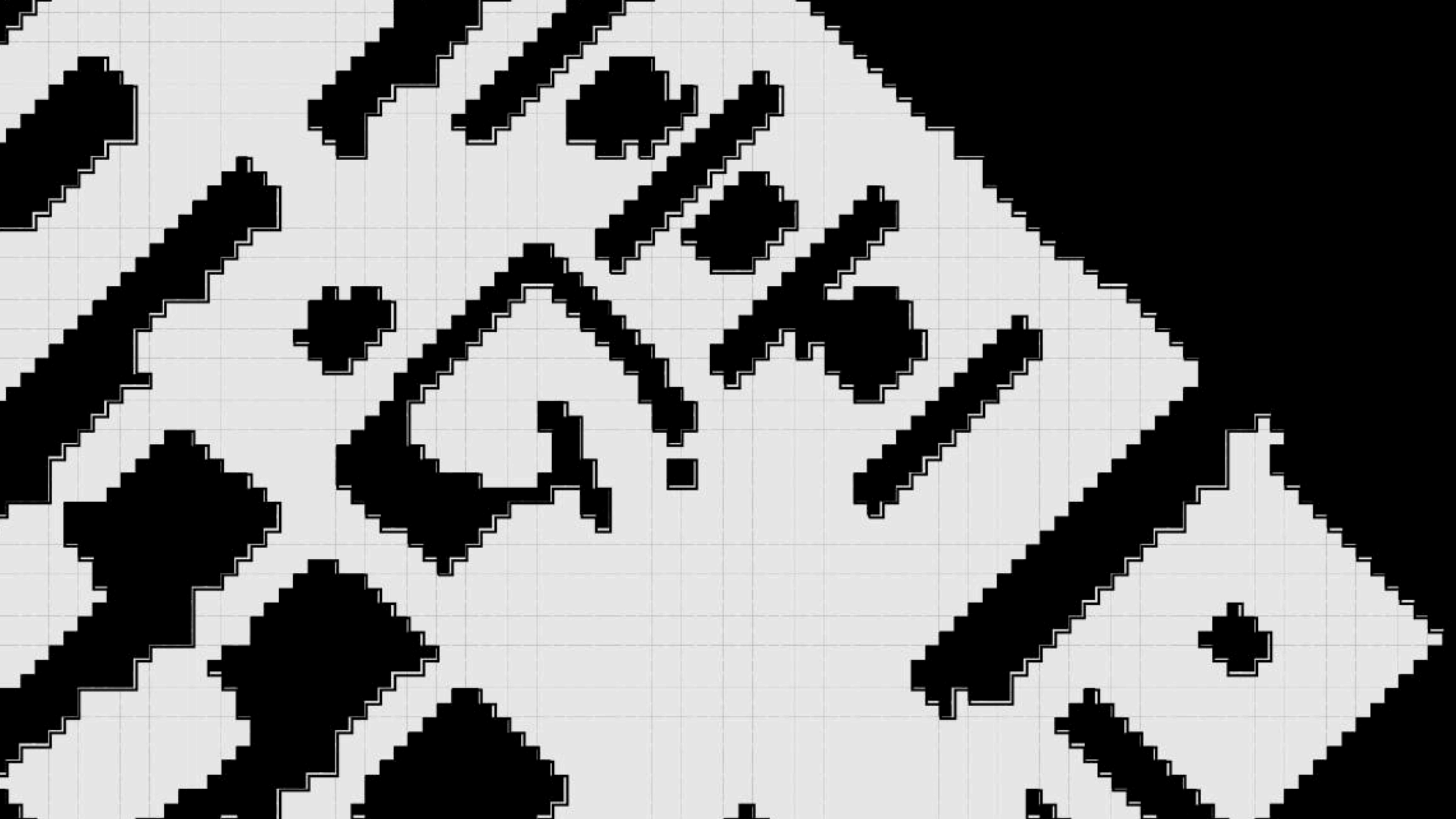
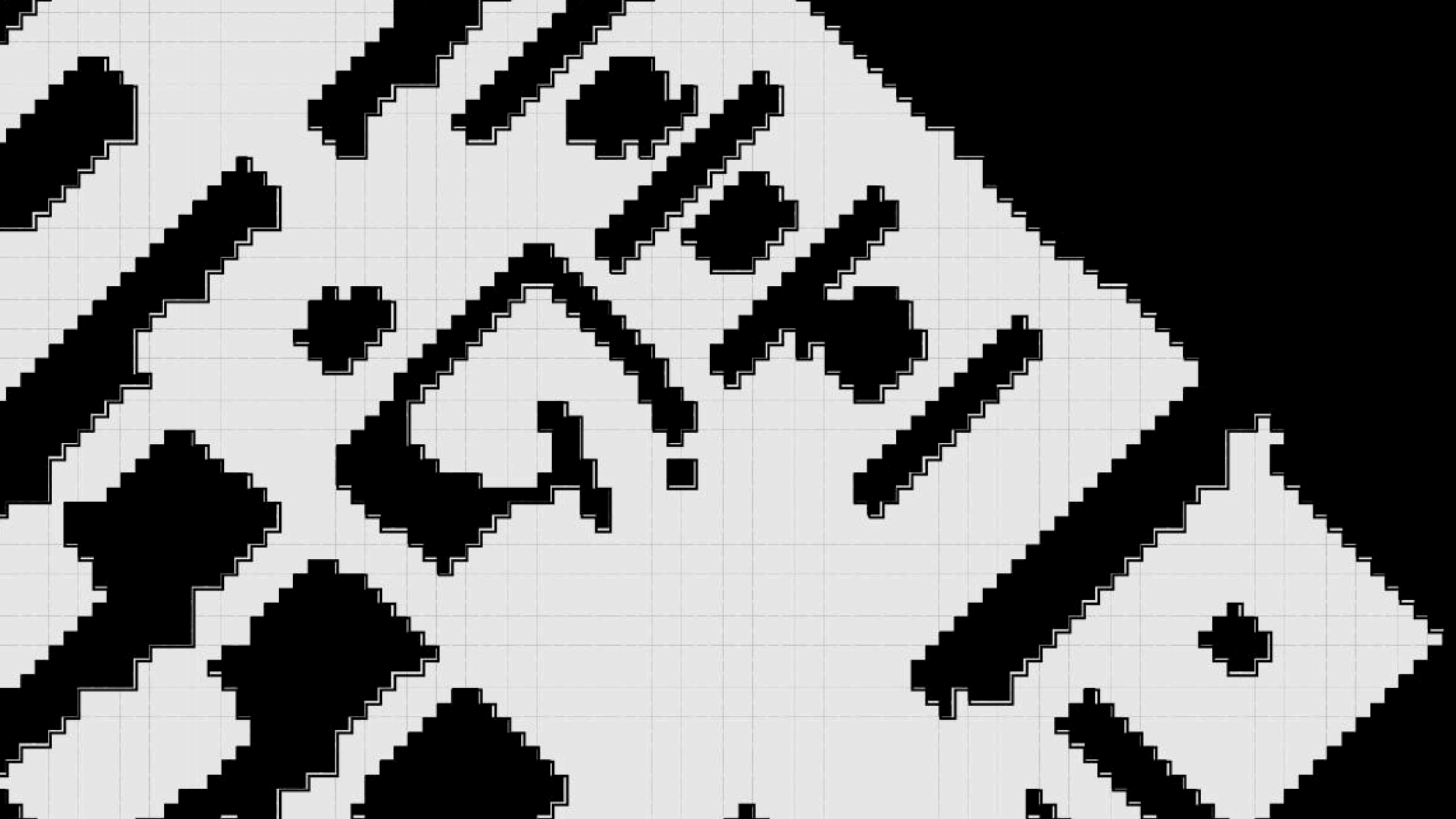
Bidirectional Search: Is It For Me?

Nathan R. Sturtevant (@nathansttt)
Associate Professor
University of Denver





Collaborators

Award 1551406

Jingwei Chen
University of Denver

Robert Holte
University of Alberta

Sneha Sawlani

Ariel Felner

Sandra Zilles

Eshed Shaham

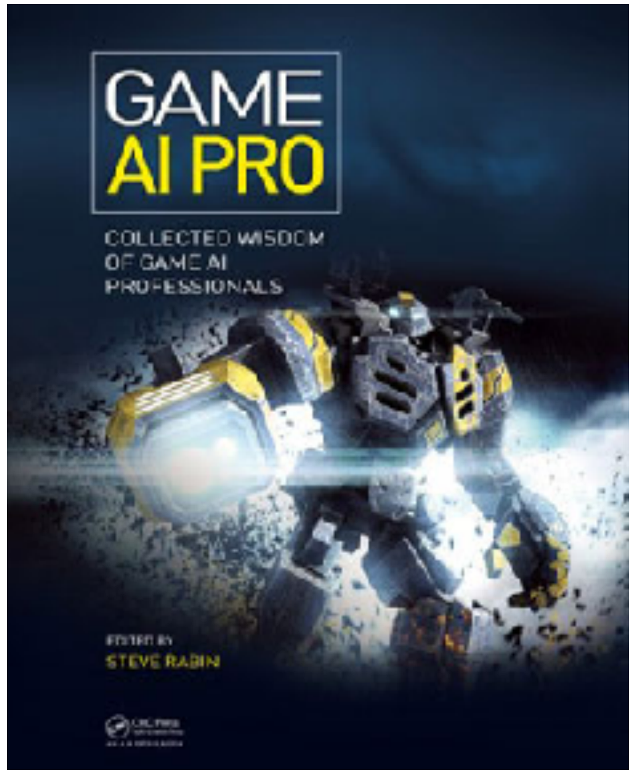
DRAGON AGE™
ORIGINS

Designed and implemented pathfinding engine

EA BIOWARE®

Lecture Takeaways

- When should I use bidirectional search?
- What algorithm should I use for bidirectional search?

