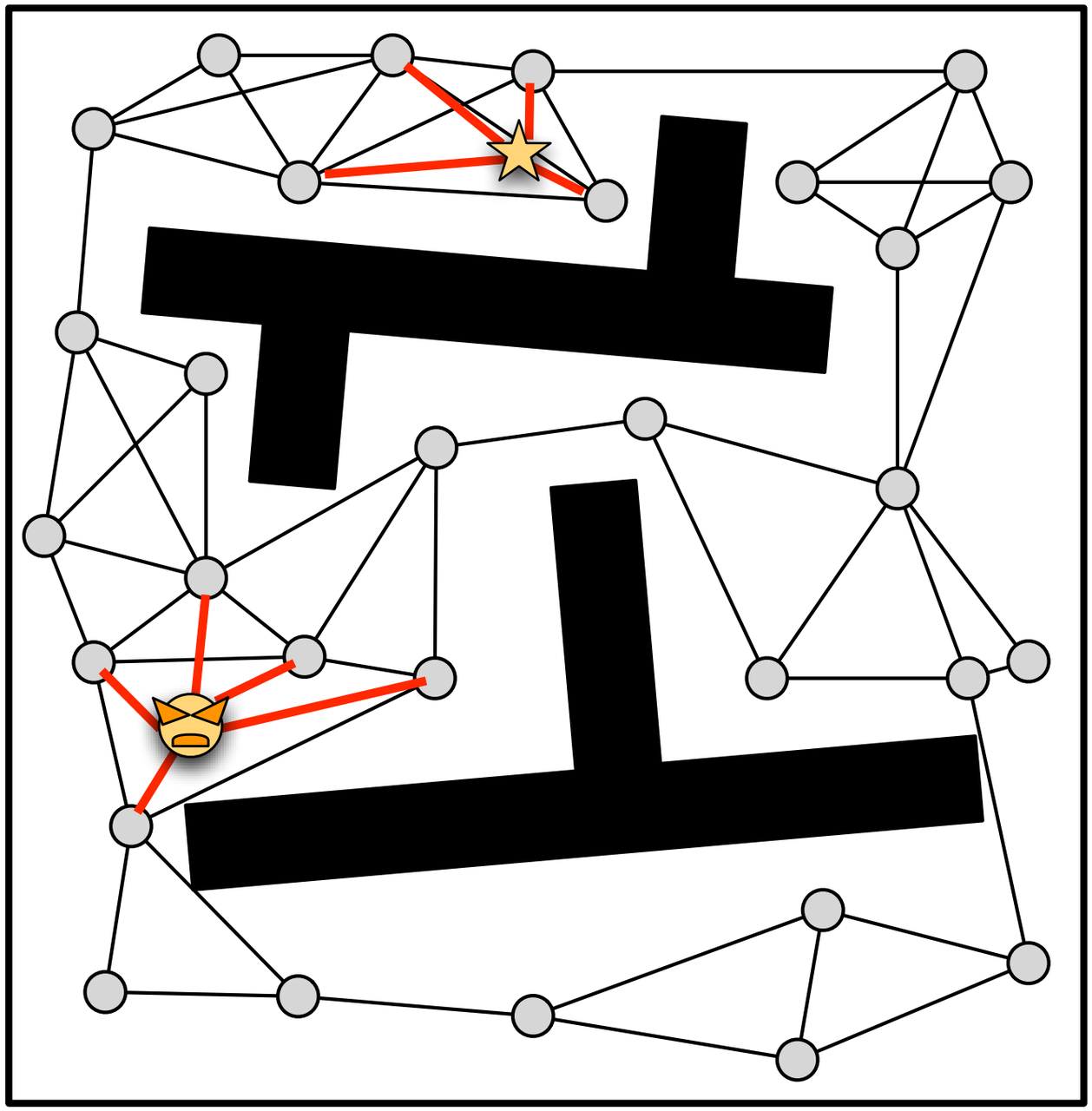
Connect nearby points to form a graph



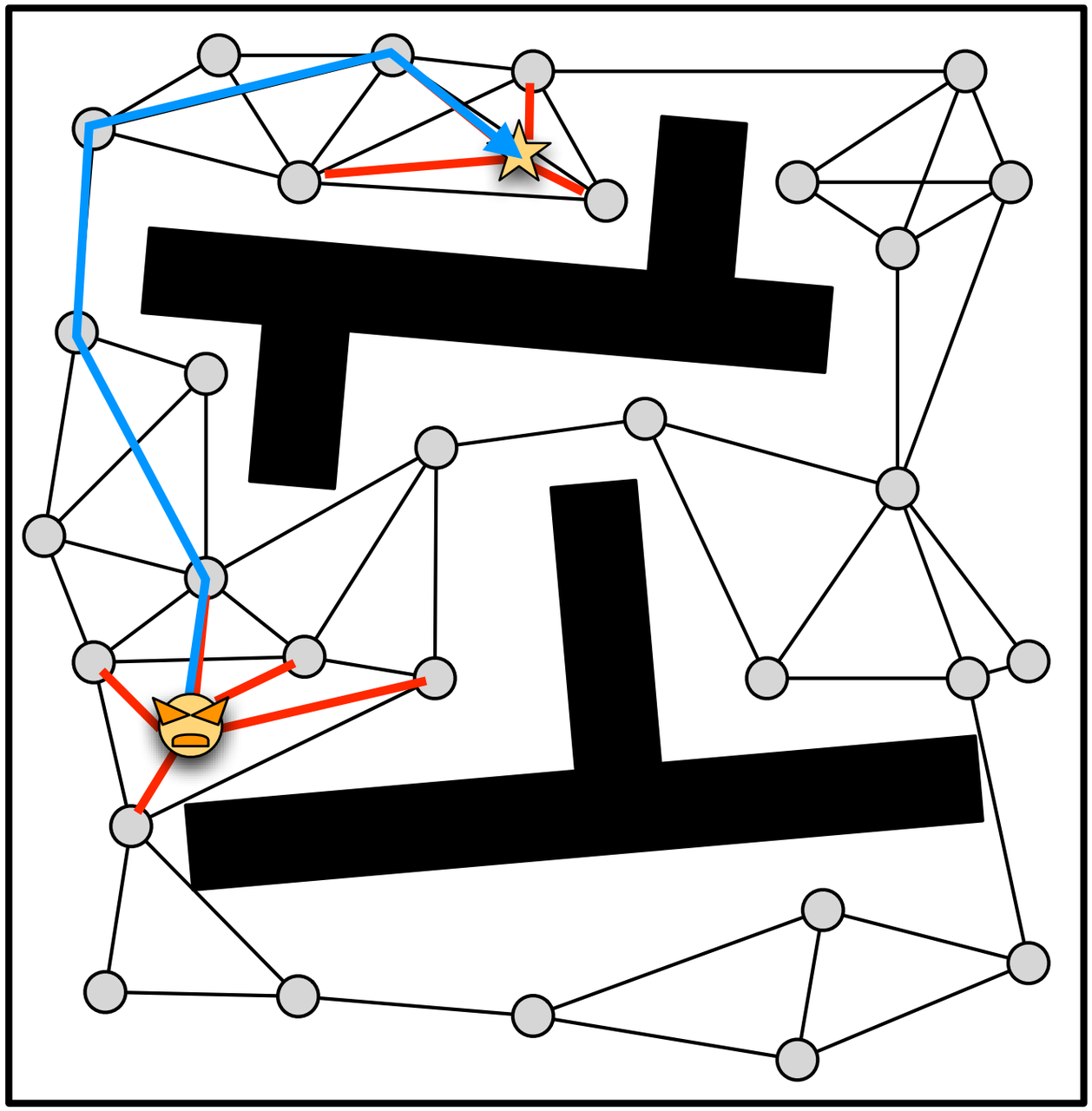
Pathfinding: Connect start and goal to rest of roadmap



