IJCAI 2018 Tutorial on Heuristic Search

Nathan Sturtevant, Daniel Harabor,
Sven Koenig, Ariel Felner

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:
    - Bidirectional Search

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:
    - Bidirectional Search
    - Heuristics and Constraints

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:
    - Bidirectional Search
    - Heuristics and Constraints
    - Optimal and suboptimal search

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:
    - Bidirectional Search
    - Heuristics and Constraints
    - Optimal and suboptimal search
    - Any-angle search

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:
    - Bidirectional Search
    - Heuristics and Constraints
    - Optimal and suboptimal search
    - Any-angle search
  - Tools for teaching search concepts

Overview

- This afternoon you will get:
  - Overview of current state-of-art in heuristic search
  - Current topics:
    - Bidirectional Search
    - Heuristics and Constraints
    - Optimal and suboptimal search
    - Any-angle search
  - Tools for teaching search concepts
    - Web demos you can use

Overview

- Exponential Domains
  - Algorithms & Heuristics
- Polynomial Domains
  - Algorithms, Heuristics & Constraints
- Bidirectional Search
  - Theory & Algorithms

Demos

https://movingai.com/SAS/

Single-Agent Search Demo Page

This page lists demos that can be used for teaching and exploring algorithms in single-agent search.

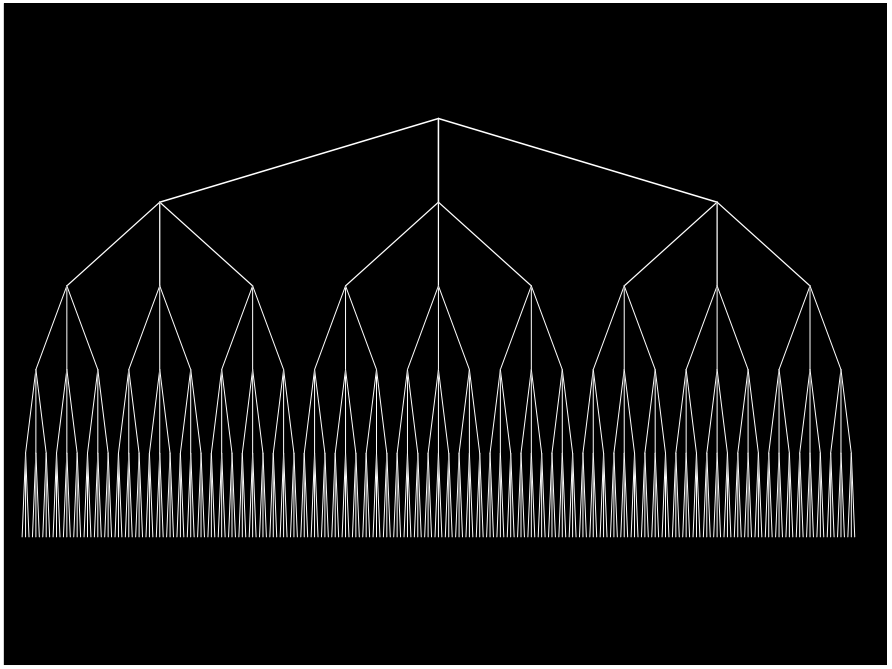
- Search Algorithms
  - DFS, BFS, DFID on an exponential tree
  - Dijkstra, A* and Weighted A* on a graph
  - A* on a grid map
  - IDA* on the 3x2 Sliding tile puzzle
  - NBS on a grid map
  - JPS on a grid map
  - Abstraction and refinment in grid maps
- Heuristics
  - IDA* with Pattern Databases
  - IDA* with Min-Compressed Pattern Databases
  - A* with inconsistent heuristics
  - Differential Heuristics
  - Transit Routing
- Constraints
  - Reach
  - Bounding Boxes
- Theory
  - Necessary expansions in unidirectional and bidirectional search
  - The Must-Expand graph (GMX) for bidirectional search
  - Fractional meeting points for bidirectional search algorithms

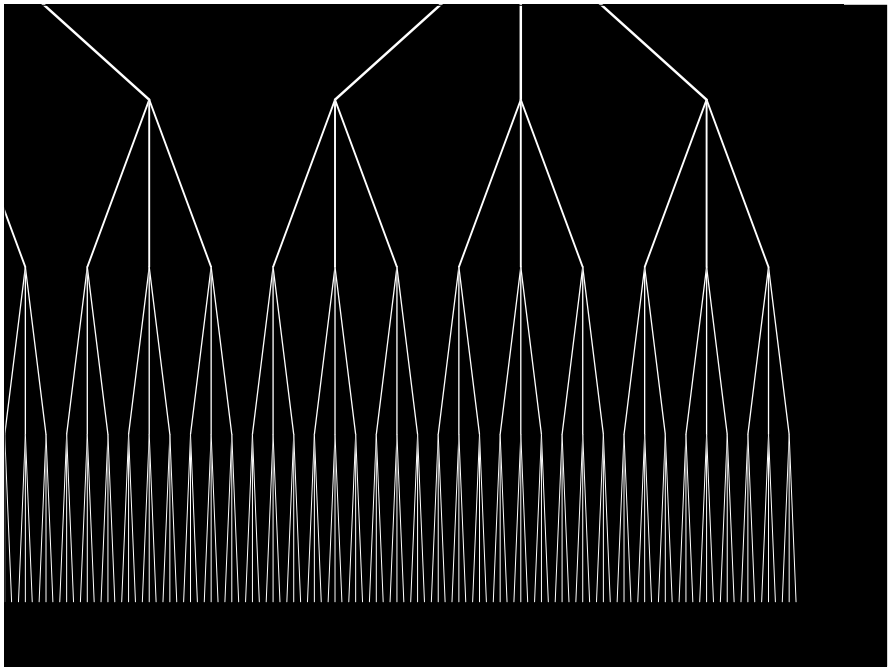
Exponential Domains

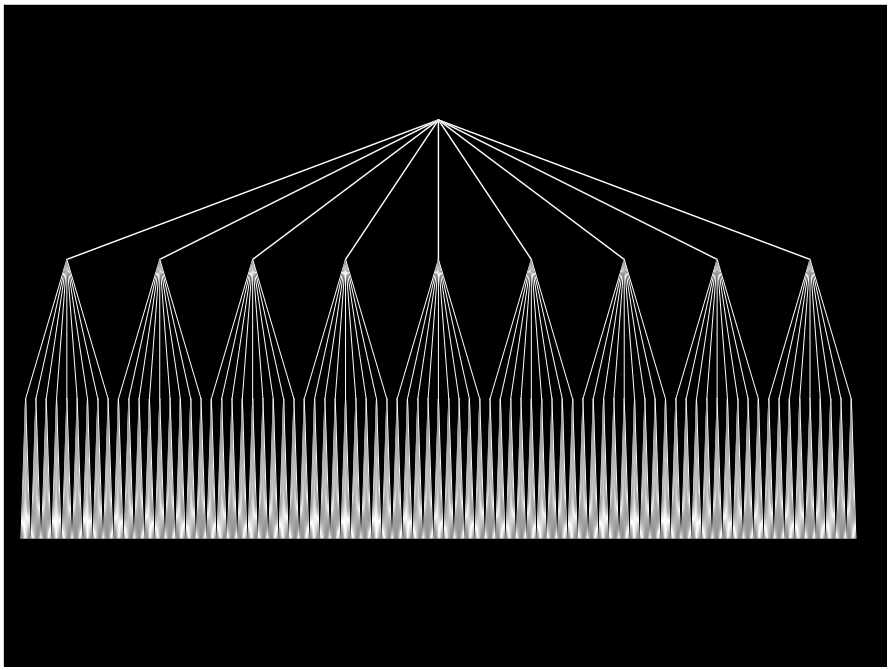
UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

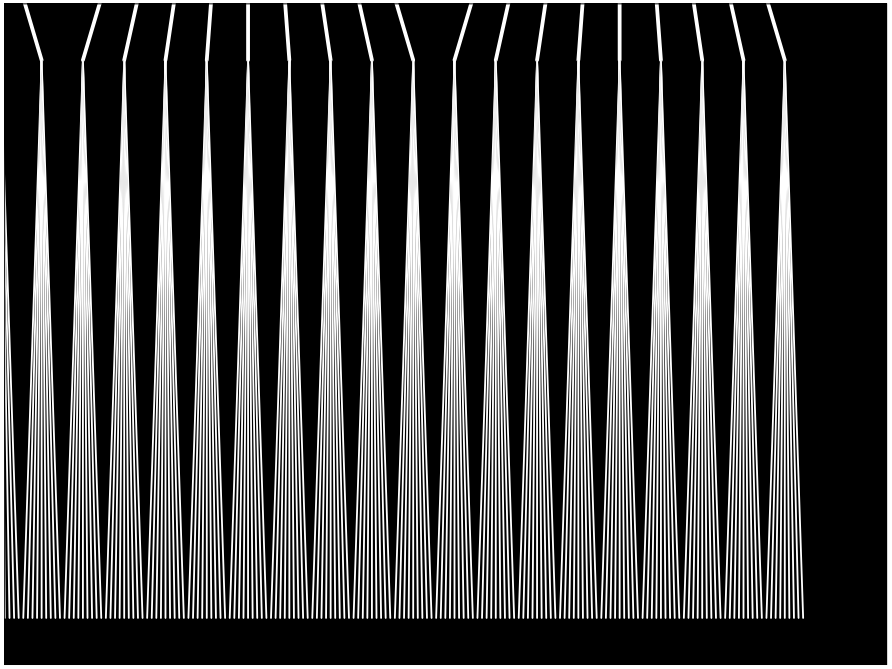
Exponential Domains

- Characterized by:
  - branching factor *b*
  - depth of tree, *d*









$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$
$$(b-1)N(b,d) = b^{d+1} - 1$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$
$$(b-1)N(b,d) = b^{d+1} - 1$$

$$N(b,d) = \frac{b^{d+1} - 1}{(b-1)}$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$
$$(b-1)N(b,d) = b^{d+1} - 1$$
$$N(b,d) = \frac{b^{d+1} - 1}{(b-1)}$$

$$N(b,d) = \frac{b(b^d) - 1}{(b-1)}$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$
$$(b-1)N(b,d) = b^{d+1} - 1$$
$$N(b,d) = \frac{b^{d+1} - 1}{(b-1)}$$
$$N(b,d) = \frac{b(b^d) - 1}{(b-1)}$$

$$N(b,d) = \frac{b}{b-1}(b^d) - \frac{1}{b-1}$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$
$$(b-1)N(b,d) = b^{d+1} - 1$$
$$N(b,d) = \frac{b^{d+1} - 1}{(b-1)}$$

$$N(b,d) = \frac{b(b^d) - 1}{(b-1)}$$

$$N(b,d) = \frac{b}{b-1}(b^d) - \frac{1}{b-1}$$
$$N(b,d) \approx \frac{b}{b-1}(b^d)$$

$$N(b,d) = 1 + b + b^2 + ... + b^{d-1} + b^d$$
$$bN(b,d) = b + b^2 + b^3 + ... + b^d + b^{d+1}$$
$$bN(b,d) - N(b,d) = b^{d+1} - 1$$
$$(b-1)N(b,d) = b^{d+1} - 1$$
$$N(b,d) = \frac{b^{d+1} - 1}{(b-1)}$$

$$N(b,d) = \frac{b(b^d) - 1}{(b-1)}$$

$$N(b,d) = \frac{b}{b-1}(b^d) - \frac{1}{b-1}$$
$$N(b,d) \approx \frac{b}{b-1}(b^d)$$

Uninformed Algorithms

- Basic graph algorithms
  - DFS
    - Not necessarily optimal if goal not at a leaf
  - BFS
    - High memory requirement

Uninformed Algorithms

- DFID
  - Depth-limited DFS
  - Increment the depth after each search
- Asymptotically optimal in node expansions and time
  - Assuming no other information (eg heuristic)

Demo

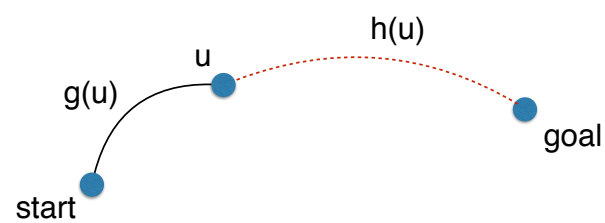
Demo tasks

- Choose a goal where DFS is faster than DFID
  - How much faster?
- Choose a goal where DFID is faster than DFS
  - How much faster?

Heuristics

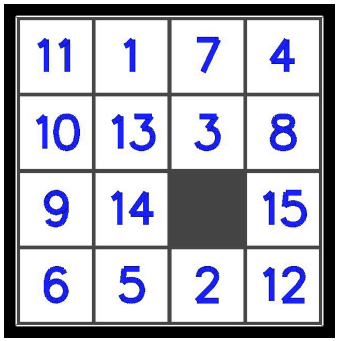
- Heuristic function maps a state to a *distance/cost*
  - Estimate of *distance/cost* to the goal
- Admissible heuristic doesn't overestimate true cost
  - h(s, g) ≤ c(s, g)
- Consistent heuristic obeys triangle inequality (undirected graphs)
  - $|h(a,g) - h(b,g)| \leq c(a,b)$

Heuristic Picture



Sample Domain/Heuristic

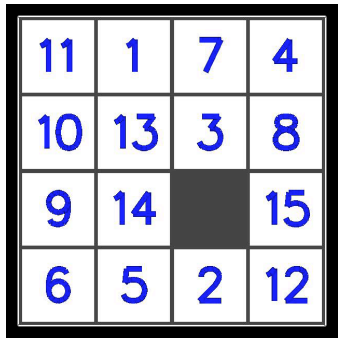
- Sliding tile puzzle
  - Manhattan distance
  - Sum of distances from each tile to its goal location



UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Sample Domain/Heuristic

- Sliding tile puzzle
  - Manhattan distance
  - Sum of distances from each tile to its goal location



UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

IDA*

- Instead of iteratively searching depth layers (DFID)
  - (Korf, 1985)
- Search f-cost layers
  - $f(n) = g(n)+h(n)$
  - Estimate of the total path length
  - Searching from shortest to longest paths

Demo

Demo Tasks

- What states have low heuristic value but are far from the goal?
- Does IDA* ever visit the same state twice in the same iteration?