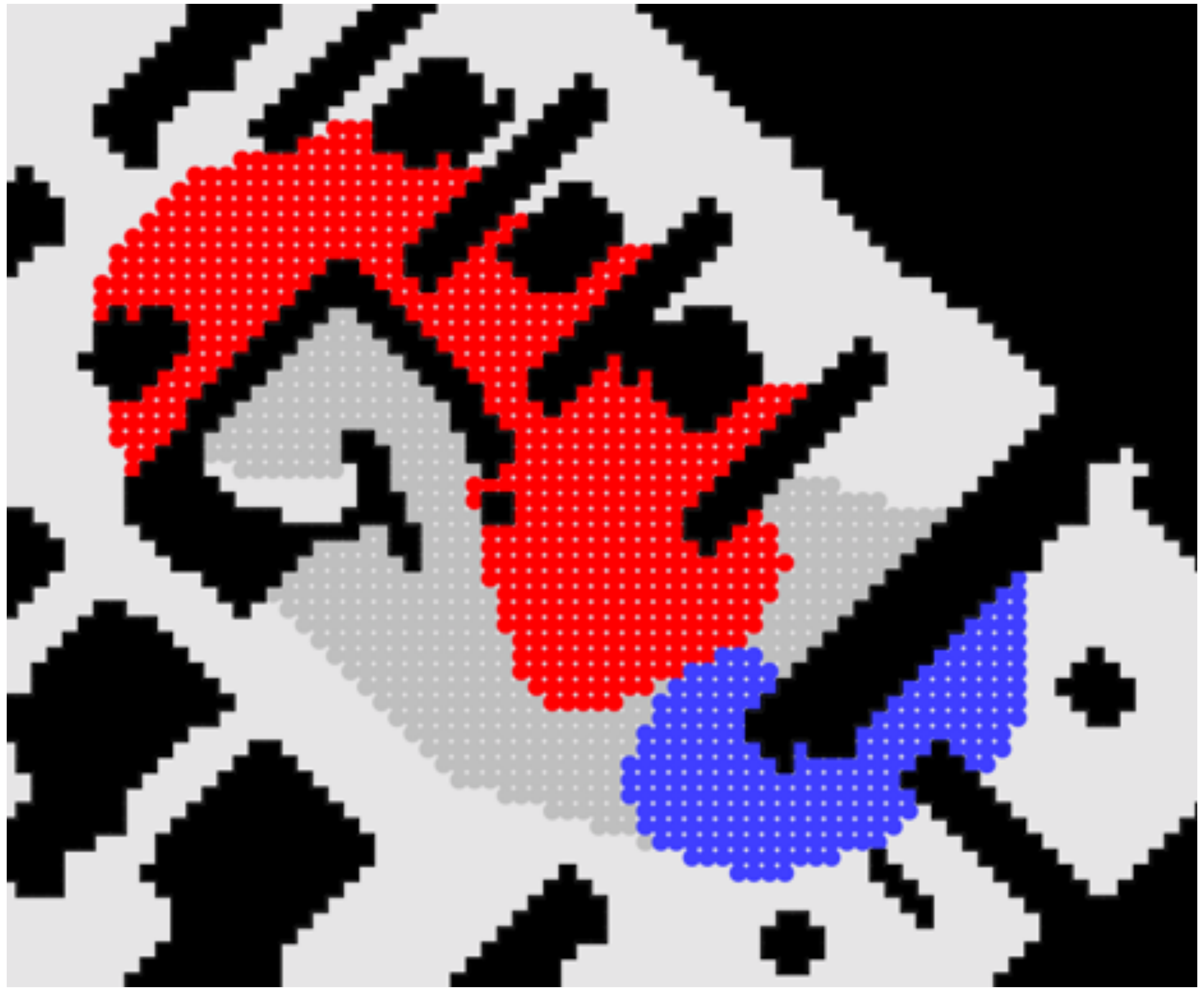


Pathfinding Architecture Optimizations

by Steve Rabin & Nathan Sturtevant

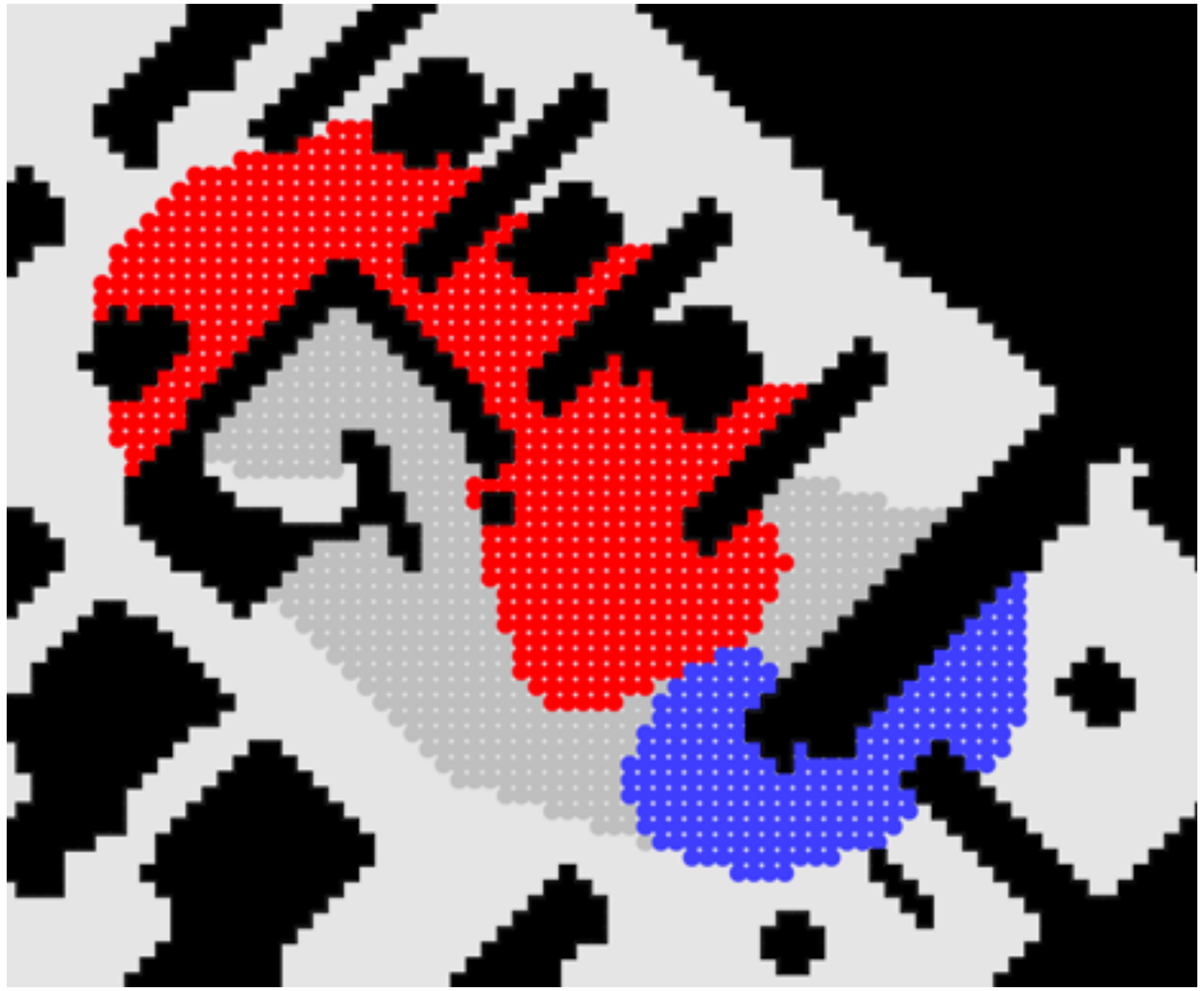
Bad Idea #2: Bidirectional Pathfinding

Optimal Bidirectional Search

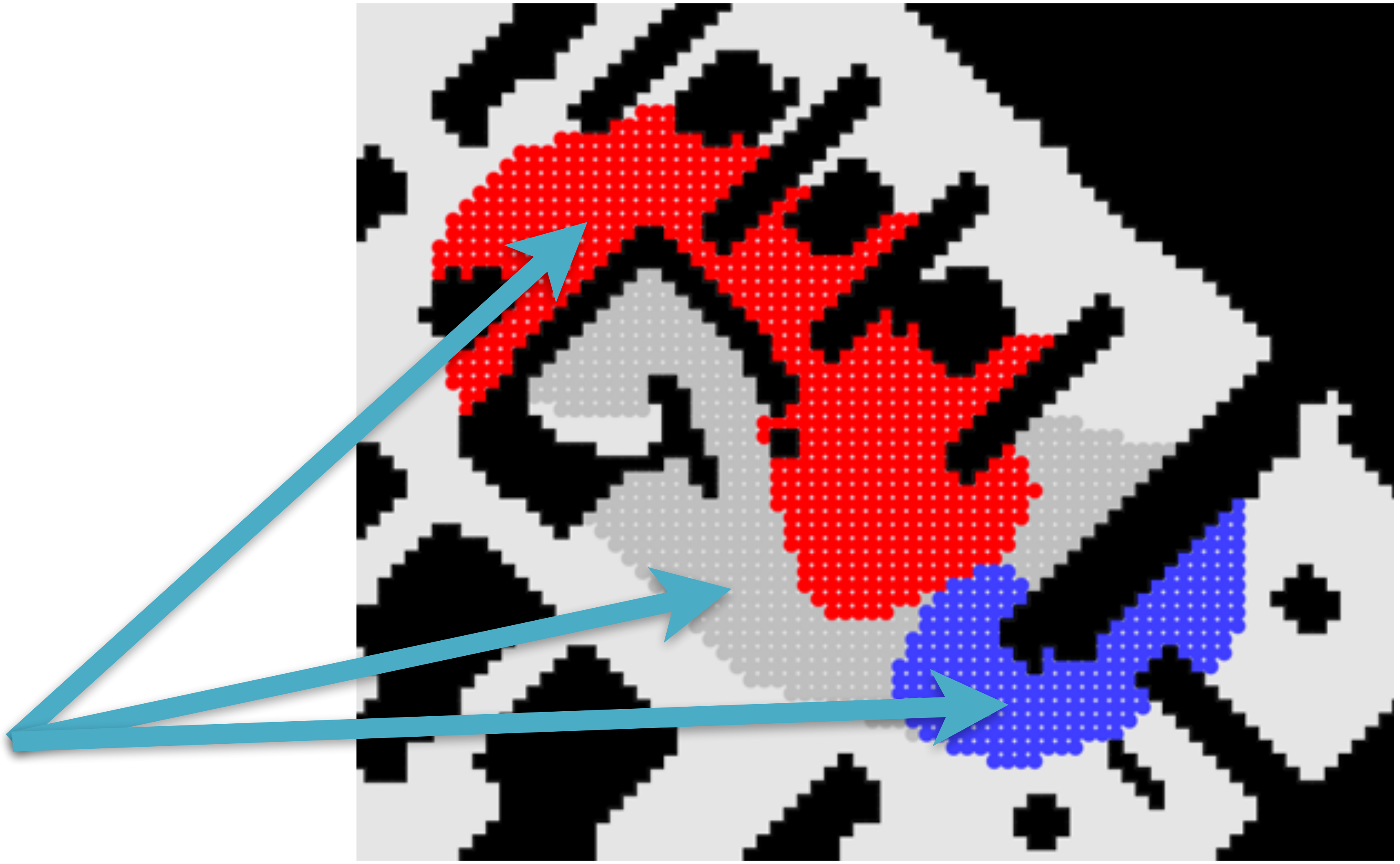


Optimal Bidirectional Search

!