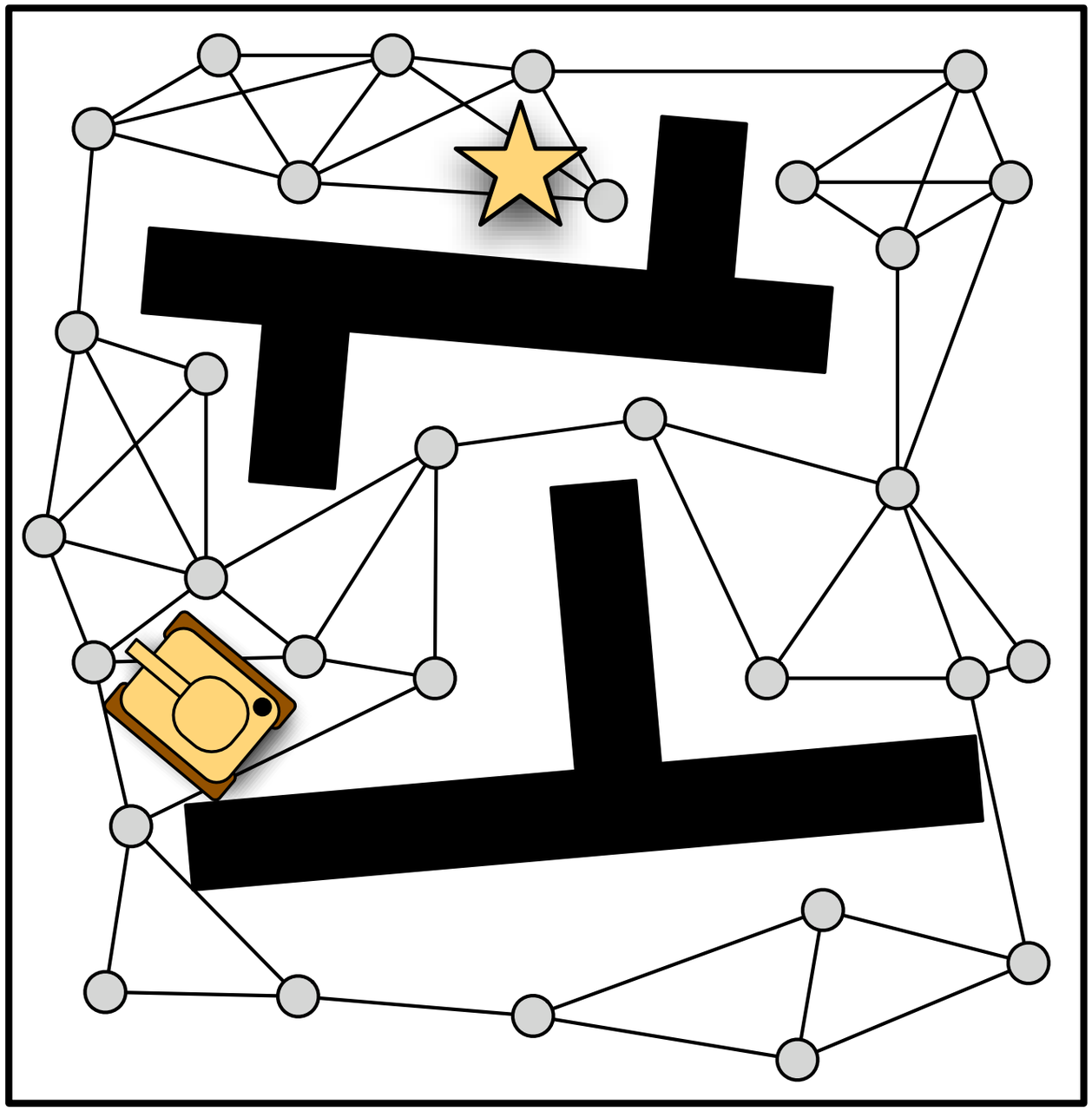
Pathfinding: Connect start and goal to rest of roadmap



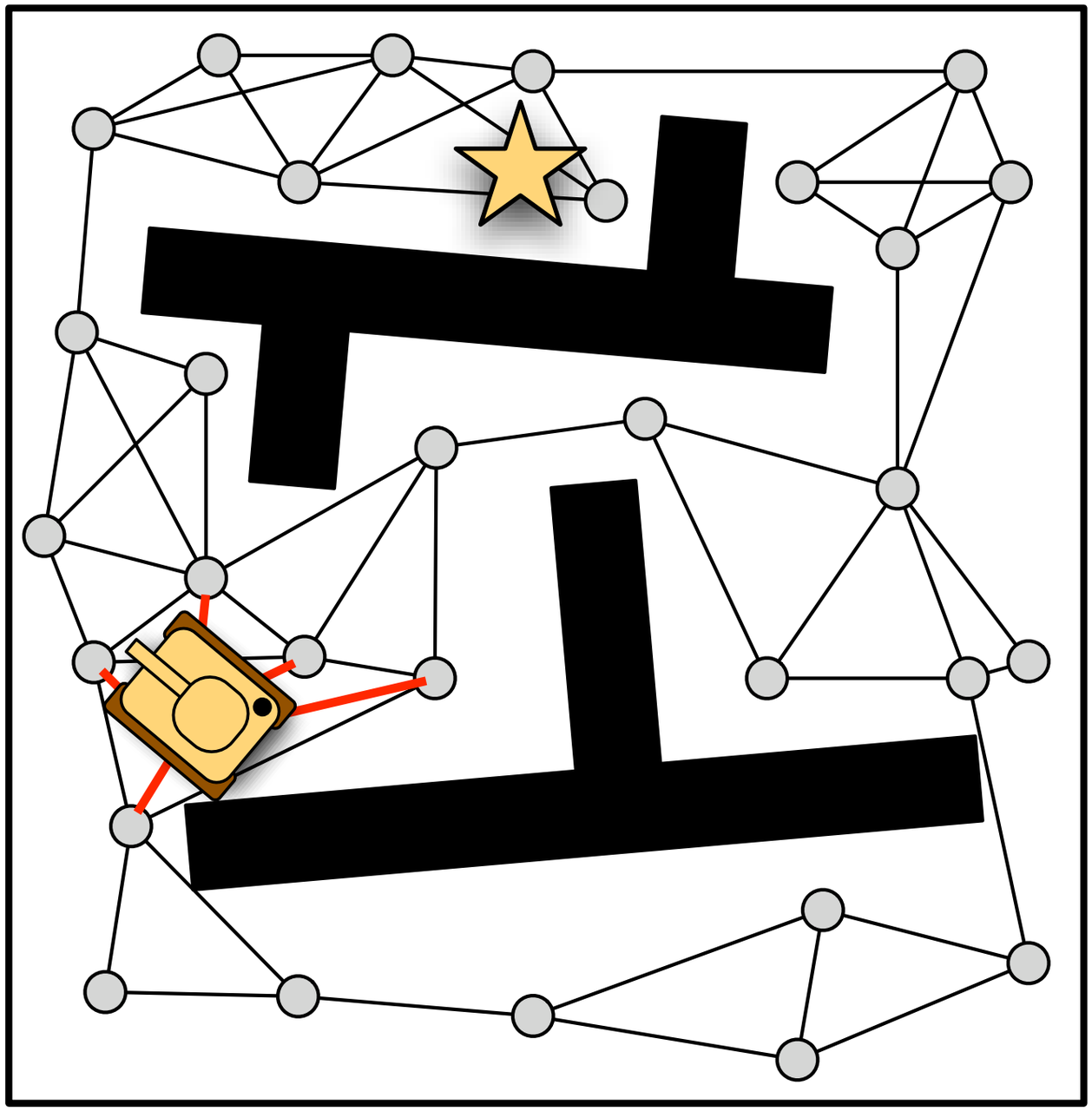
Pathfinding: Connect start and goal to rest of roadmap



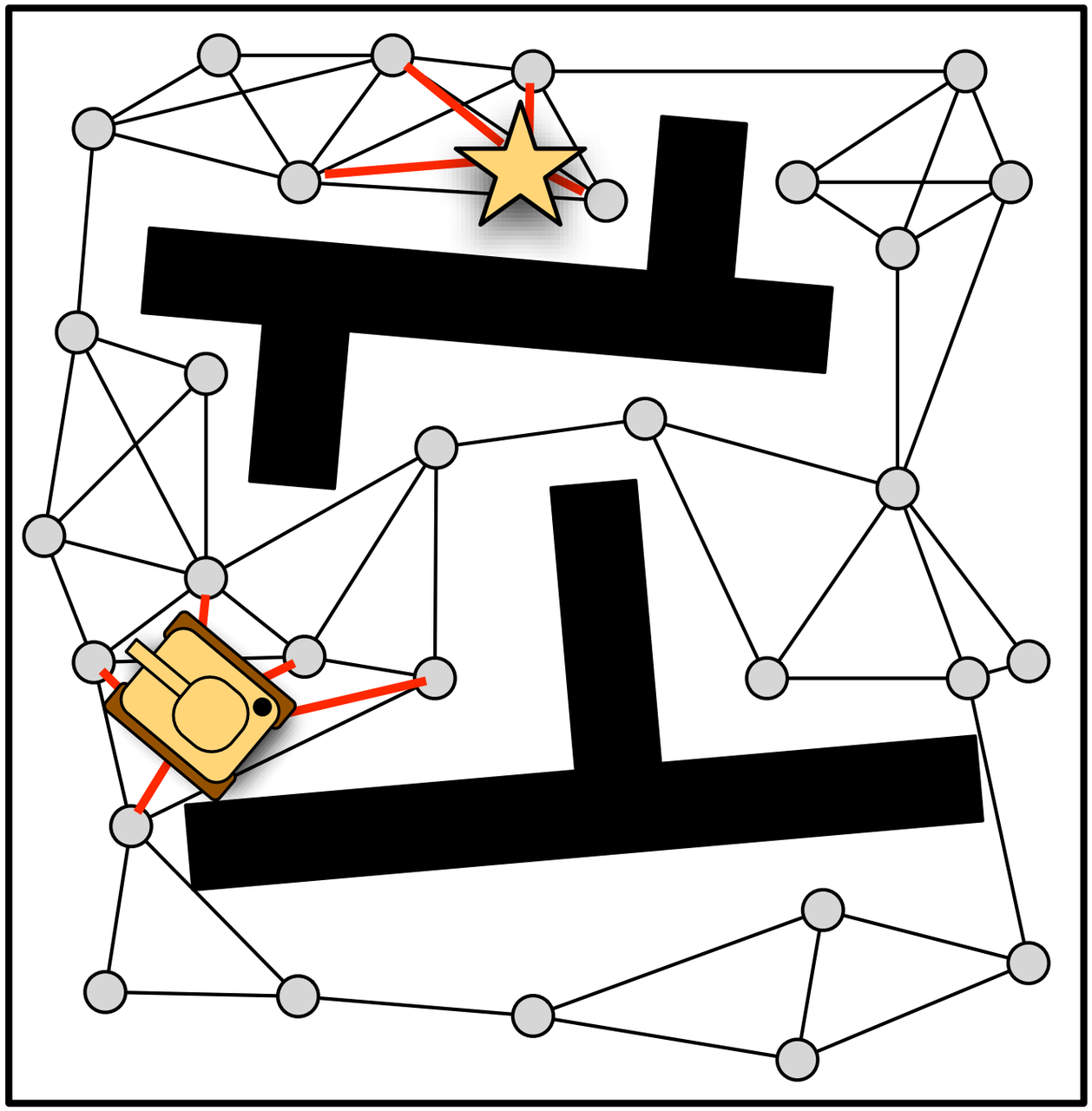
Pathfinding: Connect start and goal to rest of roadmap