Pattern Database

- Assume there is a fixed goal state
  - Precompute distances to goal state
  - (Culberson and Schaeffer, 1996)
- Pattern database
  - Build an abstract state space
  - BFS from the goal in the abstract space
  - Abstract distances are heuristics to the goal

Domain Abstraction

- Take states and replace some values with blanks / colors
  - (0 1 2 3 4 5 6 7 8 9)
  - (0 1 2 3 4 5 6 7 8 9)
  - (0 * 2 * 4 * 6 * 8 *)
- Extreme example
  - (0 * * * * * * * * *)
- ϕ( ) is this mapping function

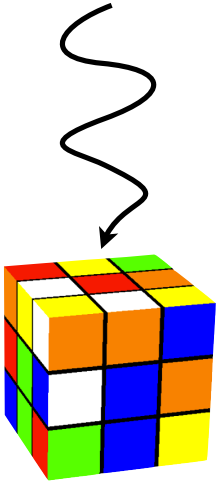
PDB Heuristic

- When we need the heuristic for a state s
  - Compute ϕ(s)
  - Lookup the distance from ϕ(s) to the goal
  - Return the distance as the heuristic value

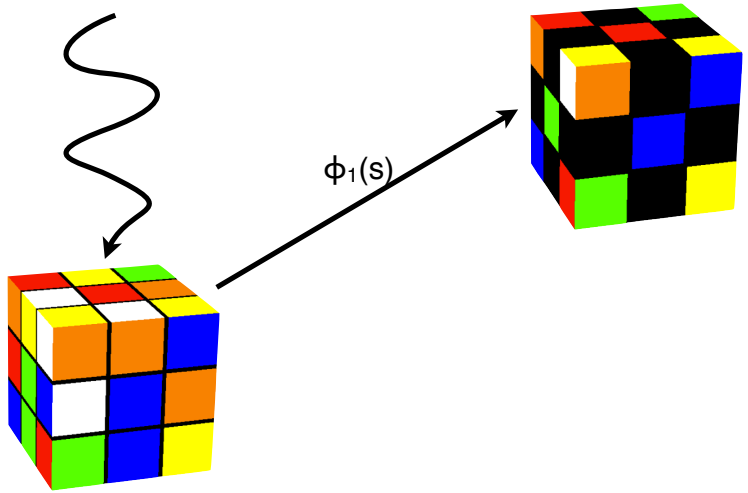
PDBs: Use during search

PDBs: Use during search

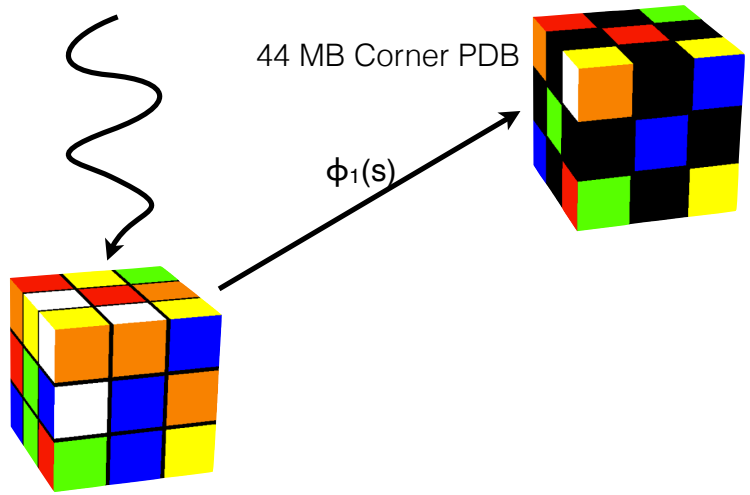
PDBs: Use during search



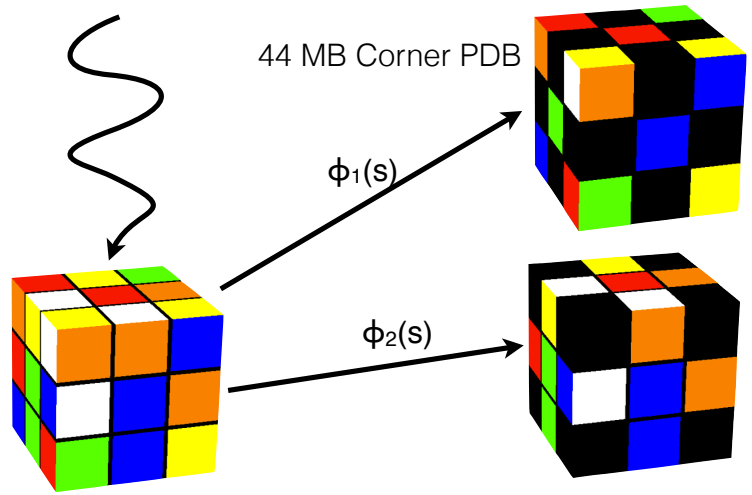
PDBs: Use during search



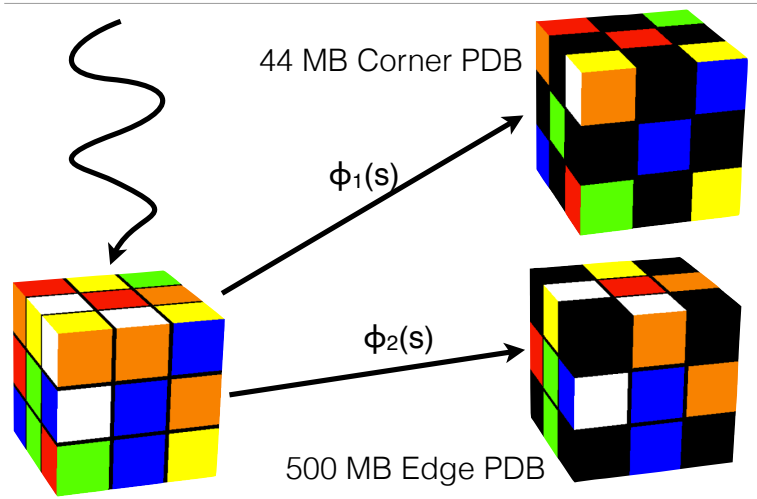
PDBs: Use during search



PDBs: Use during search



PDBs: Use during search



Demo

Demo Tasks

- What is better, a pattern database (PDB) or manhattan distance (MD)?
  - Where are PDBs weak?
  - Where is MD weak?

Maximum heuristic after abstraction

$$b^h = b^d/k$$

Maximum heuristic after abstraction

$$b^h = b^d/k$$

$$\log(b^h) = \log(b^d) - \log(k)$$

Maximum heuristic after abstraction

$$b^h = b^d/k$$

$$\log(b^h) = \log(b^d) - \log(k)$$

$$h\log(b) = d\log(b) - \log(k)$$

Maximum heuristic after abstraction

$$b^h = b^d/k$$
$$\log(b^h) = \log(b^d) - \log(k)$$
$$h\log(b) = d\log(b) - \log(k)$$
$$h = d - \log(k)/log(b)$$

Maximum heuristic after abstraction

$$b^h = b^d/k$$
$$\log(b^h) = \log(b^d) - \log(k)$$
$$h\log(b) = d\log(b) - \log(k)$$
$$h = d - \log(k)/log(b)$$
$$h = d - \log_b(k)$$

Maximum heuristic after abstraction

$$b^h = b^d/k$$
$$\log(b^h) = \log(b^d) - \log(k)$$
$$h\log(b) = d\log(b) - \log(k)$$
$$h = d - \log(k)/log(b)$$
$$h = d - \log_b(k)$$

Polynomial Domains

Polynomial Domains

- Exponential domains grow as b^d
- Polynomial domains grow according to dimension of state space
  - r^2, r^3, d^k

Polynomial Domains

- Suppose a branching factor b
  - If we search to depth d
    - b^d states or d^k states?
  - Difference is due to duplicate detection

Duplicate Detection

- Most exponential domains aren't actually trees
- Most polynomial domains contain many duplicates
- Solution:
  - *Closed list* to detect duplicates

Best First Search

- Put start onto OPEN
- While OPEN not empty / solution not found
  - Among all states on OPEN:
    - Select the state with lowest **cost**
    - Expand it
- Expand:
  - For each successor s
    - if s on CLOSED, discard
    - else place/update s on OPEN

A* / Best First Search

- Best measure is by f-cost
  - f(n) = g(n) + h(n)
- Prioritize states that have a shorter path length/cost estimate

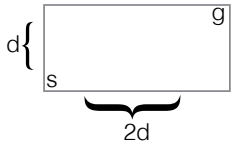
Demo

Demo task

- Build a graph where a state is first generated with suboptimal g-cost
  - (eg the f-cost of a state on open decreases)

How many duplicates?

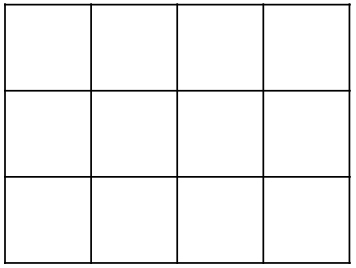
- How many unique paths between two states in an octile map?
  - Assume a 2d x d rectangle with start/goal in the corners



$$\binom{2d}{d} = \frac{(2d)!}{2(d!)}$$

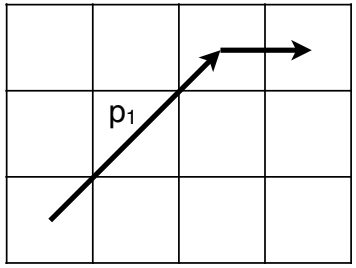
Canonical Ordering of Paths

- Order all optimal paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to cardinal actions



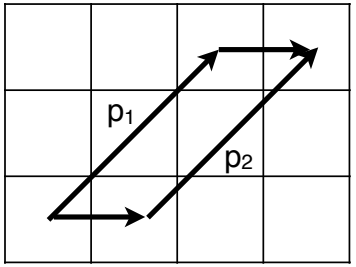
Canonical Ordering of Paths

- Order all optimal paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to cardinal actions

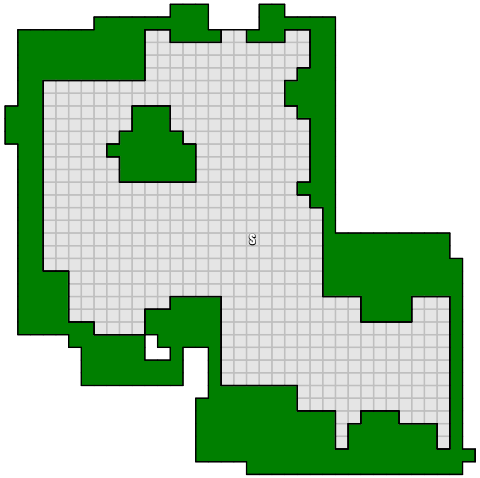


Canonical Ordering of Paths

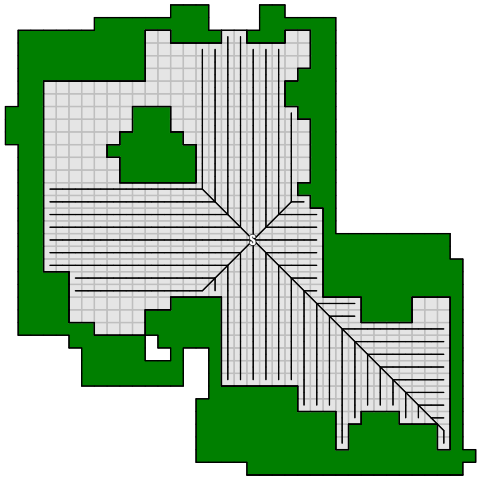
- Order all optimal paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to cardinal actions



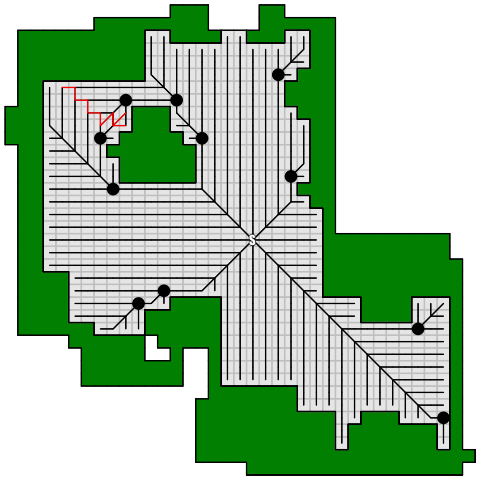
Basic Map



Simple Canonical Ordering



Canonical Ordering



Algorithms

- Canonical A* (Sturtevant & Rabin, 2016)
  - Search using canonical ordering
- Jump Point Search (JPS) (Harabor & Grastien, 2011)
  - Following canonical ordering to jump point, and only point jump points on open list
- Bounded JPS (Sturtevant & Rabin, 2016)
  - Following canonical ordering depth k

Algorithms

- Canonical A* BJPS(0)
  - Search using canonical ordering
- Jump Point Search (JPS) BJPS(∞)
  - Following canonical ordering to jump point, and only point jump points on open list
- Bounded JPS BJPS(k)
  - Following canonical ordering depth k

Algorithms

- Canonical A* BJPS(0)
  - Search using canonical ordering
- Jump Point Search (JPS) BJPS(∞)
  - Following canonical ordering to jump point, and only point jump points on open list
- Bounded JPS BJPS(k)
  - Following canonical ordering depth k
- Canonical Dijkstra
  - 4x faster single-source shortest path

Demo

Demo Tasks

- Print grid map and ask students to:
  - Draw canonical ordering
  - Label jump points
- Does JPS put a state on OPEN that A* would not?
- How does JPS generate a node on different paths?

Polynomial Heuristic

- $h^d = r^d/k$ [d = dimension of map]
- For d = 2 $h = \frac{r}{\sqrt{k}}$
- BFS in *abstract state space* will not be effective
- Need to store exact distances

Polynomial State Spaces

- Often explicit:
  - Road networks
  - Game maps
  - Social network
- Can often afford O(n) memory with *n* nodes in graph

Heuristic Approaches

- Different heuristics in polynomial/exponential domains
  - Global Network Positioning (Ng & Zhang, 2001)
  - ALT (Goldberg & Harrelson, 2005)
  - True-Distance Heuristics; **Differential Heuristic** (Sturtevant, et al, 2009)
  - 1-Dimensional Euclidean Heuristic (Rayner, et al, 2011)
  - Compressed DH (Goldenberg et al, 2011)
  - Subset Selection of Search Heuristics (Rayner et al, 2013)
  - **FastMap** (Cohen et al, 2018)

Distance Approximation

- Suppose undirected graph
- Suppose we know $d(p_1, x)$ for all x
- How can we use this to estimate the distance between any states a & b?