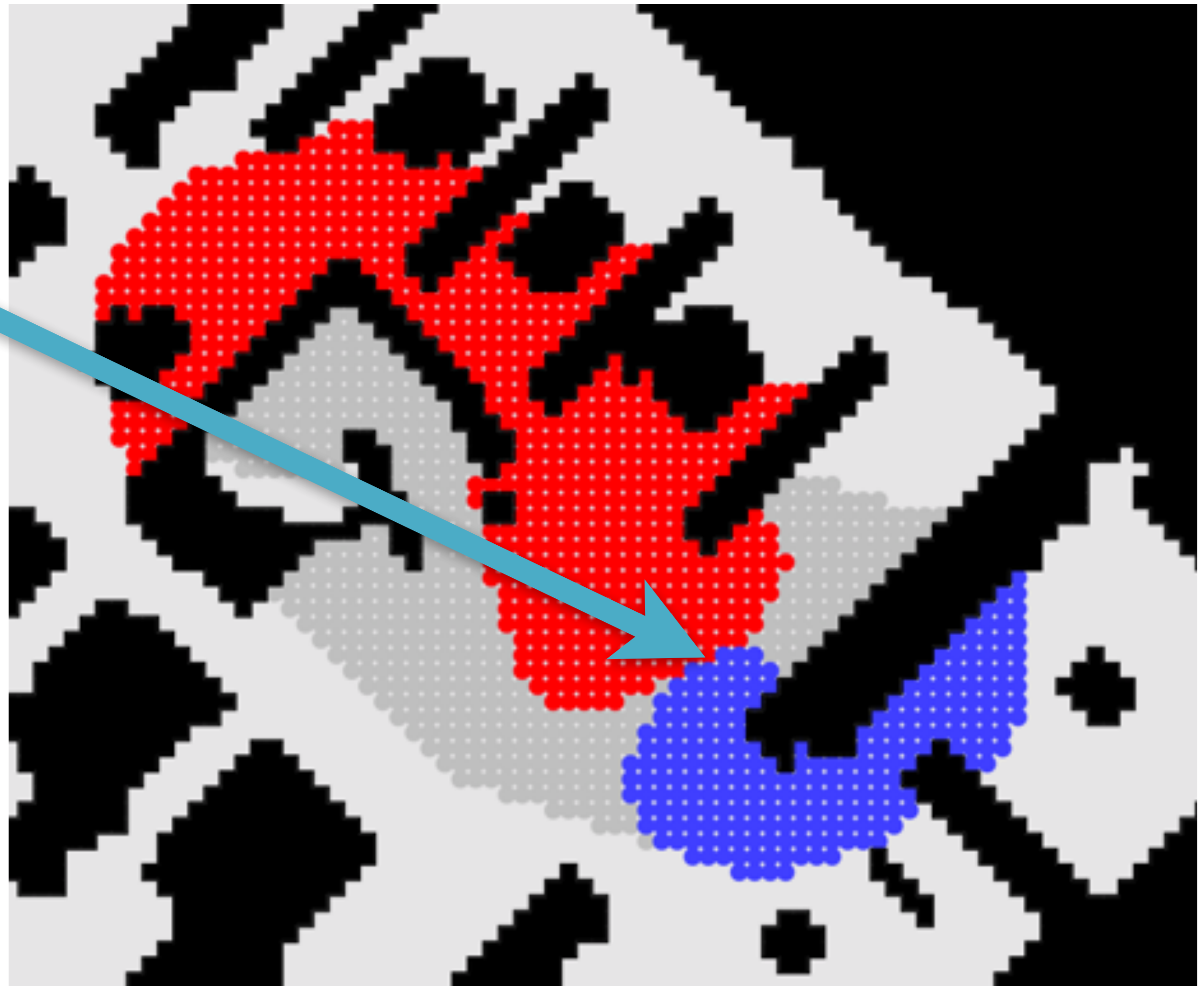
Optimal Bidirectional Search



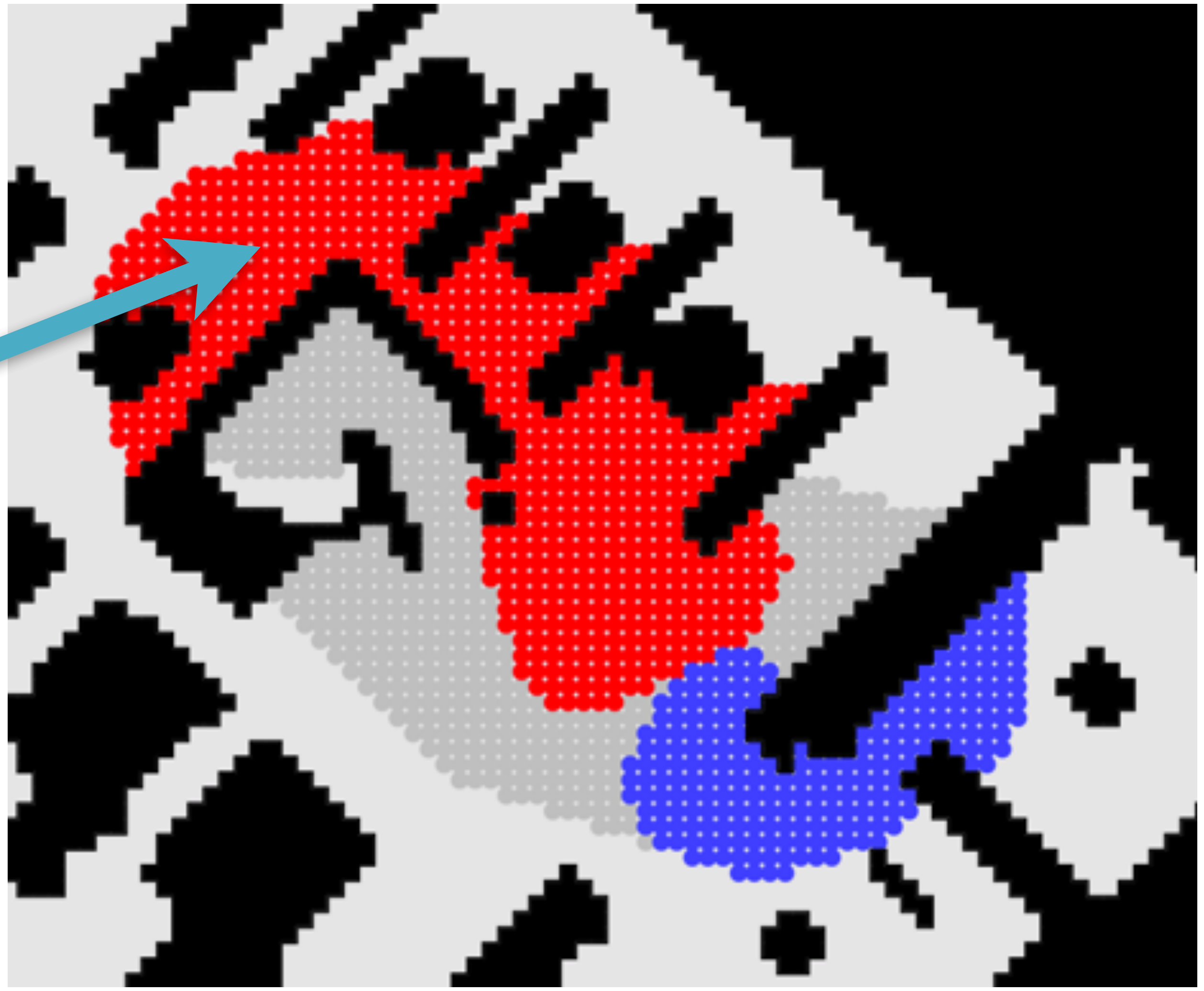
All states that could be expanded

Optimal Bidirectional Search

Choose a meeting point

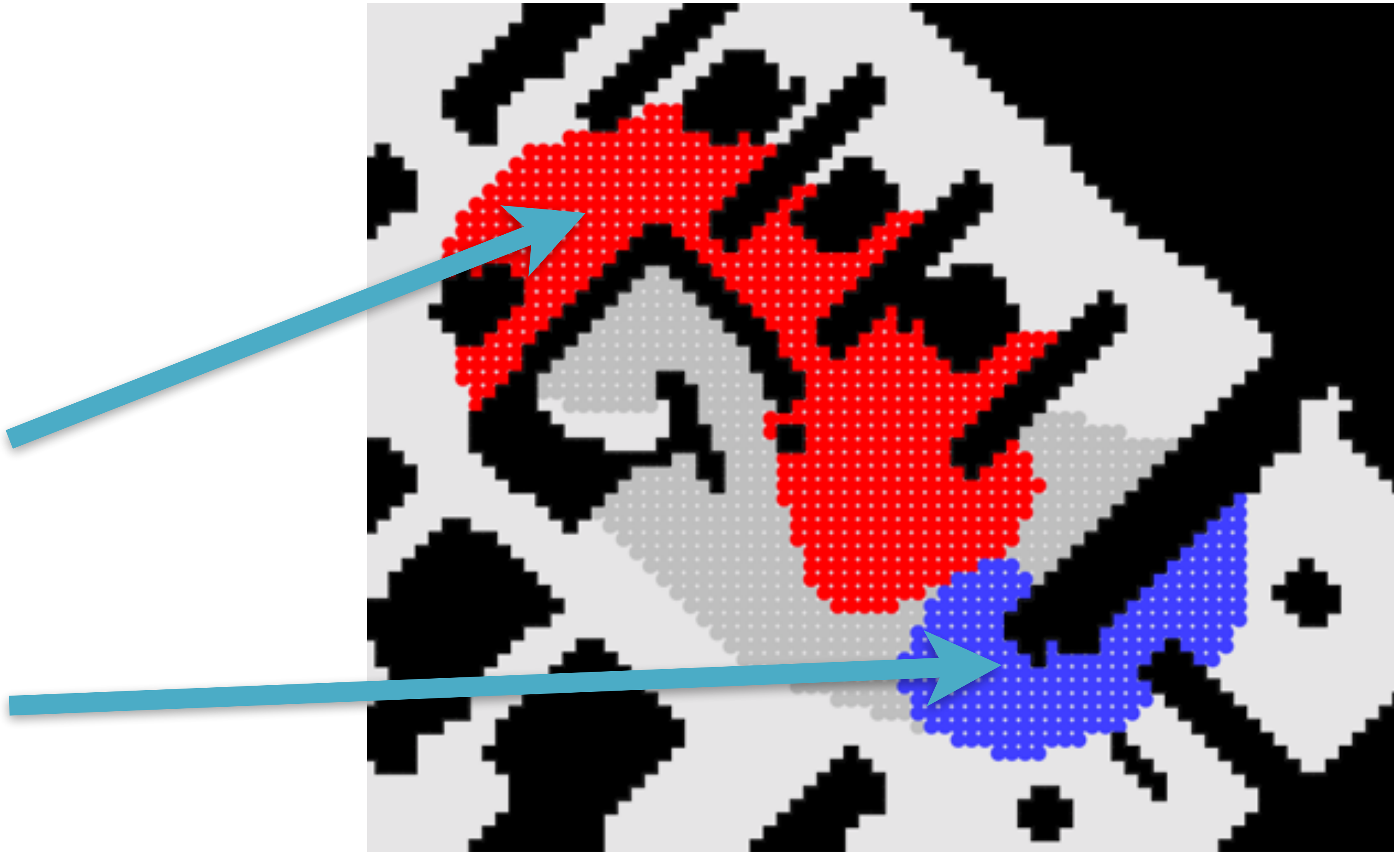


Optimal Bidirectional Search



Expand up to that point forward

Optimal Bidirectional Search



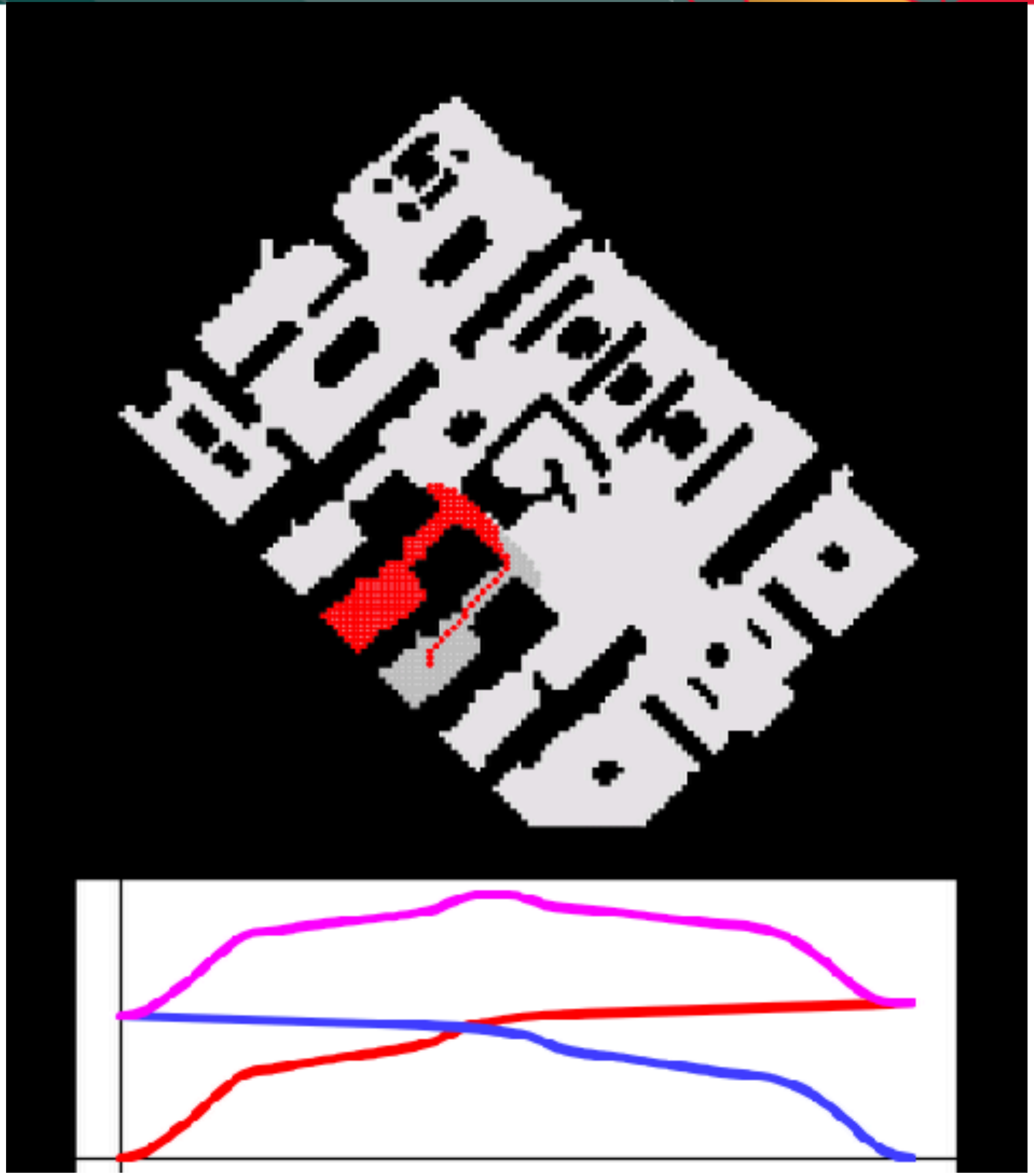
Expand up to that point forward

Expand up to that point backward

Demo

Explanation

- Perfect heuristic near goal
    - Open space
- Symmetric



New Algorithm: NBS

- NBS never expands more than 2x the states expanded by the **best possible** algorithm
  - *In our theoretical framework*