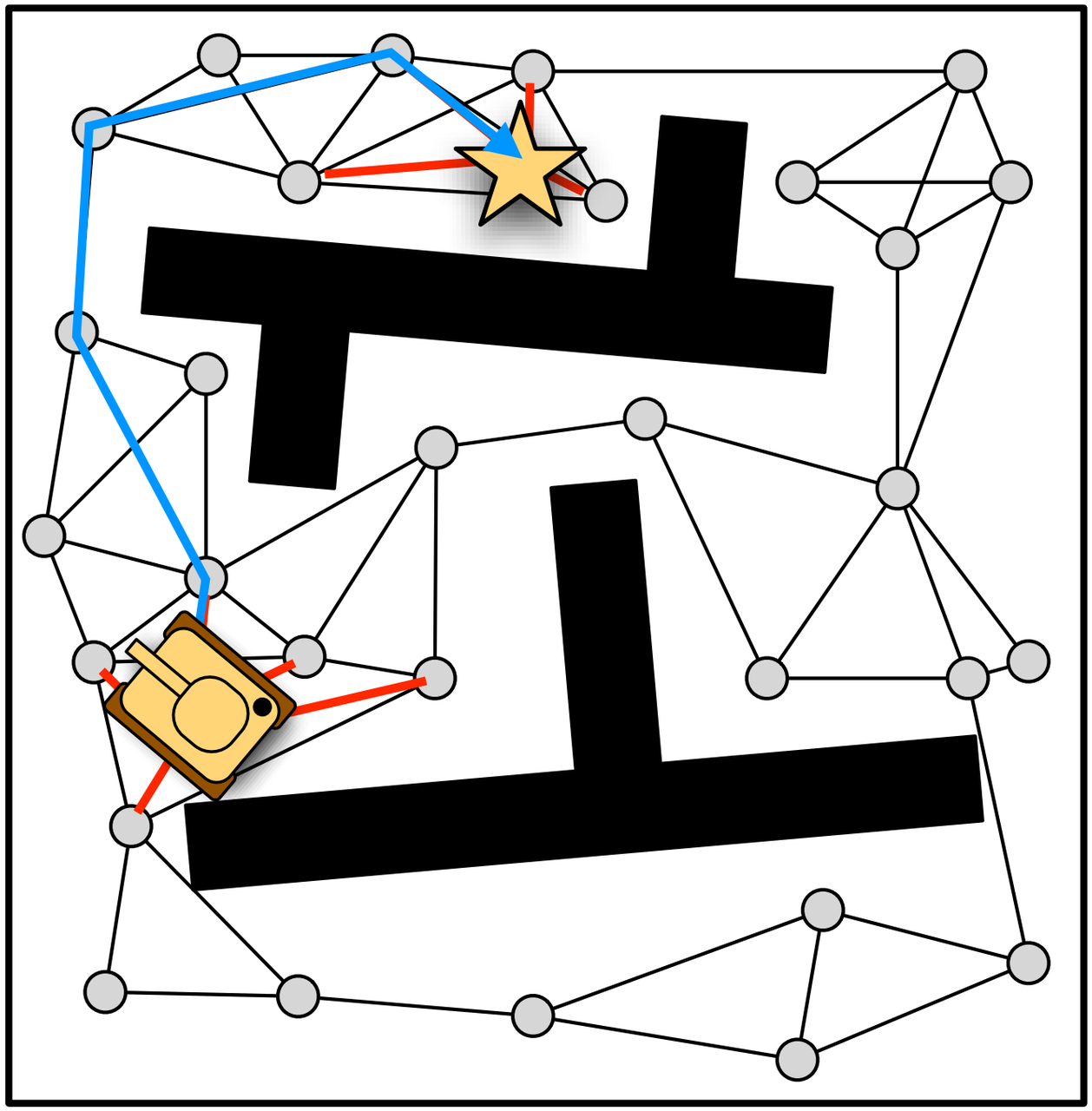
Most PRMs are only useful for agent/configuration used during construction.



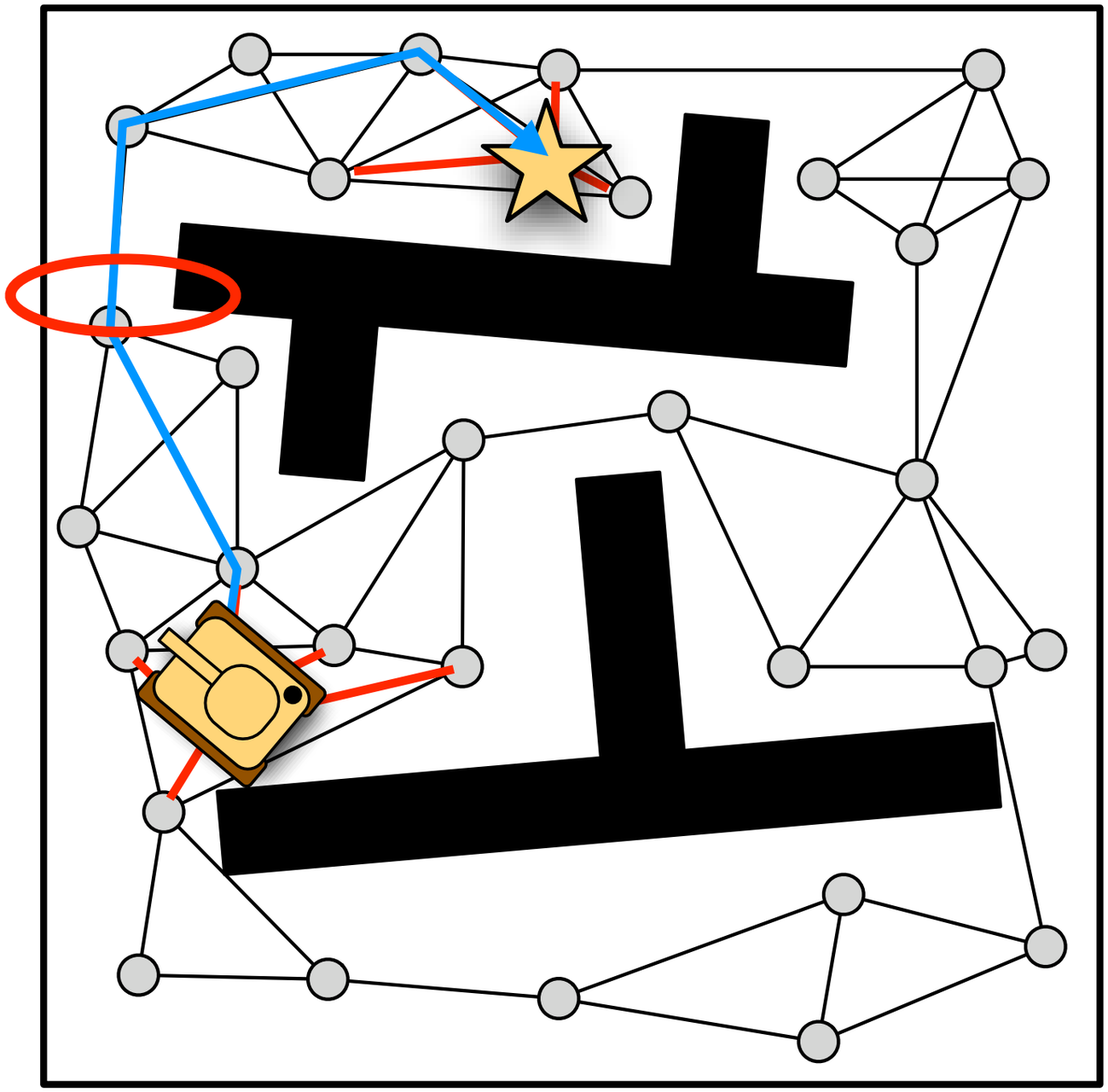
Most PRMs are only useful for agent/configuration used during construction.