Distance Approximation

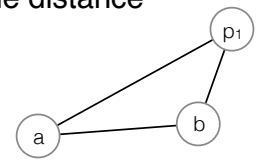
- Suppose undirected graph
- Suppose we know $d(p_1, x)$ for all x
- How can we use this to estimate the distance between any states a & b?
  - Triangle inequality:

$$|d(a, p_1) - d(b, p_1)| \leq d(a, b)$$

Distance Approximation

- Suppose undirected graph
- Suppose we know $d(p_1, x)$ for all x
- How can we use this to estimate the distance between any states a & b?
  - Triangle inequality:

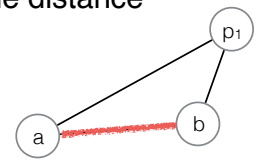
$$|d(a, p_1) - d(b, p_1)| \leq d(a, b)$$



Distance Approximation

- Suppose undirected graph
- Suppose we know $d(p_1, x)$ for all x
- How can we use this to estimate the distance between any states a & b?
  - Triangle inequality:

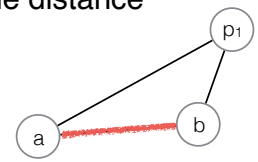
$$|d(a, p_1) - d(b, p_1)| \leq d(a, b)$$



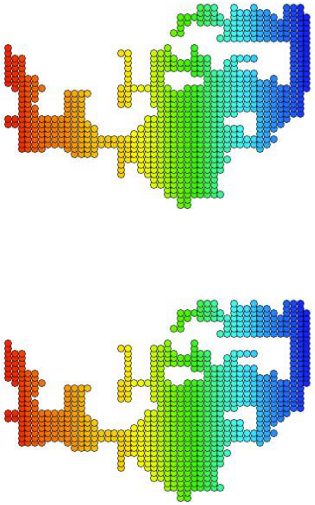
Distance Approximation

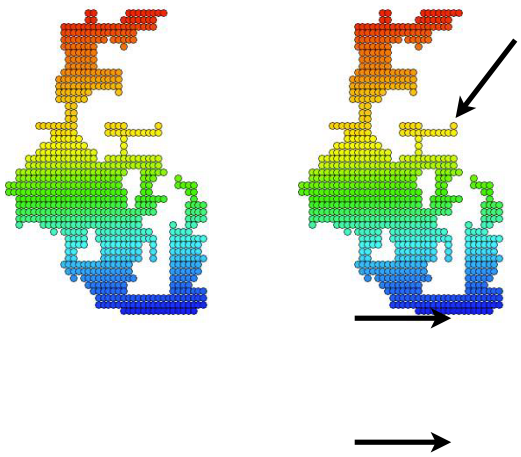
- Suppose undirected graph
- Suppose we know $d(p_1, x)$ for all x
- How can we use this to estimate the distance between any states a & b?
  - Triangle inequality:

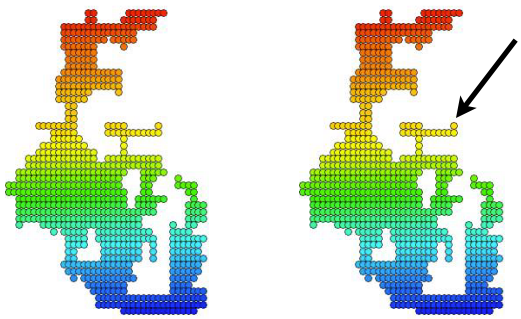
$$|d(a, p_1) - d(b, p_1)| \leq d(a, b)$$

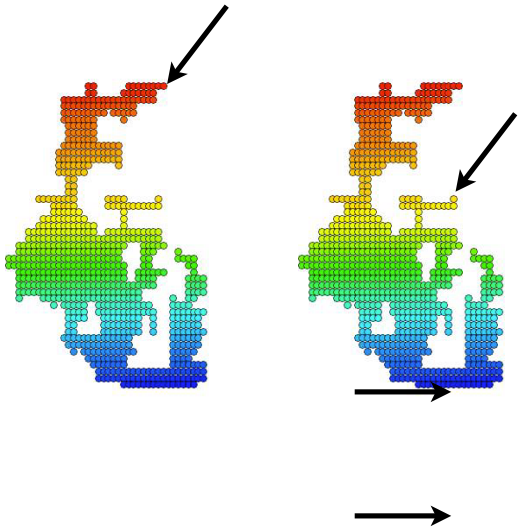


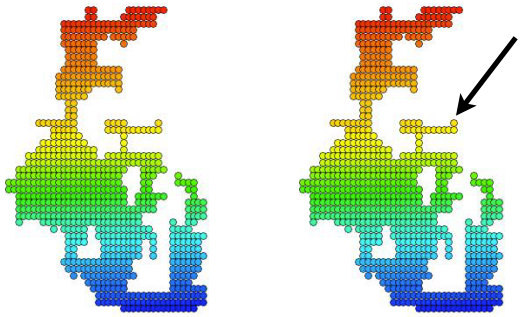
- Key question:
  - Where to place p_i

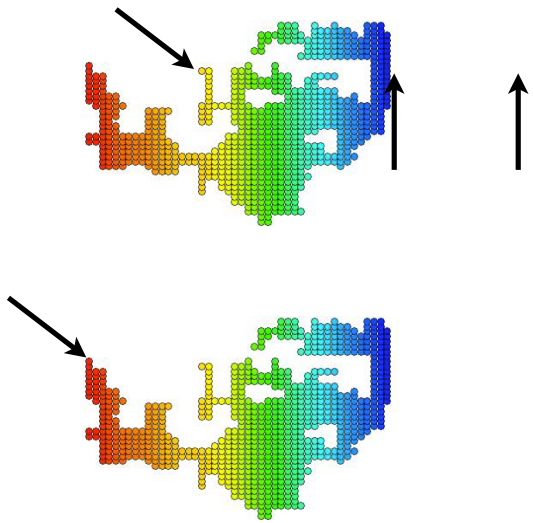


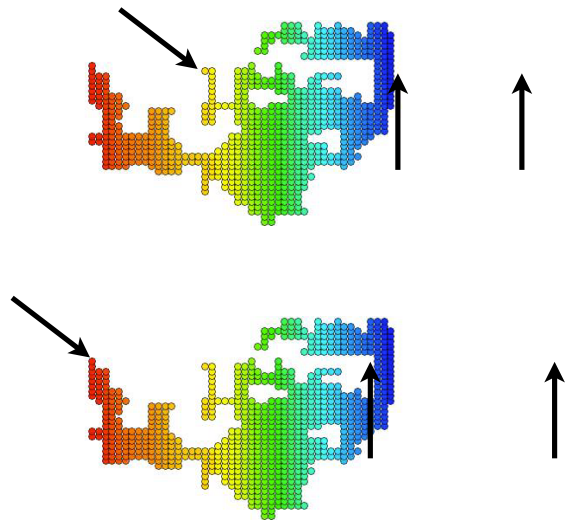












UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Selecting the next pivot

- Use greedy approach (Rayer et al, 2013)
  - Choose several possible pivots
  - Add the one that maximizes the information gain (increase in heuristic value)

Demo

Demo Tasks

- For a given pivot:
  - What path would be easy?
  - What path would be hard?
- For a given path:
  - What is a good pivot?
  - What is a bad pivot?

Constraints

- Heuristic knowledge is contained in distance function
- Some information is not easily represented in distances
- Given start, goal, current state, etc
  - Should we generate a particular successor?
  - Only if the constraints allow us to

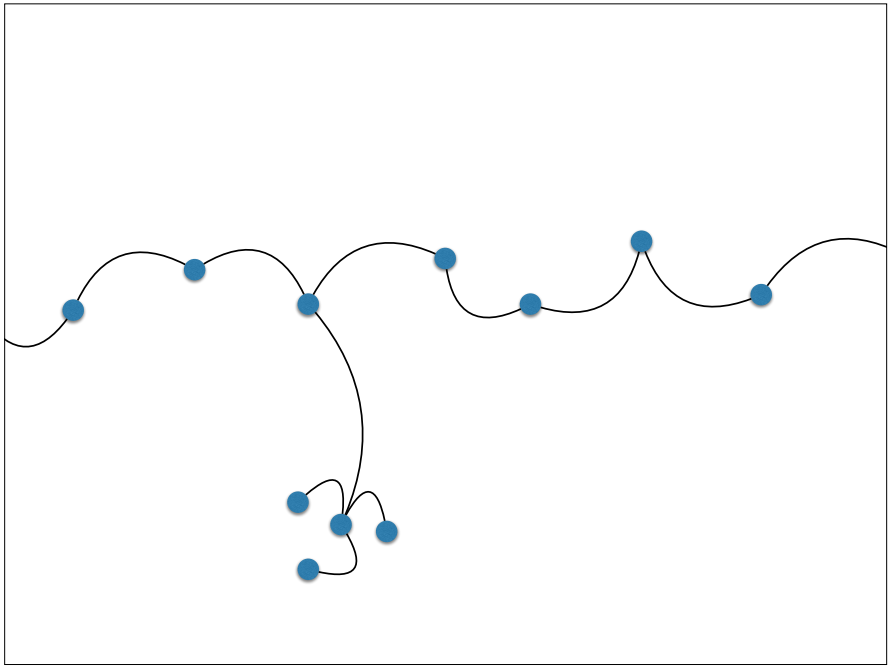
Reach

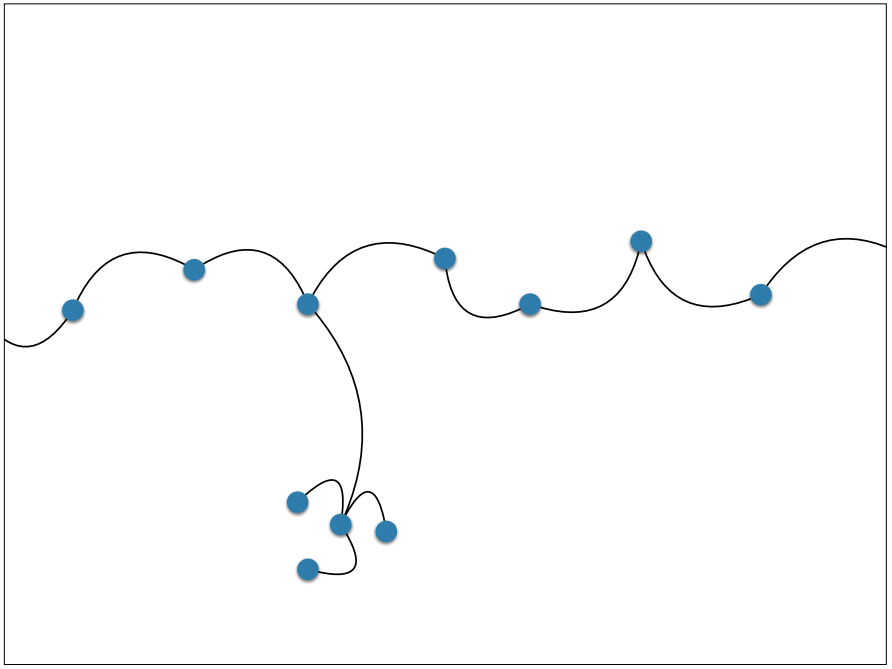
- Given a path P from *s* to *t* and a vertex *v* on P, the reach of *v* with respect to P is the minimum of the length of the prefix of P (the subpath from *s* to *v*) and the length of the suffix of P (the subpath from *v* to *t*).
- The reach of *v*, r(*v*), is the maximum, over all shortest paths P through *v*, of the reach of *v* with respect to P.
- (Goldberg, Kaplan & Werneck, 2005)

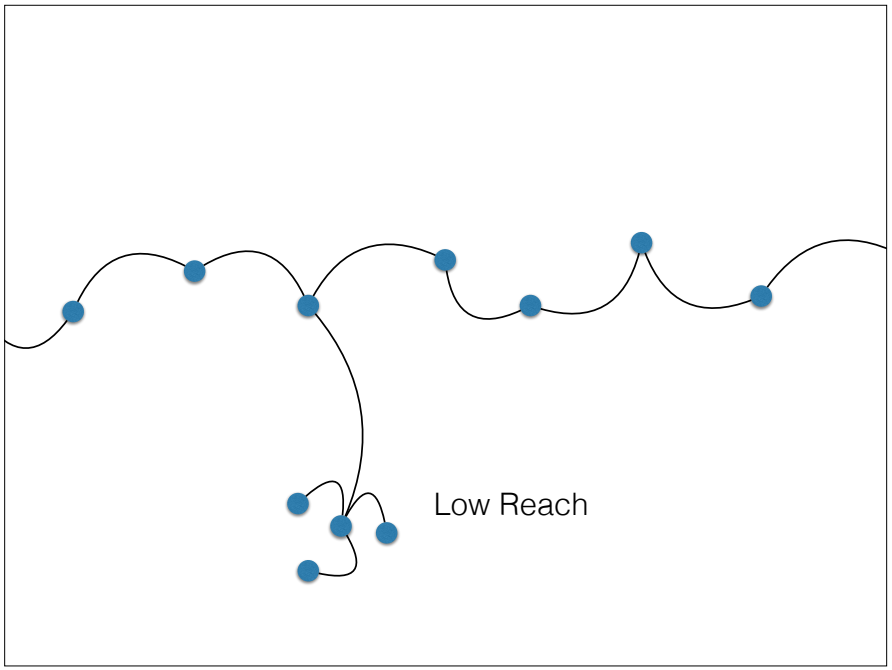
(Unidirected) Reach

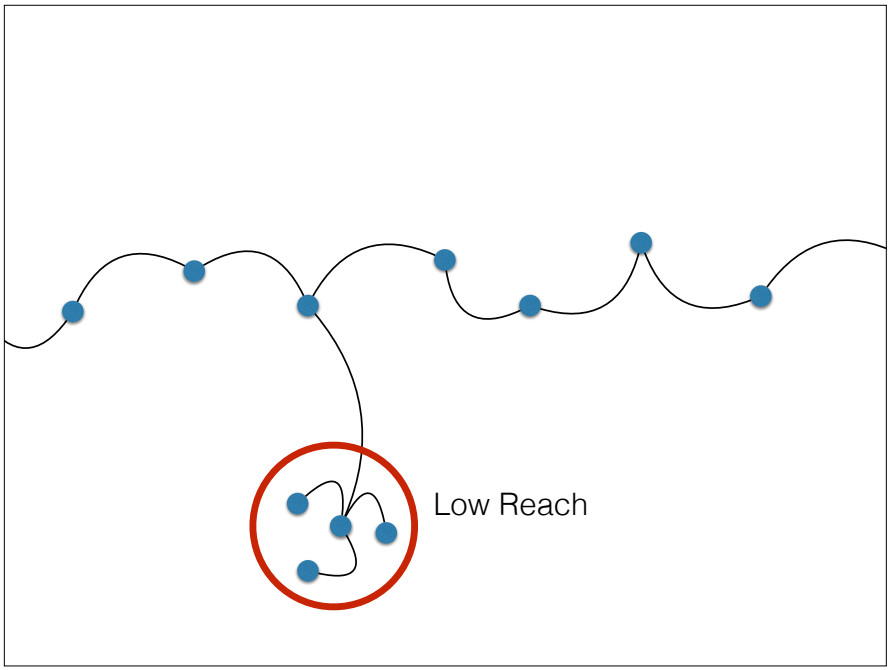
- Each node has reach *r*
- For an optimal path to go through node n
  - dist(start, n) ≤ r

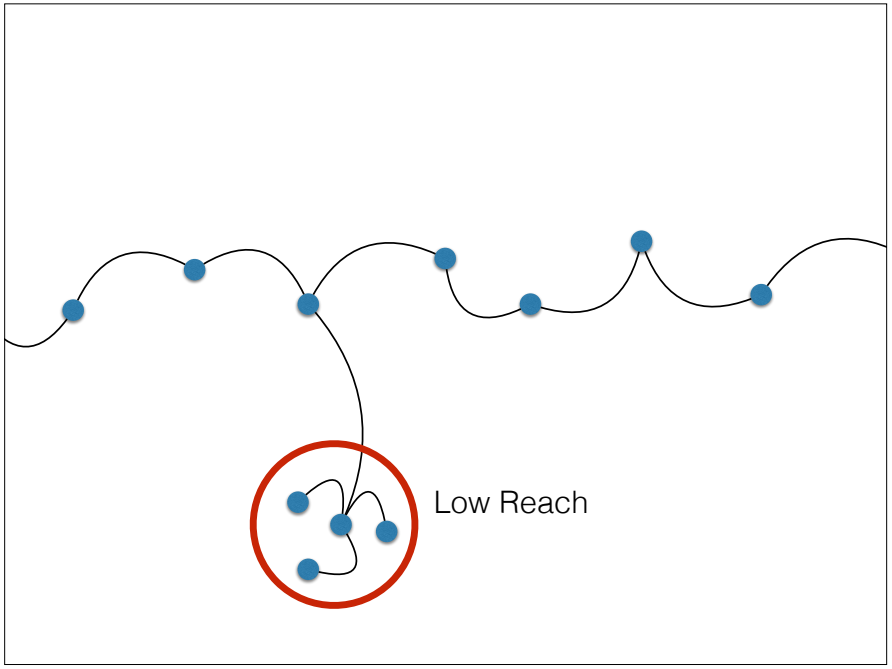
    or
  - dist(n, goal) ≤ r
- (Goldberg, Kaplan & Werneck, 2005)

