A Brief History and Recent Achievements in Bidirectional Search

Nathan R. Sturtevant, University of Denver
Ariel Felner, Ben Gurion University

Other major contributors:
Jingwei Chen, University of Denver
Eshed Shaham, Ben Gurion University
Robert Holte, University of Alberta
Sandra Zilles, University of Regina
Key Related Work

• 1959 - Dijkstra’s Algorithm
Key Related Work

• 1959 - Dijkstra’s Algorithm
• 1966 - Bidirectional Search (Nicholson & Doran)
Key Related Work

• 1959 - Dijkstra’s Algorithm
• 1966 - Bidirectional Search (Nicholson & Doran)
• 1968 - Heuristic Search (Hart, Nilsson & Raphael)
Key Related Work

• 1959 - Dijkstra’s Algorithm
• 1966 - Bidirectional Search (Nicholson & Doran)
• 1968 - Heuristic Search (Hart, Nilsson & Raphael)
• 1969 - Bidirectional Heuristic Search (Pohl)
Key Related Work

• 1959 - Dijkstra’s Algorithm
• 1966 - Bidirectional Search (Nicholson & Doran)
• 1968 - Heuristic Search (Hart, Nilsson & Raphael)
• 1969 - Bidirectional Heuristic Search (Pohl)
• 1985 - A* Theory (Dechter & Pearl)
Overview

• Bidirectional Theory
  • Eckerle et al, ICAPS 2017

• Optimal algorithm (offline)
  • Shaham et al, SoCS 2017

• Near-optimal algorithm (online)
  • Chen et al, IJCAI 2017
Assumptions

- Front-to-end bidirectional search
- Admissible algorithms
  - Performance with consistent heuristics
- Deterministic, black box algorithm
Unidirectional Theory

• ANY admissible unidirectional search algorithm:
  • Must expand ALL states with:
    • $f(s) = g(s) + h(s) < C^*$
  • Otherwise we can construct instances on which it won’t find the optimal solution
What states must be expanded by all bidirectional algorithms?
Conclusion

• Given a single state $s$
  • There exists a bidirectional algorithm that does not expand $s$
Conclusion

• Given a single state $s$
  • There exists a bidirectional algorithm that does not expand $s$

• Given some pairs of states $(u, v)$
  • We can avoid expanding $u$
  • We can avoid expanding $v$
  • We can’t avoid expanding BOTH $u$ and $v$
High-Level Picture

start

goal
High-Level Picture

start —— u —— goal

start —— v —— goal
High-Level Picture

start

u

v

goal
High-Level Picture

$g_F(u)$
High-Level Picture

\[ g_F(u) \]

\[ h(u, \text{goal}) \]
High-Level Picture

\[ f_F(u) = g_F(u) + h(u, \text{goal}) \]
High-Level Picture

start \quad u \quad v \quad goal
High-Level Picture

\[ h(\text{start}, v) \]

\[ g_B(v) \]
High-Level Picture

\[ f_B(v) = g_B(v) + h(\text{start}, v) \]
High-Level Picture

\text{start} \rightarrow u \rightarrow v \rightarrow \text{goal}
High-Level Picture

start \rightarrow g_F(u) \rightarrow u \rightarrow v \rightarrow \text{goal}
High-Level Picture

\[ g_F(u) \quad \quad \quad \quad g_B(v) \]

start \quad \quad u \quad \quad v \quad \quad goal
Theorem

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

- If \( lb(u, v) < C^* \) then we must expand either \( u \) or \( v \)
- Leads implicitly to termination conditions

$f_F(u) < C^*$
\[ f_B(v) < C^* \]
\[ g_F(u) + g_B(v) < C^* \]
$g_F(u) + g_B(v) < C^*$
\[ g_F(u) + g_B(v) < C^* \]
$g_F(u) + g_B(v) < C^*$

$C^* = 10.5$
\( C^* = 10.5 \)

\[
g_F(u) + g_B(v) < C^* \]
\[ C^* = 10.5 \]

\[ g_F(u) + g_B(v) < C^* \]
$C^* = 10.5$
$C^* = 10.5$
Total Work: 25

$C^* = 10.5$
Total Work: 27

C* = 10.5
Total Work: 28

C* = 10.5
Total Work: 27

C* = 10.5
Total Work: 22

\[ C^* = 10.5 \]
Total Work: 21

C* = 10.5
Total Work: 18

C* = 10.5
Total Work: 17

C* = 10.5
Total Work: 15

C* = 10.5
Total Work: 14

C* = 10.5
Fractional MM

• Takes a parameter $f$
  • Cost of the state space to explore in each direction
  • Costs correspond to different vertex covers

• We can (offline) compute the best algorithm for a given search problem
Vertex Cover on a Bipartite Graph

• Approximation algorithm:
  • Repeat until all vertices covered
    • Choose any edge/line with uncovered vertex
    • Place both states into vertex cover

• Gives 2x approximation to optimal vertex cover
  • (Papadimitriou & Steiglitz, 1982)
Using this algorithm

• We don’t know the full graph ahead of time
Using this algorithm

• We don’t know the full graph ahead of time
  • Build the graph as we go
Using this algorithm

- We don’t know the full graph ahead of time
  - Build the graph as we go
- We don’t know the optimal solution cost
Using this algorithm

• We don’t know the full graph ahead of time
  • Build the graph as we go

• We don’t know the optimal solution cost
  • Must estimate $C^*$
Using this algorithm

• We don’t know the full graph ahead of time
  • Build the graph as we go

• We don’t know the optimal solution cost
  • Must estimate C*

• We must avoid re-expanding states
Using this algorithm

• We don’t know the full graph ahead of time
  • Build the graph as we go

• We don’t know the optimal solution cost
  • Must estimate C*

• We must avoid re-expanding states
  • Carefully order state expansions
Using this algorithm

• We don’t know the full graph ahead of time
  • Build the graph as we go

• We don’t know the optimal solution cost
  • Must estimate $C^*$

• We must avoid re-expanding states
  • Carefully order state expansions

• Computing $lb(u, v)$ could be expensive
Using this algorithm

• We don’t know the full graph ahead of time
  • Build the graph as we go

• We don’t know the optimal solution cost
  • Must estimate C*

• We must avoid re-expanding states
  • Carefully order state expansions

• Computing \( l_b(u, v) \) could be expensive
  • Efficient data structures
NBS

- Put start/goal onto forward/backward priority queues
- While forward/backward not empty
  - Among all state on queues:
    - Select the pair with lowest $lb$
    - Expand both of them
  - Terminate when $lb \geq$ best path
- Gives $2x$ bound on optimal number of expansions
  - Bound is tight

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
Necessary Node Expansions (brc203d)

- ○ Instance
- dashed line $y=2x$
- dotted line $y=x/2$

<table>
<thead>
<tr>
<th>Necessary Expansions by NBS</th>
<th>Necessary Expansions by A*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5k</td>
<td>20k</td>
</tr>
<tr>
<td>10k</td>
<td>15k</td>
</tr>
<tr>
<td>15k</td>
<td>10k</td>
</tr>
<tr>
<td>20k</td>
<td>5k</td>
</tr>
</tbody>
</table>
Summary

• Theory
  • First definition of necessary node expansions
  • fMM - implements optimal bidirectional search
• Practice
  • Near-optimal approach (NBS)
  • Node expansions are bounded by 2x optimal
• Demos & videos will appear at:
  • https://www.movingai.com
Open Questions

• What can we learn about bidirectional search from the minimum vertex cover?

• Is there an algorithm with better average performance?

• Efficient front-to-front search?