Most PRMs are only useful for agent/configuration used during construction.



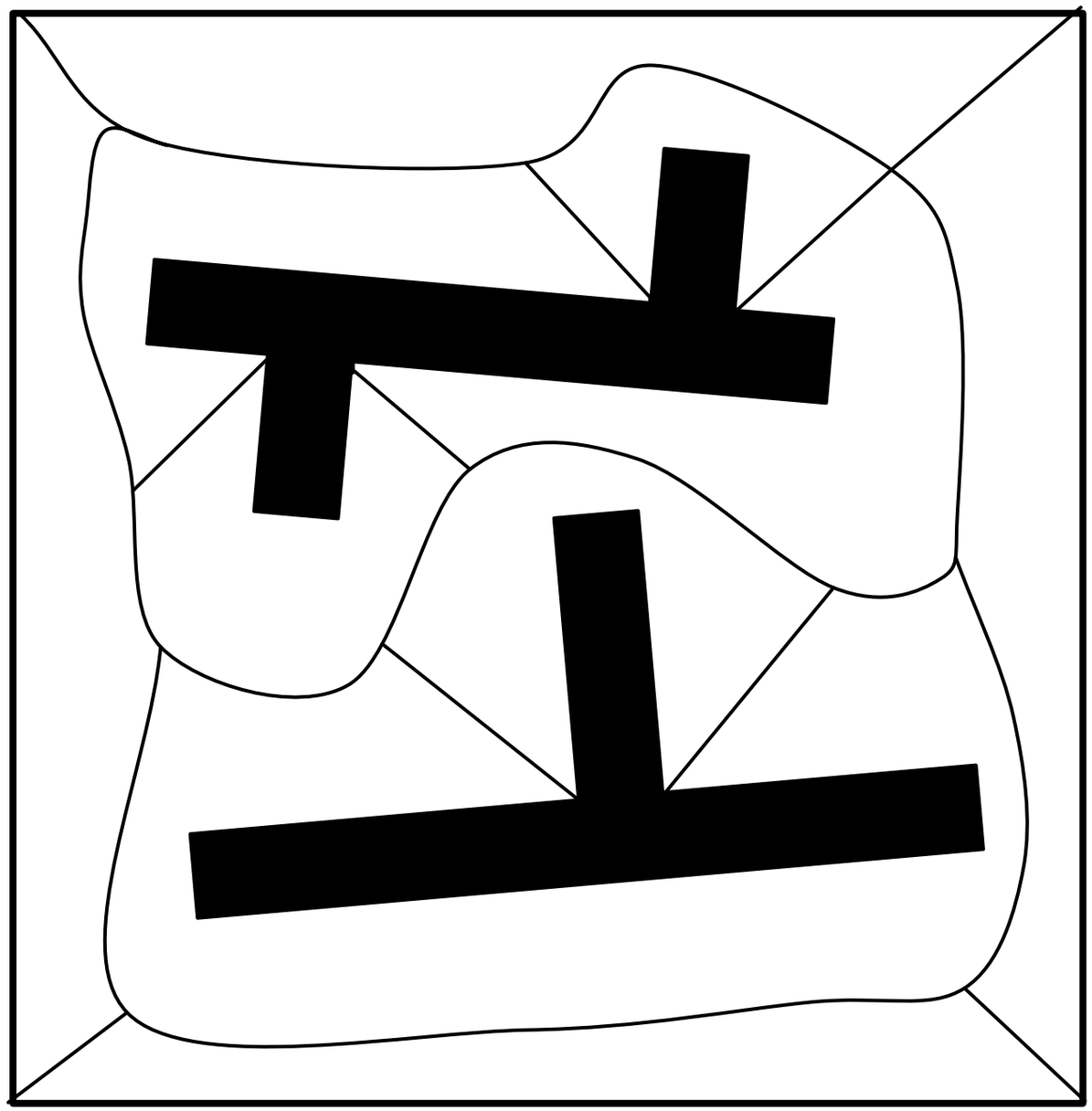
Most PRMs are only useful for agent/configuration used during construction.



Most PRMs are only useful for agent/configuration used during construction.

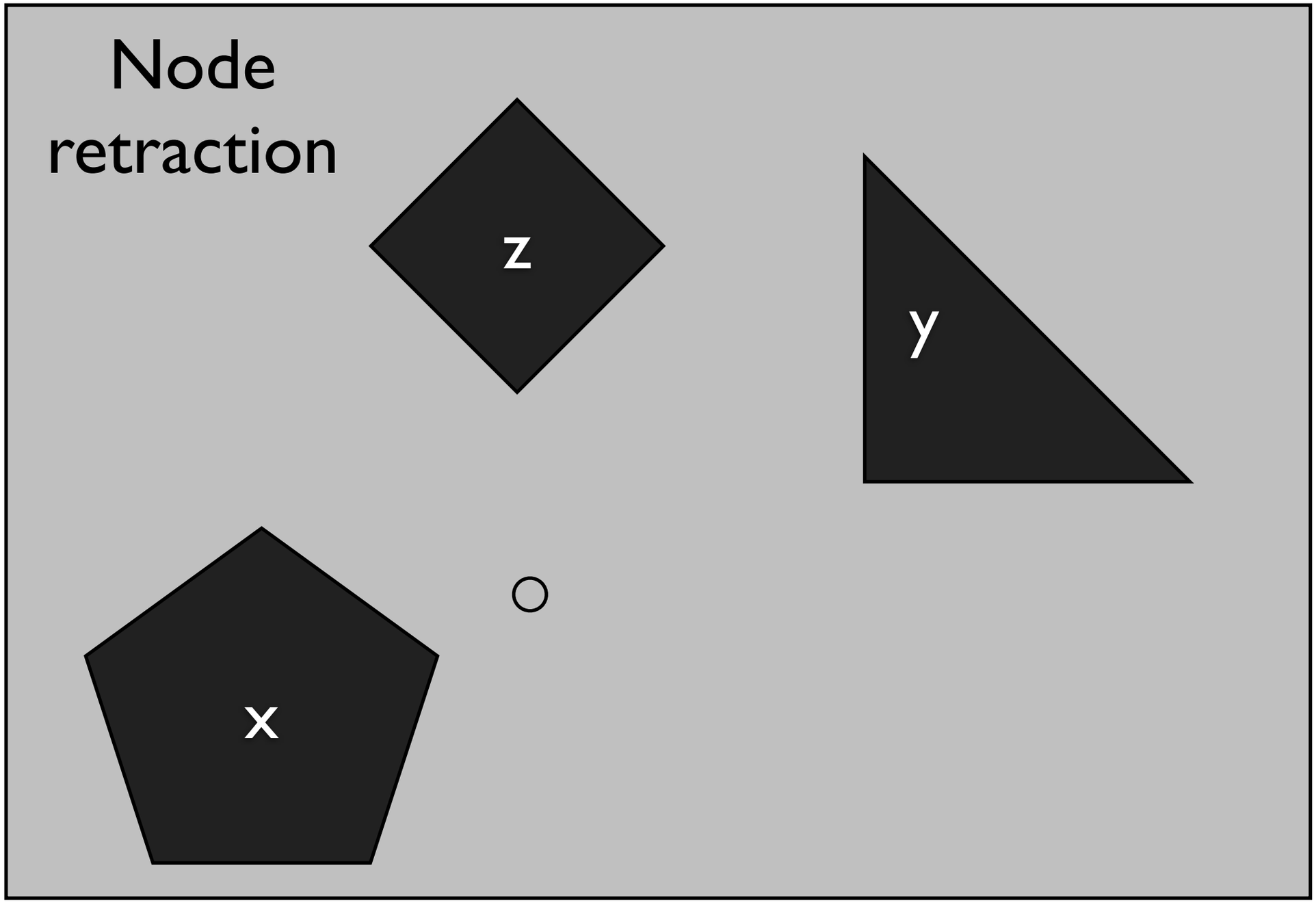
2 options to fix this problem:

- Create a PRM for each unit type
- Opt for a (generalised) Voronoi-based PRM

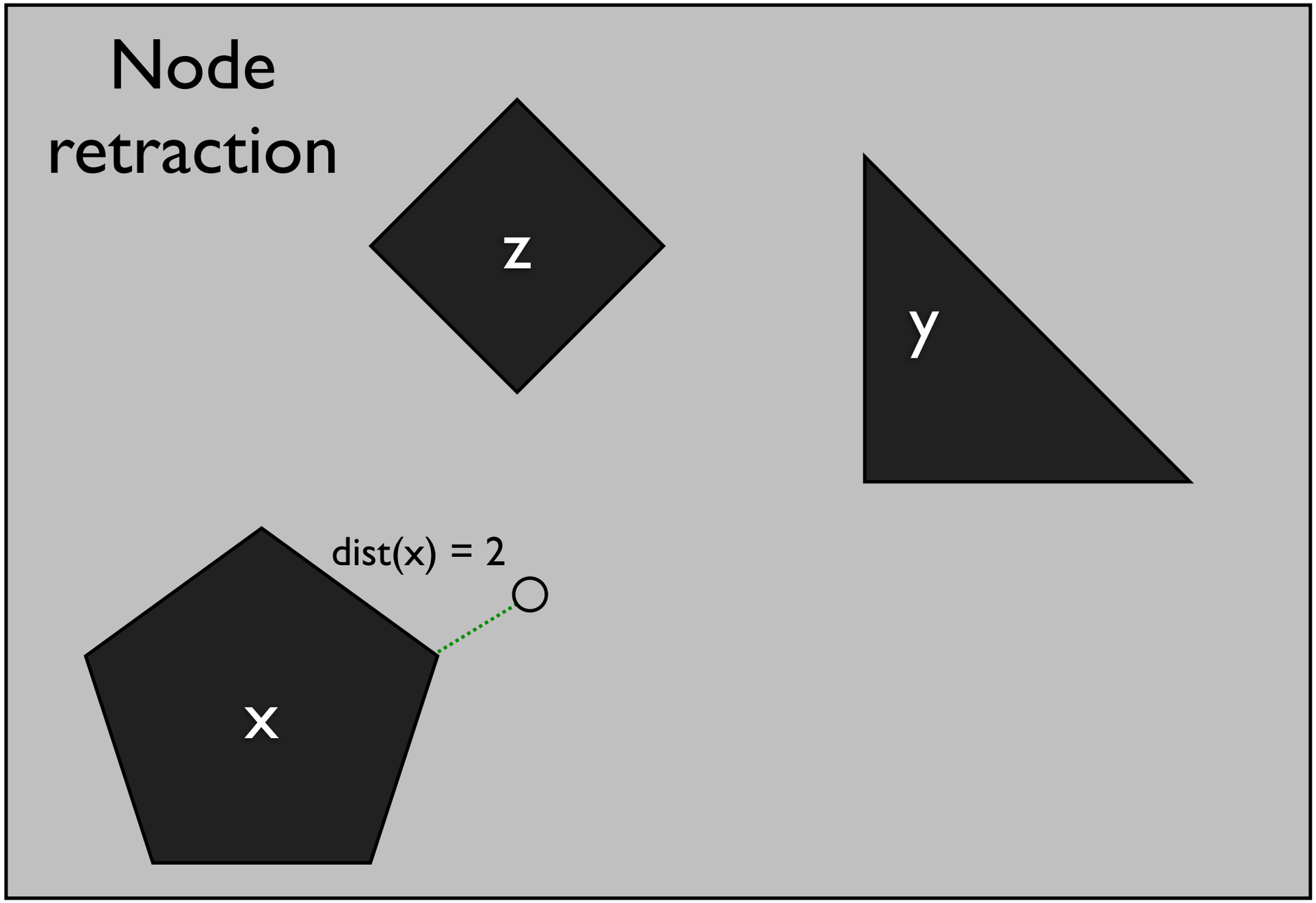


Voronoi diagram: Each point on the medial axis maximises clearance.

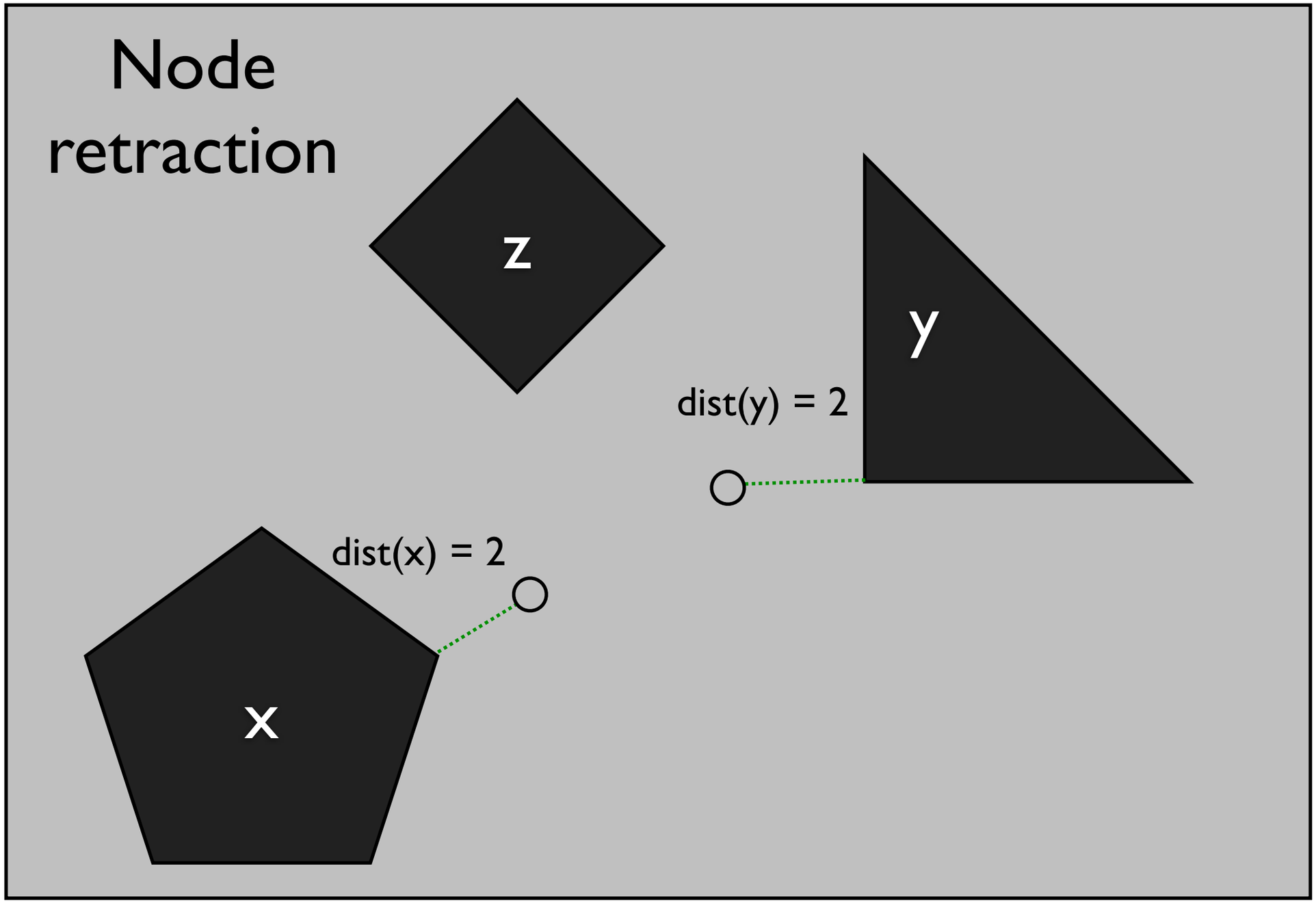
Intuition: retract PRM (nodes and edges) to the medial axis.