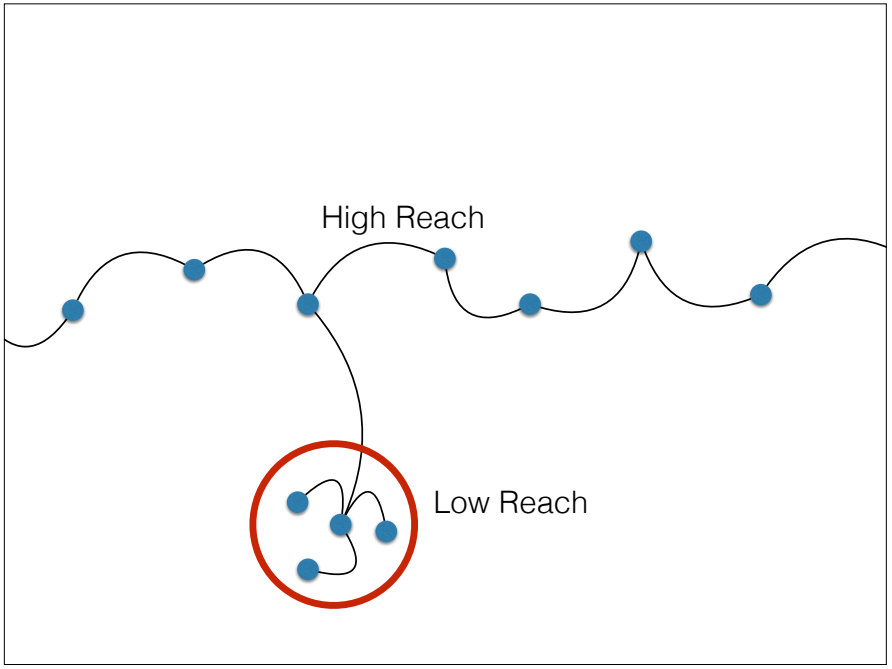


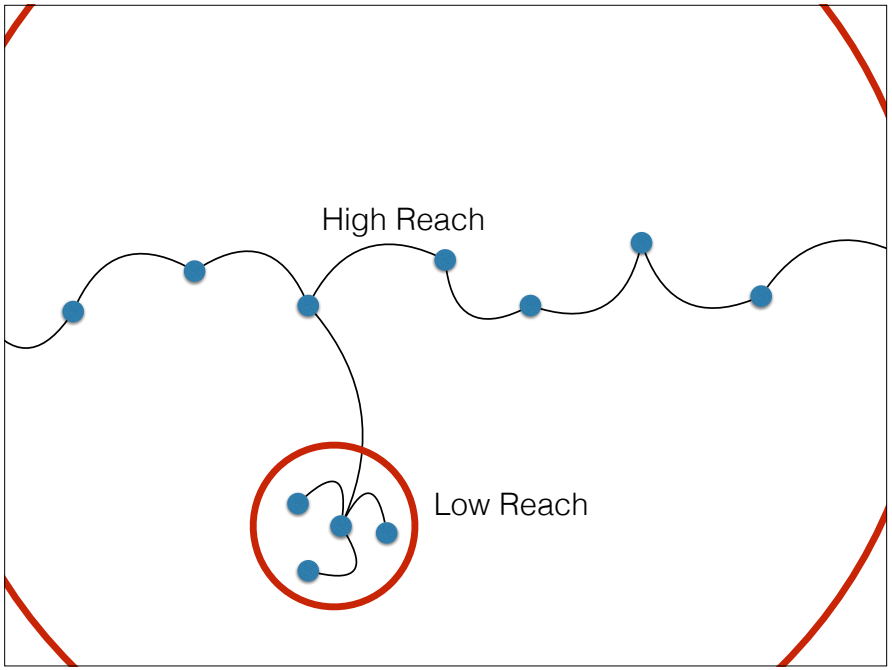



Low Reach


Low Reach







Demo

Demo Tasks

- What states have high reach?
- What states have low reach?

Bounding Boxes

- Assume you are at state *n* and and take action *a*
- What states can we reach on an optimal path?
  - Instead of storing states, represent with bounding box
- Geometric containers - Wagner et al (2005)
  - Only tested with Dijkstra's algorithm
- Bounding Boxes - Rabin and Sturtevant (2016)

Demo

Demo Tasks

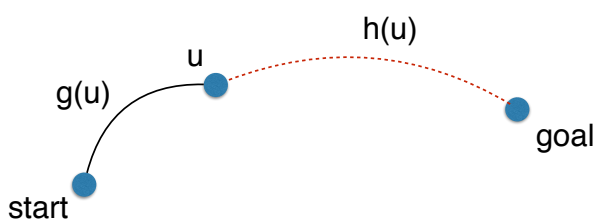
- What is the bounding box for a particular state/ action?
- Why does the canonical ordering not improve result for diagonal actions?
- In what sort of maps will bounding boxes be ineffective?

Bidirectional Search

A* Theory

- All nodes with f(n) < C* must be expanded
  - Dechter & Pearl, 1985
- Termination condition:
  - Once you have a solution and all states with f(n) < f(solution) have been expanded, you can terminate

A* Theory



$g(u) + h(u) < C^*$

Bidirectional Search Types

- Front-to-end
  - Heuristics are from frontier to goal/start
- Front-to-front
  - Heuristics are between two frontiers
    - SFBDS (Felner et al, 2010)

Theory

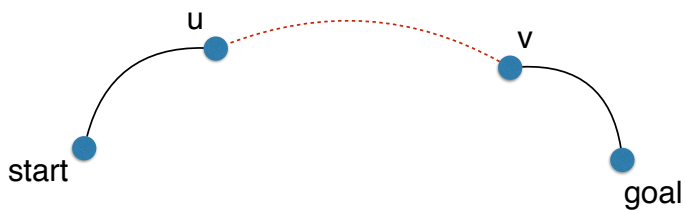
- What states must be expanded by a bidirectional search algorithm?
- Important theory question recently answered
  - (Eckerle et al, 2017)
  - Need the theory to design good algorithms

Demo (Game)

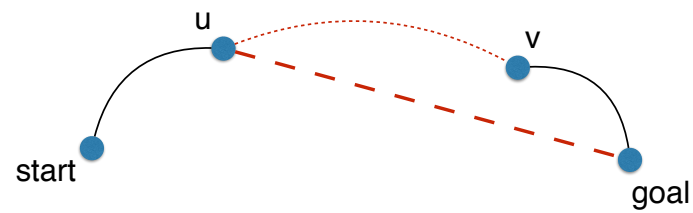
High-Level Picture

High-Level Picture

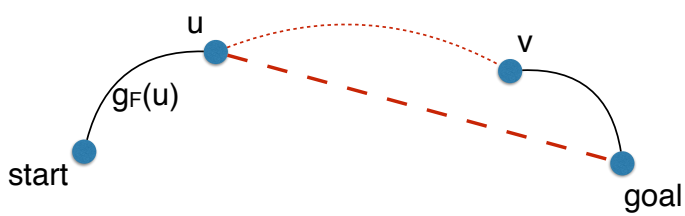
High-Level Picture



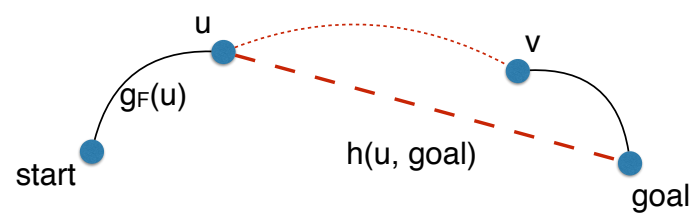
High-Level Picture



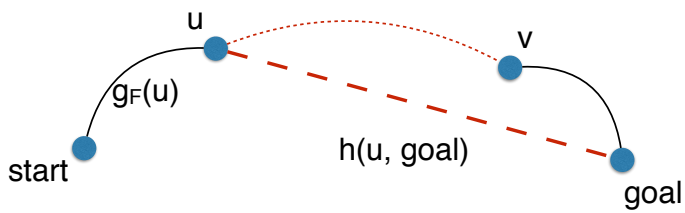
High-Level Picture



High-Level Picture

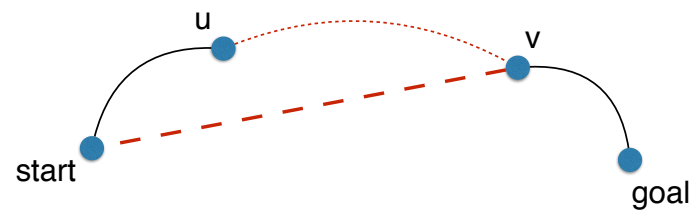


High-Level Picture

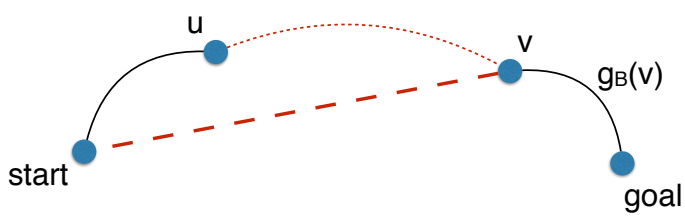


$f_F(u) = g_F(u) + h(u, goal)$

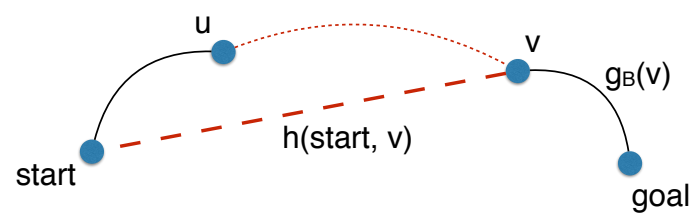
High-Level Picture



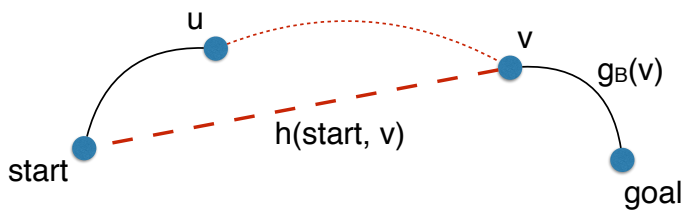
High-Level Picture



High-Level Picture

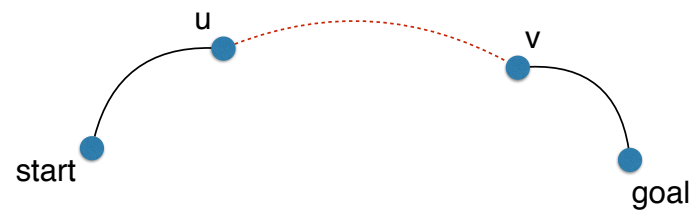


High-Level Picture

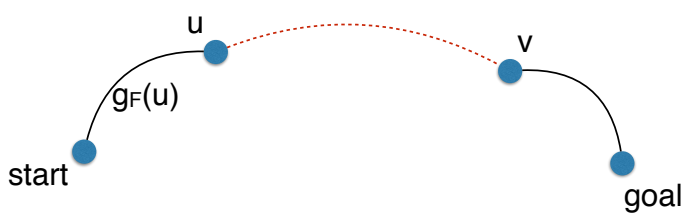


$f_B(v) = g_B(v) + h(\text{start}, v)$

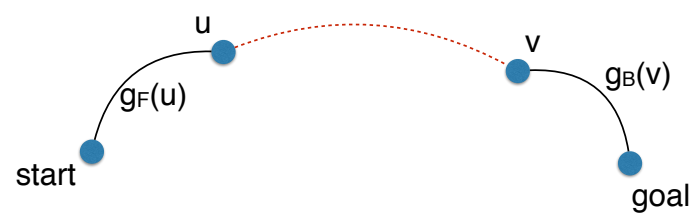
High-Level Picture



High-Level Picture



High-Level Picture



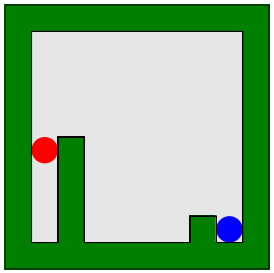
Theorem

f-cost:
estimate of total path length

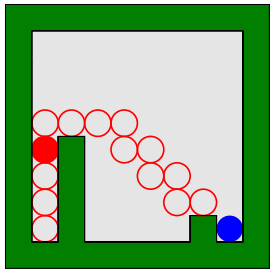
$$\mathrm{lb}(u,v) = max(f_F(u), f_B(v), g_F(u) + g_B(v))$$

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

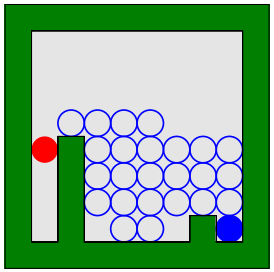
Sufficient Conditions for Node Expansion in Bidirectional Heuristic Search, Jurgen Eckerle, Jingwei Chen, Nathan Sturtevant, Sandra Zilles and Robert Holte, International Conference on Automated Planning and Scheduling (ICAPS), **2017**

