AAAI 2020 Tutorial on Heuristic Search

Ariel Felner, Sven Koenig, Nathan Sturtevant

Part I
Introduction and Overview

Motivation

- Suppose you need a path…
- …for multiple agents…
- …as quickly as possible…
- …but with quality guarantees

- What are the basic approaches that you might consider when building an approach to solve the problem?

Introduction

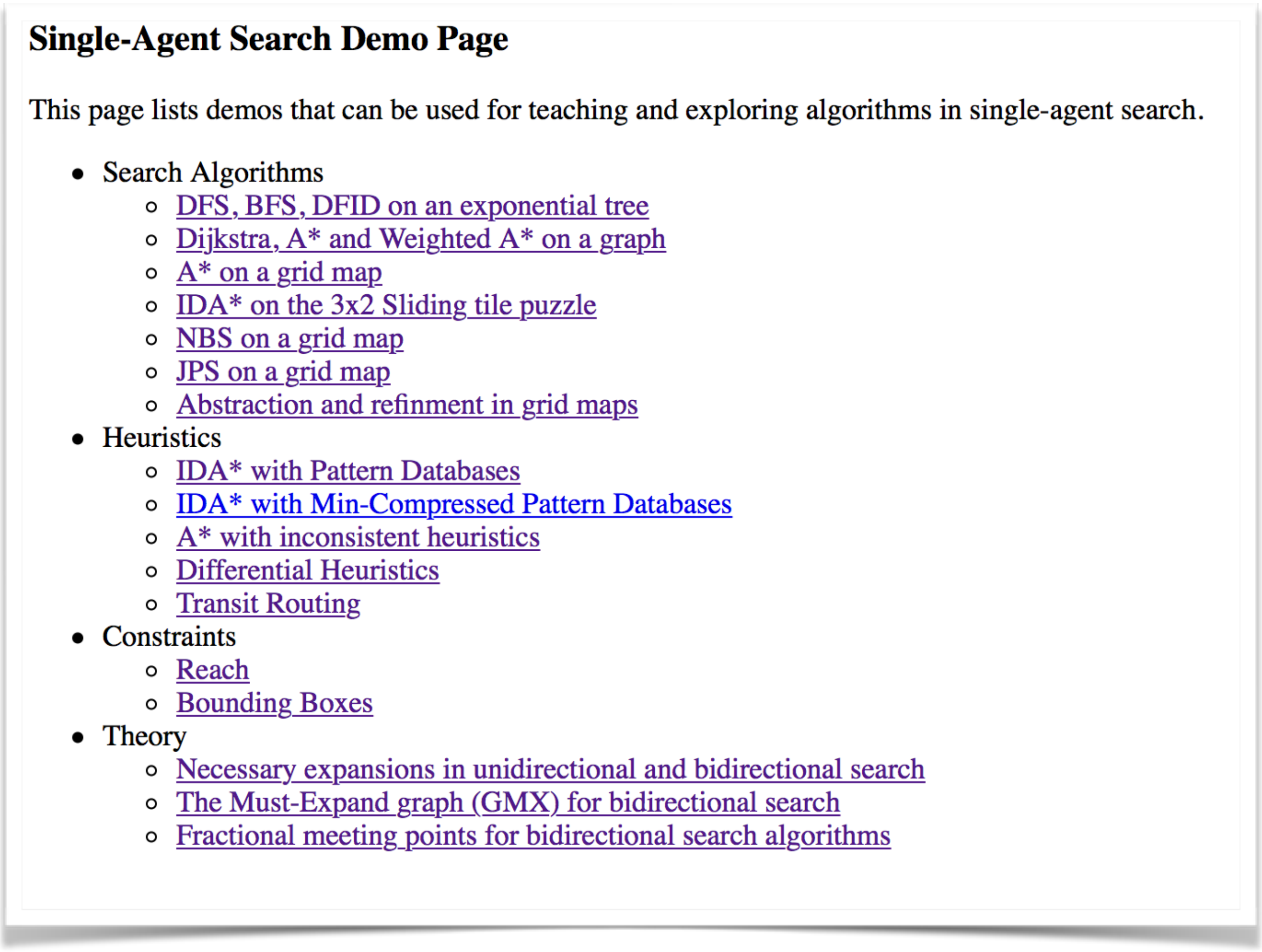
- **Baseline Assumptions of Heuristic Search**
  - Exponential & Polynomial Algorithms
  - Bidirectional Search
  - Exponential & Polynomial Heuristics
- **Examples & case studies**
  - Constraints
  - Any-Angle Search
  - Multi-Agent Path Planning
- **Tools for teaching/learning search concepts**

Materials

- Workshop materials:
  - http://movingai.com/AAAI-HS20/
- Outline, Slides, Demos

Demos

https://movingai.com/SAS/



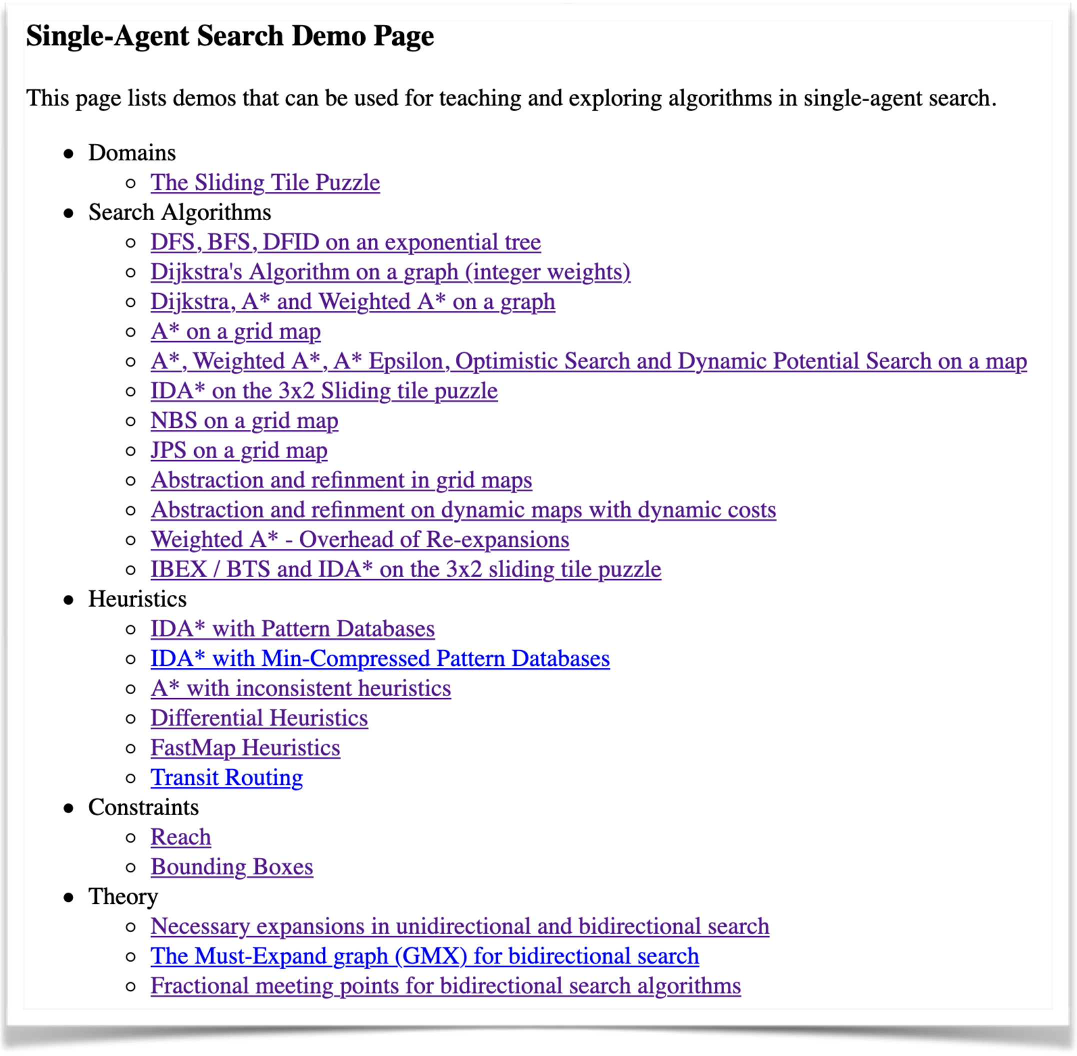
Single-Agent Search Demo Page

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

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.

- Domains
  - The Sliding Tile Puzzle
- Search Algorithms
  - DFS, BFS, DFID on an exponential tree
  - Dijkstra's Algorithm on a graph (integer weights)
  - Dijkstra, A* and Weighted A* on a graph
  - A* on a grid map
  - A*, Weighted A*, A* Epsilon, Optimistic Search and Dynamic Potential Search on a map
  - IDA* on the 3x2 Sliding tile puzzle
  - NBS on a grid map
  - JPS on a grid map
  - Abstraction and refinment in grid maps
  - Abstraction and refinment on dynamic maps with dynamic costs
  - Weighted A* - Overhead of Re-expansions
  - IBEX / BTS and IDA* on the 3x2 sliding tile puzzle
- Heuristics
  - IDA* with Pattern Databases
  - IDA* with Min-Compressed Pattern Databases
  - A* with inconsistent heuristics
  - Differential Heuristics
  - FastMap 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

Your Speakers

Ariel Fener - *Ben Gurion University*

Sven Koenig - *University of Southern California*

Nathan Sturtevant - *University of Alberta*
Alberta Machine Intelligence Institute (Amii)
Canada Institute for Advanced Research

Acknowledgments

- For Sven’s research, see idm-lab.org/projects or send an email to skoenig@usc.edu
- Thanks to the many students and colleagues who contributed to this research!
- Thanks to NSF and Amazon for funding!
  - The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government.

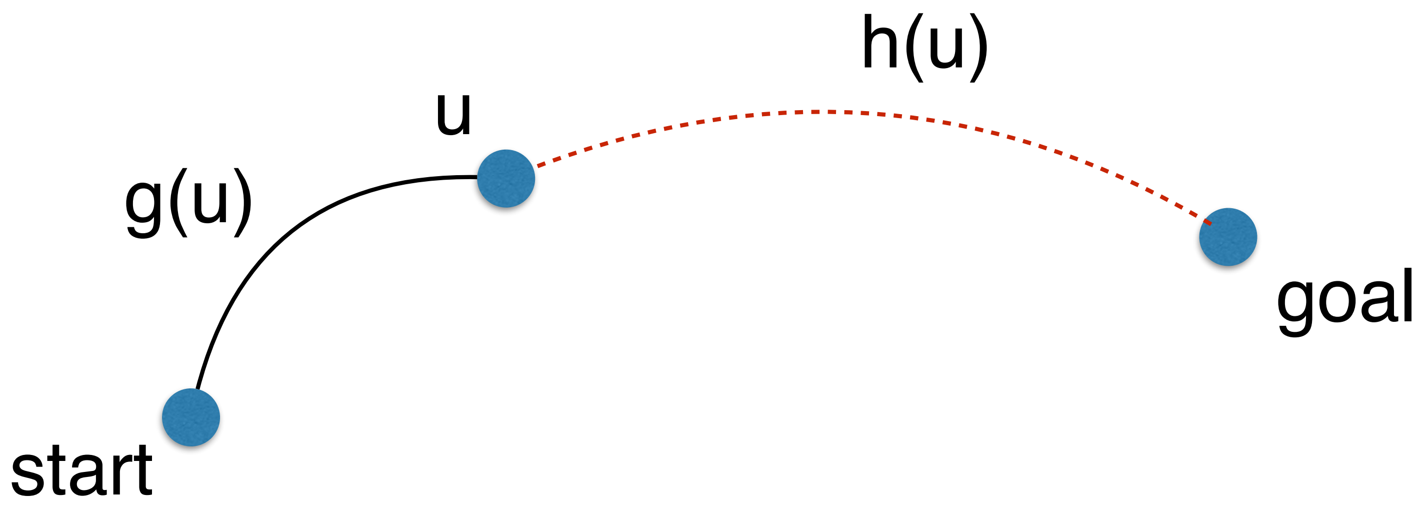
Problem Definition

- Given:
  - A start state & goal state (or goal test)
  - A successor & cost function
  - A heuristic function
- Return:
  - A path from the start to the goal
  - *With optimal cost*
- Black box assumption

Terminology

- g(*n*) - Cost from start to *n* in the search
- h(*n*) - Estimated cost from *n* to the goal
- c(*a*, *b*) - optimal cost from *a* to *b*