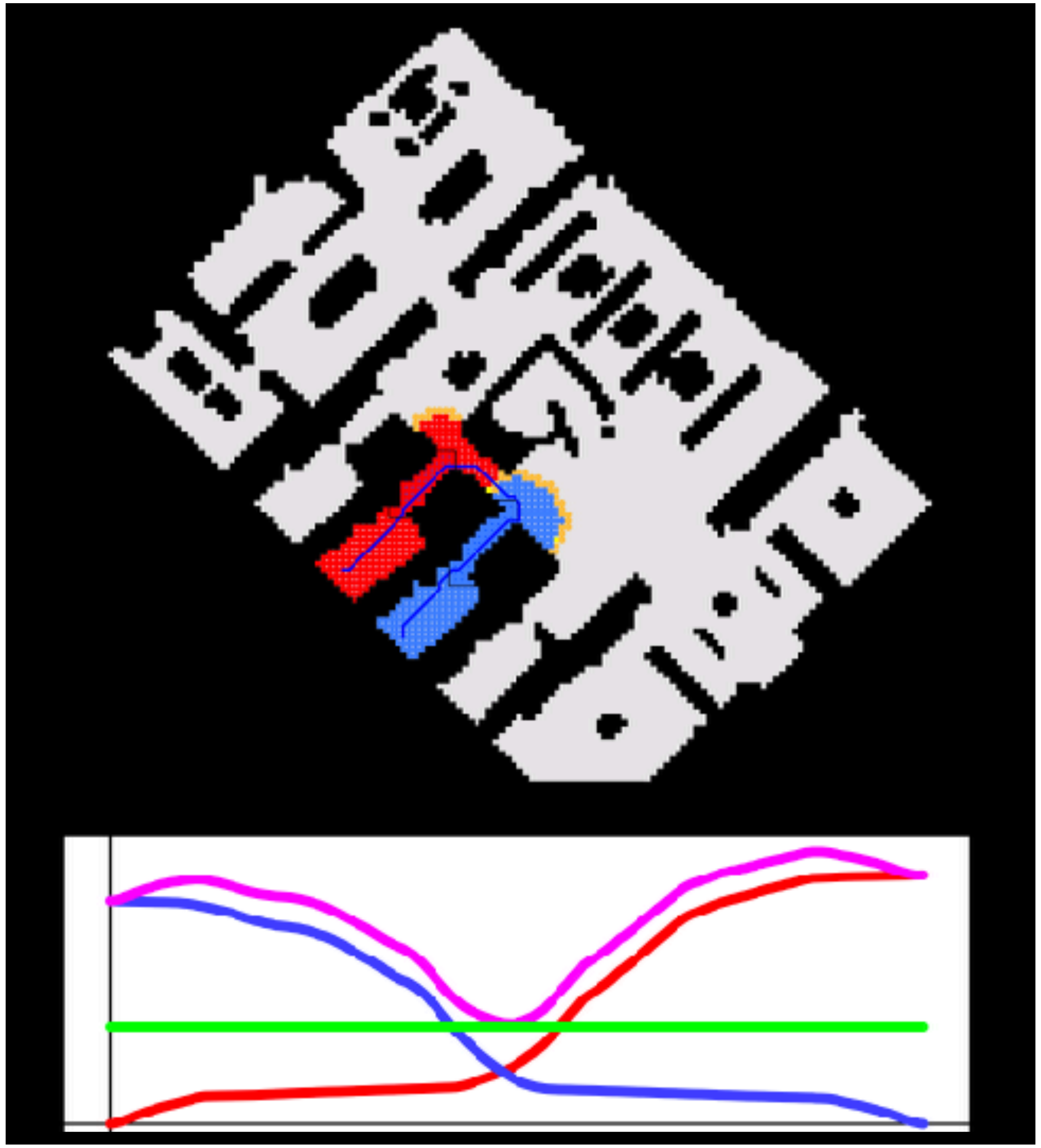
- NBS does equal work in each direction

When should we use NBS?

Scenario 1: Weighted terrain

Weighted terrain

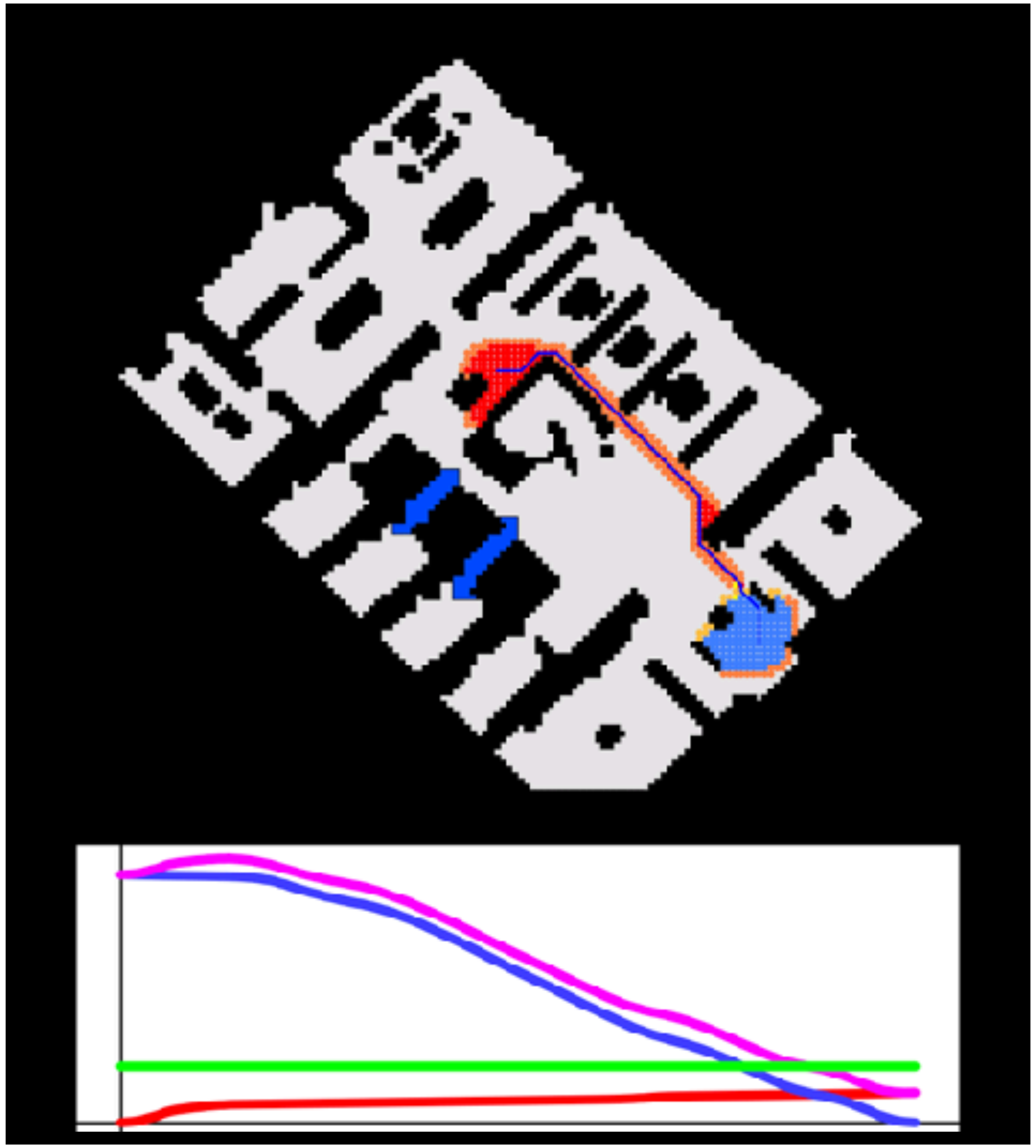
- Costly to look for alternate paths around weighted terrain



Scenario 2:
Problem Asymmetry

Problem Asymmetry

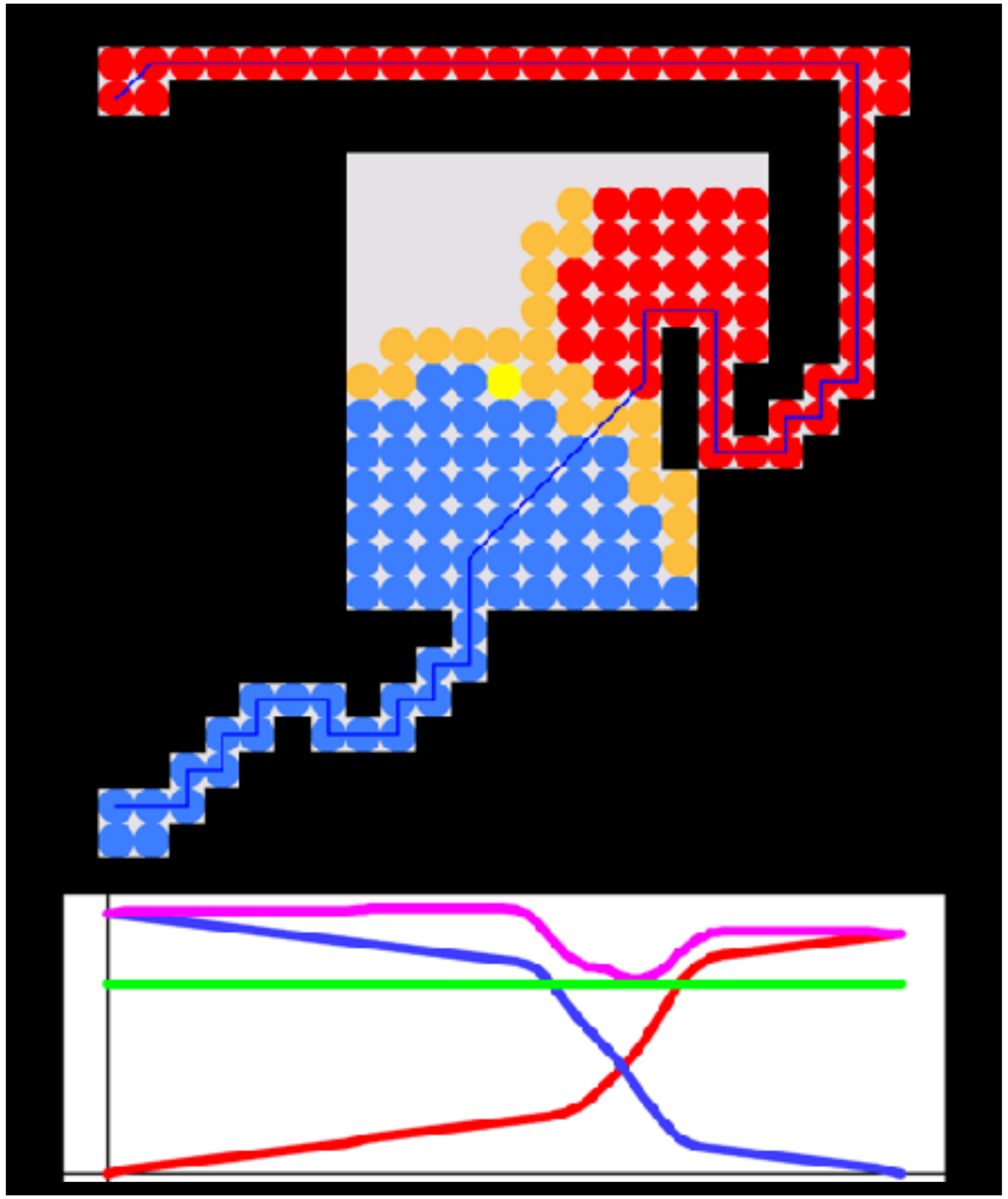
- When forward is much more expensive than backwards
  - 3x worse on average
- Also happens with weighted terrain



Scenario 3: Map Asymmetry

Map Asymmetry

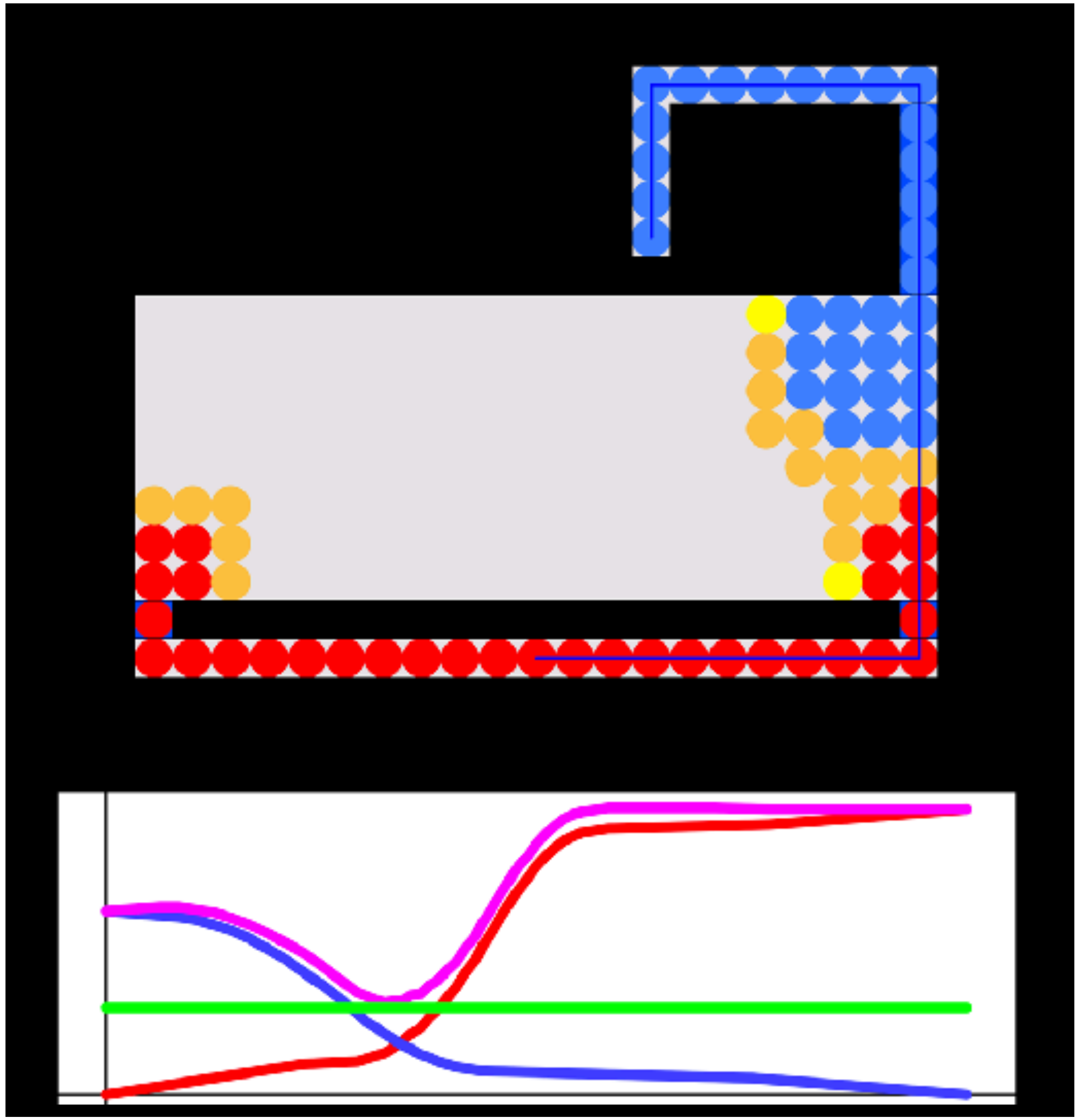
- Common in city maps
  - Dense regions of pathfinding nodes
- Bidirectional search will avoid the densest region