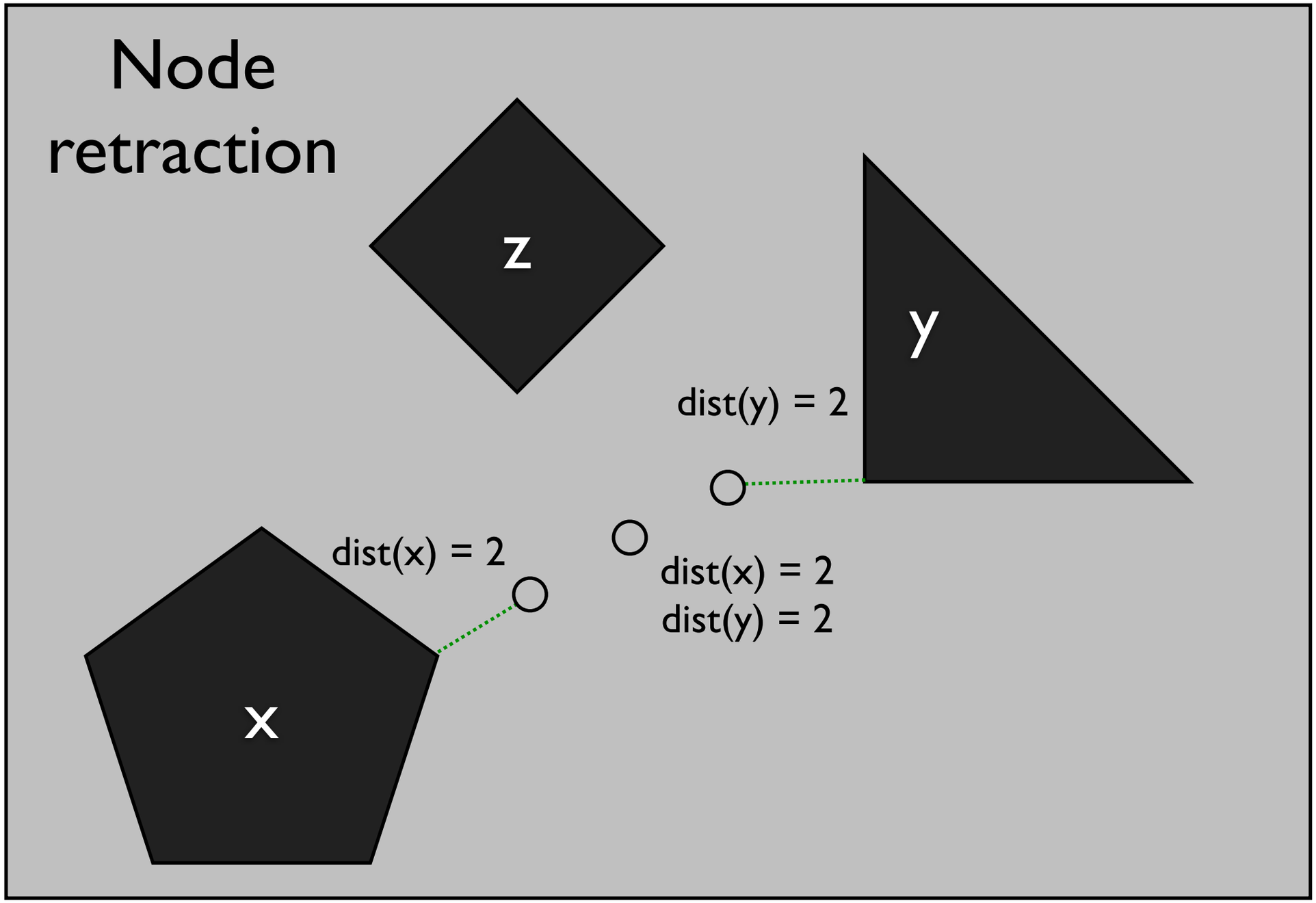

Identify the closest obstacle then retract the point to a position equidistant to 2 obstacles.



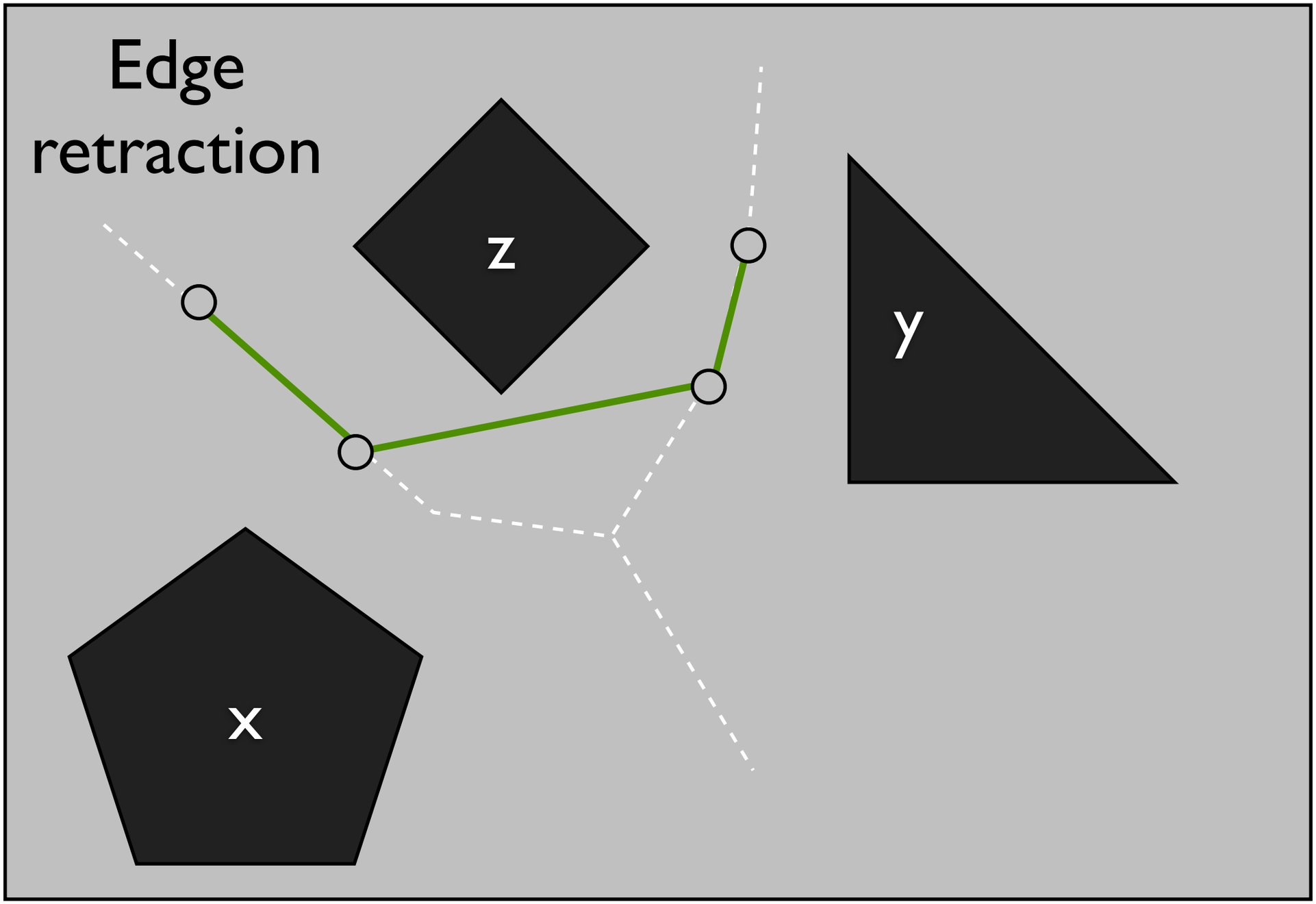

Identify the closest obstacle then retract the point to a position equidistant to 2 obstacles.



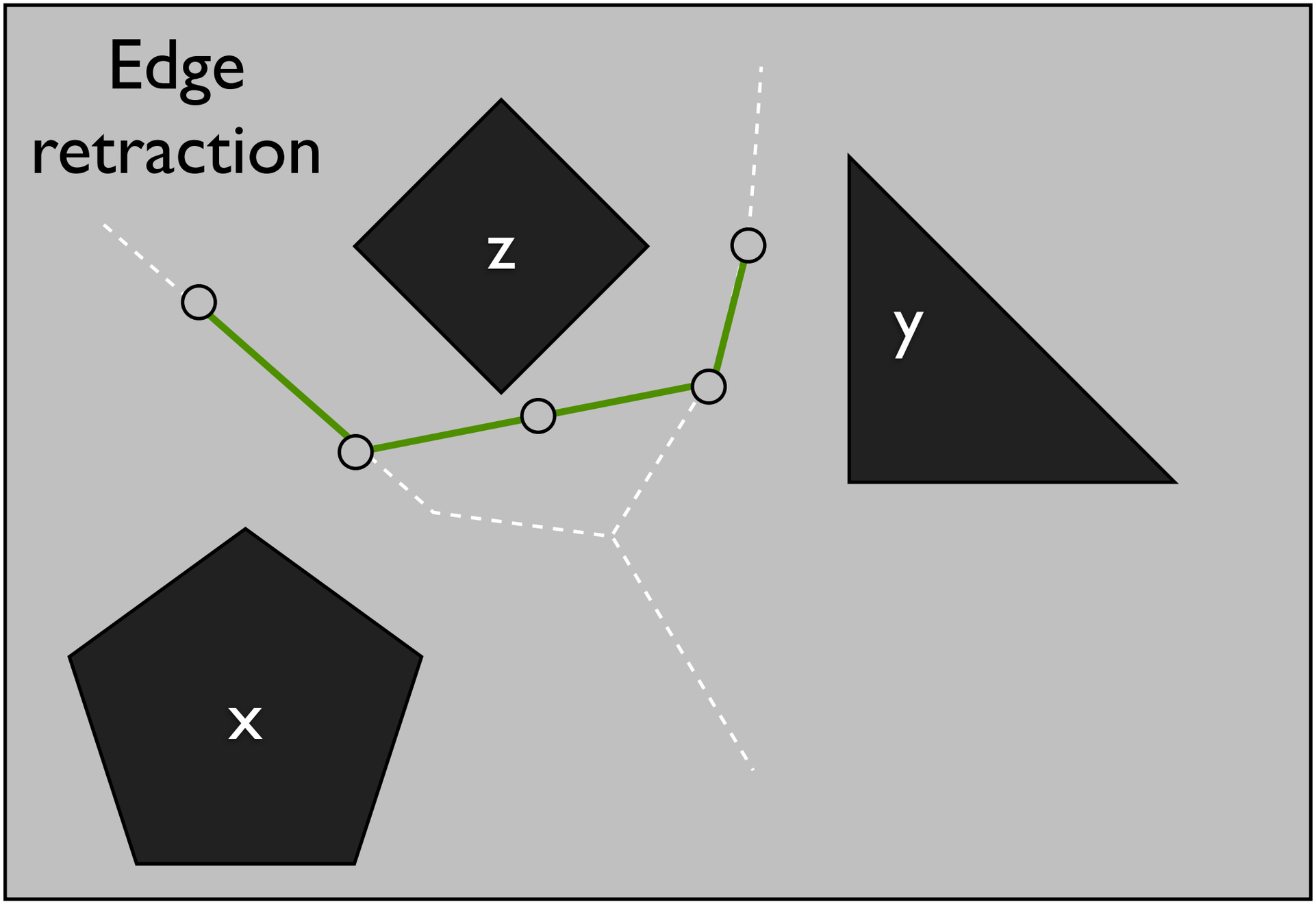

Identify the closest obstacle then retract the point to a position equidistant to 2 obstacles.