Heuristic Picture



Heuristics

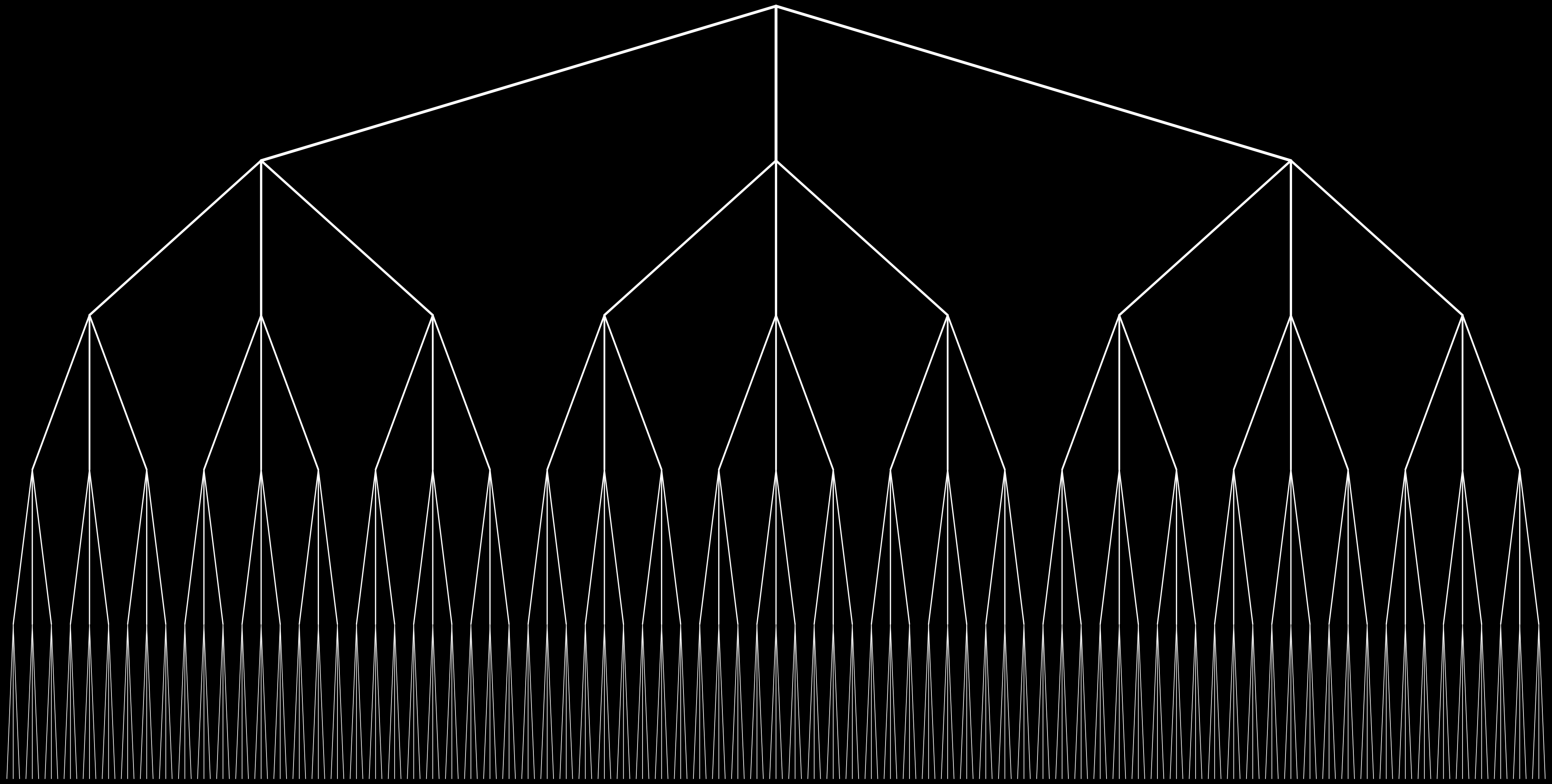
- Heuristic function maps a state to a *distance/cost*
  - Estimate of *distance/cost* to the goal
- Admissible heuristic doesn't overestimate true cost
  - $h(s, g) \leq c(s, g)$
- Consistent heuristic obeys triangle inequality (undirected graphs)
  - $|h(a,g) - h(b,g)| \leq c(a,b)$

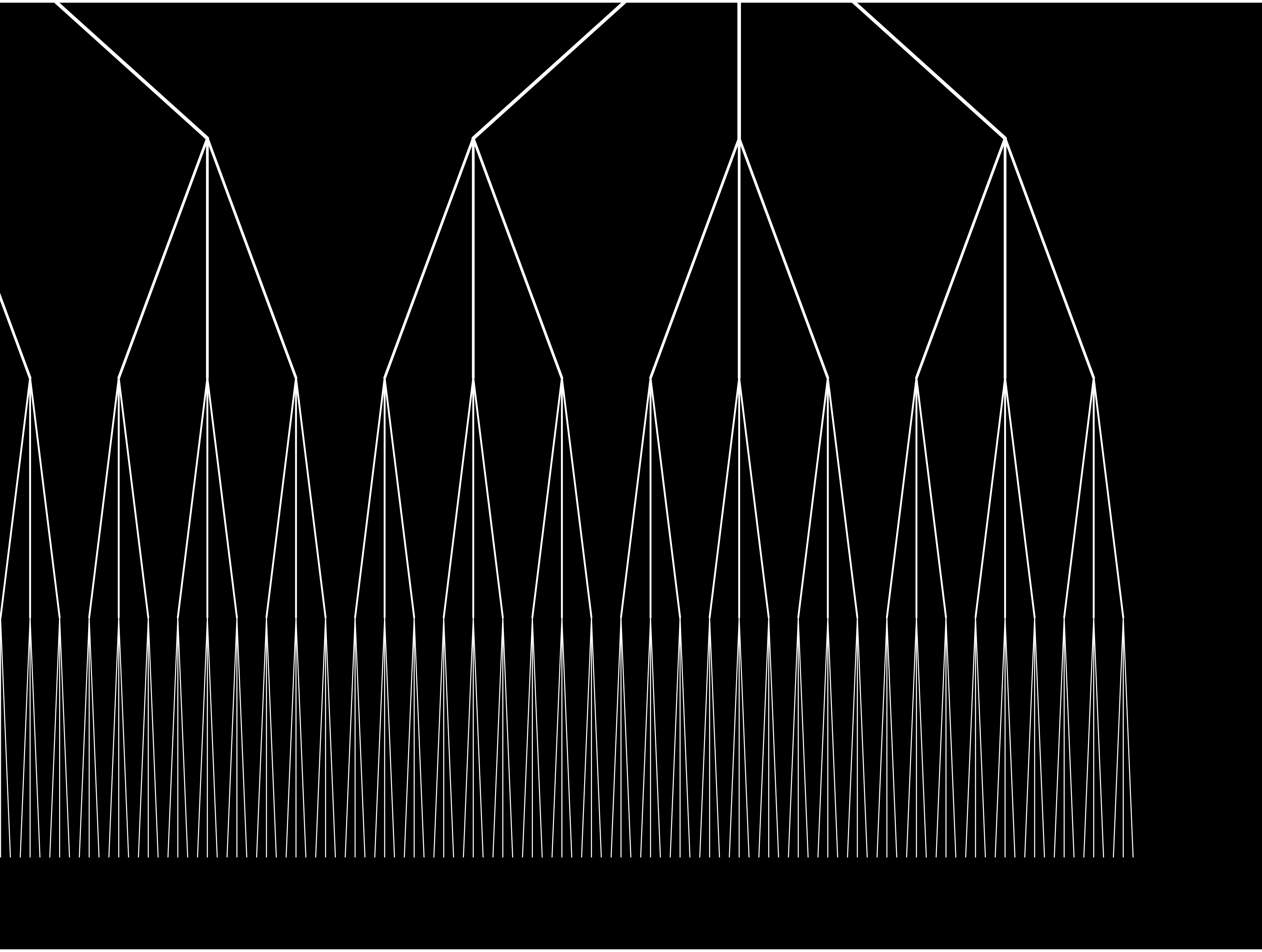
Part II
Baseline Algorithms

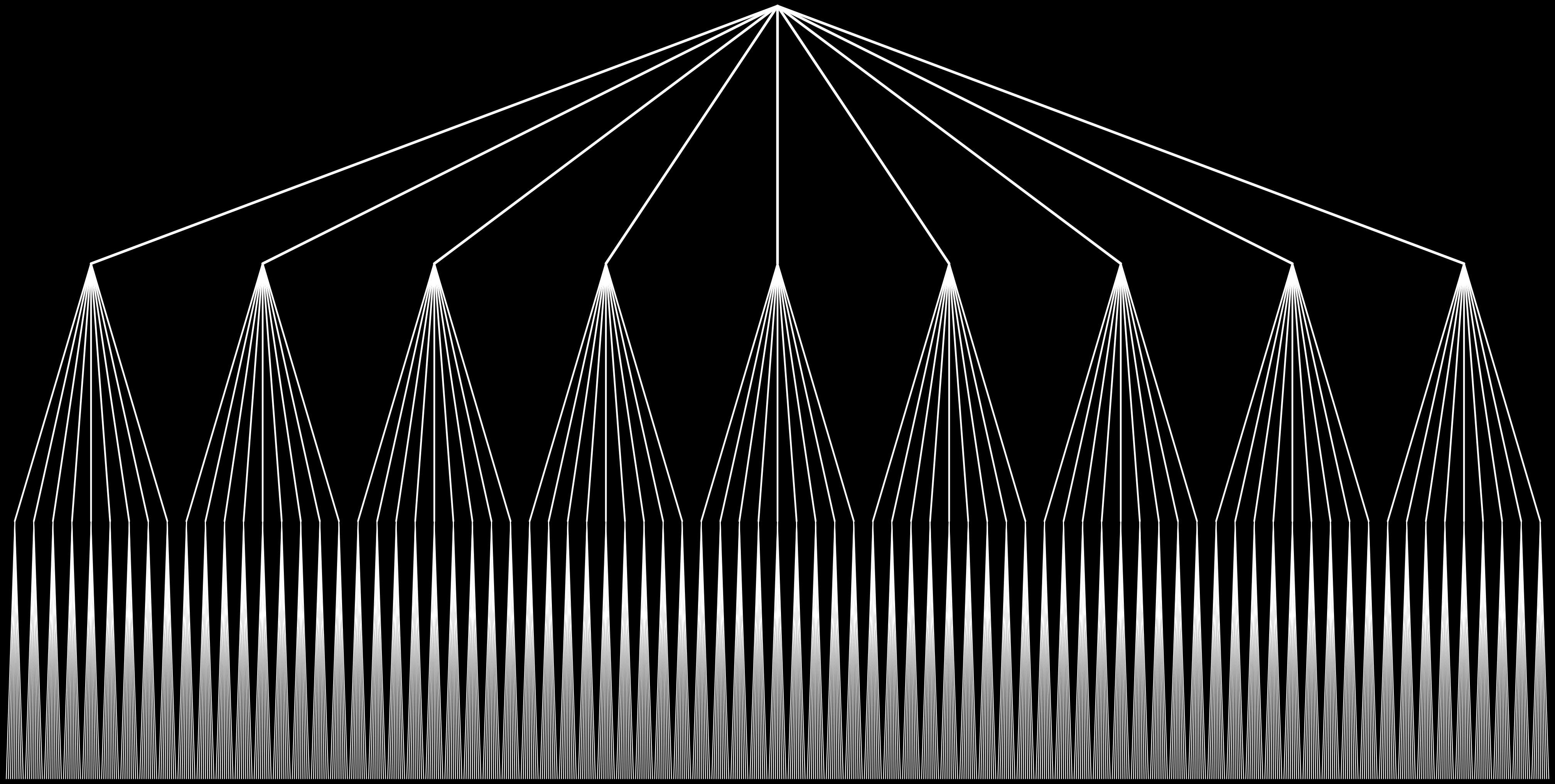
Exponential Algorithms

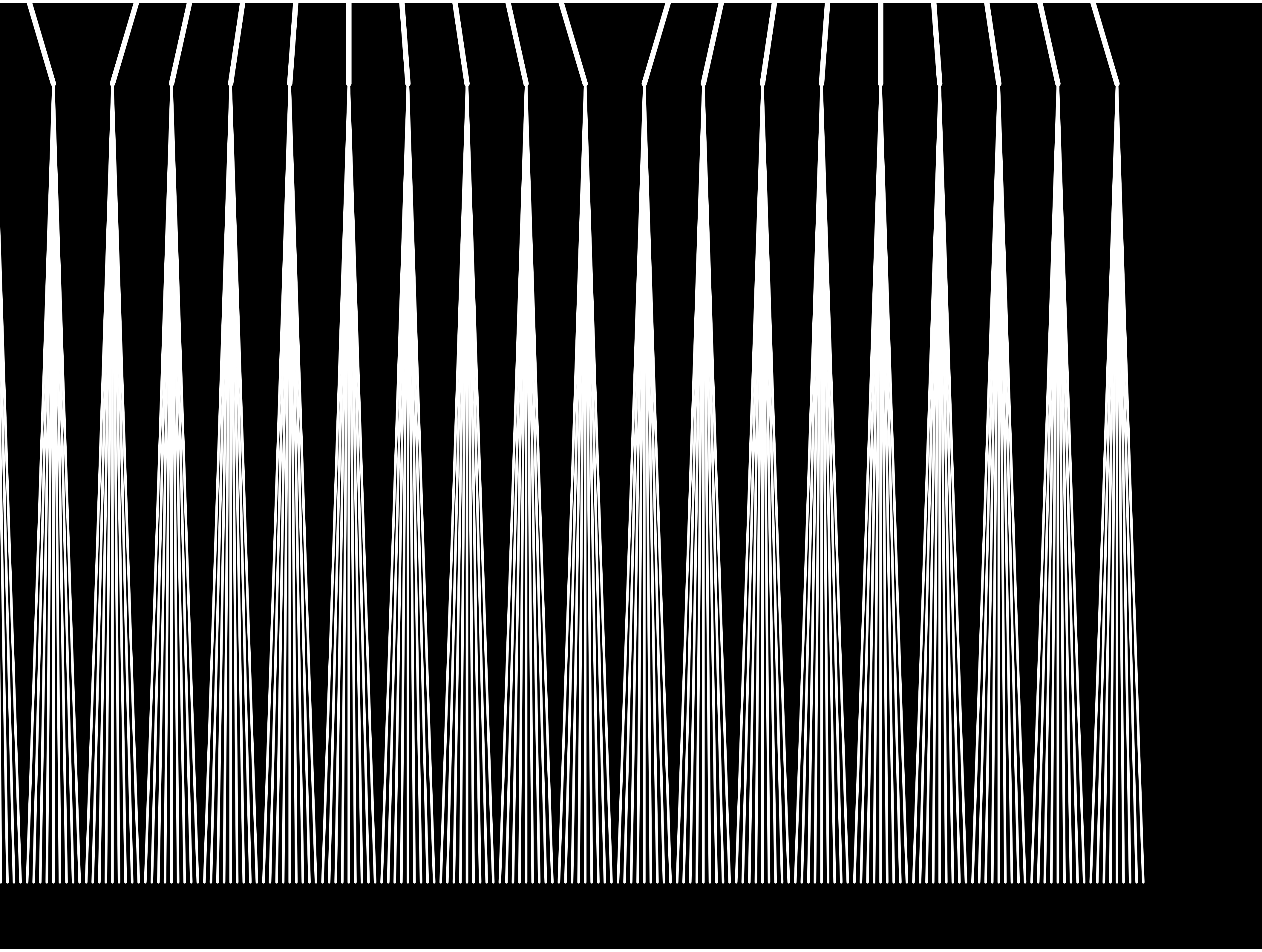
Exponential Domains

- Characterized by:
  - Branching factor *b*
  - Depth of tree, *d*
- Assume a tree not a graph









$$N(b,d) = 1 + b + b^2 + \ldots + b^{d-1} \boxed{+ b^d}$$

$$bN(b,d) = b + b^2 + b^3 + \ldots + b^d + b^{d+1}$$

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

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

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

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

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

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

Depth-First Iterative Deepening,
An Optimal Admissible Tree Search
Richard Korf
Artificial Intelligence, **1985**

Uninformed Algorithms

- Basic graph algorithms
  - DFS
    - Not necessarily optimal if goal not at a leaf
  - BFS
    - High memory requirement
    - Can use external-memory
      - DEMBFS - IJCAI 2019 (Hu & Sturtevant, 2019)