Scenario 4:
Local Minima

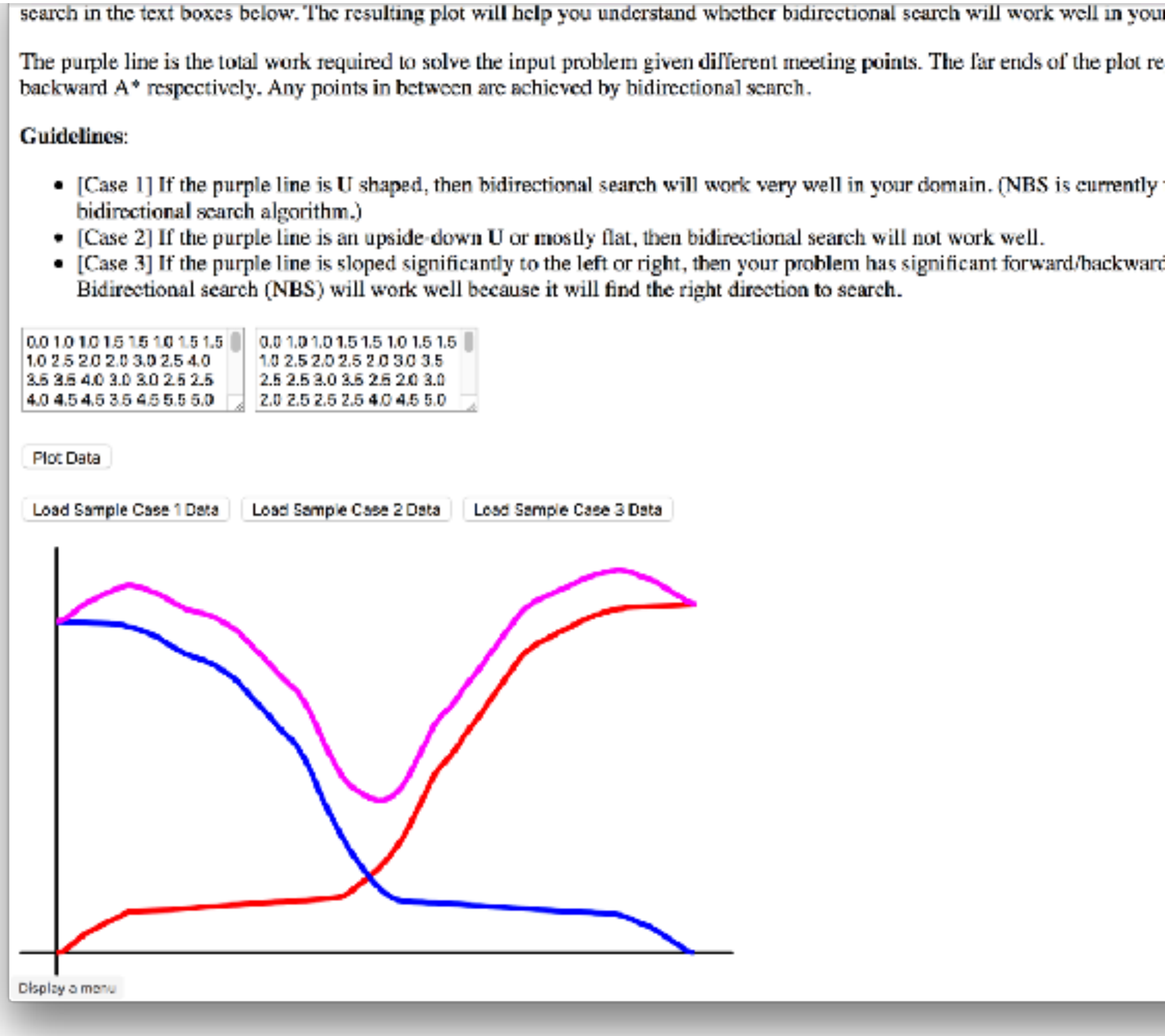
Local Minima

- Many states look close, but aren't
  - Could be fixed by a better heuristic



Testing in practice

- Web tool available for analysis
- http://www.movingai.com/GDC18/test.html



search in the text boxes below. The resulting plot will help you understand whether bidirectional search will work well in your

The purple line is the total work required to solve the input problem given different meeting points. The far ends of the plot re
backward A* respectively. Any points in between are achieved by bidirectional search.

Guidelines:

- [Case 1] If the purple line is U shaped, then bidirectional search will work very well in your domain. (NBS is currently
bidirectional search algorithm.)
- [Case 2] If the purple line is an upside-down U or mostly flat, then bidirectional search will not work well.
- [Case 3] If the purple line is sloped significantly to the left or right, then your problem has significant forward/backward
Bidirectional search (NBS) will work well because it will find the right direction to search.

```
0.0 1.0 1.0 1.5 1.5 1.0 1.5 1.5
1.0 2.5 2.0 2.0 3.0 2.5 4.0
3.5 3.5 4.0 3.0 3.0 2.5 2.5
4.0 4.5 4.5 3.5 4.5 5.5 5.0
```

```
0.0 1.0 1.0 1.5 1.5 1.0 1.5 1.5
1.0 2.5 2.0 2.5 2.0 3.0 3.5
2.5 2.5 3.0 3.5 2.5 2.0 3.0
2.0 2.5 2.5 2.5 4.0 4.5 5.0
```

NBS Details

A*

A*

- Put start onto priority queue

A*

- Put start onto priority queue
- While queue not empty / solution not found

A*

- Put start onto priority queue
- While queue not empty / solution not found
    - Among all states on queue:

A*

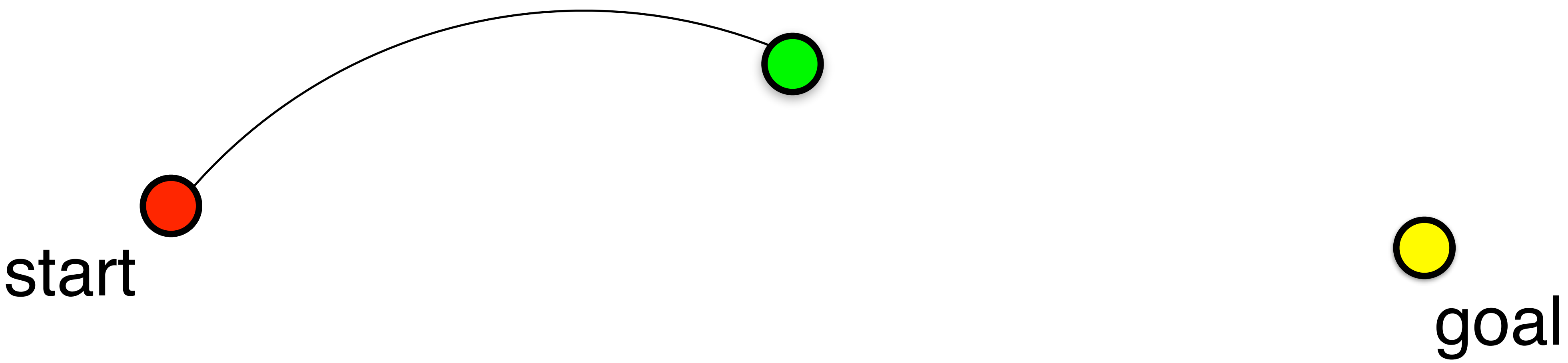
- Put start onto priority queue
- While queue not empty / solution not found
  - Among all states on queue:
    - Select the state with lowest **f-cost**

A*

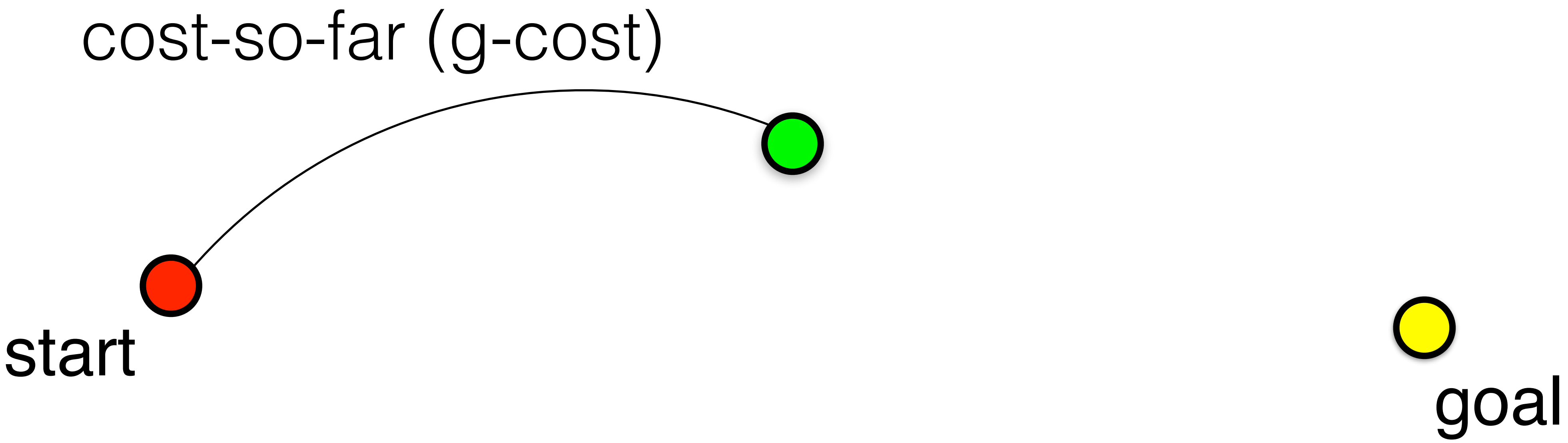
- Put start onto priority queue
- While queue not empty / solution not found
  - Among all states on queue:
    - Select the state with lowest **f-cost**
    - Expand it