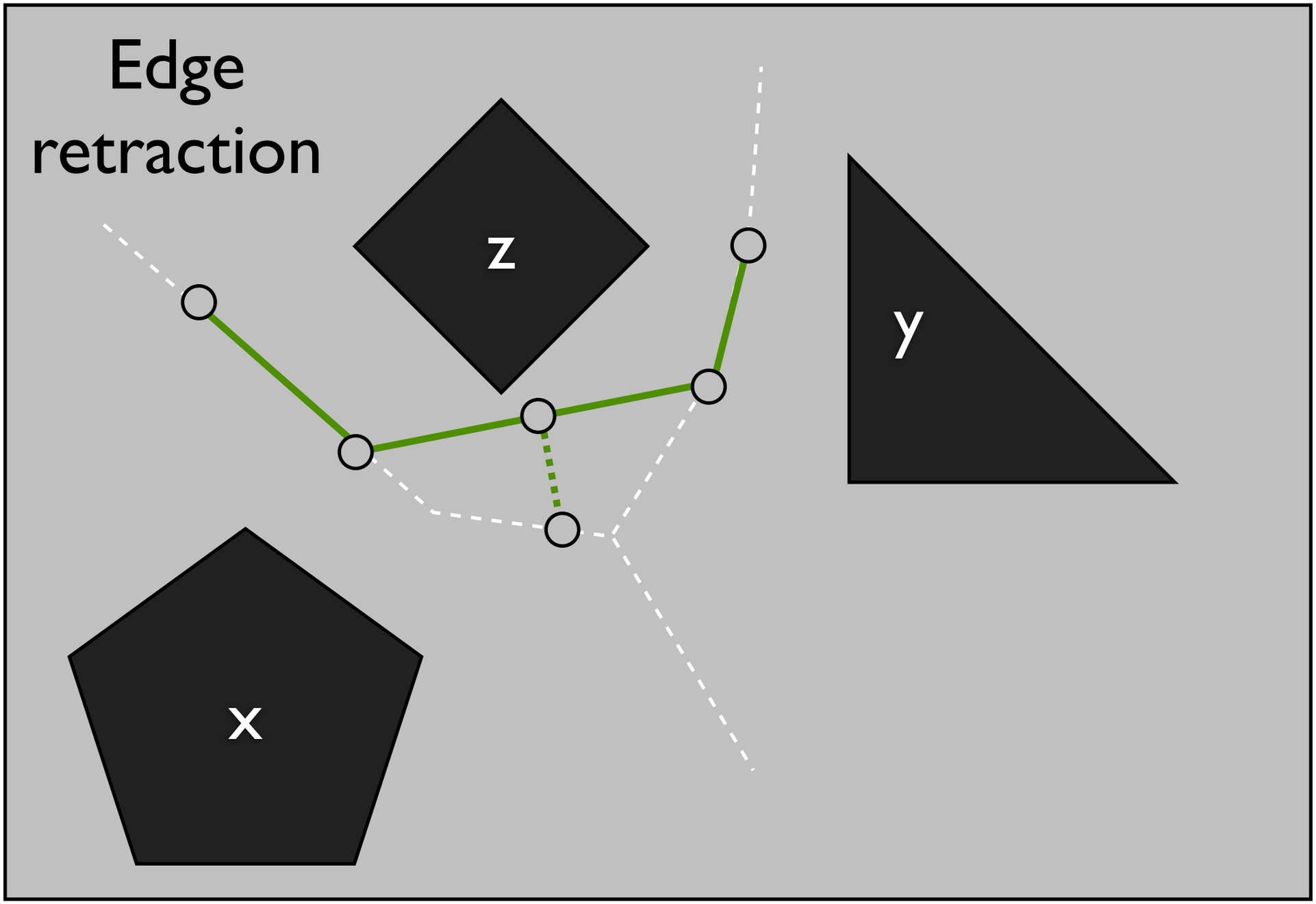

Identify the closest obstacle then retract the point to a position equidistant to 2 obstacles.



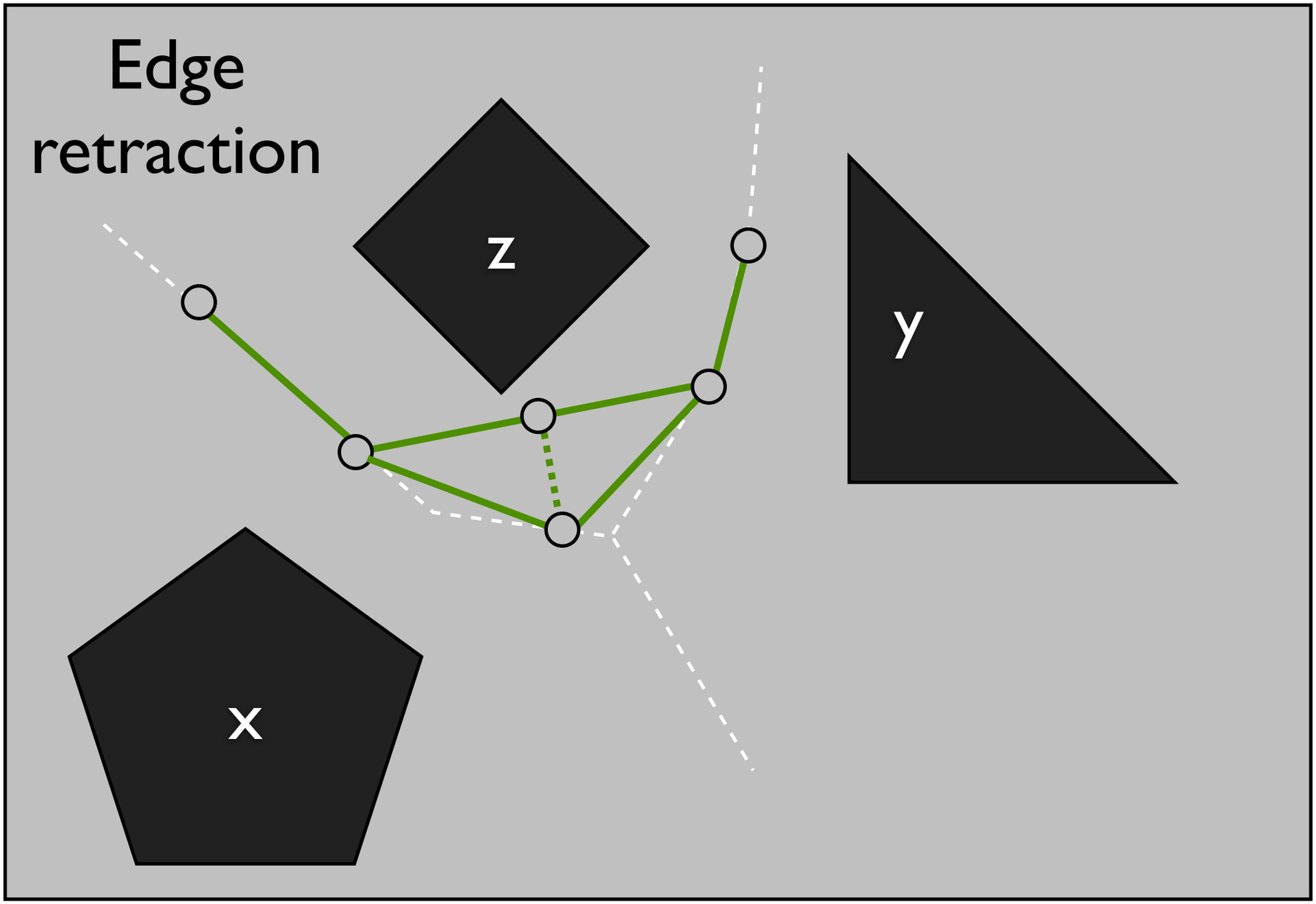

Edges too close to obstacles are retracted by splitting.




Edges too close to obstacles are retracted by splitting.