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

http://www.movingai.com/SAS/EXP

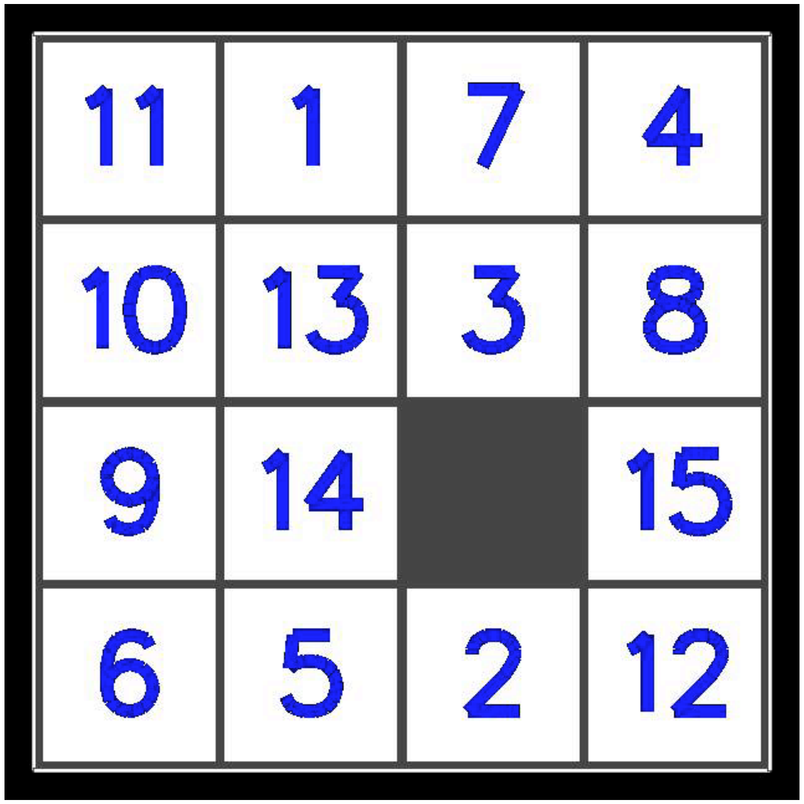
Demo tasks

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

UNIVERSITY OF ALBERTA

Sample Domain/Heuristic

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



IDA*

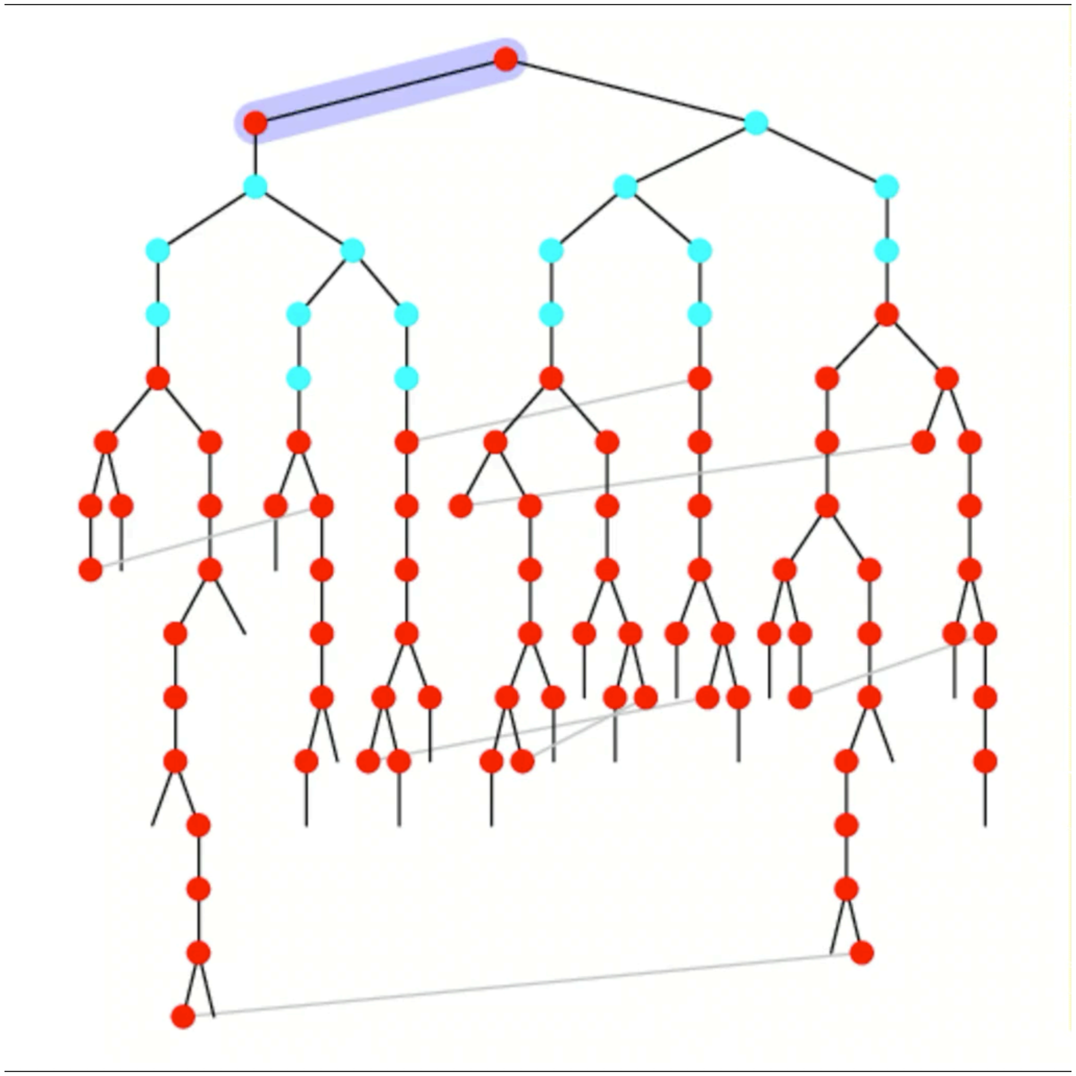
- Instead of iteratively searching depth layers (DFID)
- Search f-cost layers
    - f(*n*) = g(*n*)+h(*n*)
    - Estimate of the total path length
    - Searching from shortest to longest paths
- Expected running time O(b^d)

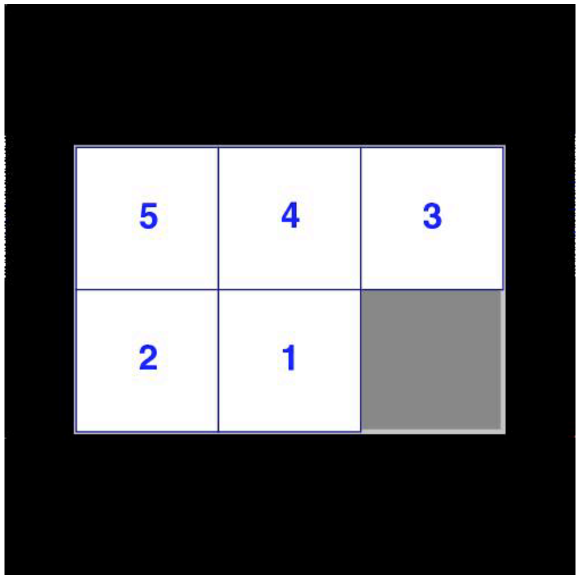
Depth-First Iterative Deepening, An Optimal Admissible Tree Search
Richard Korf
Artificial Intelligence, **1985**

Demo

http://www.movingai.com/SAS/IDA

IDA* - Unit Costs





UNIVERSITY OF ALBERTA

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?

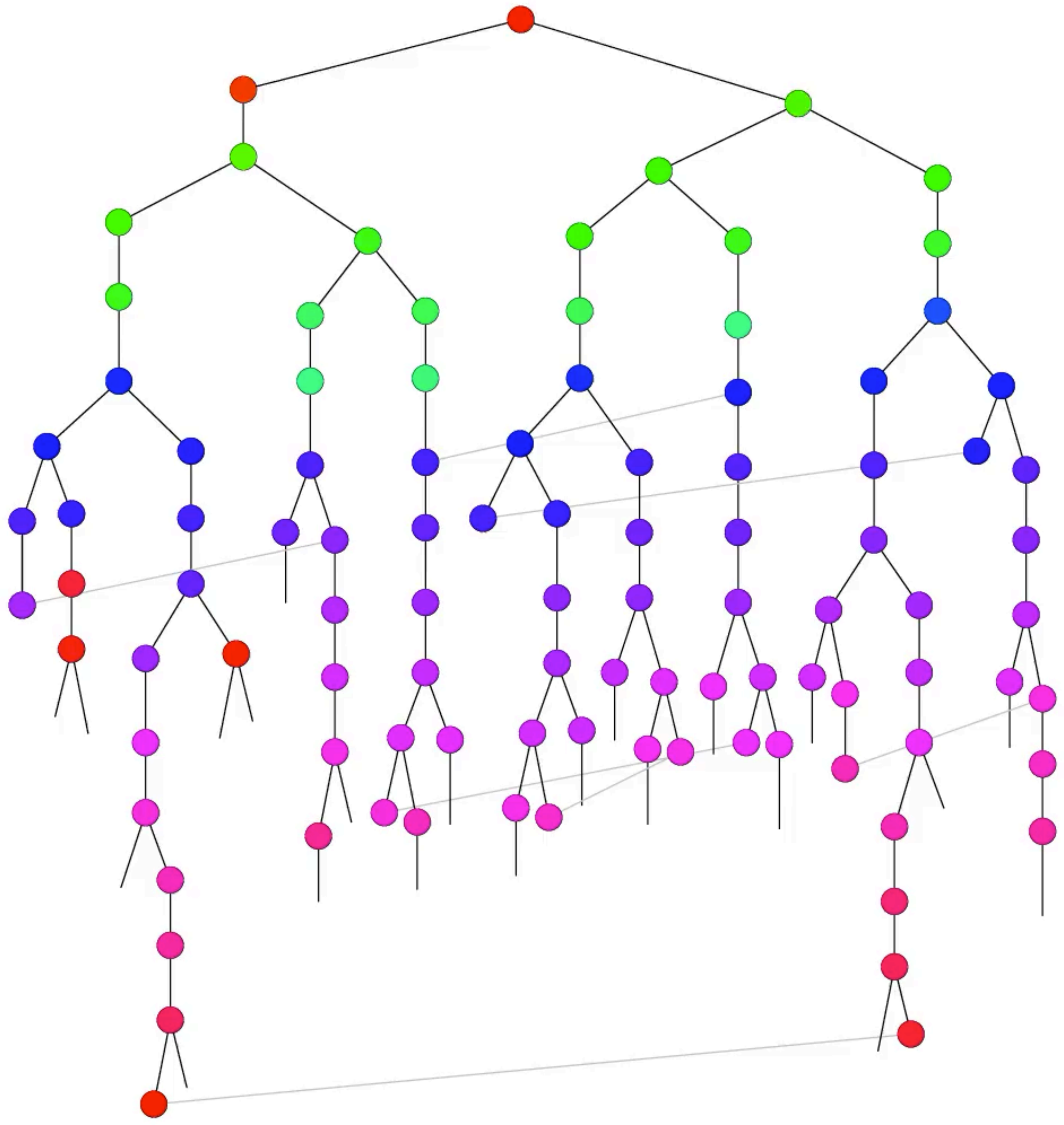
UNIVERSITY OF ALBERTA

IDA* Worst Case

- Analysis assumes tree grows exponentially
- Otherwise might expand (in tree with N nodes):

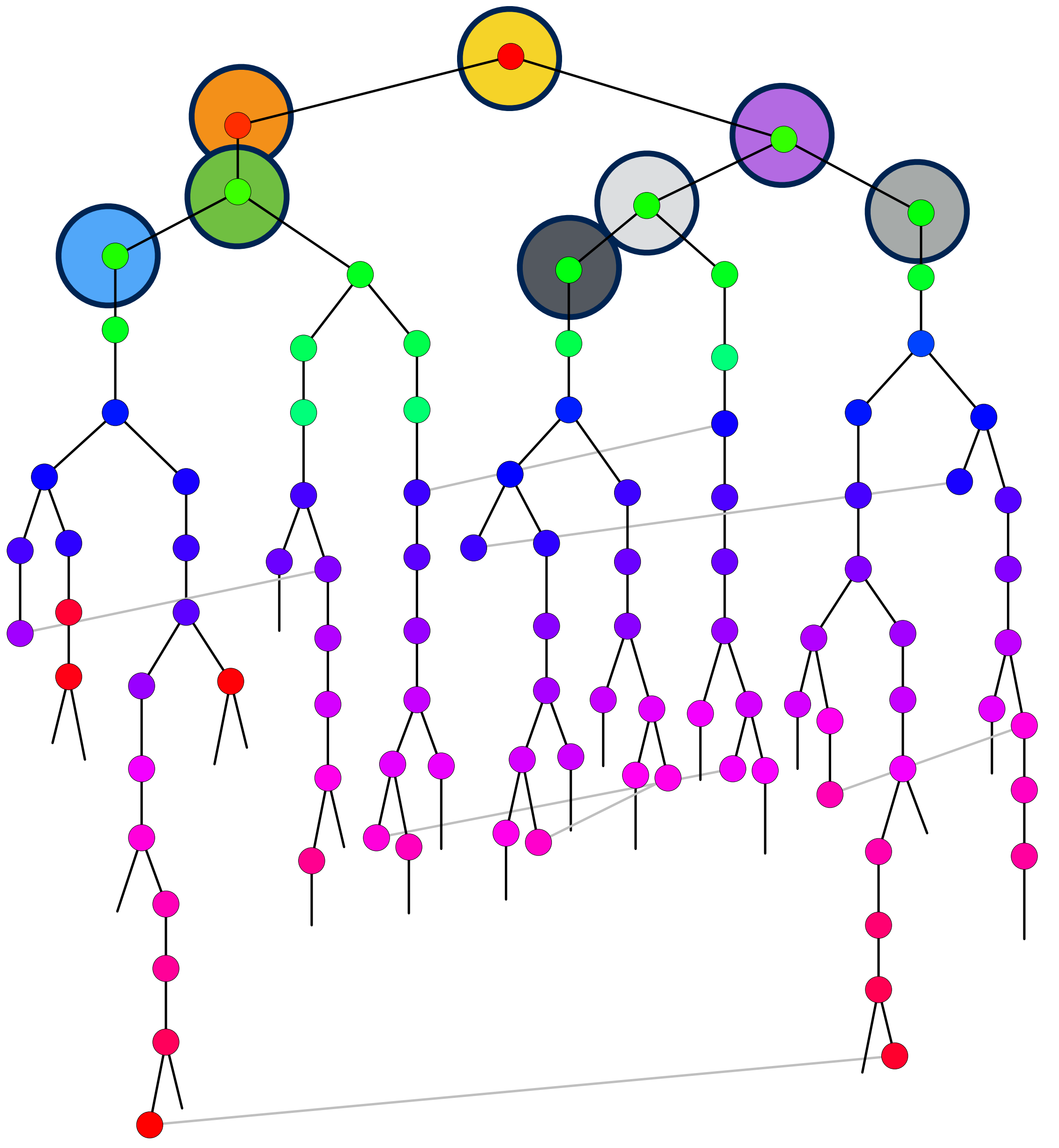
$$1 + 2 + 3 + \cdots + N = \frac{(N)(N+1)}{2}$$

- If $N = b^d$, this is b^{2d} expansions!