A*: f-cost

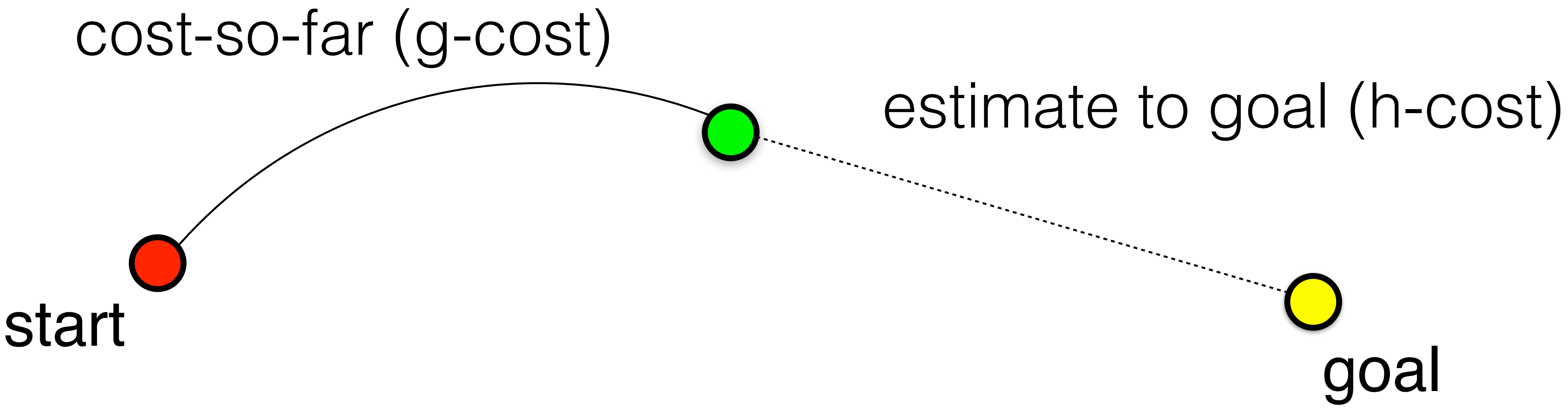
A*: f-cost



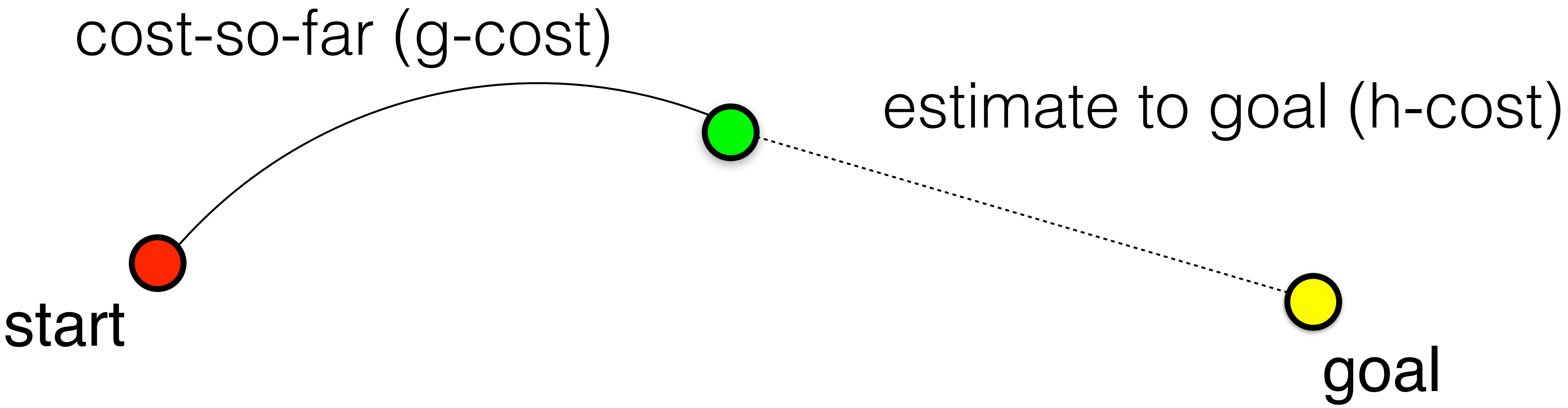
A*: f-cost



A*: f-cost



A*: f-cost



f-cost = g-cost + h-cost = estimated path length

A*

- Put start onto priority queue
- While queue not empty / solution not found
  - Among all states on queue:
    - Select the state with lowest **f-cost**
    - Expand it

A* → NBS

- Put start onto priority queue
- While queue not empty / solution not found
  - Among all states on queue:
    - Select the state with lowest **f-cost**
    - Expand it

Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions, Jingwei Chen, Robert C. Holte, Sandra Zilles and Nathan R. Sturtevant, International Joint Conference on Artificial Intelligence (IJCAI), **2017**

A* → NBS

- Put start/**goal** onto forward/**backward** priority queues
- While queue not empty / solution not found
  - Among all states on queue:
    - Select the state with lowest **f-cost**
    - Expand it

Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions,
Jingwei Chen, Robert C. Holte, Sandra Zilles and Nathan R. Sturtevant,
International Joint Conference on Artificial Intelligence (IJCAI), **2017**

A* → NBS

- Put start/**goal** onto forward/**backward** priority queues
- While queue**s** not empty / solution not found
  - Among all states on queue:
    - Select the state with lowest **f-cost**
    - Expand it

Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions, Jingwei Chen, Robert C. Holte, Sandra Zilles and Nathan R. Sturtevant, International Joint Conference on Artificial Intelligence (IJCAI), **2017**

A* → NBS

- Put start/**goal** onto forward/**backward** priority queues
- While queue**s** not empty / solution not found
  - Among all states on queue**s**:
    - Select the state with lowest **f-cost**
    - Expand it

Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions,
Jingwei Chen, Robert C. Holte, Sandra Zilles and Nathan R. Sturtevant,
International Joint Conference on Artificial Intelligence (IJCAI), **2017**

A* → NBS

- Put start/**goal** onto forward/**backward** priority queues
- While queue**s** not empty / solution not found
    - Among all states on queue**s**:
        - Select the **pair** with lowest ***lower bound***
        - Expand it

Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions,
Jingwei Chen, Robert C. Holte, Sandra Zilles and Nathan R. Sturtevant,
International Joint Conference on Artificial Intelligence (IJCAI), **2017**

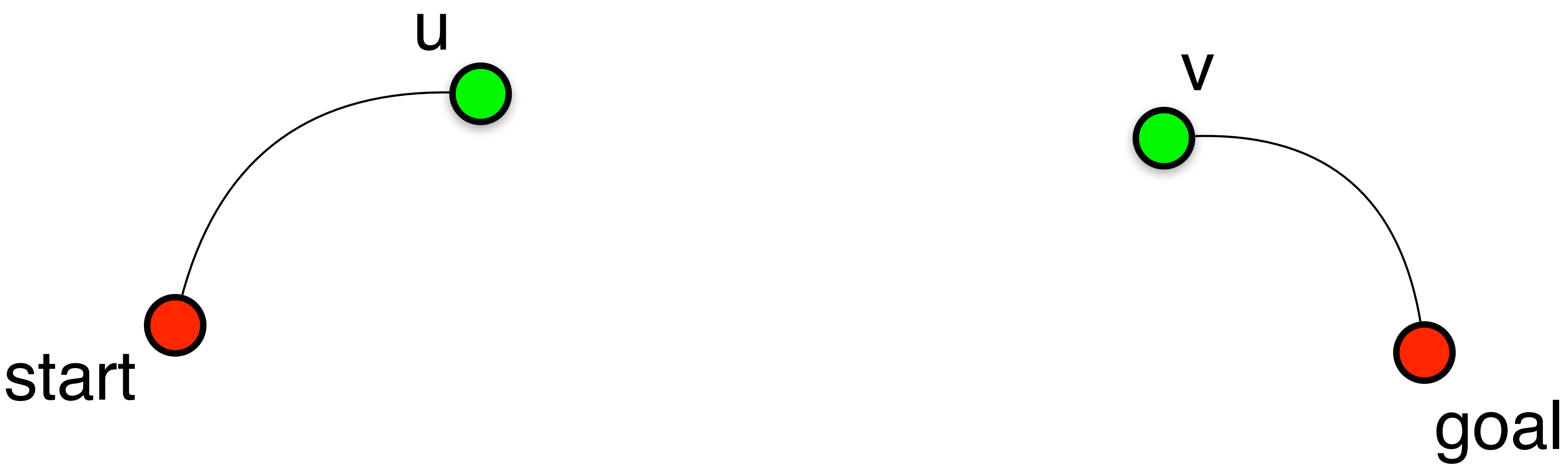
A* → NBS

- Put start**/goal** onto forward**/backward** priority queues
- While queue**s** not empty / solution not found
  - Among all states on queue**s**:
    - Select the **pair** with lowest ***lower bound***
    - Expand **both** of them

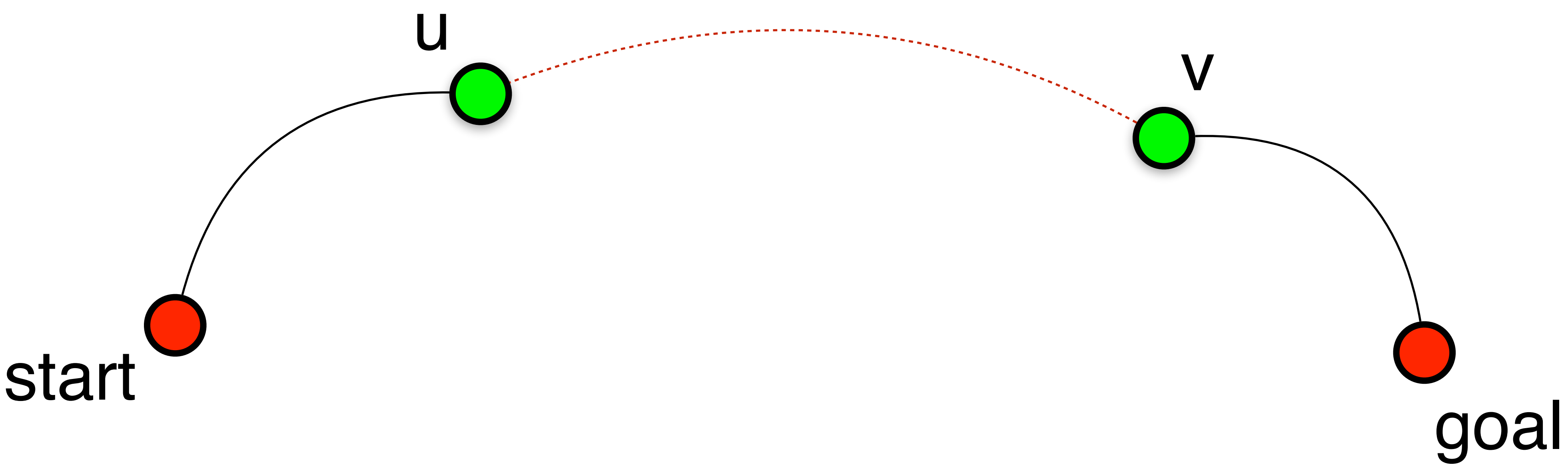
Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions,
Jingwei Chen, Robert C. Holte, Sandra Zilles and Nathan R. Sturtevant,
International Joint Conference on Artificial Intelligence (IJCAI), **2017**

