Bidirectional Search: Is It For Me?

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

Jingwei Chen  
University of Denver

Robert Holte  
University of Alberta

Sneha Sawlani  
Ariel Felner

Sandra Zilles  
Eshed Shaham
Designed and implemented pathfinding engine
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](http://www.movingai.com/GDC18/test.html)
NBS Details
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

cost-so-far (g-cost)

start

goal
A*: f-cost

cost-so-far (g-cost)

start

estimate to goal (h-cost)

goal
$A^*: f\text{-cost}$

- cost-so-far (g-cost)
- estimate to goal (h-cost)

$f\text{-cost} = g\text{-cost} + h\text{-cost} = \text{estimated path length}$
**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
A* → NBS

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

- Put start/goal onto forward/backward priority queues
- While queues not empty / solution not found
  - Among all states on queues:
    - 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 queues not empty / solution not found
  - Among all states on queues:
    - 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 queues not empty / solution not found
  - Among all states on queues:
    - 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

$g_F(u)$

start

$u$

$v$

goal
NBS: lower bound
NBS: lower bound

\[ f_F(u) = g_F(u) + h(u, \text{goal}) \]
NBS: lower bound
NBS: lower bound
NBS: lower bound

start $\rightarrow$ u $\rightarrow$ v $\rightarrow$ goal

$h(\text{start}, v)$

$g_B(v)$

B
NBS: lower bound

\[ f_B(v) = g_B(v) + h(\text{start}, v) \]
NBS: lower bound
NBS: lower bound

\[ g_F(u) \]
NBS: lower bound
NBS: lower bound

\[ g_F(u) + g_B(v) \]
NBS: lower bound

\[ lb(u, v) = \max(f_F(u), f_B(v), g_F(u) + g_B(v)) \]
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?

  - Open-source implementation of NBS
  - Demo from this lecture*
  - Offline analyzer for analyzing pathfinding
  - Technical reference papers
- Find me on twitter:
  - @nathanstttt