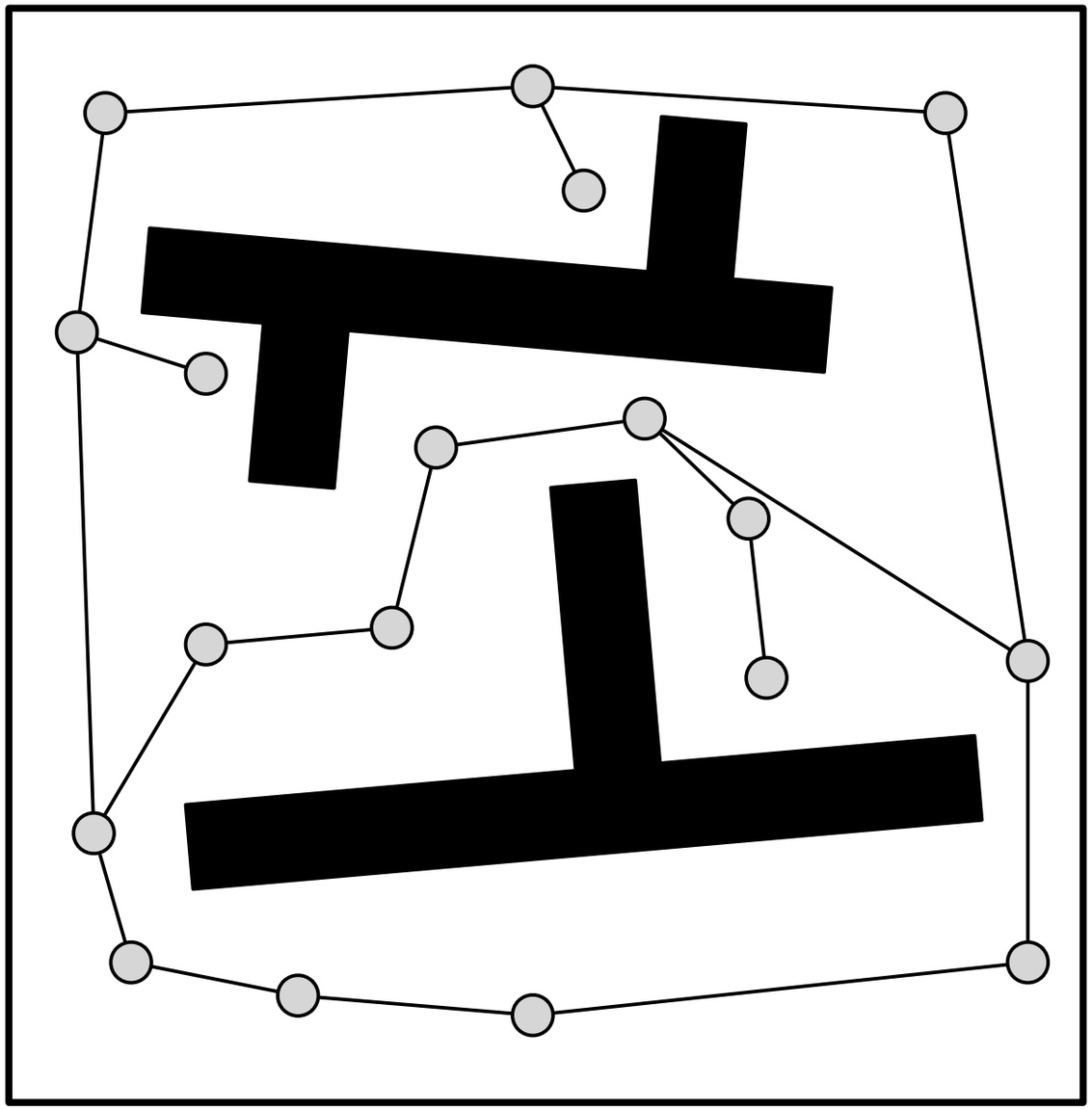

Edges too close to obstacles are retracted by splitting.



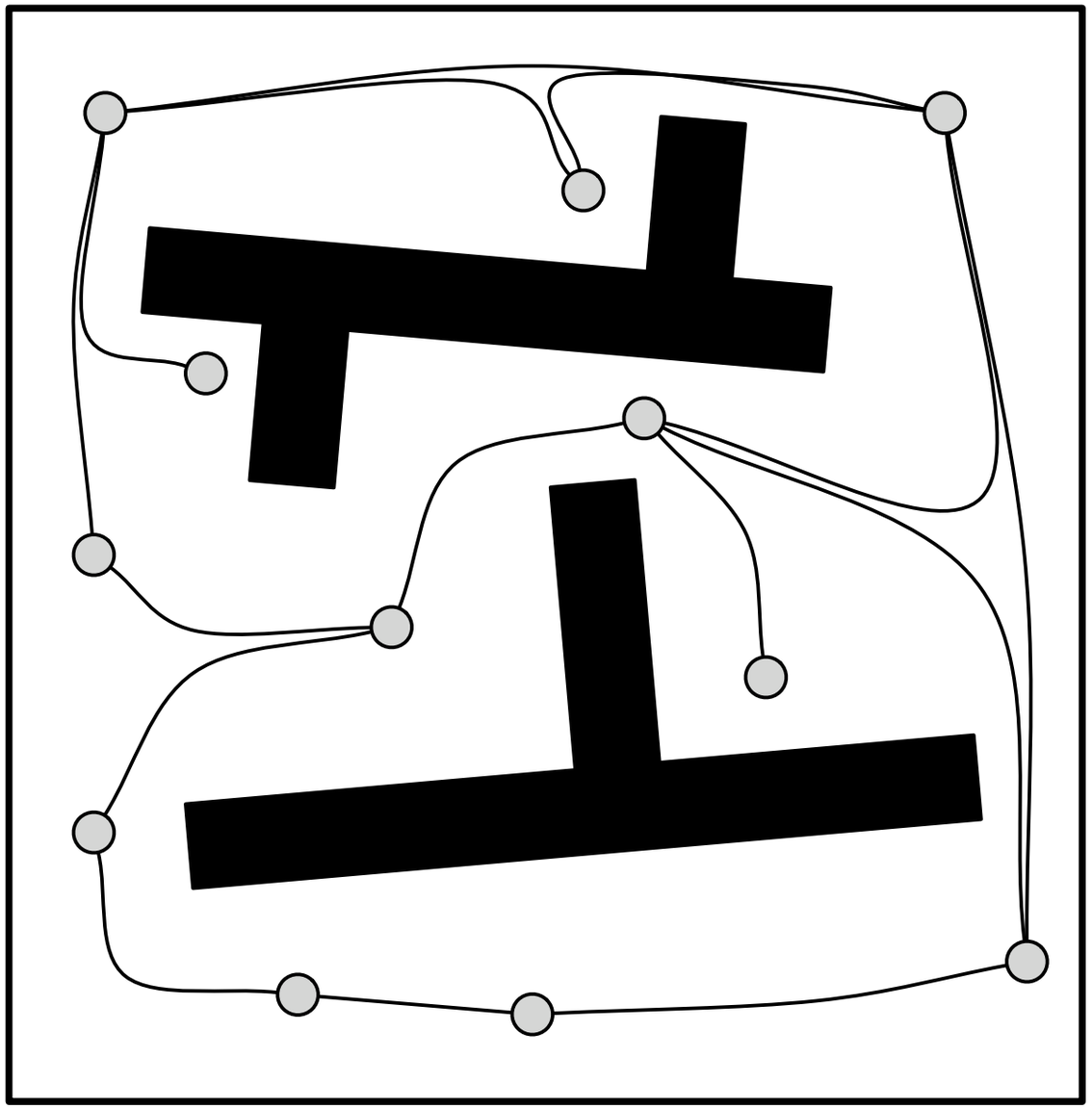

Edges too close to obstacles are retracted by splitting.

- Other improvements:
  - Remove overlapping edges (similar idea to weak dominance)
  - Smooth edges using circular blends

# Other improvements



Remove overlapping edges
(similar to weak dominance)



Apply circular blends

# PRM Advantages

- Fast to construct.
- Low overhead (especially Vornoi PRMs)
- Facilitate pathfinding in n-dimensional spaces
- Generalisable to different unit sizes (Voronoi PRMs only)
- Nice alternative to navigation meshes

# PRM Disadvantages

- Only probabilistically complete*
- Might need several PRMs (to support units with different capabilities).
- Non Voronoi PRMs produce ugly paths.

# Conclusion

- There's more to pathfinding than A*!
- Many open problems; some with nice solutions
  - HPA*, HAA*, Roadmaps
- Pathfinding with abstraction is very effective.

# References


- A Formal Basis for the Heuristic Determination of Minimum Cost Paths
  P.E. Hart, N.J Nilsson, B. Raphael
  IEEE Transactions on Systems Science and Cybernetics, 1968, Vol 4, Issue 2, pp. 100-107

- Near-Optimal Hierarchical Pathfinding
  A. Botea, M. Muller, and J. Schaeffer
  Journal of Game Development, Volume 1, 2004

- Hierarchical path planning for multi-size agents in heterogeneous environments
  Harabor D. and Botea, A.
  IEEE Symposium on Computational Intelligence & Games, 2008

- Principles of Robot Motion
  H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun
  MIT Press, 2004

- Automatic Construction of Roadmaps for Path Planning in Games
  D. Nieuwenhuisen, A. Kamphuis, M. Mooijekind and M. H. Overmars,
  International Conference on Computer Games: Artificial Intelligence, Design and Education, 2004

- Roland Geraerts and Mark H. Overmars. Creating Small Roadmaps for Solving Motion Planning Problems.
  IEEE International Conference on Methods and Models in Automation and Robotics (MMAR'05), 2005,
  pp. 531-536.

# Questions?