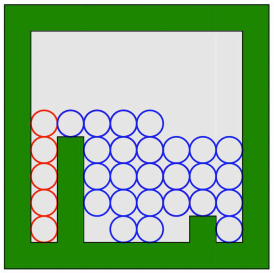


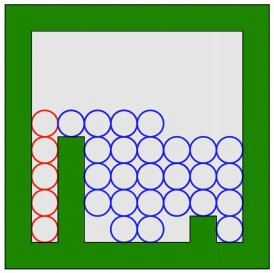
$$f_F(u) < C^*$$



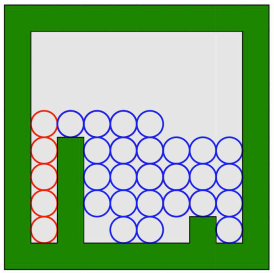
$$f_B(v) < C^*$$



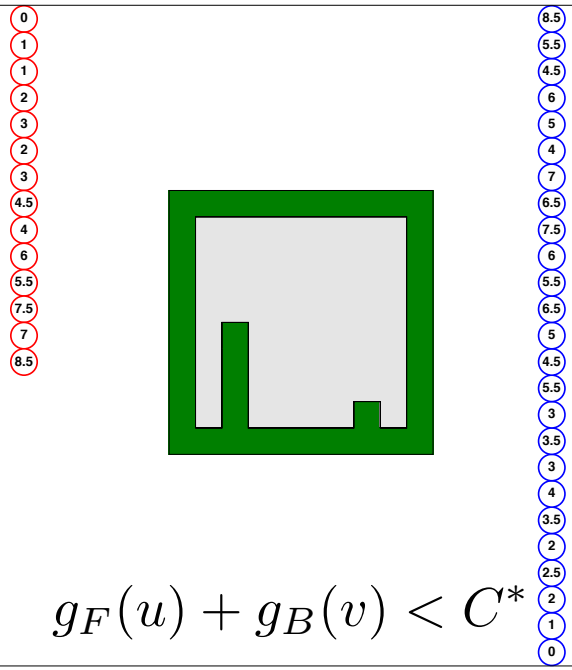




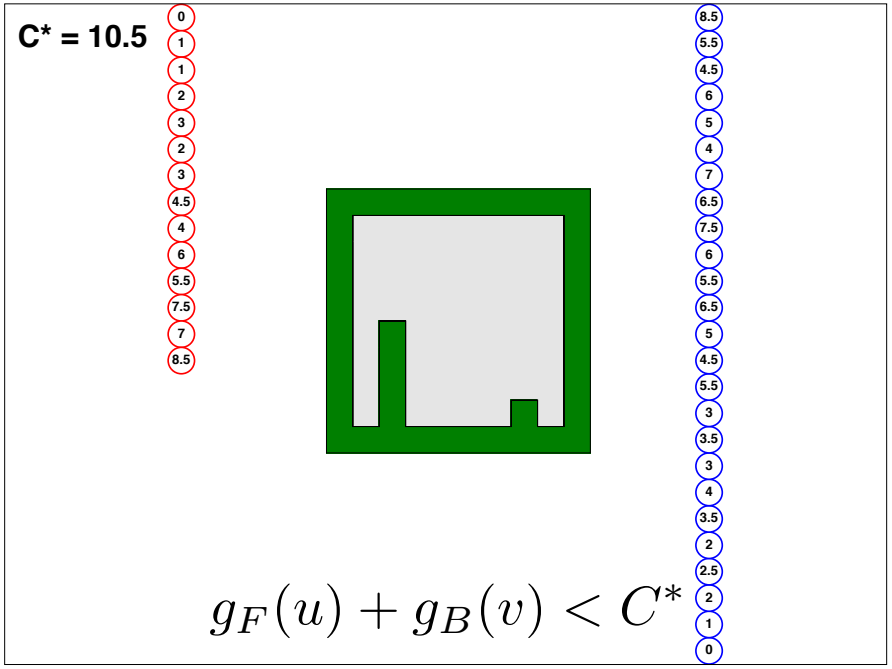
$$g_F(u) + g_B(v) < C^*$$

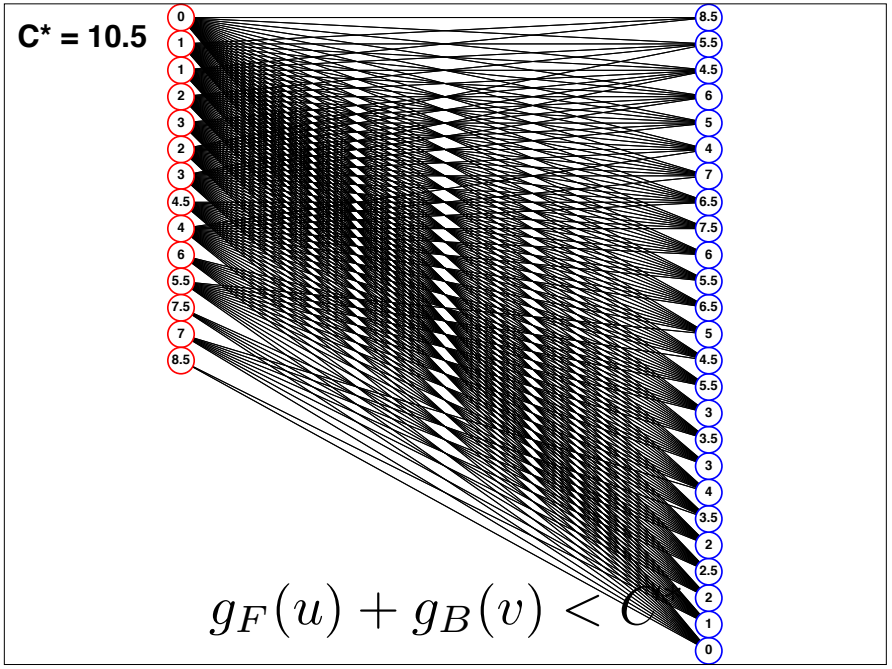


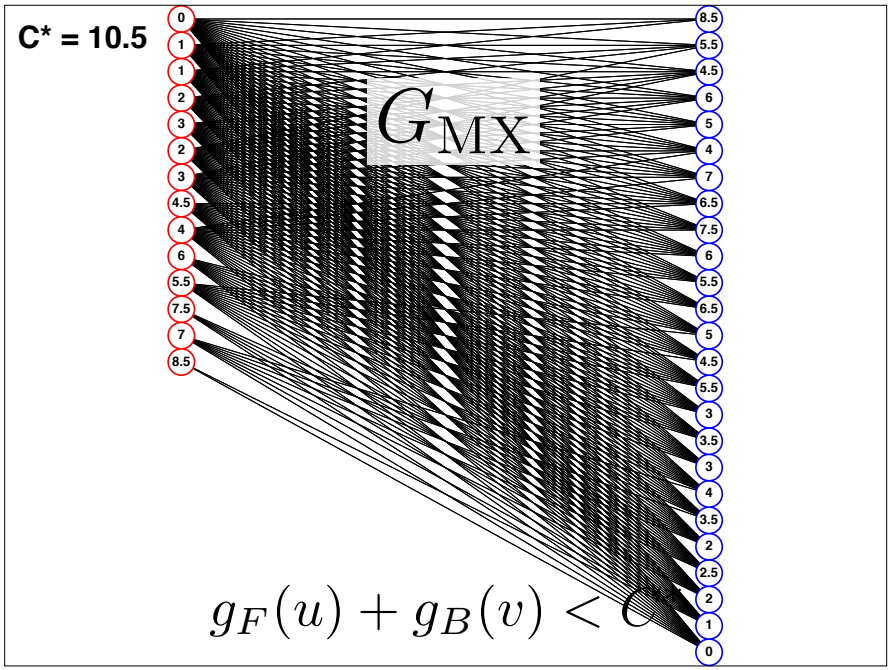
$$g_F(u) + g_B(v) < C^*$$

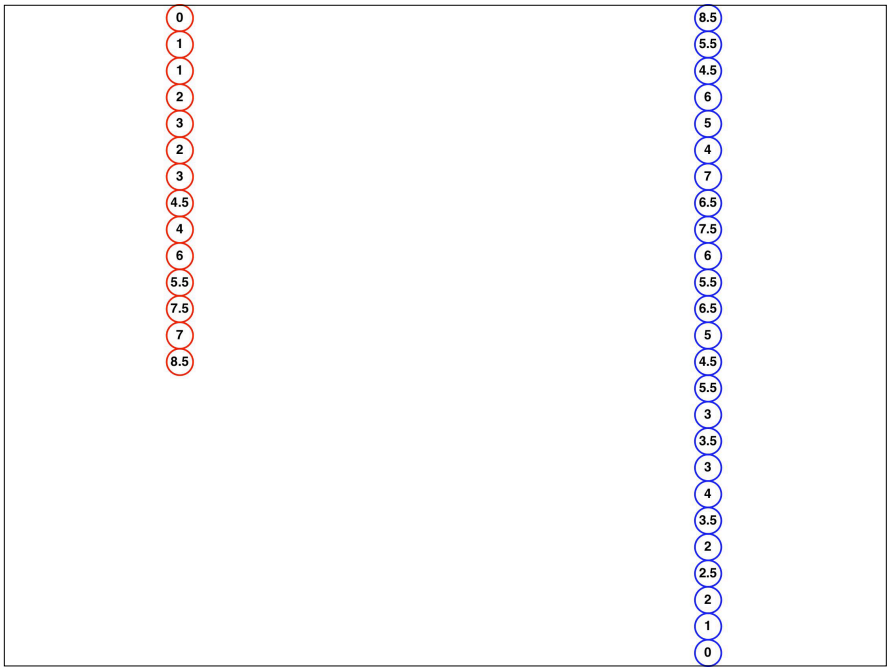


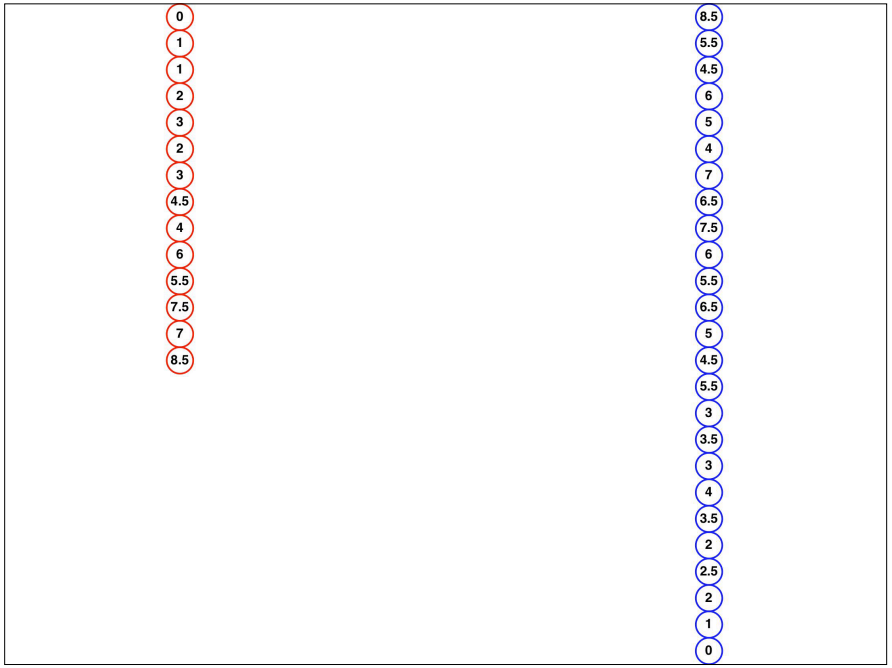
$$g_F(u) + g_B(v) < C^*$$

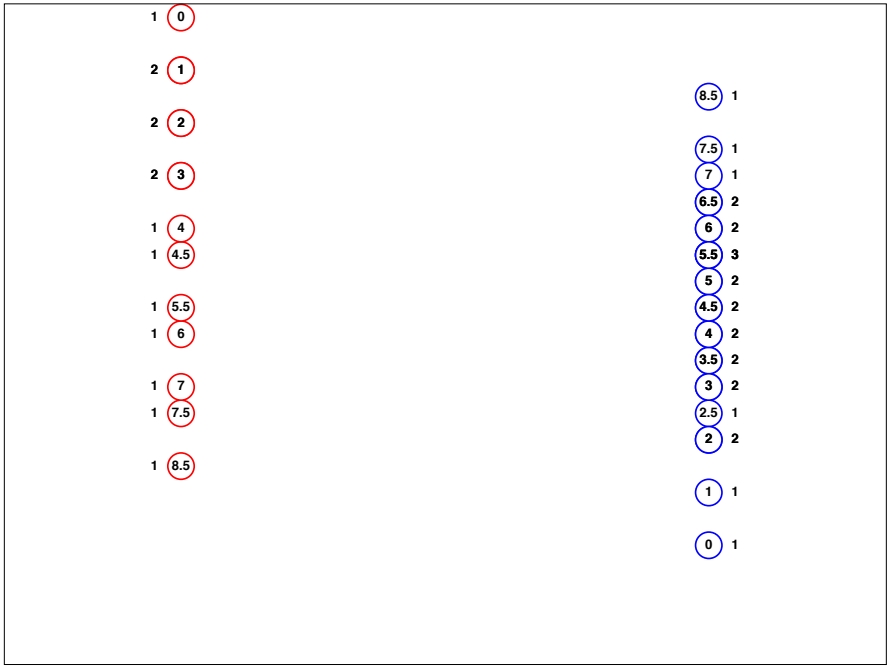

C* = 10.5
$g_F(u) + g_B(v) < C^*$

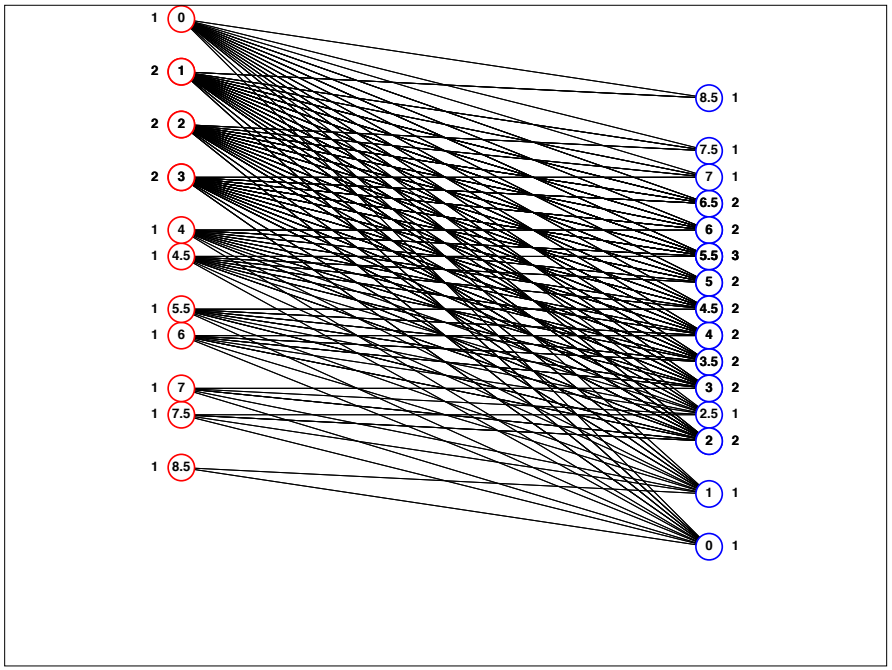

C* = 10.5
$g_F(u) + g_B(v) < C^*$

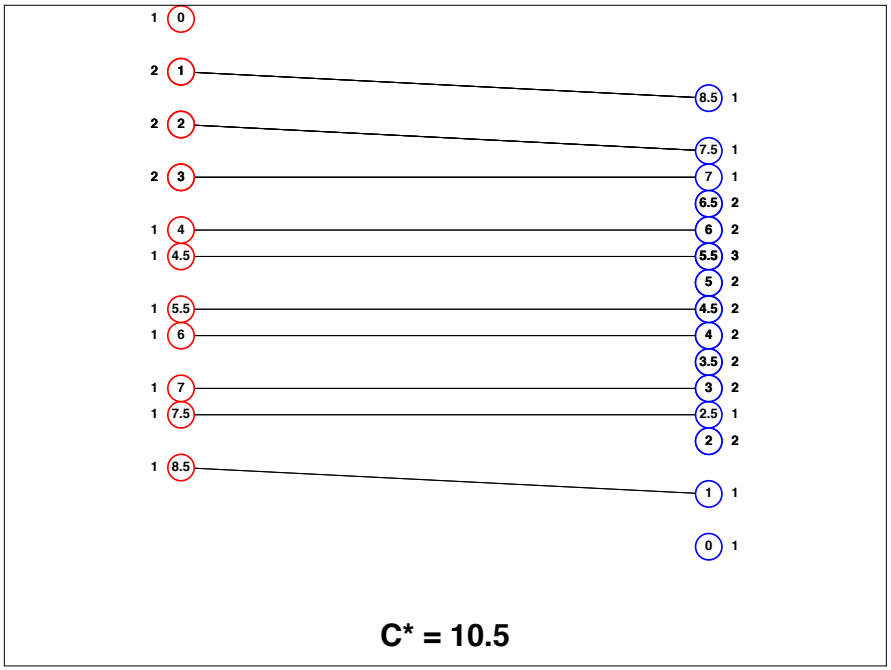

C* = 10.5
G_{MX}
$g_F(u) + g_B(v) < C^*$

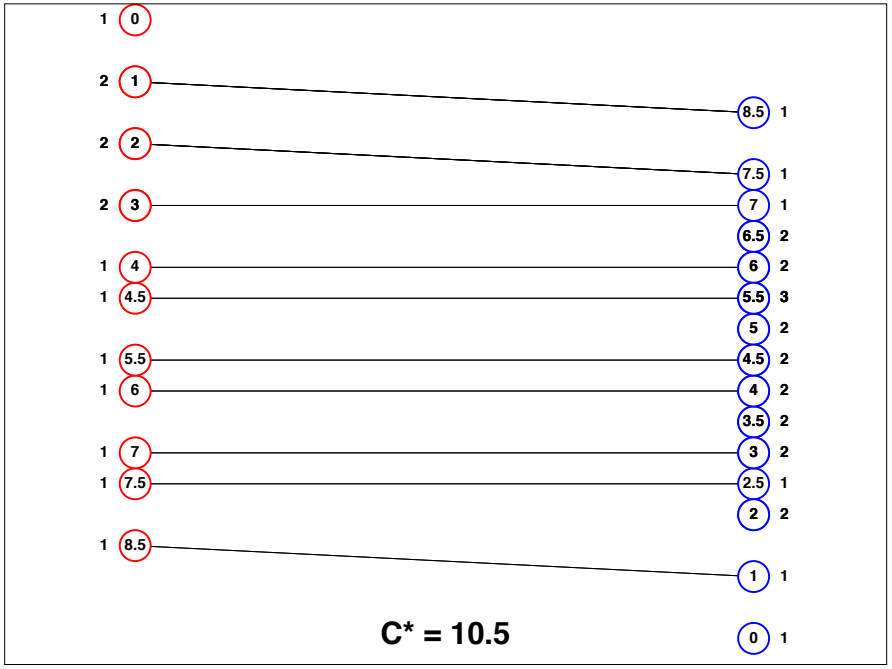


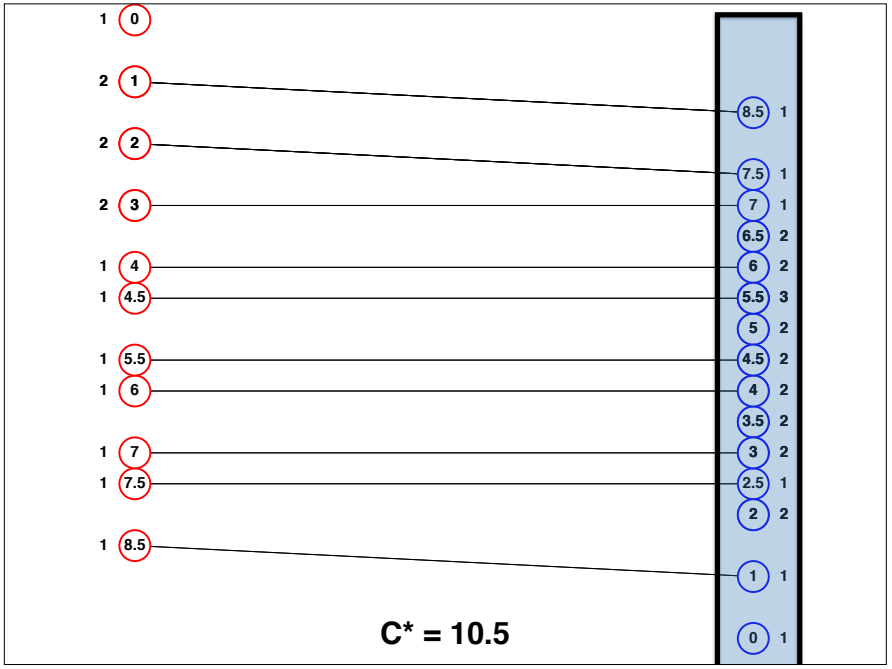

0 1 1 2 3 2 3 4.5 4 6 5.5 7.5 7 8.5