NBS: lower bound

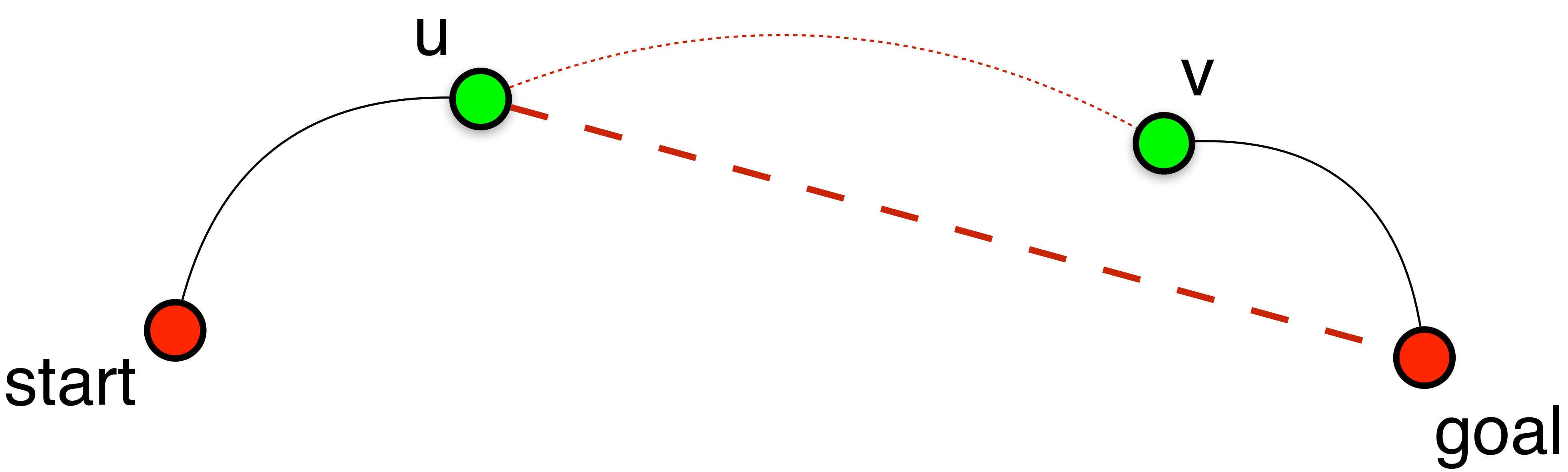
NBS: lower bound



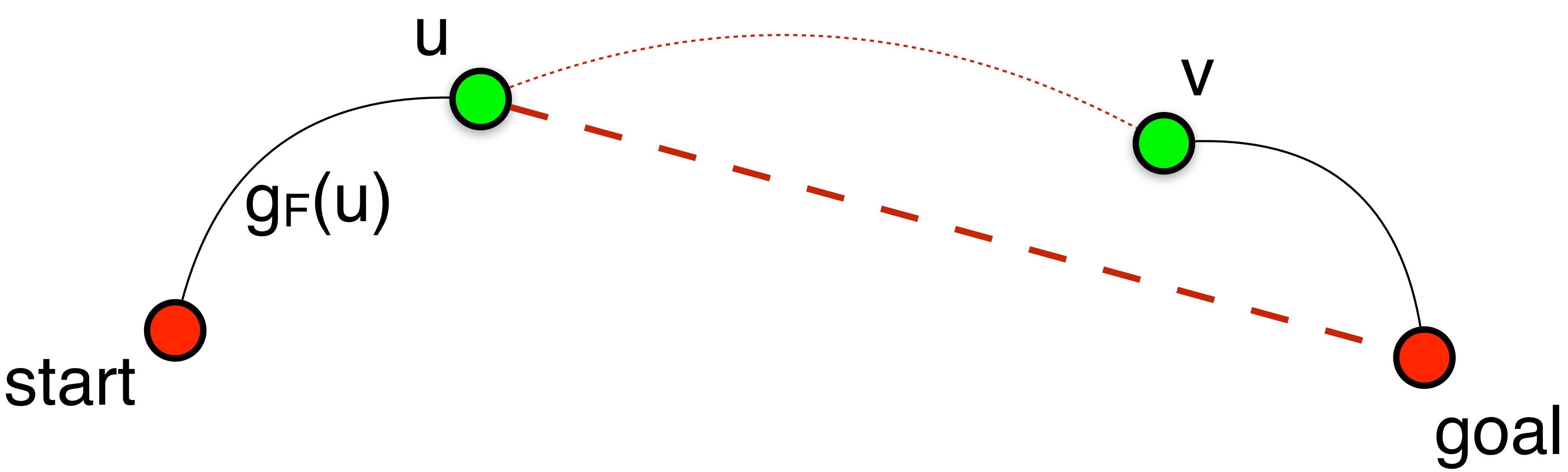
NBS: lower bound



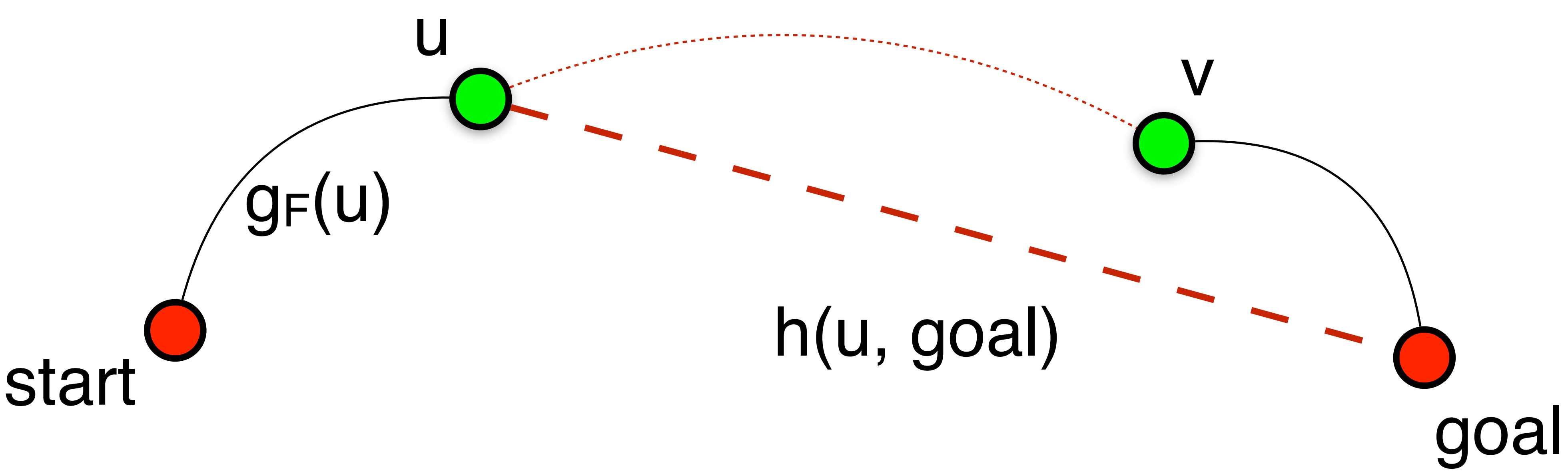
NBS: lower bound



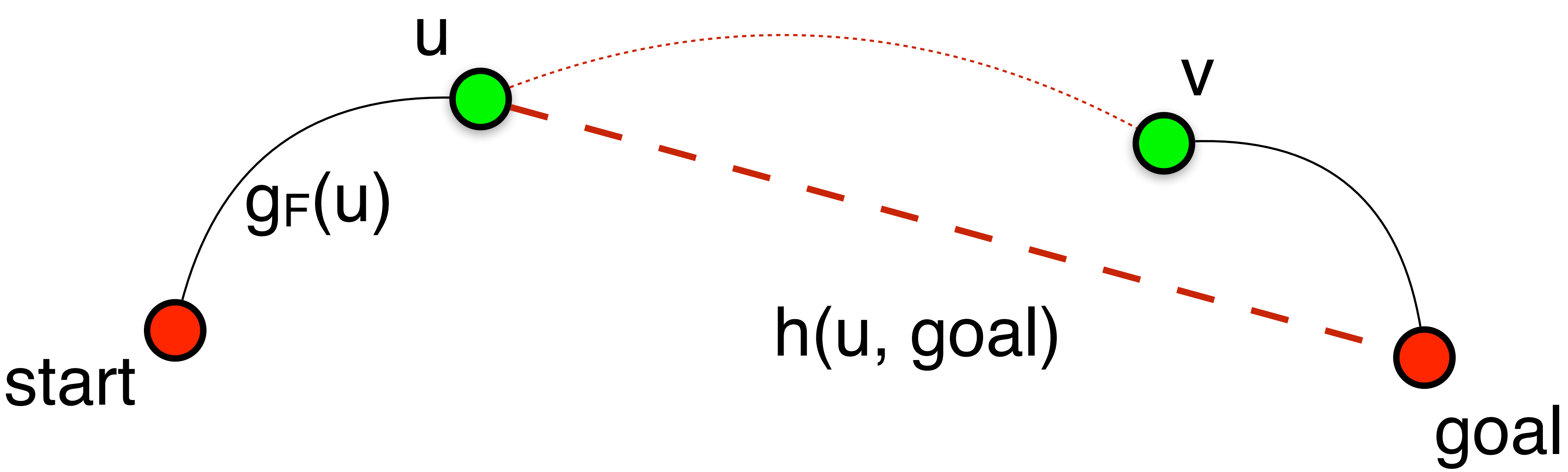
NBS: lower bound



NBS: lower bound

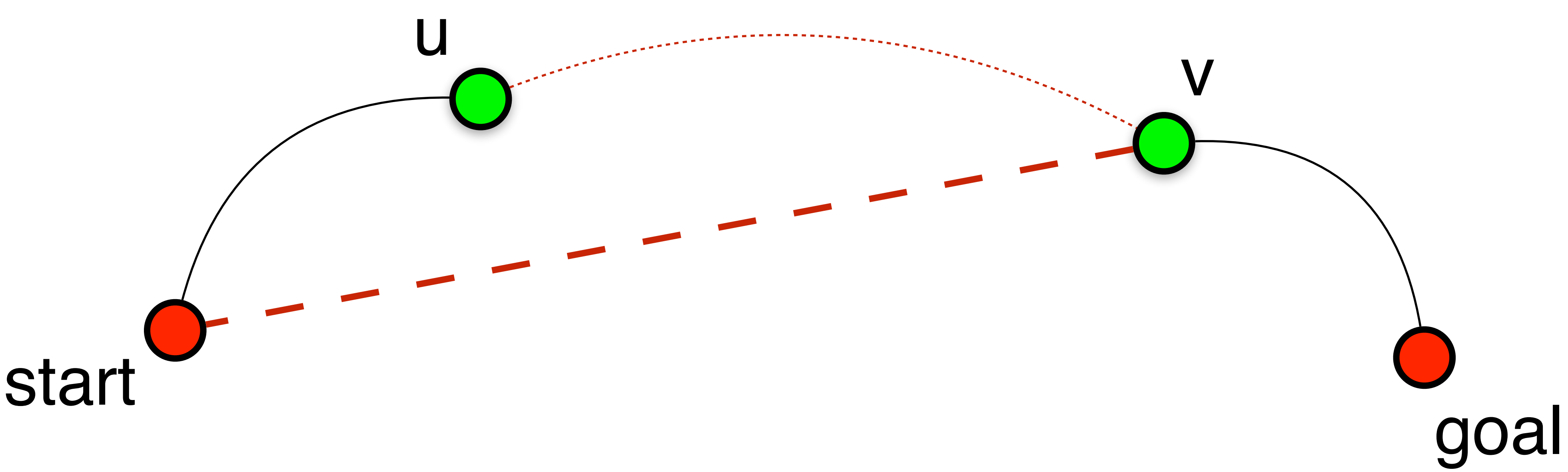


NBS: lower bound

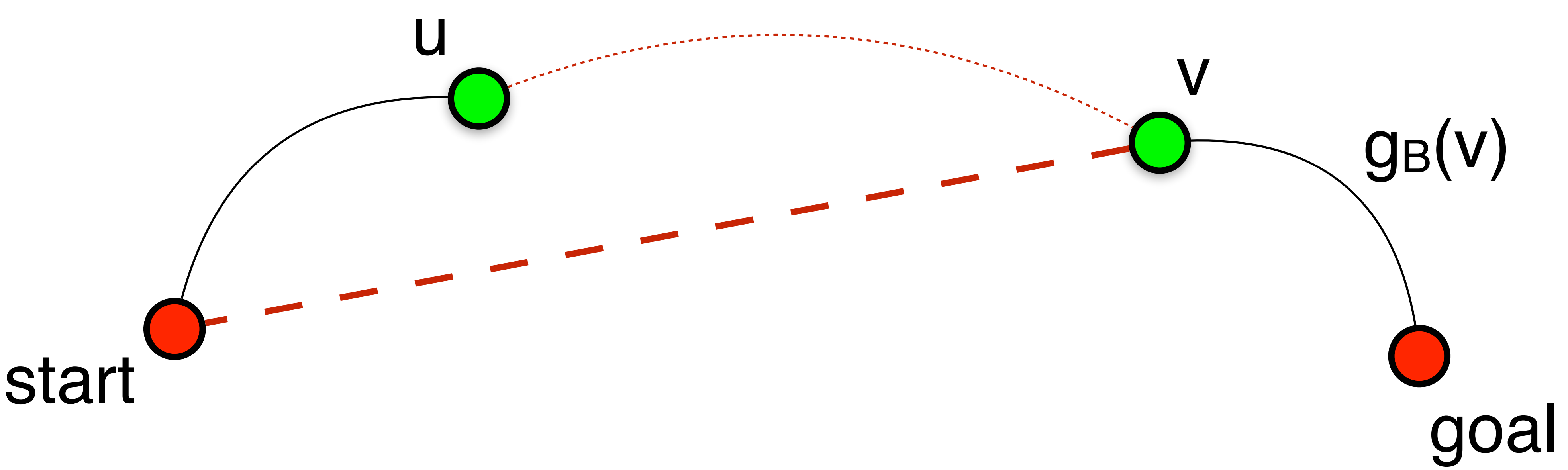


$$f_F(u) = g_F(u) + h(u, goal)$$

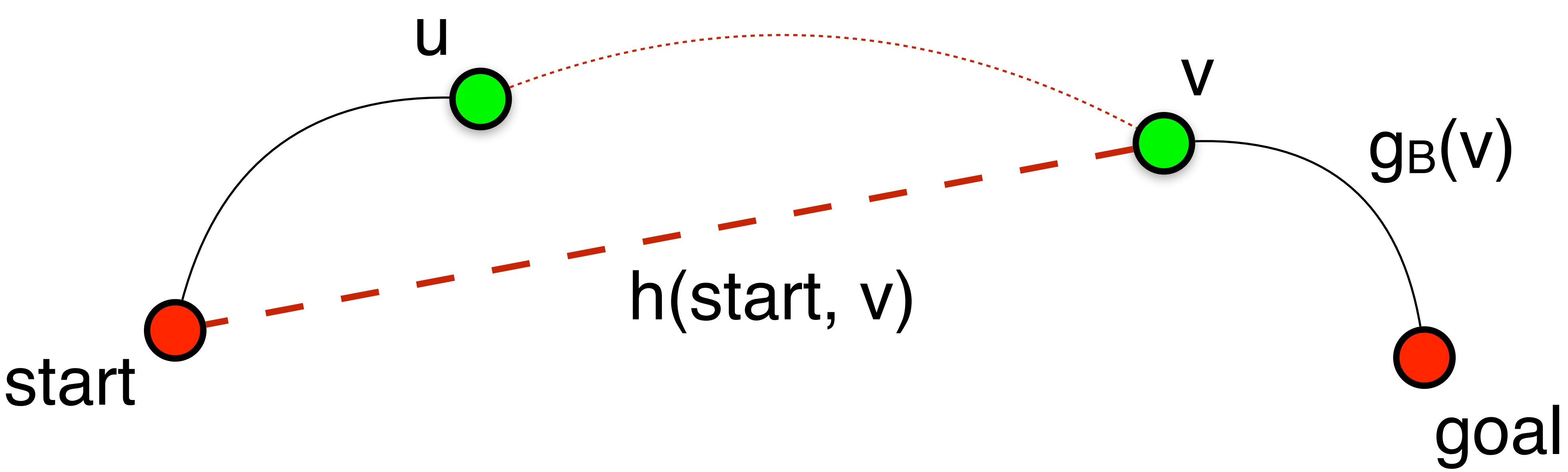
NBS: lower bound



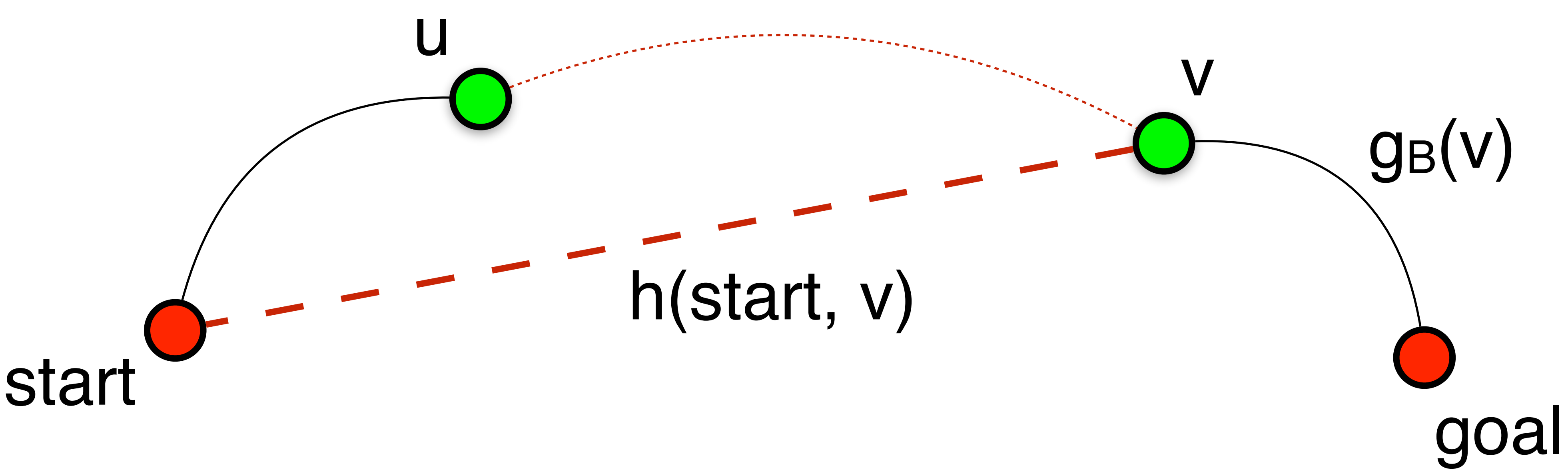
NBS: lower bound



NBS: lower bound

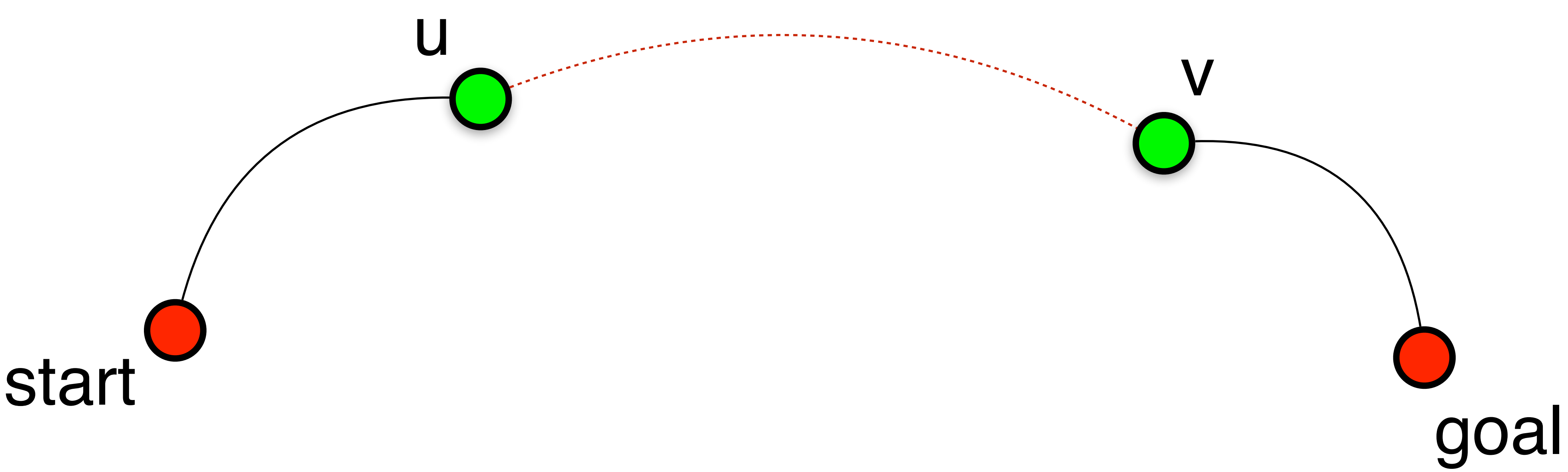


NBS: lower bound

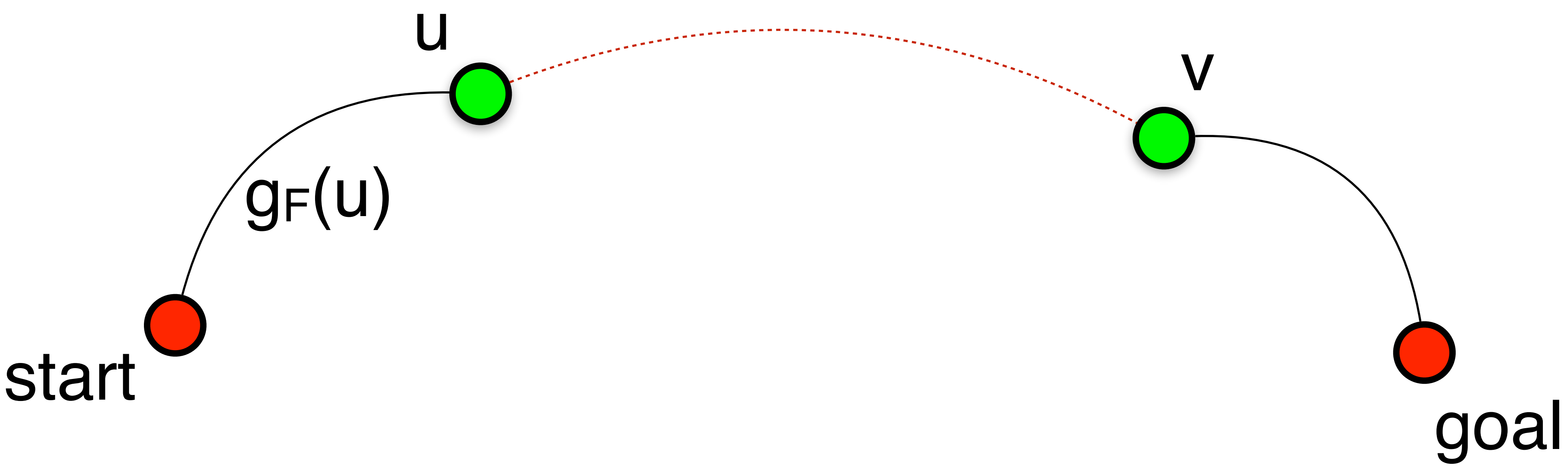


$$f_B(v) = g_B(v) + h(start, v)$$

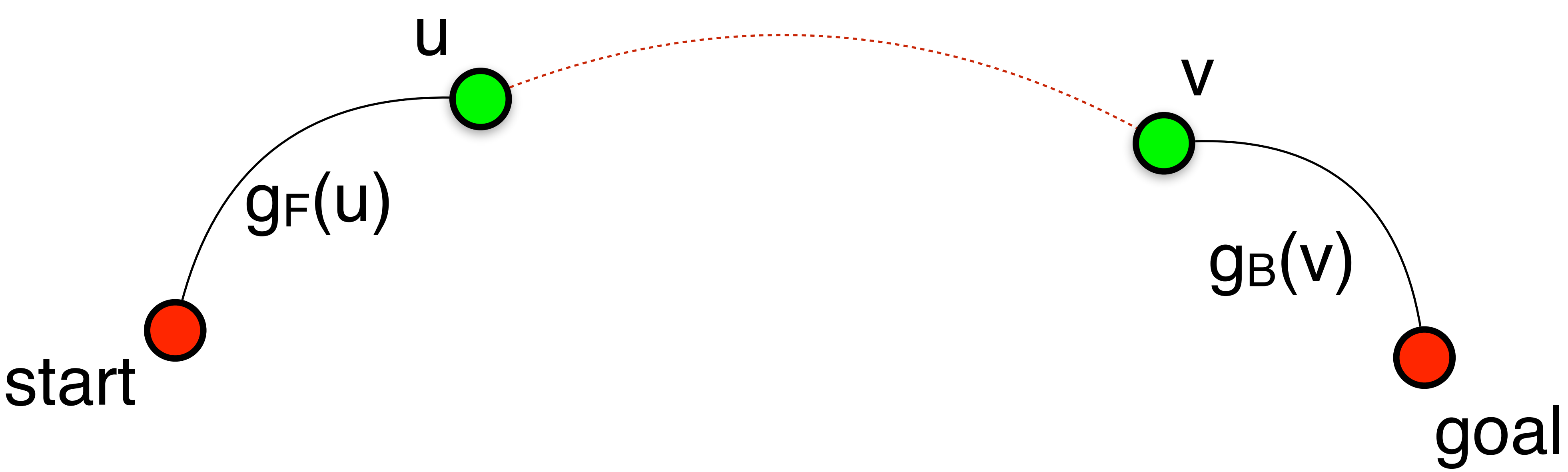
NBS: lower bound



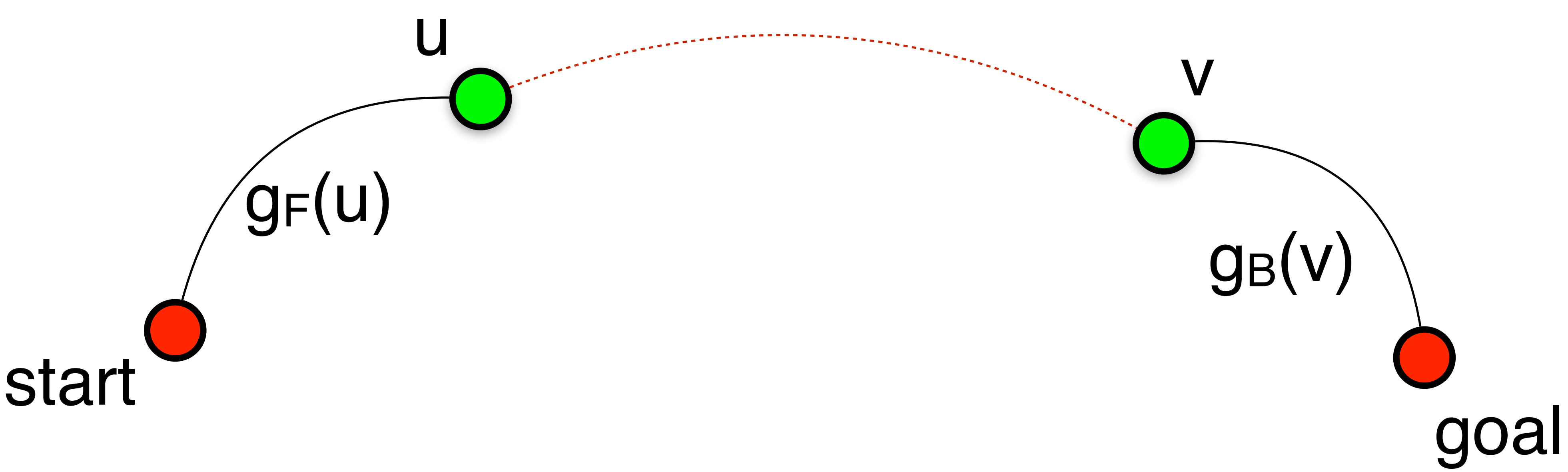
NBS: lower bound



NBS: lower bound



NBS: lower bound



$g_F(u) + g_B(v)$

NBS: lower bound

$$\begin{aligned}\mathrm{lb}(u,v) = max(&f_F(u),\\ &f_B(v),\\ &g_F(u) + g_B(v))\end{aligned}$$

NBS Data Structure

- Can efficiently find pair with minimum lower bound
  - Filter by f-cost then by g-cost

NBS Data Structure

- Can efficiently find pair with minimum lower bound
    - Filter by f-cost then by g-cost

- Cannot just select by f-cost (A*) or g-cost (Dijkstra)

NBS Guarantee

- NBS never expands more than 2x the states expanded by the **best possible** algorithm
  - *In our theoretical framework*
- NBS does equal work in each direction

Suboptimal Solutions

- Use weighted A* if path quality doesn't matter
- Terminate the search when the first solution is found in bidirectional search

Summary / Conclusions

- Use NBS for bidirectional search
- May want bidirectional search for:
  - Weighted terrain
  - Problem Asymmetry
  - Map Asymmetry
  - Local Minima

Questions?

- http://www.movingai.com/GDC18/
  - Open-source implementation of NBS
  - Demo from this lecture*
  - Offline analyzer for analyzing pathfinding
  - Technical reference papers
- Find me on twitter:
  - @nathansttt