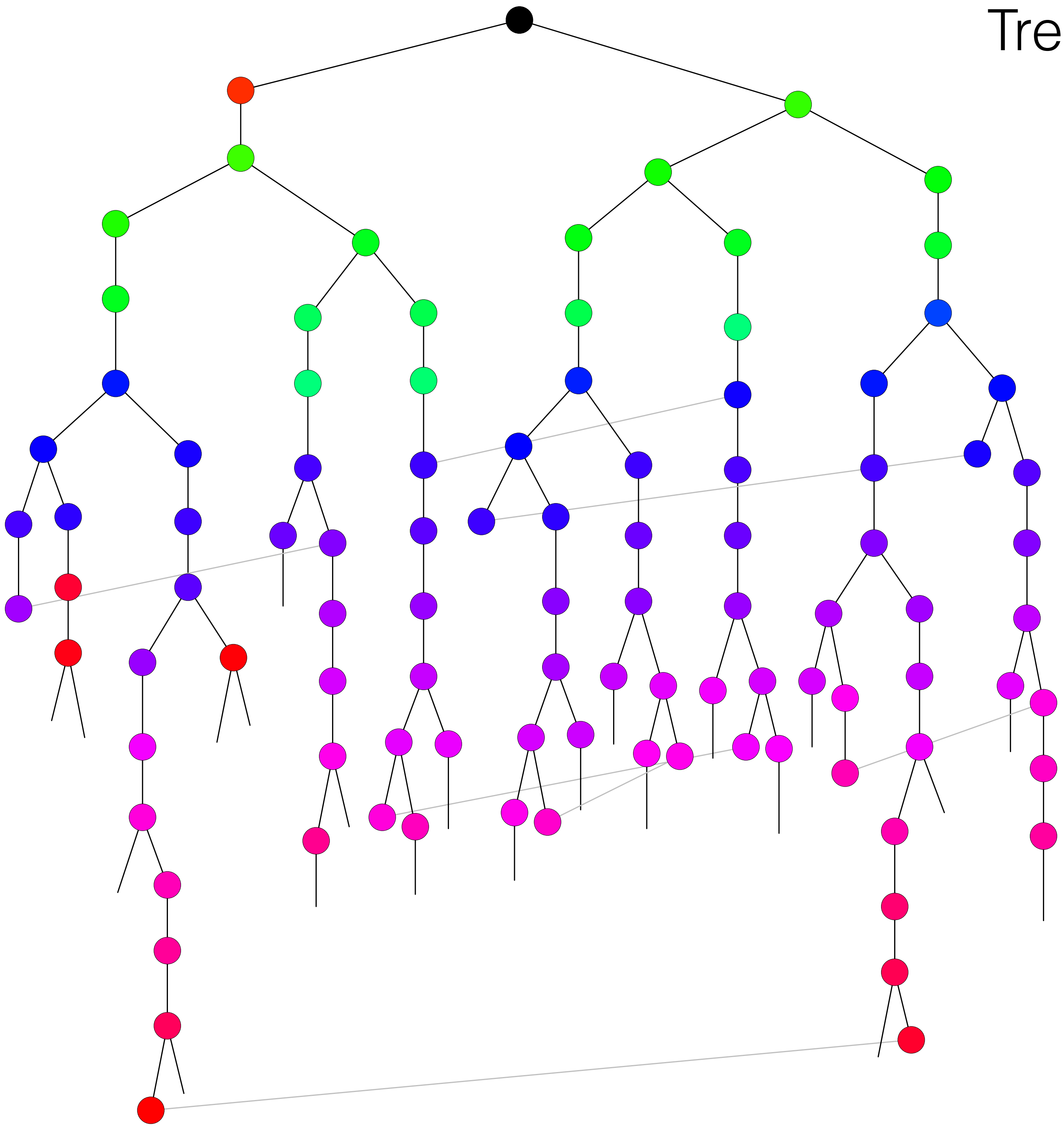
f-cost 11
11.25
13.45
13.5
13.62
13.7
13.83
13.87