8.5 5.5 4.5 6 5 4 7 6.5 7.5 6 5.5 6.5 5 4.5 5.5 3 3.5 3 4 3.5 2 2.5 2 1 0

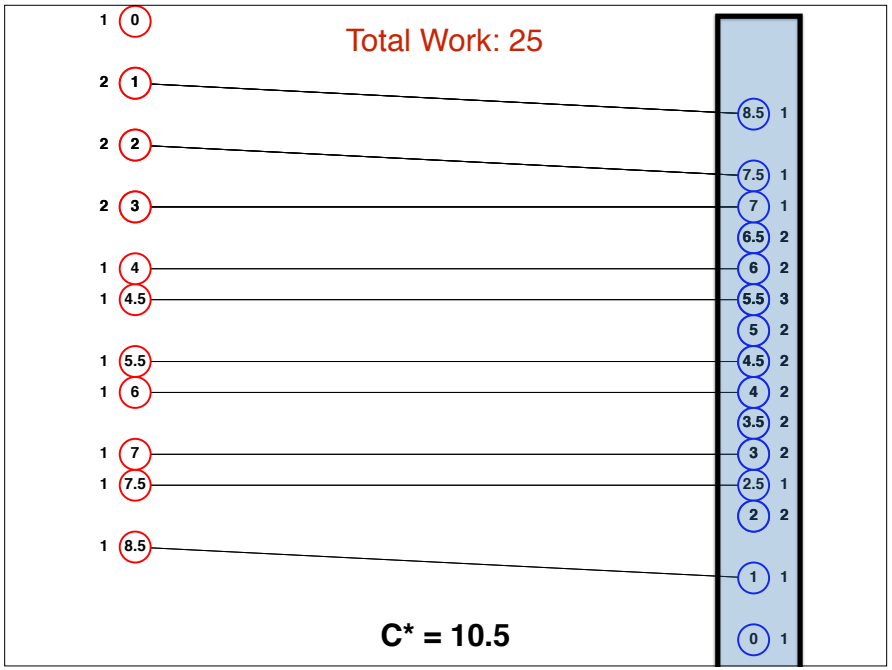

1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1

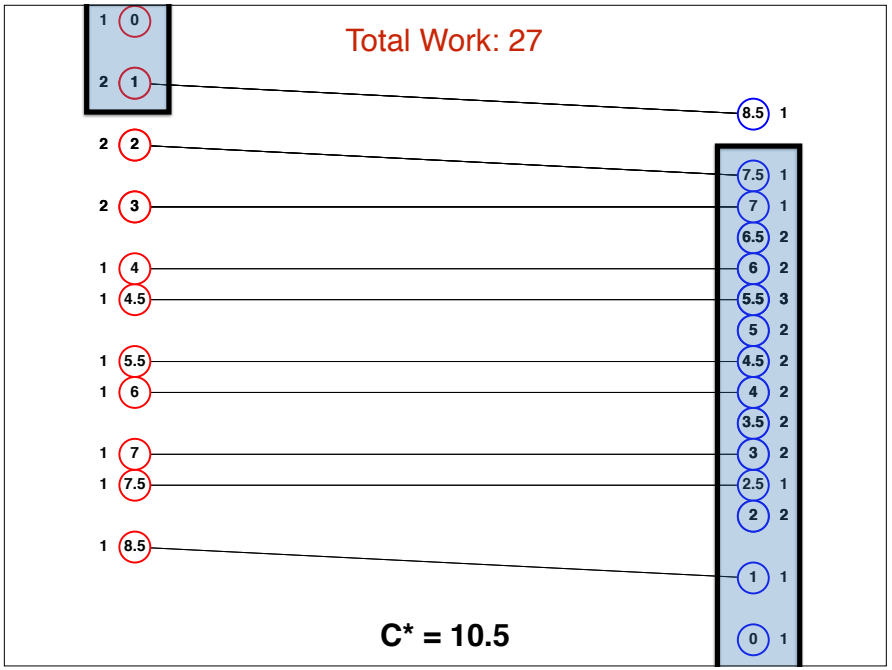

1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1

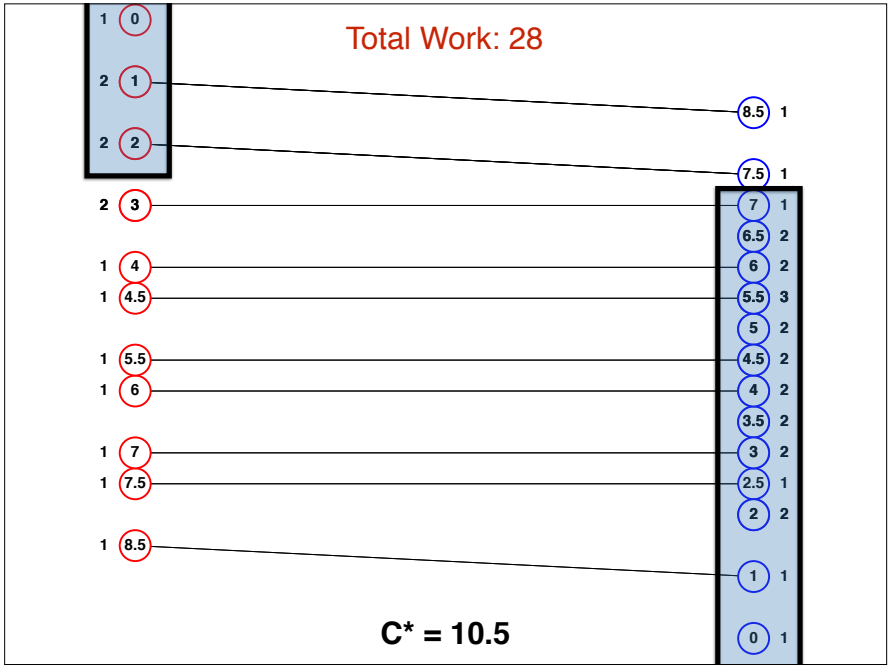

1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1
C* = 10.5

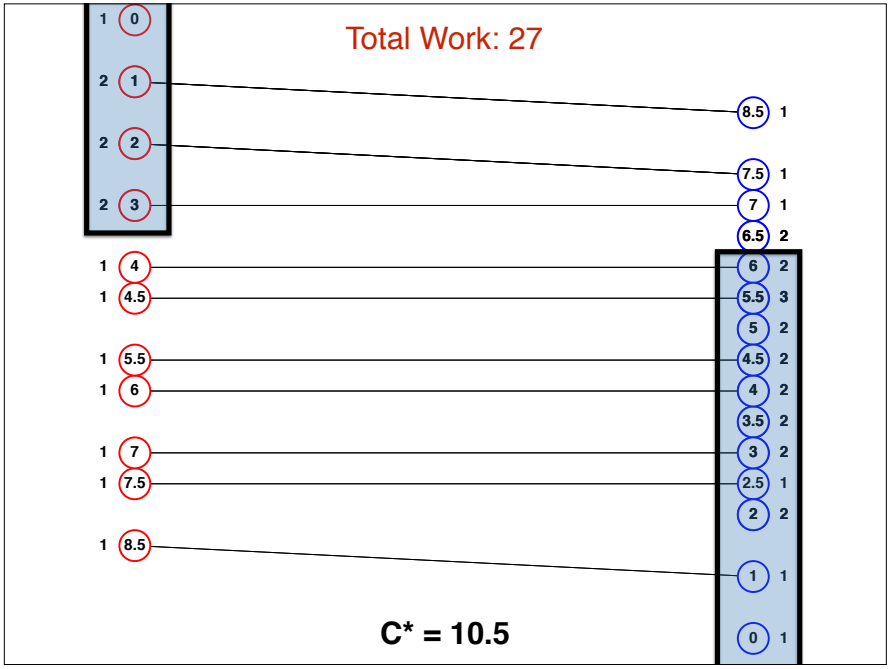

1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1
C* = 10.5

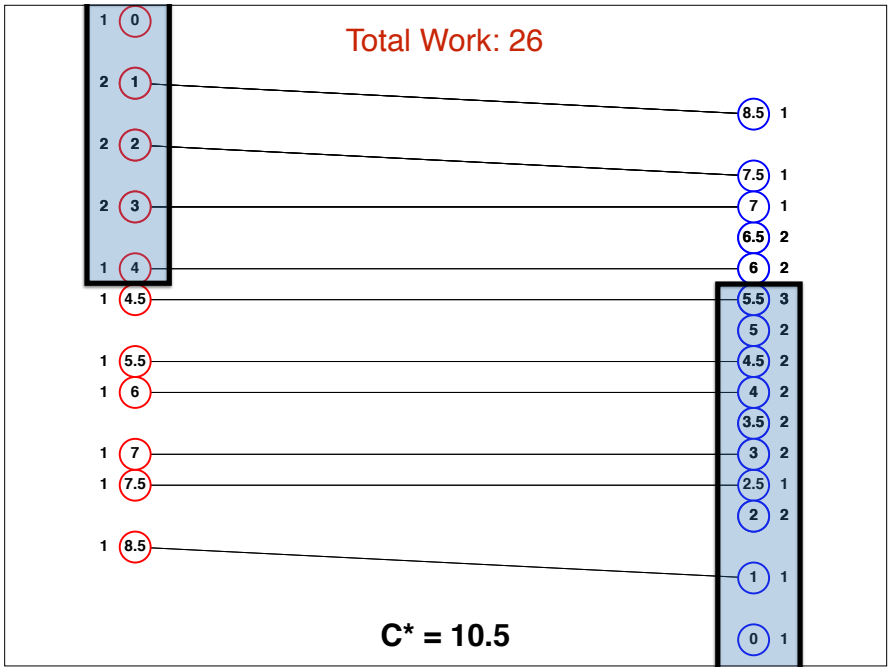

1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1
C* = 10.5

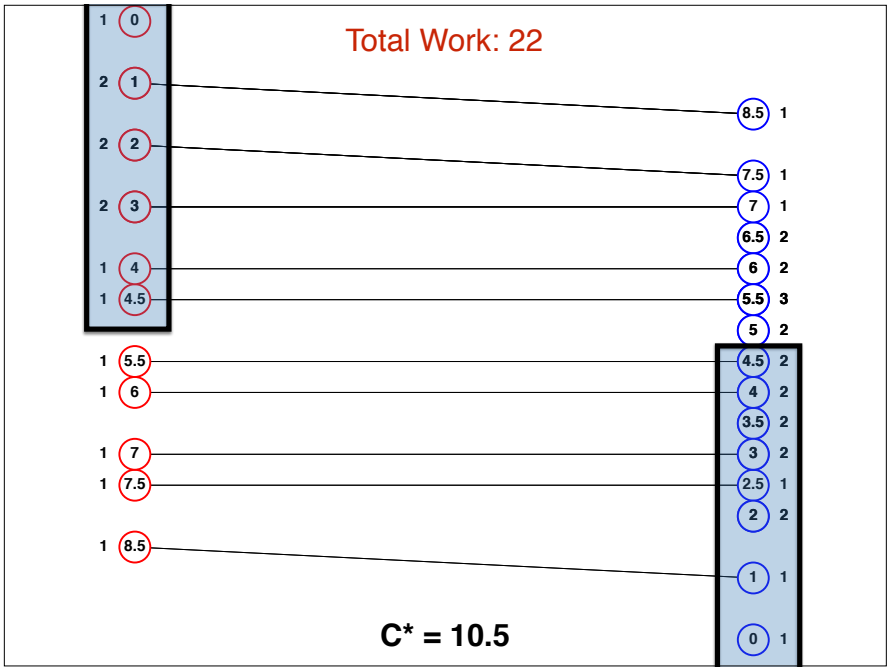

Total Work: 25
1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1
C* = 10.5

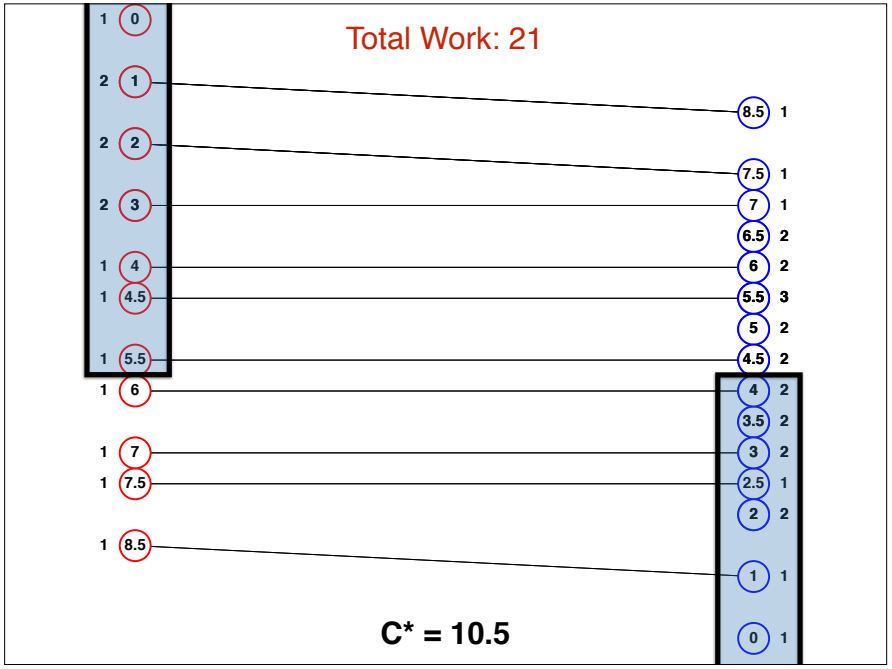

Total Work: 27
1 0
2 1
2 2
2 3
1 4
1 4.5
1 5.5
1 6
1 7
1 7.5
1 8.5
8.5 1
7.5 1
7 1
6.5 2
6 2
5.5 3
5 2
4.5 2
4 2
3.5 2
3 2
2.5 1
2 2
1 1
0 1
C* = 10.5

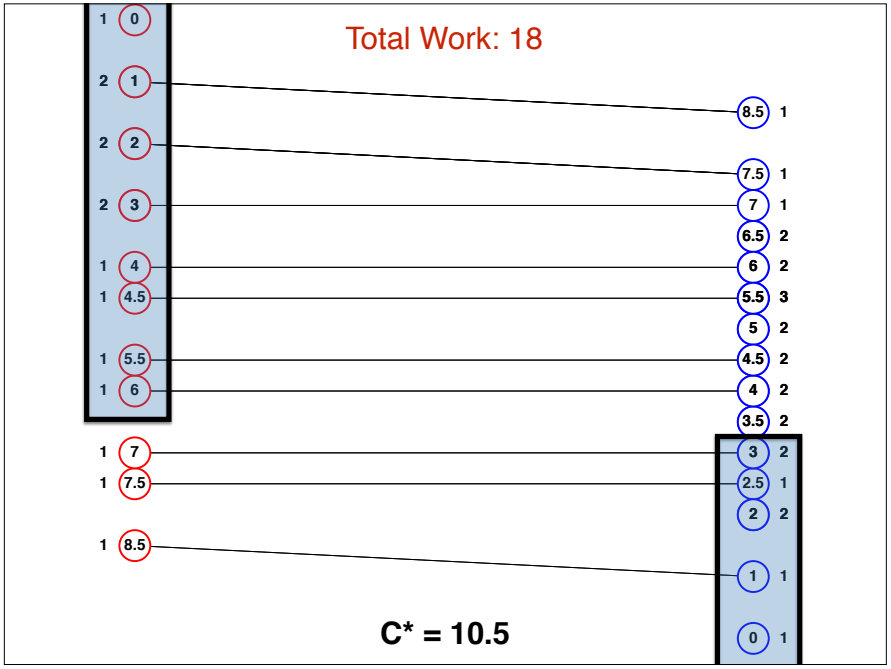

Total Work: 28
C* = 10.5

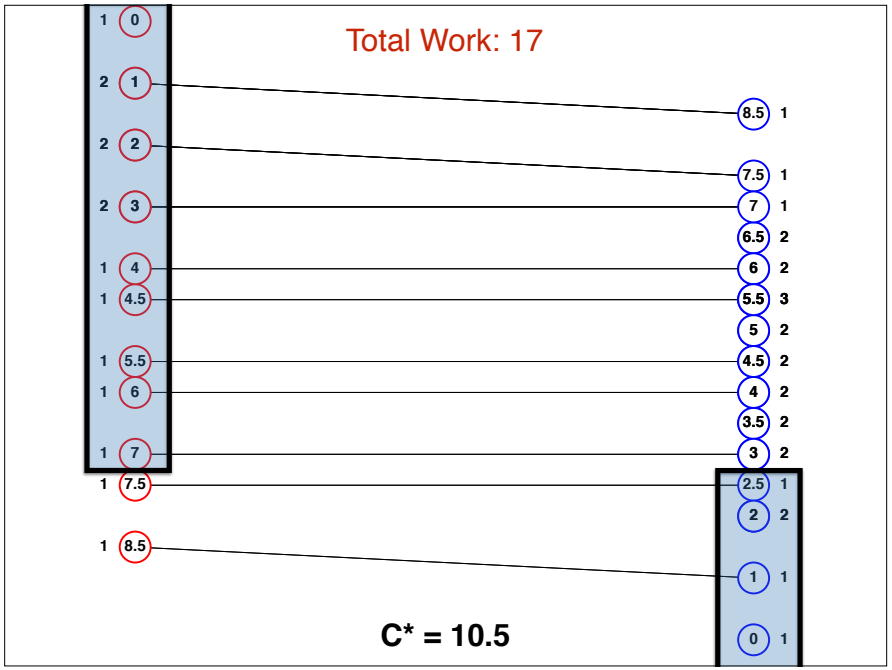

Total Work: 27
C* = 10.5

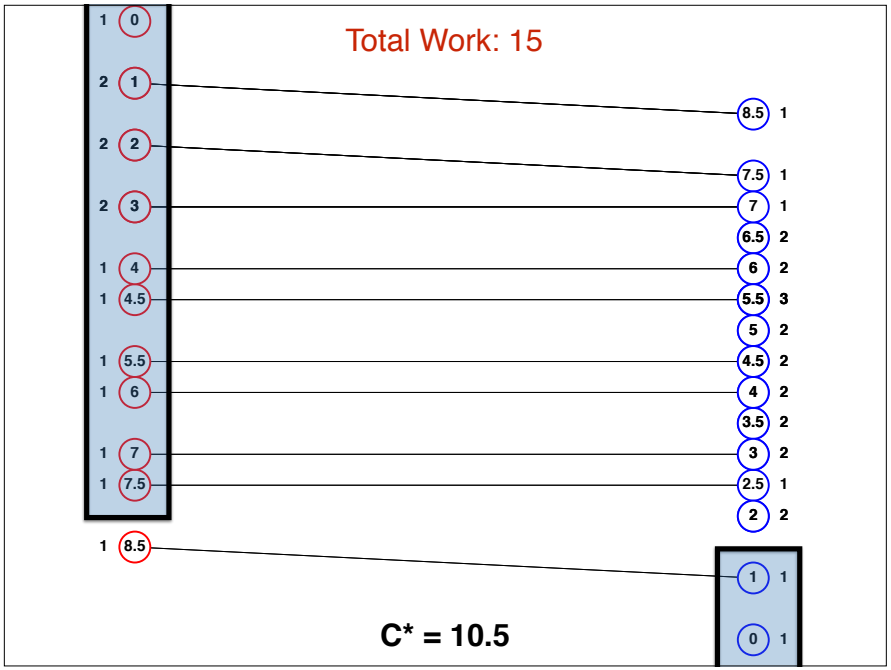

Total Work: 26
C* = 10.5

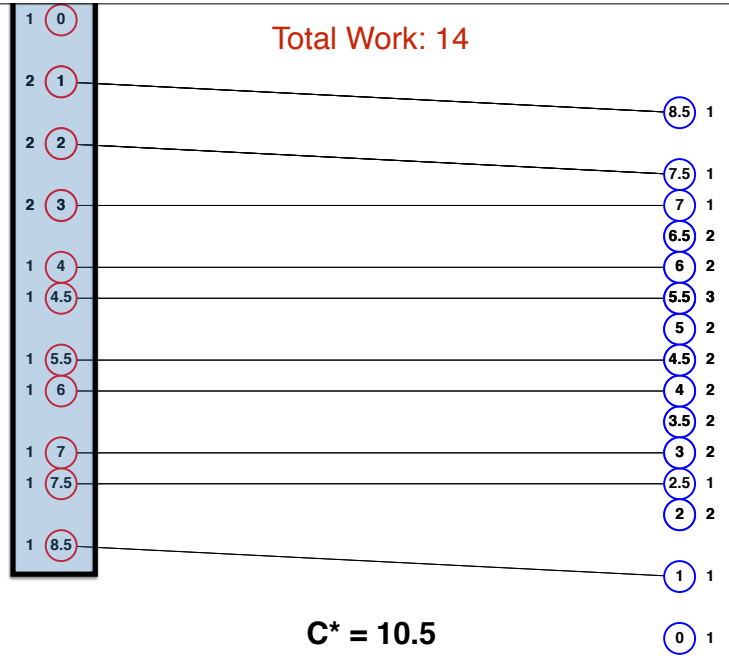

Total Work: 22
C* = 10.5


Total Work: 21
C* = 10.5


Total Work: 18
C* = 10.5


Total Work: 17
C* = 10.5


Total Work: 15
C* = 10.5

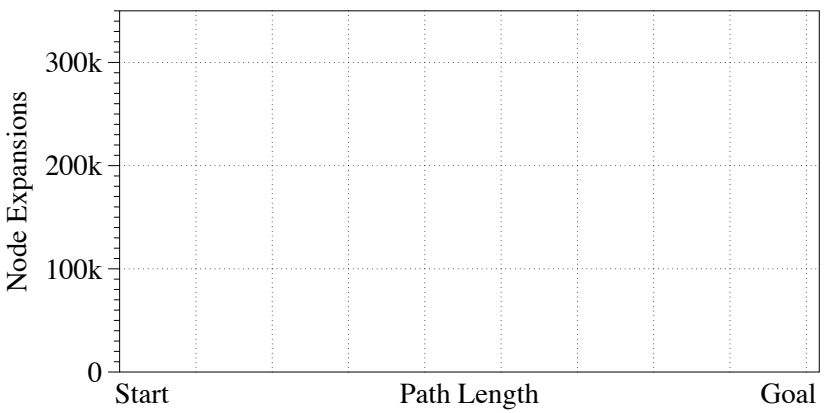


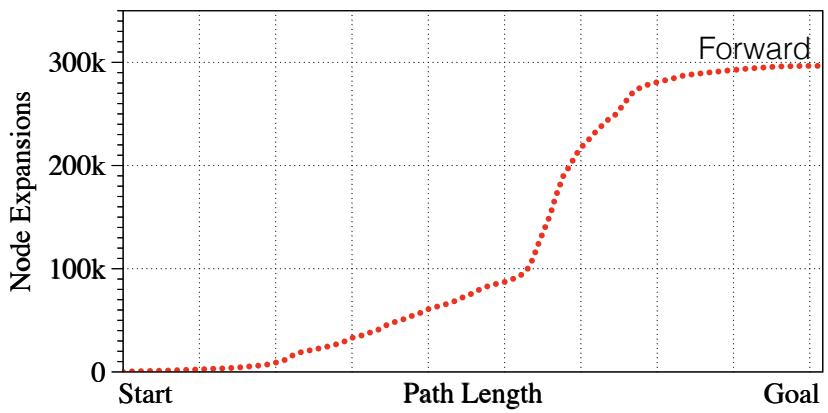
UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

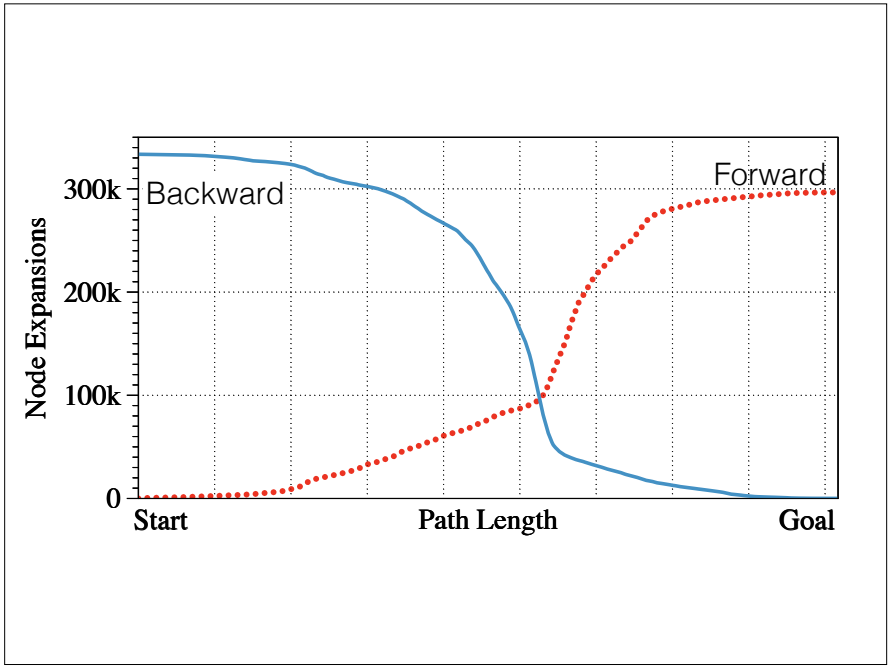
Fractional MM

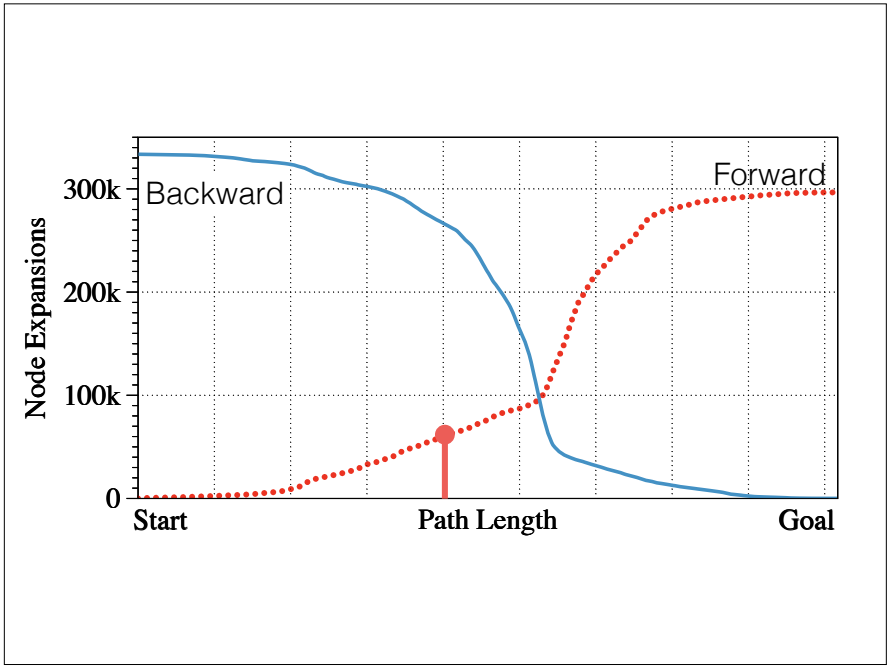
- Takes a parameter *f*
  - Cost of the state space to explore in each direction
  - Costs correspond to different vertex covers
- *We can (offline) compute the best algorithm for a given search problem*