Tree Size: 1
2
3
4
5
6
7
8



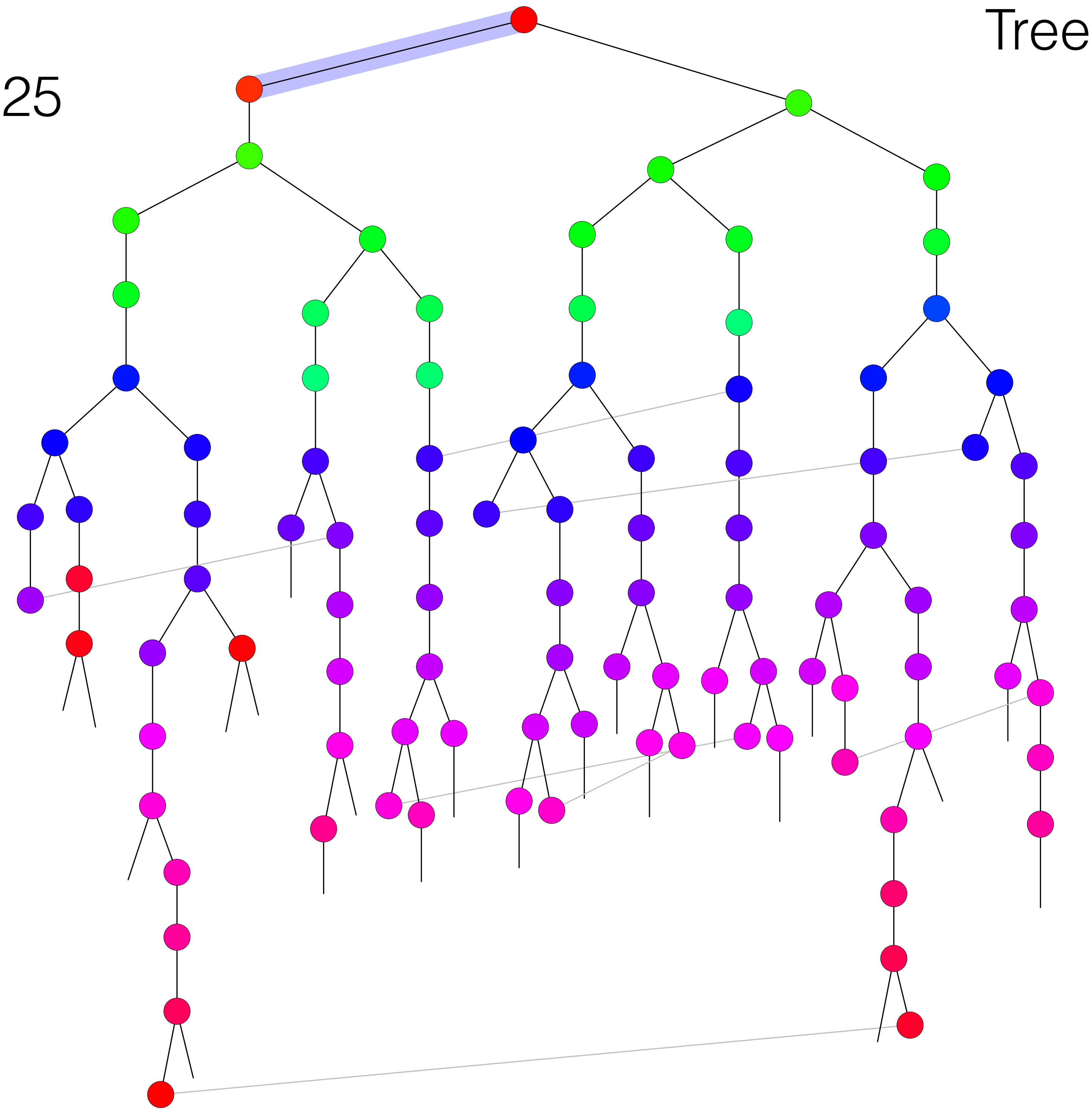
f-cost 11

Tree Size: 1



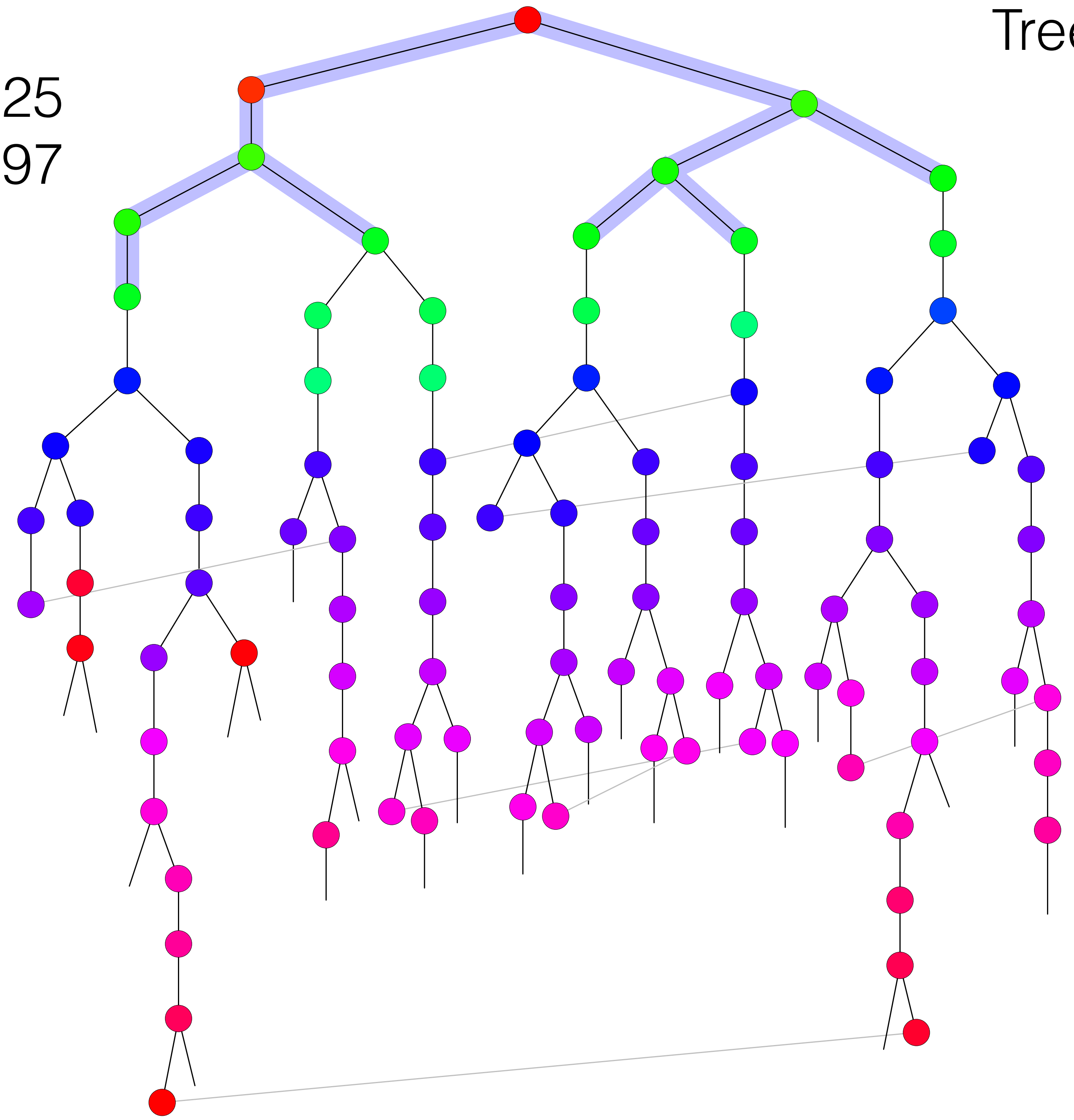
f-cost 11
11.25

Tree Size: 1
2

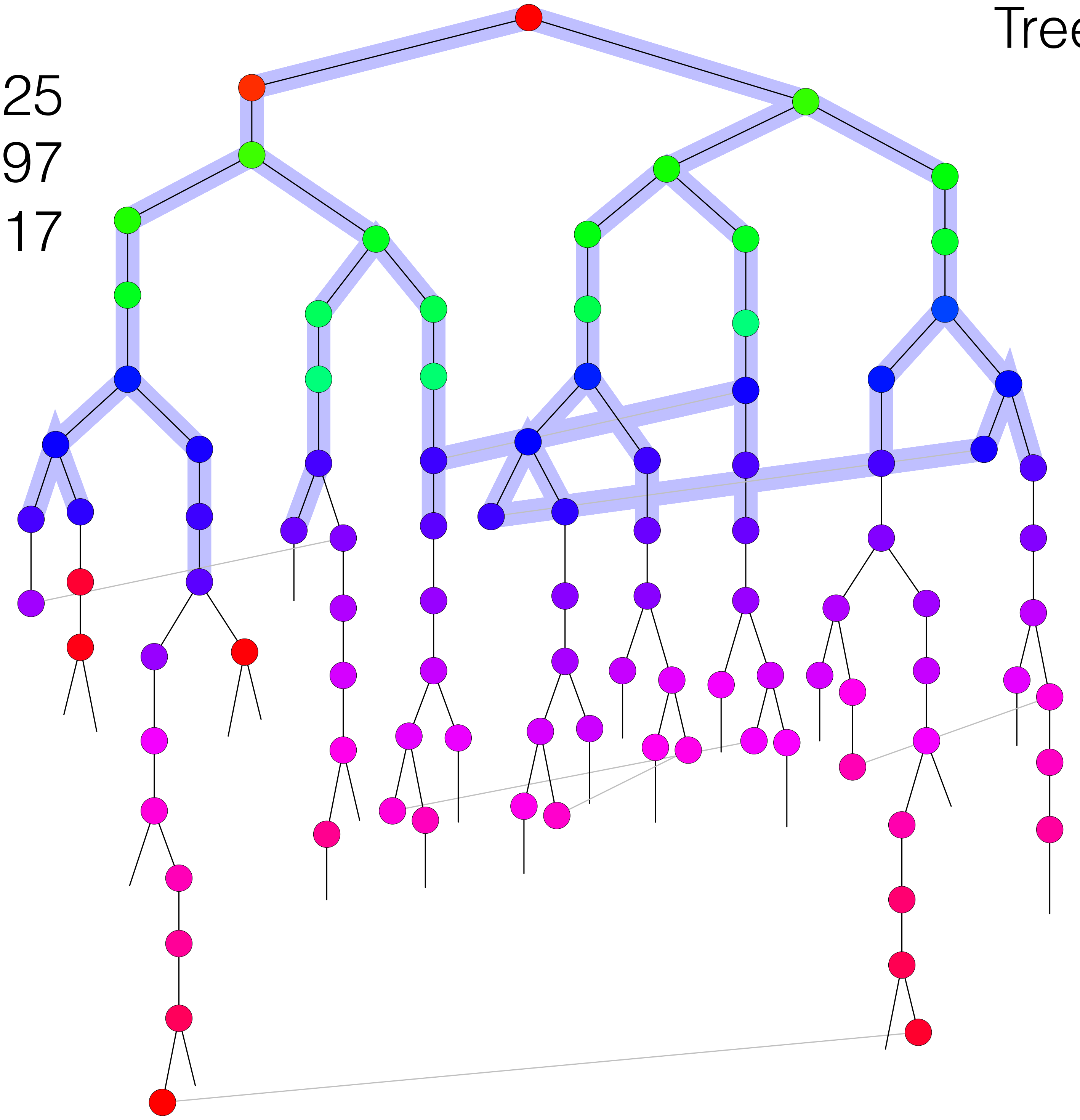


f-cost 11
11.25
13.97

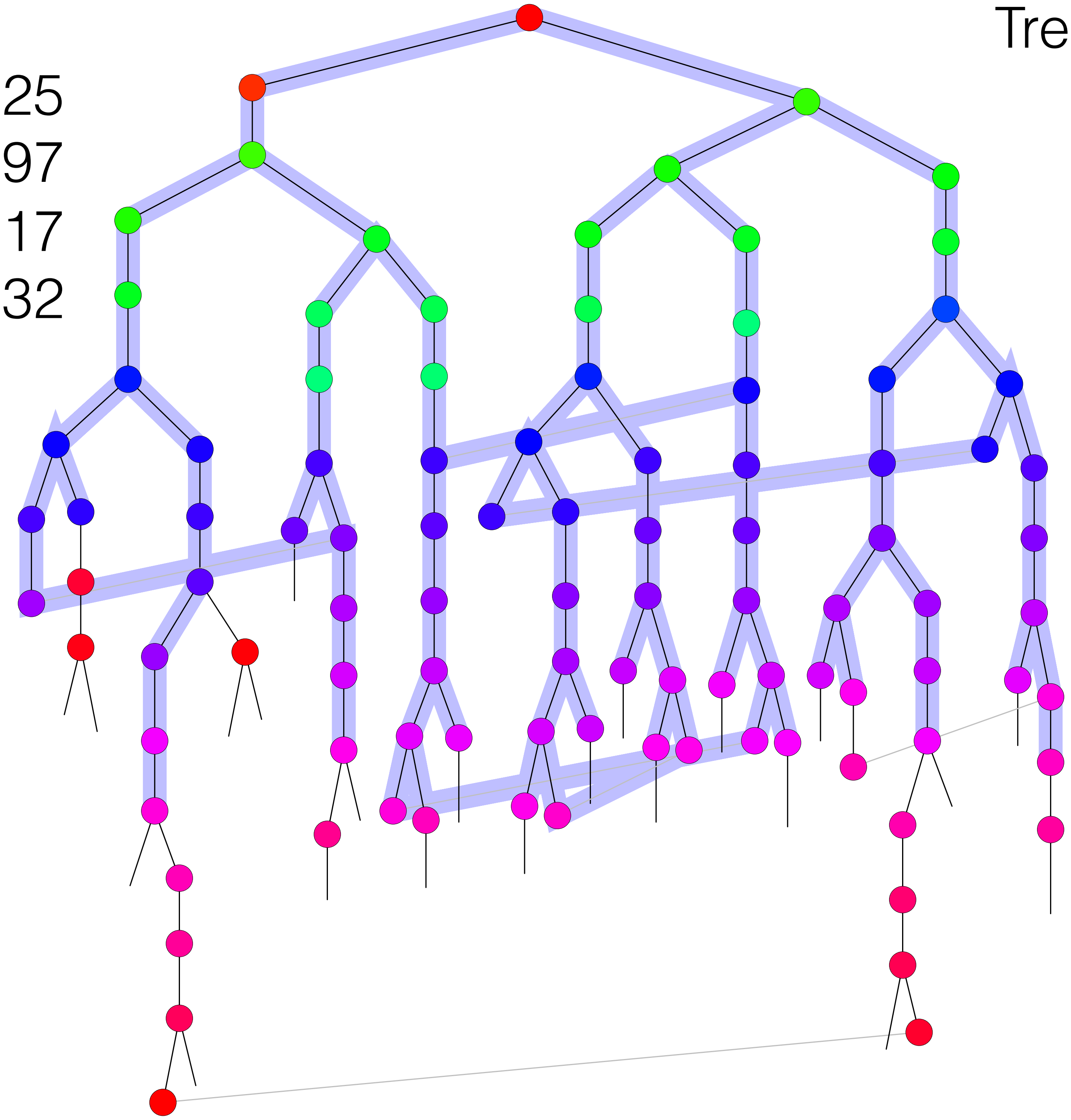
Tree Size: 1
2
11

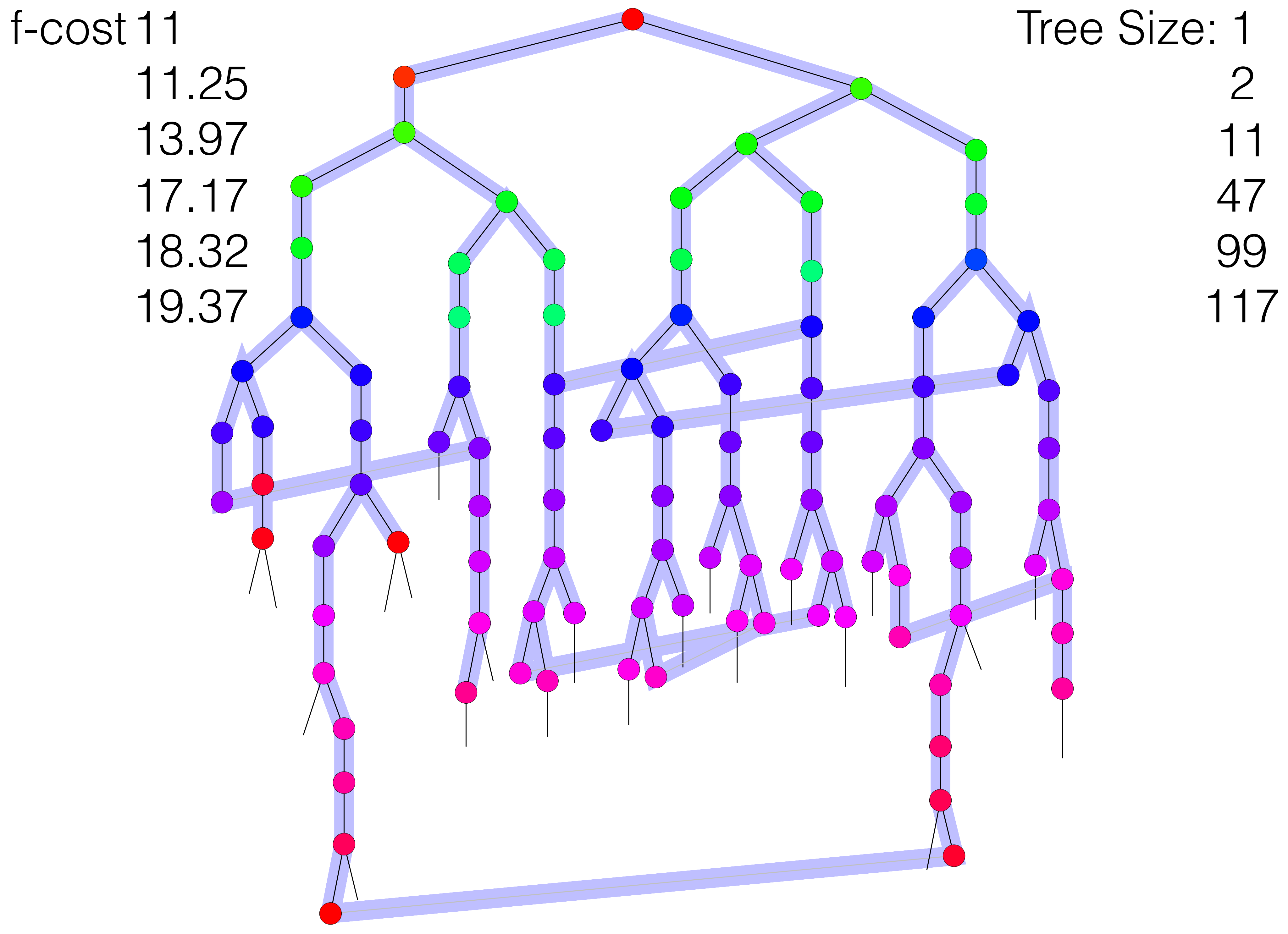


f-cost	Tree Size:
11	1
11.25	2
13.97	11
17.17	47

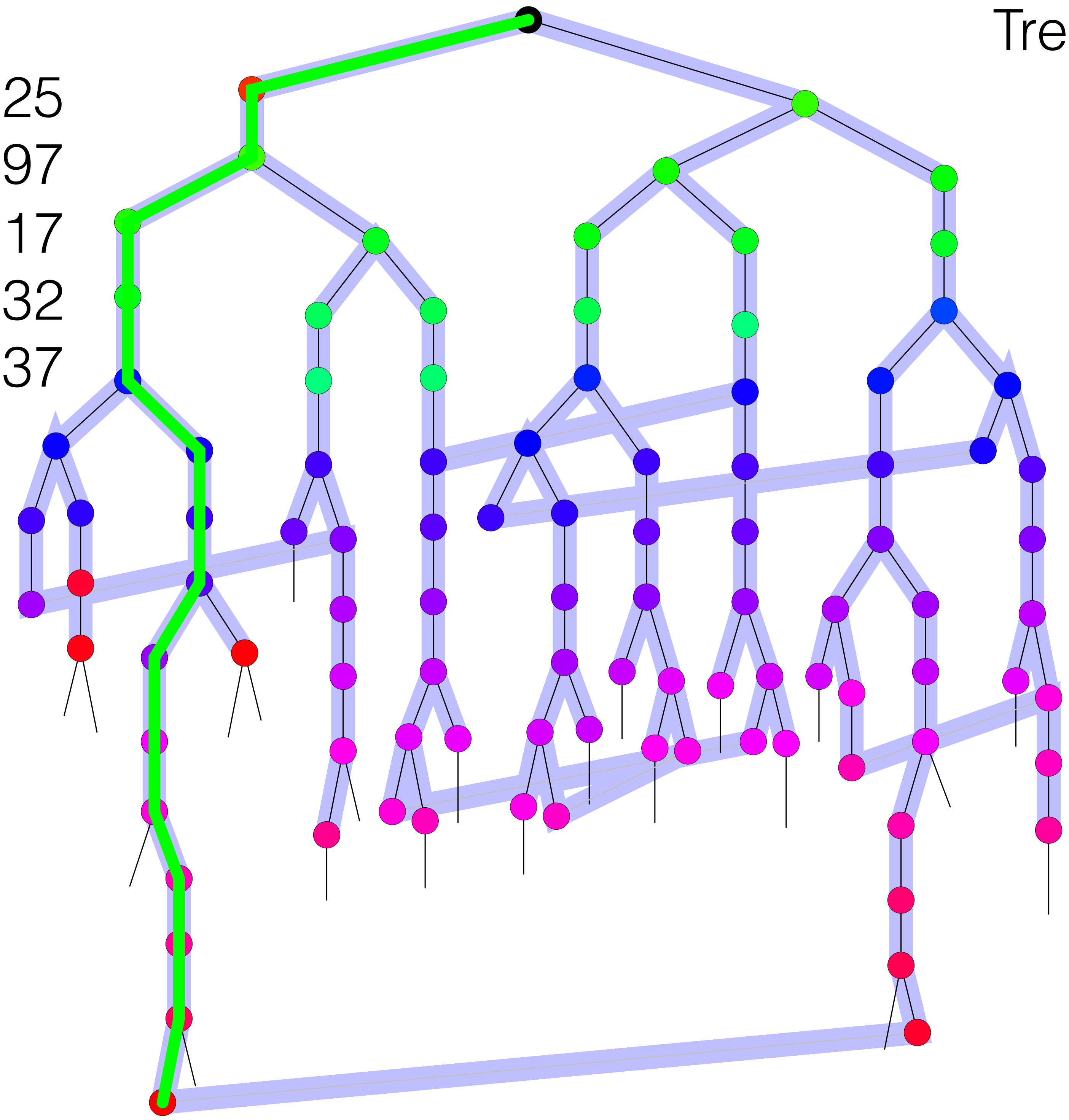


f-cost	Tree Size:
11	1
11.25	2
13.97	11
17.17	47
18.32	99




f-cost 11
11.25
13.97
17.17
18.32
19.37
Tree Size: 1
2
11
47
99
117

f-cost	Tree Size:
11	1
11.25	2
13.97	11
17.17	47
18.32	99
19.37	117



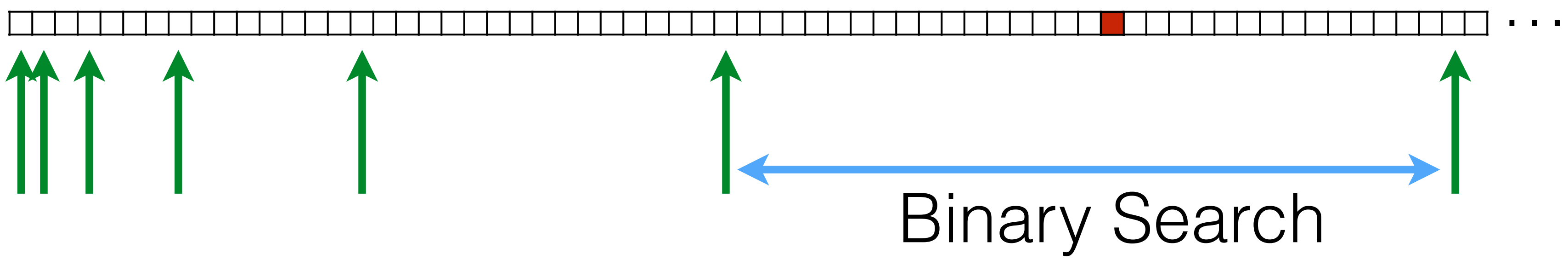
Getting the next bound

- Can be conservative:
  - IDA* (Korf, 1985)
- Can try to build a predictor based on past:
  - IDA*$_{CR}$ (Sarkar et al, 1990)
  - IDA*$_{IM}$ (Burns & Ruml, 2013)
- Can model the state space growth:
  - EDA* (Sharon et al, 2014)
- Want to guarantee exponential growth in expansions

UNIVERSITY OF ALBERTA

Exponential Search

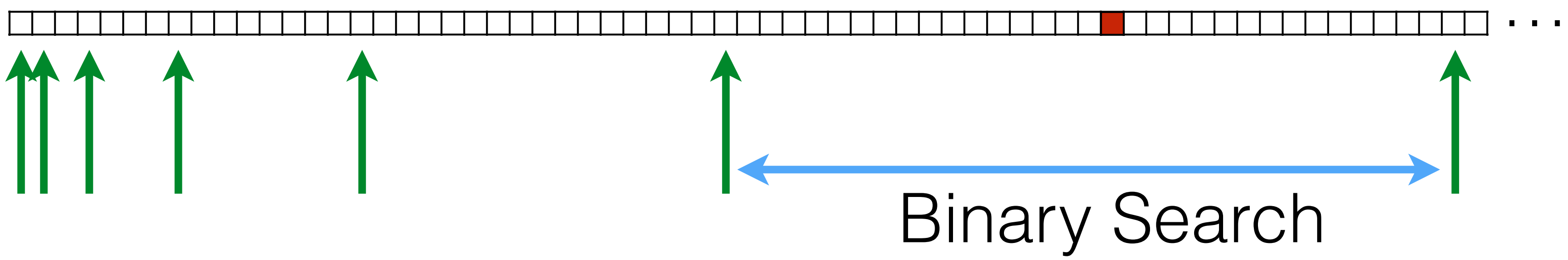
- Bentley and Yao, 1976
- Algorithm for searching sorted/unbounded array



- Running time: $\log(i)$

Exponential Search for Bounds

- Imagine array as all floating point numbers
- Algorithm for searching sorted/unbounded array

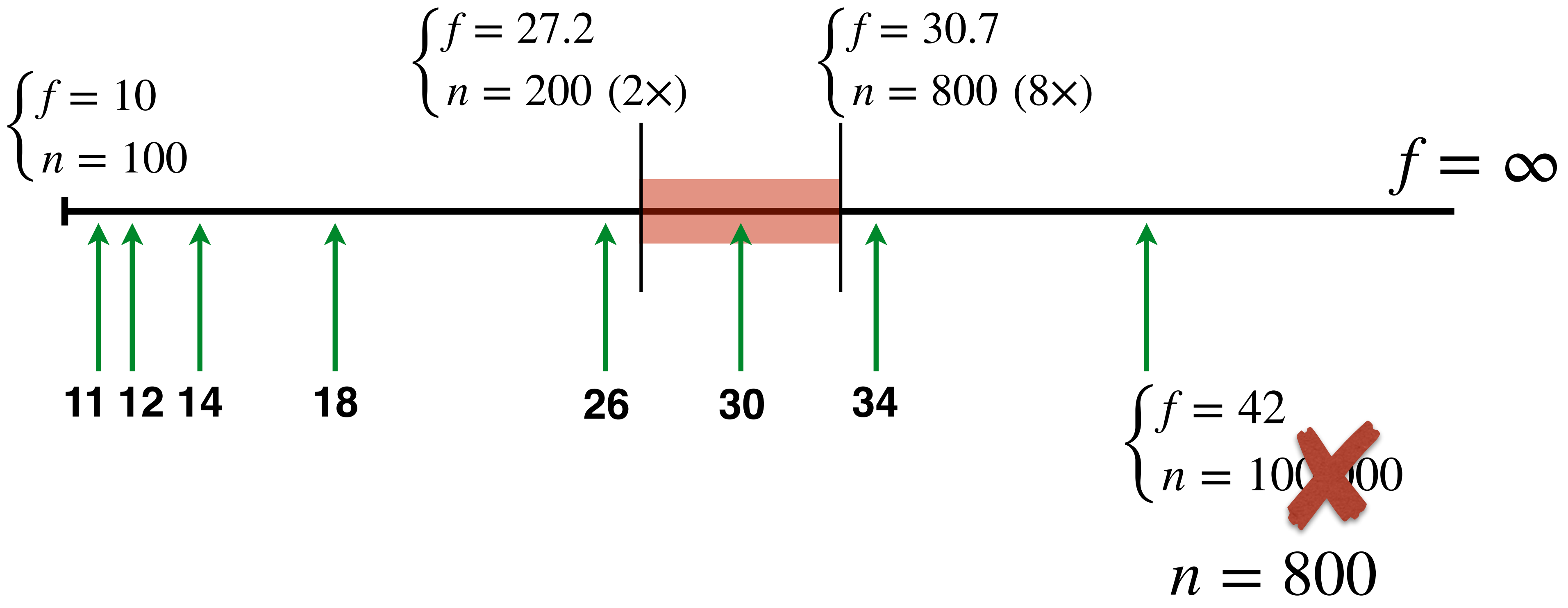


- Problem: Searching with large *f* does extra work

UNIVERSITY OF ALBERTA

Budgeted search

- Like exponential search on f-costs


$\begin{cases} f = 10 \\ n = 100 \end{cases}$
$\begin{cases} f = 27.2 \\ n = 200\ (2\times) \end{cases}$
$\begin{cases} f = 30.7 \\ n = 800\ (8\times) \end{cases}$
$f = \infty$
11 12 14 18 26 30 34
$\begin{cases} f = 42 \\ n = 100{,}000 \end{cases}$
$n = 800$

UNIVERSITY OF ALBERTA

IBEX - BTS

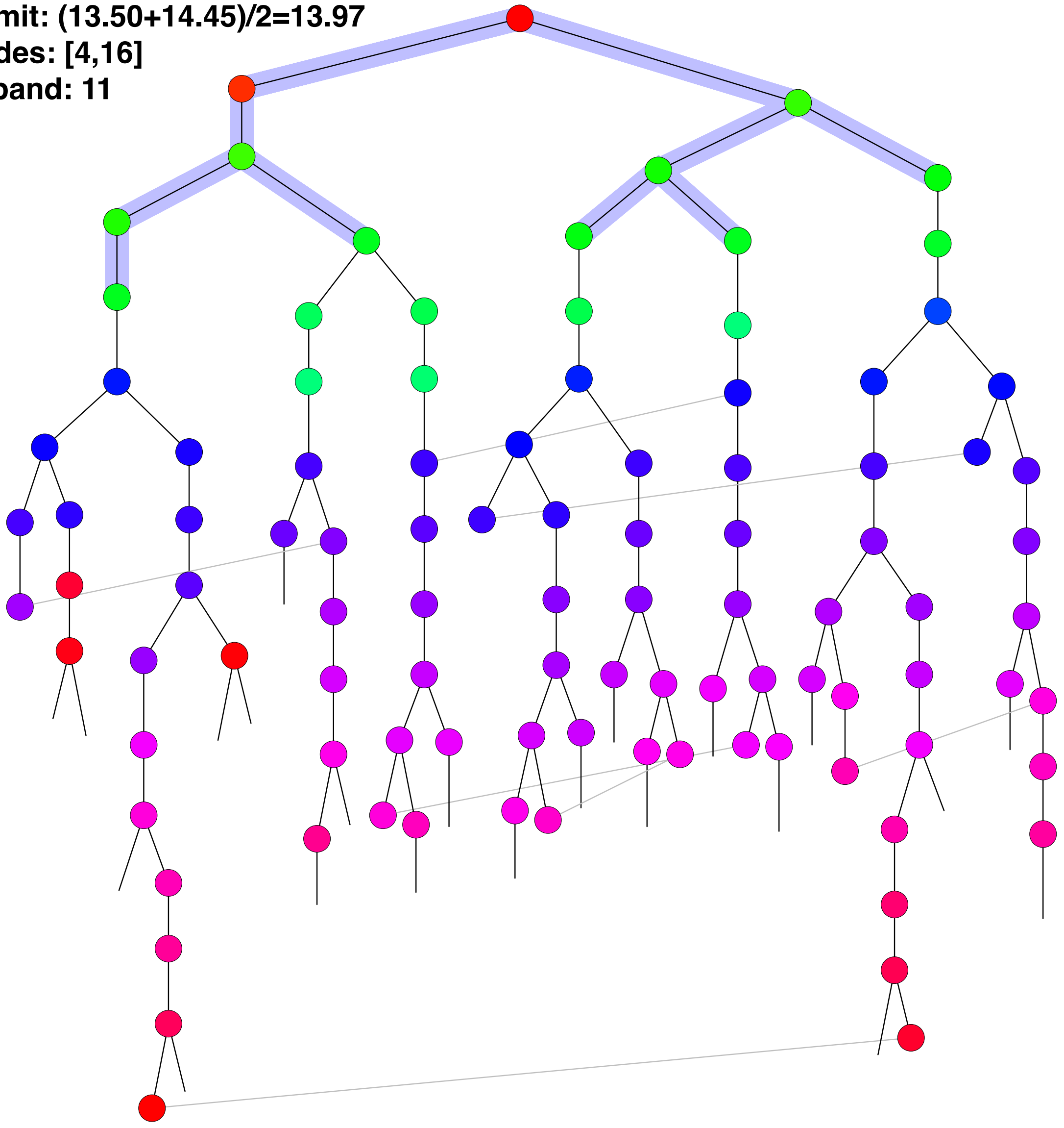
Find next f-cost bound:

1. Search with conservative f, ∞ budget (IDA*)
2. Grow f bound exponentially
3. Do binary search on f

Constant budget

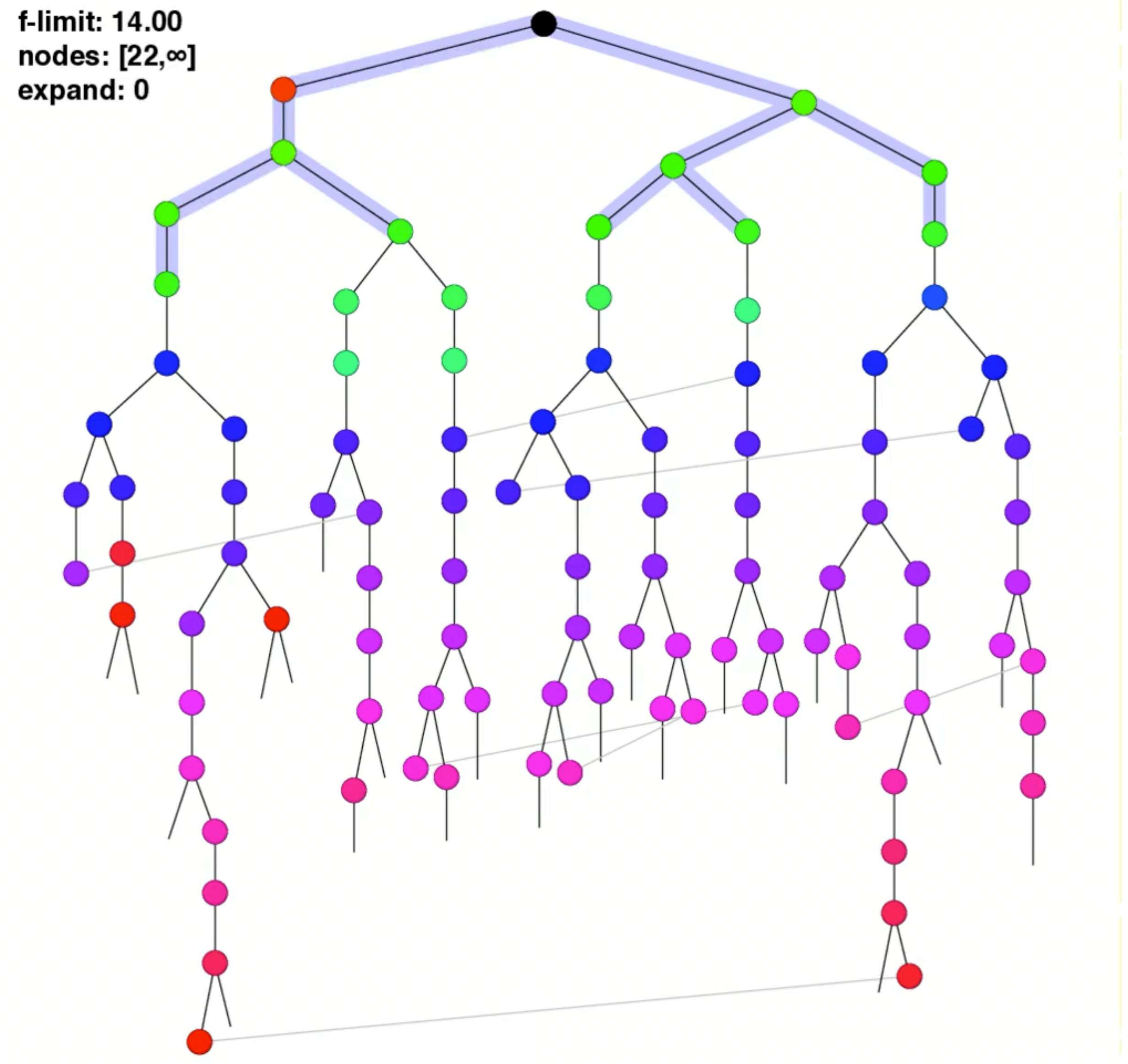
Iterative Budgeted Exponential Search
Malte Helmert, Tor Lattimore, Levi H. S. Lelis, Laurent Orseau, Nathan R. Sturtevant
IJCAI, **2019**