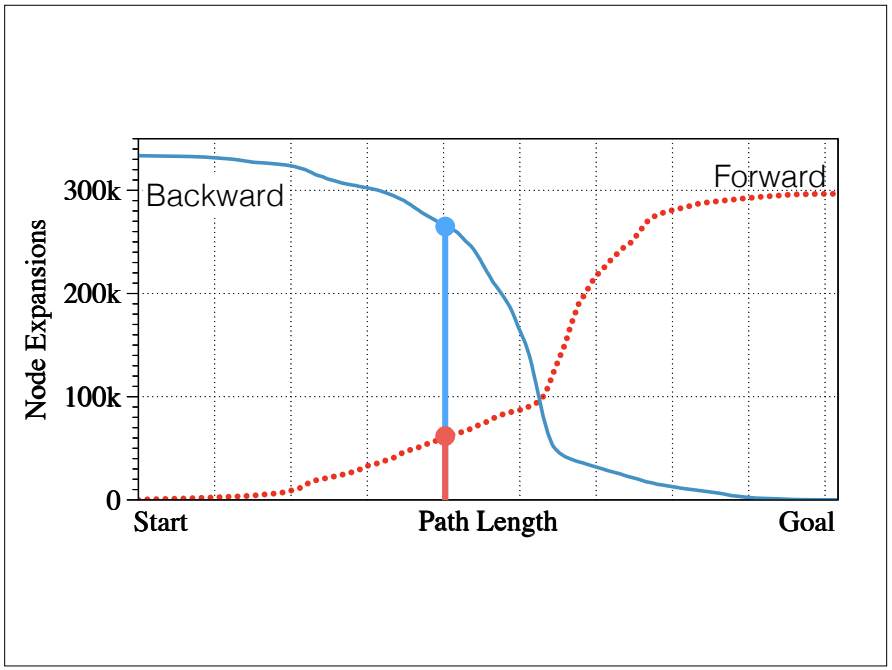
The Minimal Set of States that Must be Expanded in a Front-to-end Bidirectional Search, Eshed Shaham, Ariel Felner, Jingwei Chen and Nathan R. Sturtevant, Symposium on Combinatorial Search (SoCS), **2017**

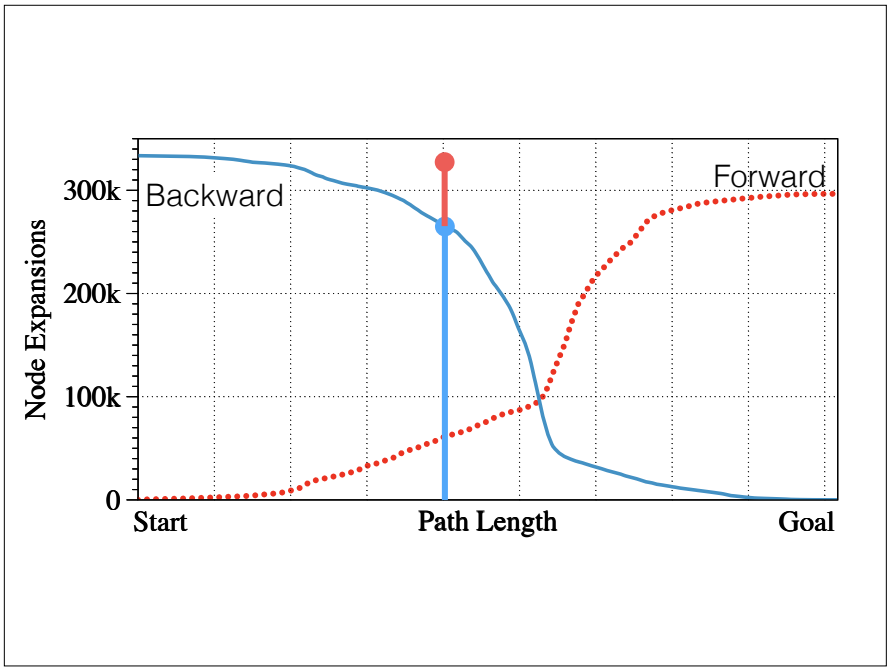


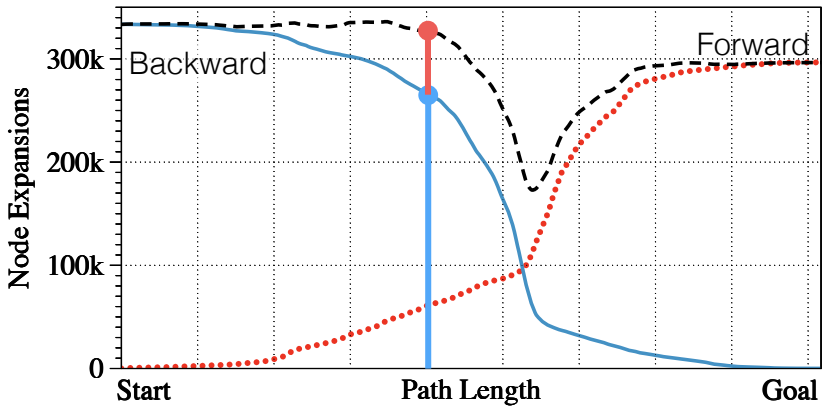


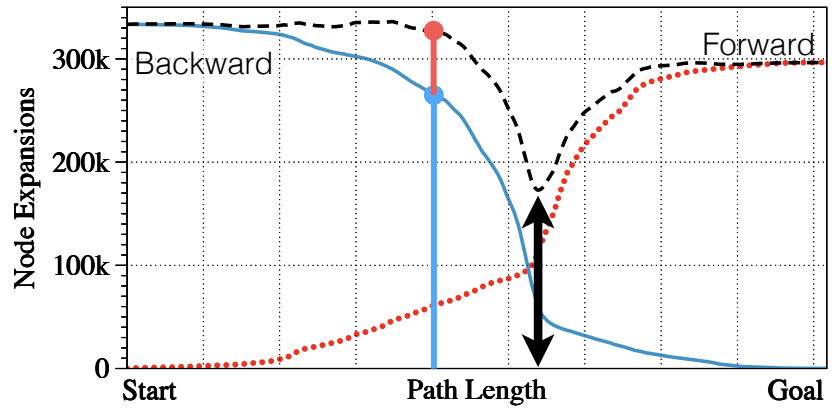

Backward
Forward
Node Expansions
300k
200k
100k
0
Start
Path Length
Goal


Backward
Forward
Node Expansions
300k
200k
100k
0
Start
Path Length
Goal


Backward
Forward
Node Expansions
300k
200k
100k
0
Start
Path Length
Goal


Backward
Forward
Node Expansions
300k
200k
100k
0
Start
Path Length
Goal





Demo

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Full Theory

- Must expand* any pairs with lb(u, v) < C*

Full Theory

- Must expand* any pairs with lb(u, v) < C*
- There is no online optimal algorithm

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Full Theory

- Must expand* any pairs with lb(u, v) < C*
- There is no online optimal algorithm
- Any **DXBB** algorithm will have instances where it is 2x worse than the optimal solution

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Vertex Cover on a Bipartite Graph

- Approximation algorithm:
  - Repeat until all vertices covered
    - Choose any edge/line with uncovered vertices
    - Place both states into vertex cover
- Gives 2x approximation to optimal vertex cover
  - (Papadimitriou & Steiglitz, 1982)

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

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

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

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

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

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*

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

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

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

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

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Using this algorithm

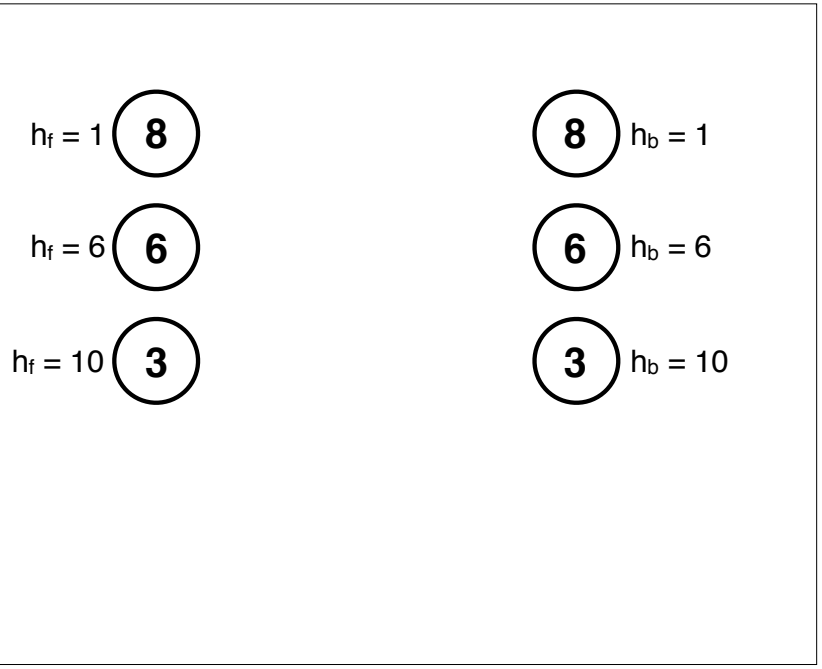
- We don't know the full graph ahead of time
  - Build the graph as we go
- We don't know the optimal solution cost
  - Must estimate C*
- We must avoid re-expanding states
  - Carefully order state expansions
- Computing lb(u, v) could be expensive
  - Efficient data structures

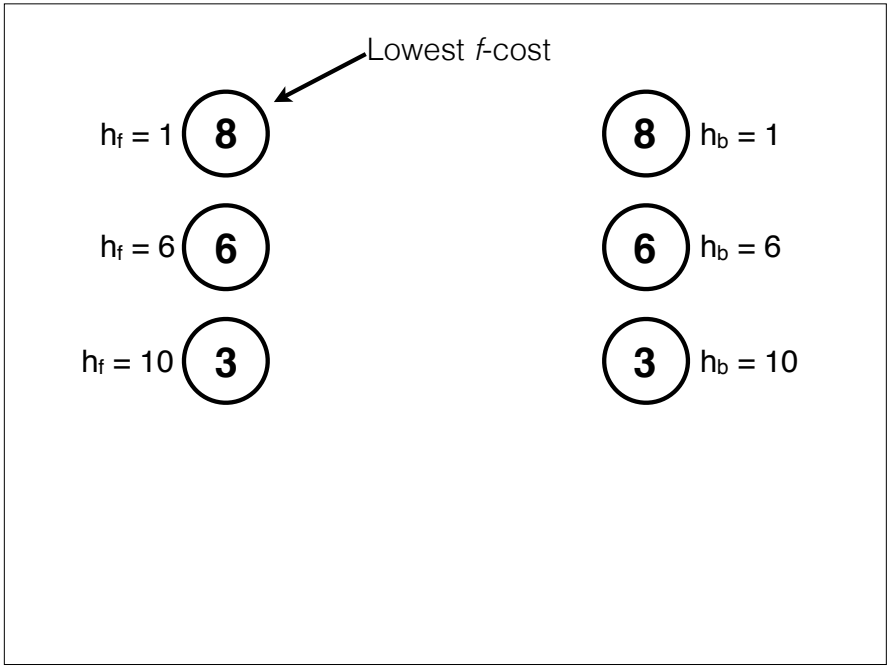
UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

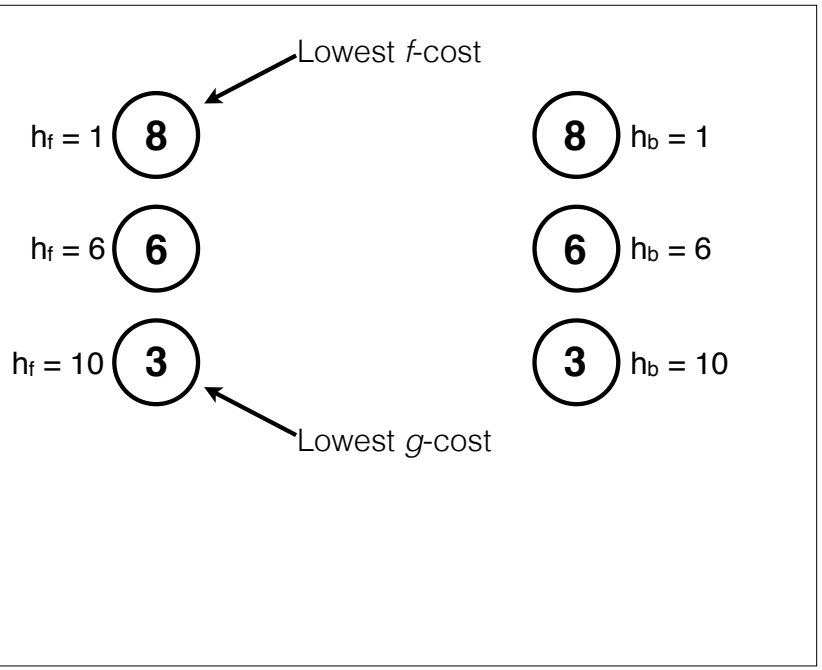
NBS

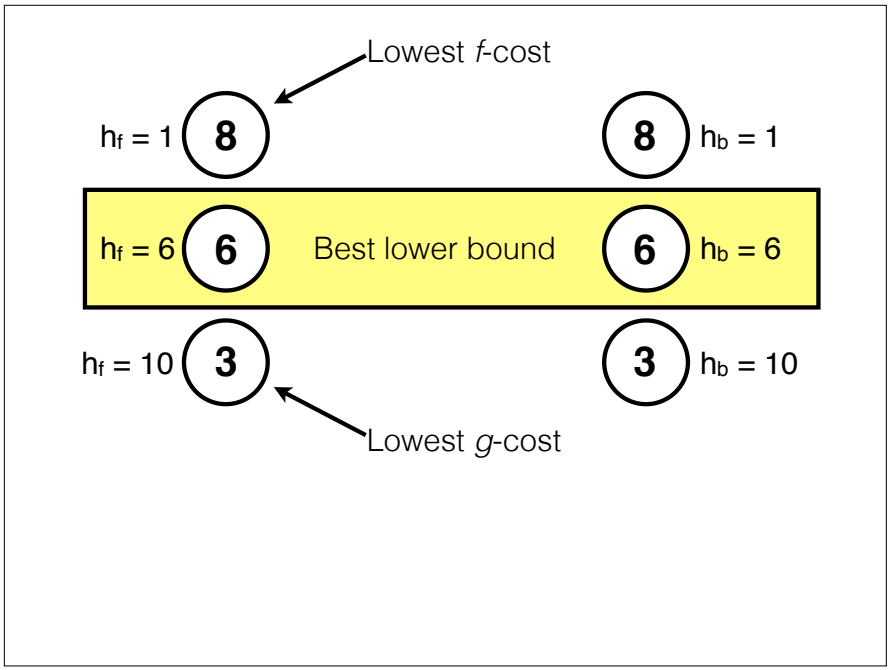
- Put start/goal onto forward/backward priority queues
- While forward/backward not empty
  - Among all state on queues:
    - Select the pair with lowest *lb*
    - Expand **both** of them
- Gives 2x bound on optimal number of expansions

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**


$h_f = 1$ 8
$h_f = 6$ 6
$h_f = 10$ 3
8 $h_b = 1$
6 $h_b = 6$
3 $h_b = 10$


Lowest *f*-cost
$h_f = 1$ 8
$h_f = 6$ 6
$h_f = 10$ 3
8 $h_b = 1$
6 $h_b = 6$
3 $h_b = 10$


Lowest *f*-cost
$h_f = 1$ 8
$h_f = 6$ 6
$h_f = 10$ 3
Lowest *g*-cost
8 $h_b = 1$
6 $h_b = 6$
3 $h_b = 10$


Lowest *f*-cost
$h_f = 1$ 8
$h_f = 6$ 6
Best lower bound
$h_f = 10$ 3
Lowest *g*-cost
8 $h_b = 1$
6 $h_b = 6$
3 $h_b = 10$

Demo

Demo tasks

- When is NBS 2x suboptimal?
- When should you use NBS even if it is 2x suboptimal?

Bidirectional Summary

- Everything has changed!
- Completely new theory
  - lb equivalent of f-cost in bidirectional search
- New algorithm: NBS
  - All* algorithms 2x worse than optimal on some instances
  - NBS no more than 2x worse* on any instance

*Theoretical restrictions apply

Summary

- Exponential Domains & approaches
- Polynomial Domains
  - Canonical Orderings
  - Heuristics
  - Constraints
- Bidirectional Search
  - New theory and lb function
  - NBS