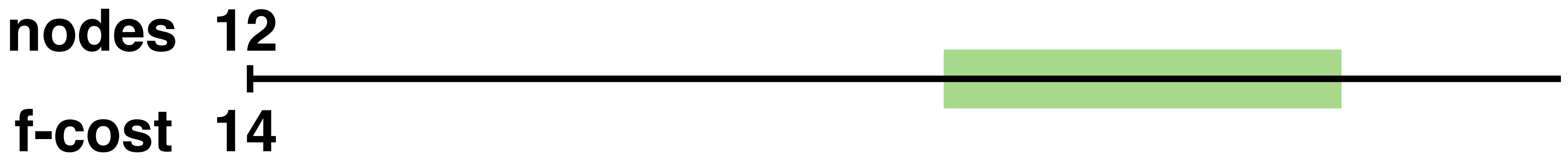
Previous Iteration



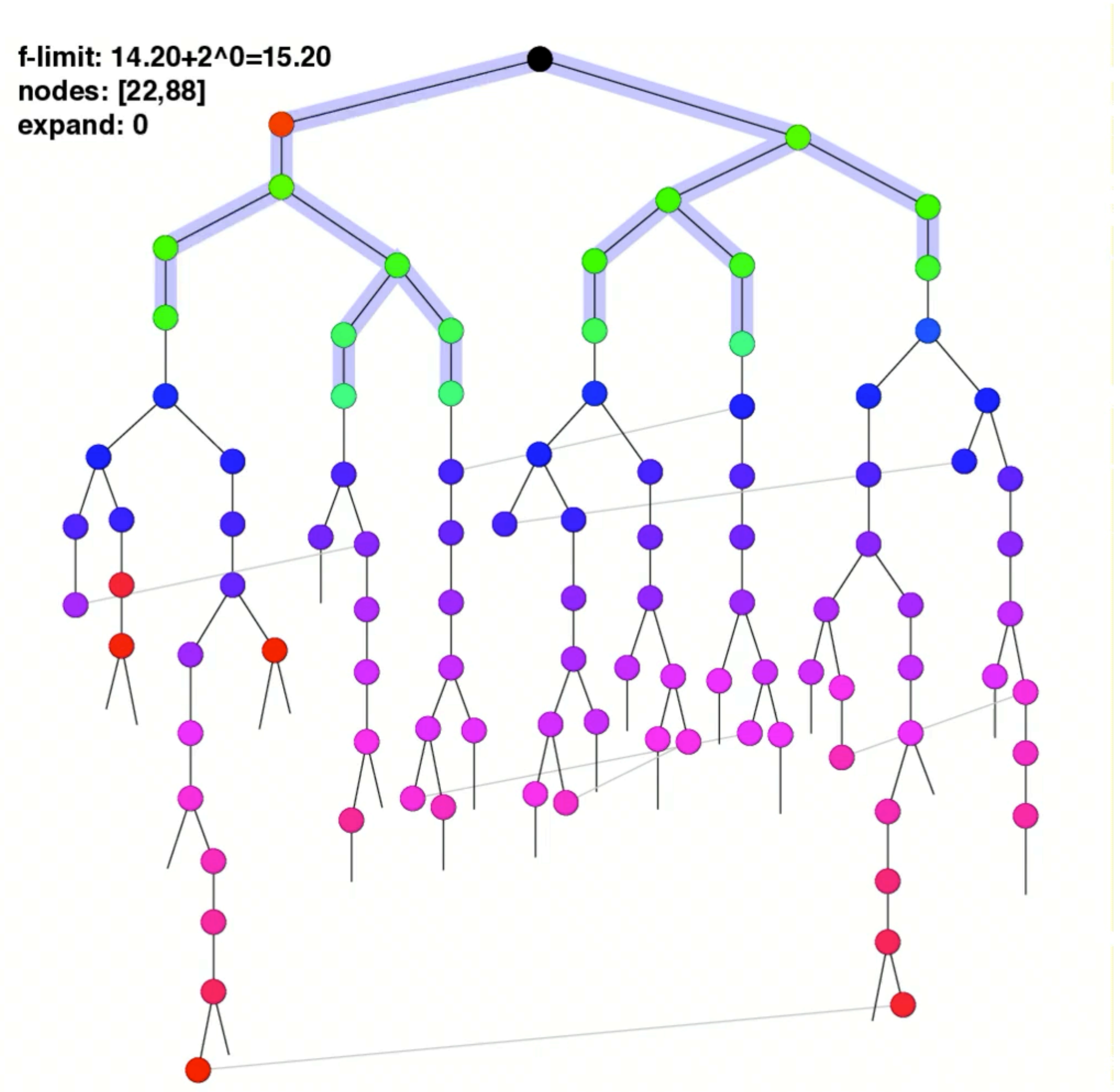
f: 13.97 nodes: 11

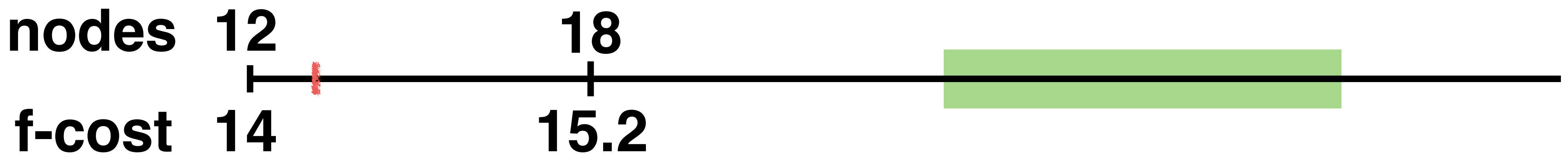
IDA*



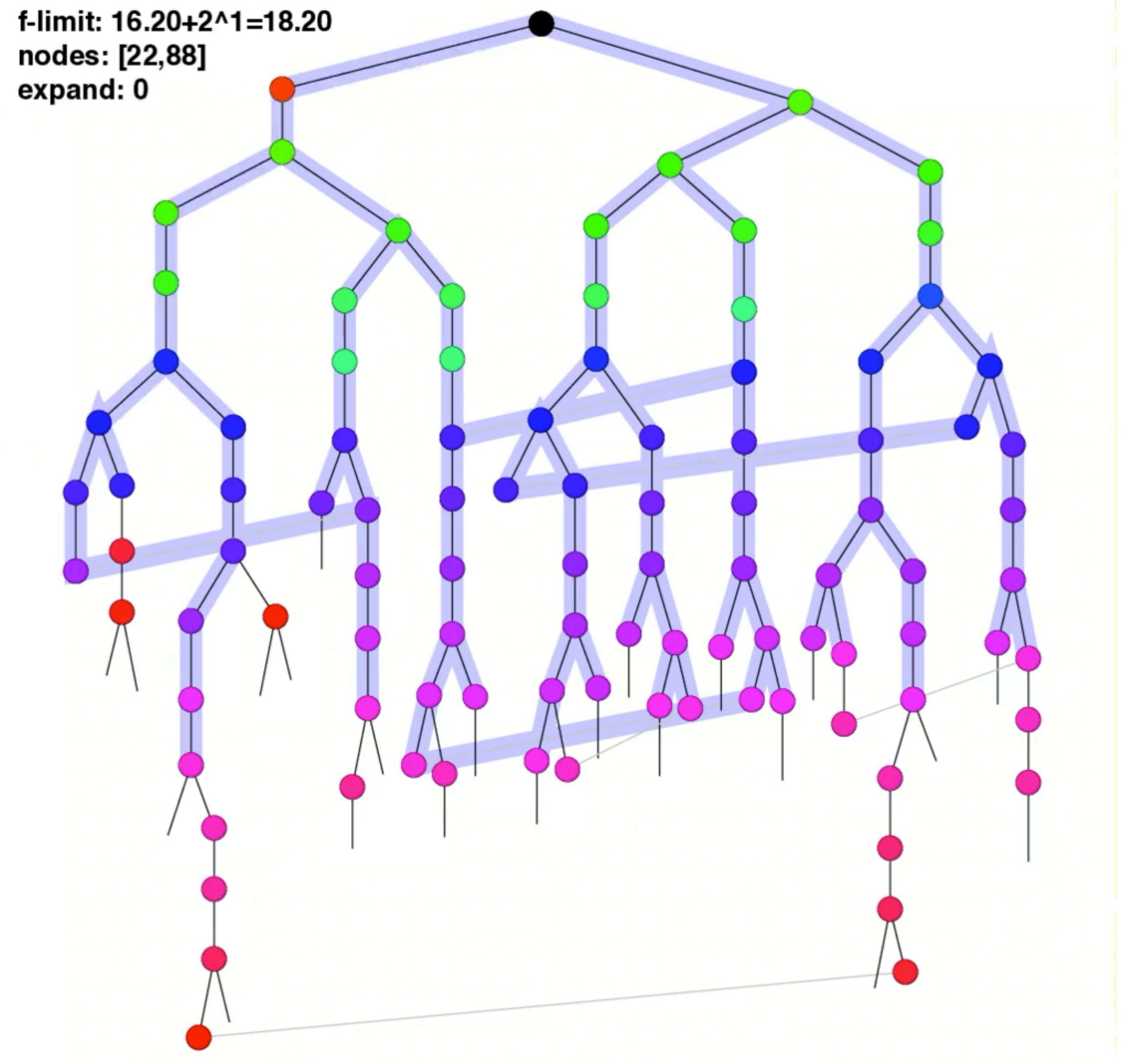


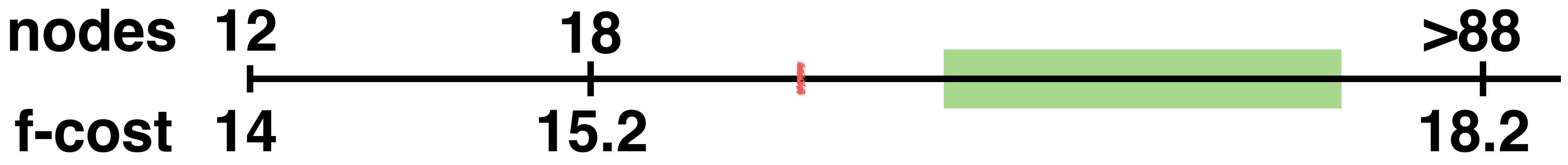
EXP


f-limit: 14.20+2^0=15.20
nodes: [22,88]
expand: 0

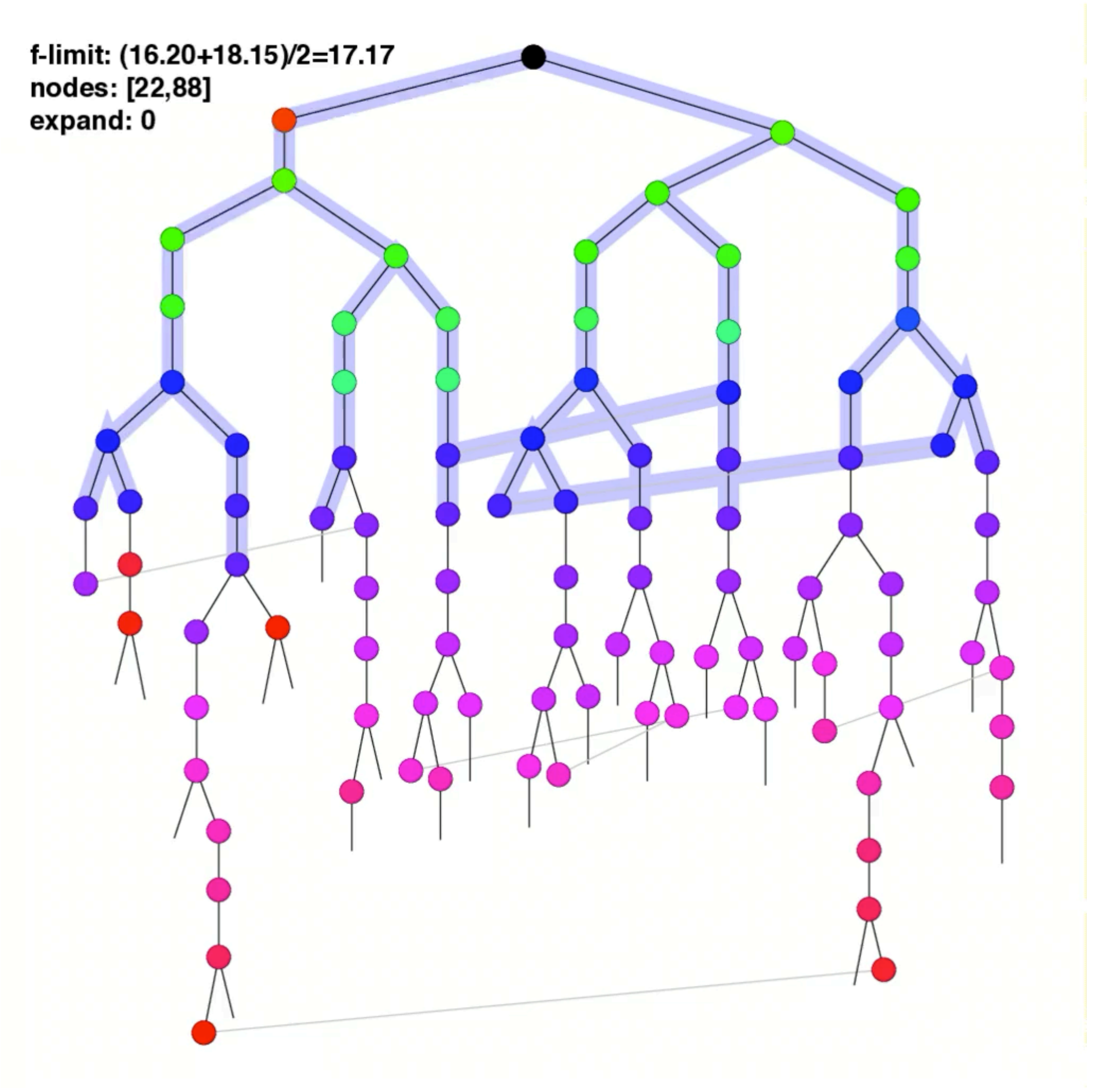

nodes
12
18
f-cost
14
15.2

EXP

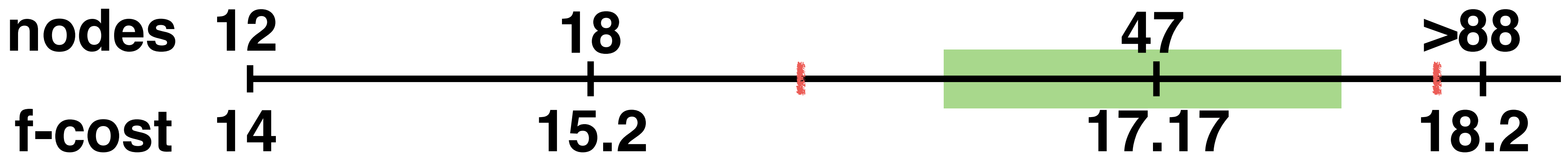

f-limit: 16.20+2^1=18.20
nodes: [22,88]
expand: 0


nodes
12
18
>88
f-cost
14
15.2
18.2

BIN



f-limit: (16.20+18.15)/2=17.17
nodes: [22,88]
expand: 0



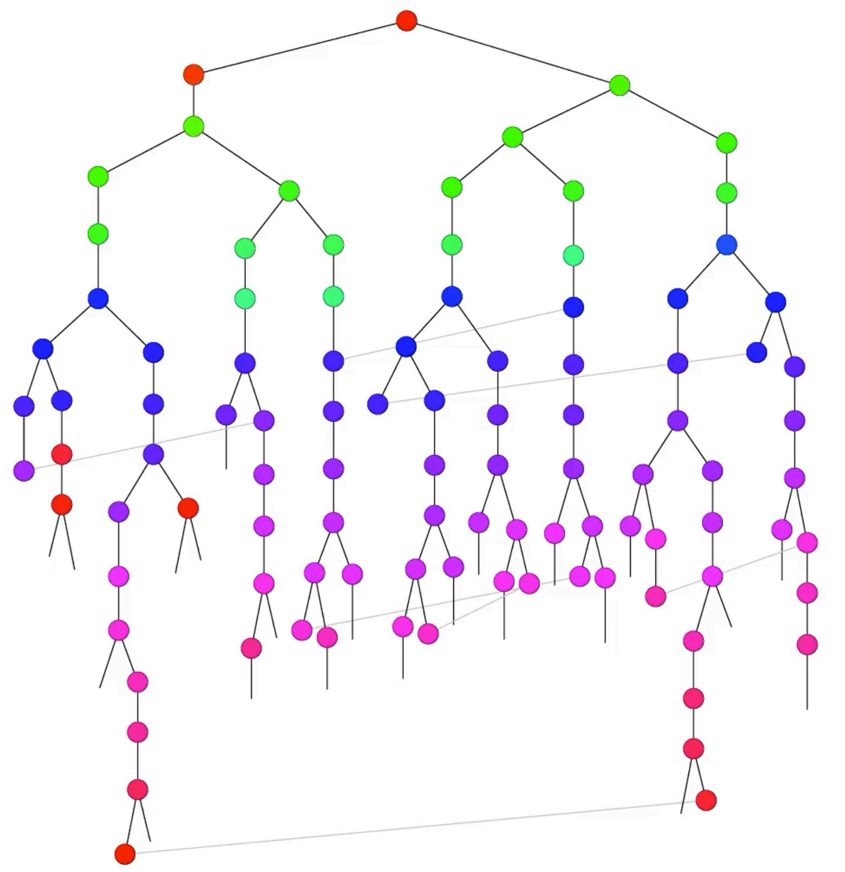
nodes
12
18
47
>88
f-cost
14
15.2
17.17
18.2

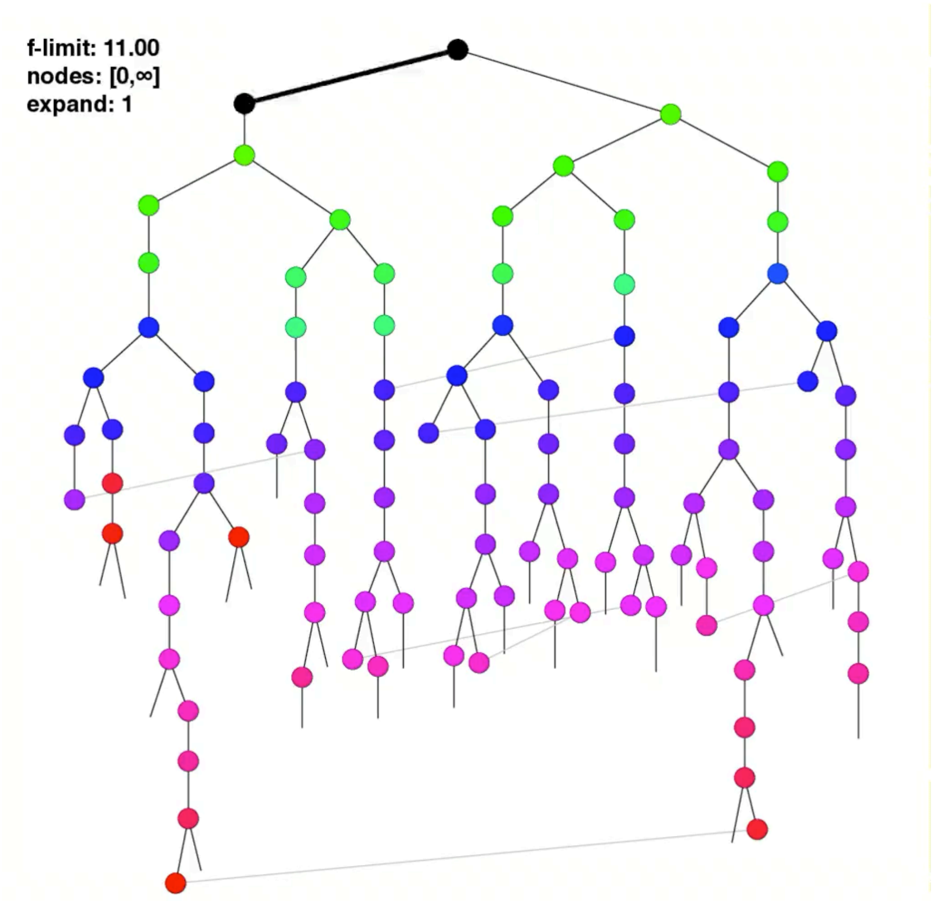
Demo

https://www.movingai.com/SAS/BTS

IDA*

BTS





86.6s

12.7s

Summary

- IBEX/BTS
    - Avoids worst case of IDA* - O(N log C*)
    - Exactly IDA* in best case

Polynomial Algorithms

Recall: Exponential Domains

- Characterized by:
  - branching factor *b*
  - depth of tree, *d*
- Usually assume a tree not a graph

Polynomial Domains

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

Polynomial Domains

- Often still have constant branching factor b
  - If we search to depth d
    - b^d states?
    - 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

UNIVERSITY OF ALBERTA

Terminology

- OPEN - Priority queue of states
- CLOSED - Hash table

- EXPAND - Take from OPEN and generate successors
- GENERATE - Get state from parent and *process*
- OPEN - States that are GENERATED but not EXPANDED
- CLOSED - States that have been EXPANDED

UNIVERSITY OF ALBERTA

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:
  - GENERATE 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 cost estimate
- With a consistent heuristic,
  states never taken off of CLOSED

Demo

http://www.movingai.com/SAS/ASG

UNIVERSITY OF ALBERTA

Demo task

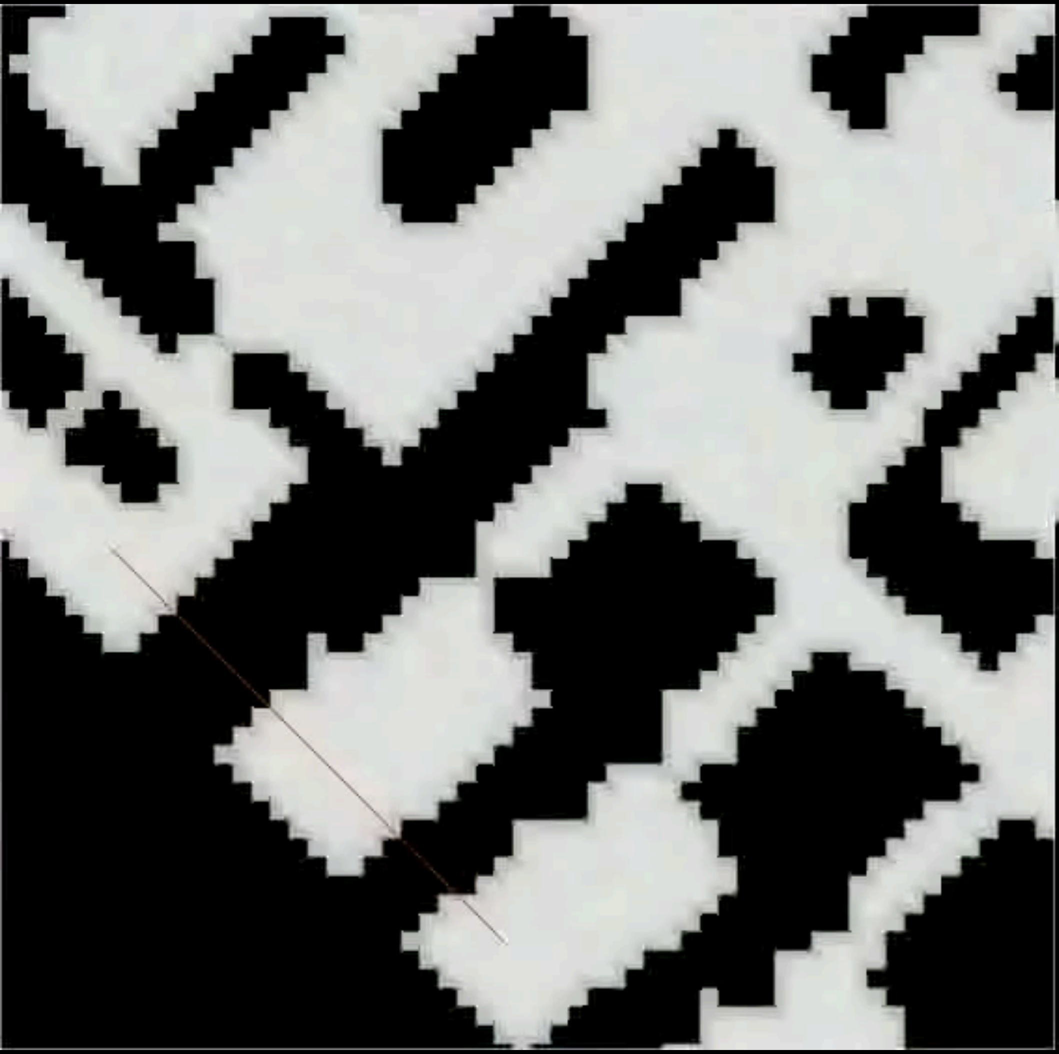
- Build a graph where a state is first generated with suboptimal g-cost
  - (eg the f-cost of a state on open decreases)

A* (B/B')

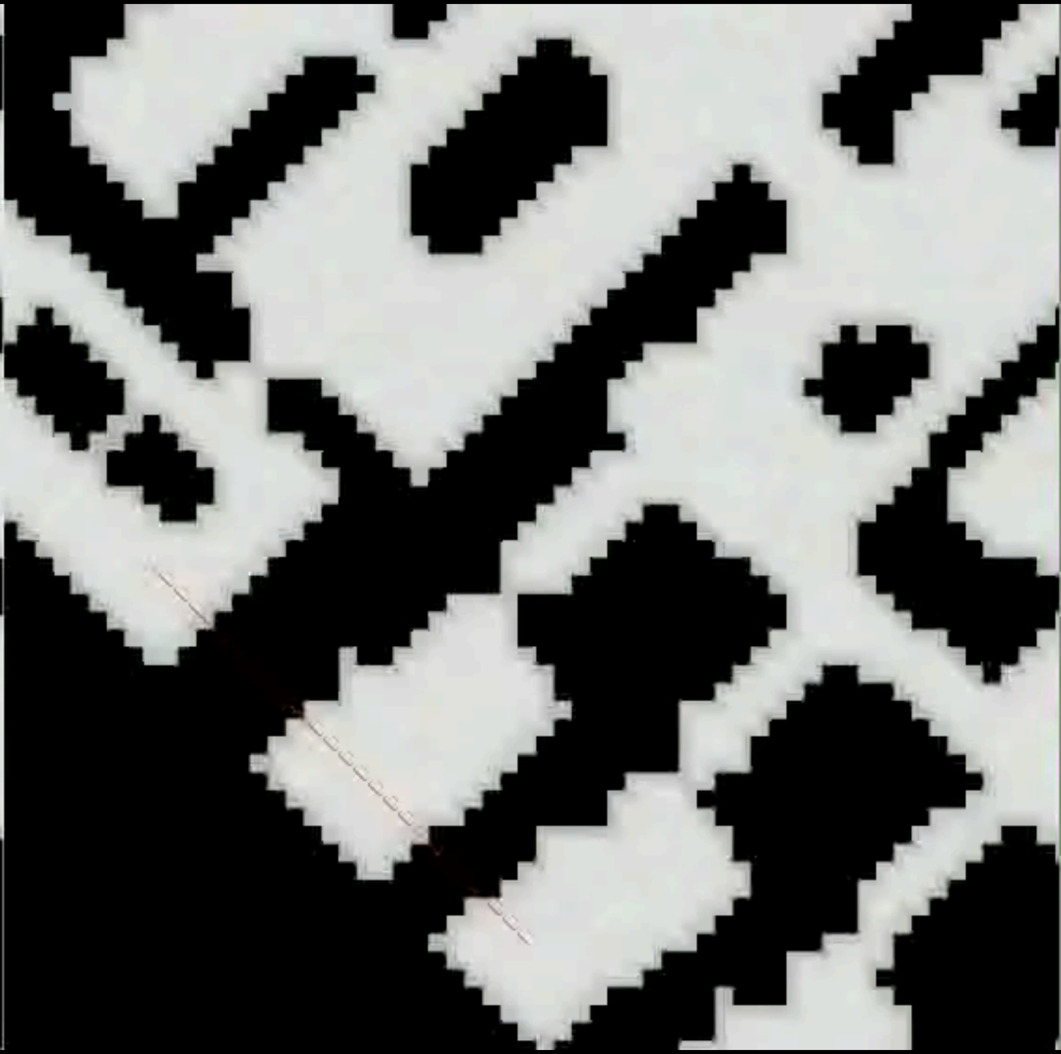
- A* with a consistent heuristic is optimal*
- With an inconsistent heuristic A* might do 2^N expansions of N states
    - Inconsistency comes from compressed or learned heuristics
- B and B' immediately propagate shorter paths
    - Will do at most N^2 expansions of N states

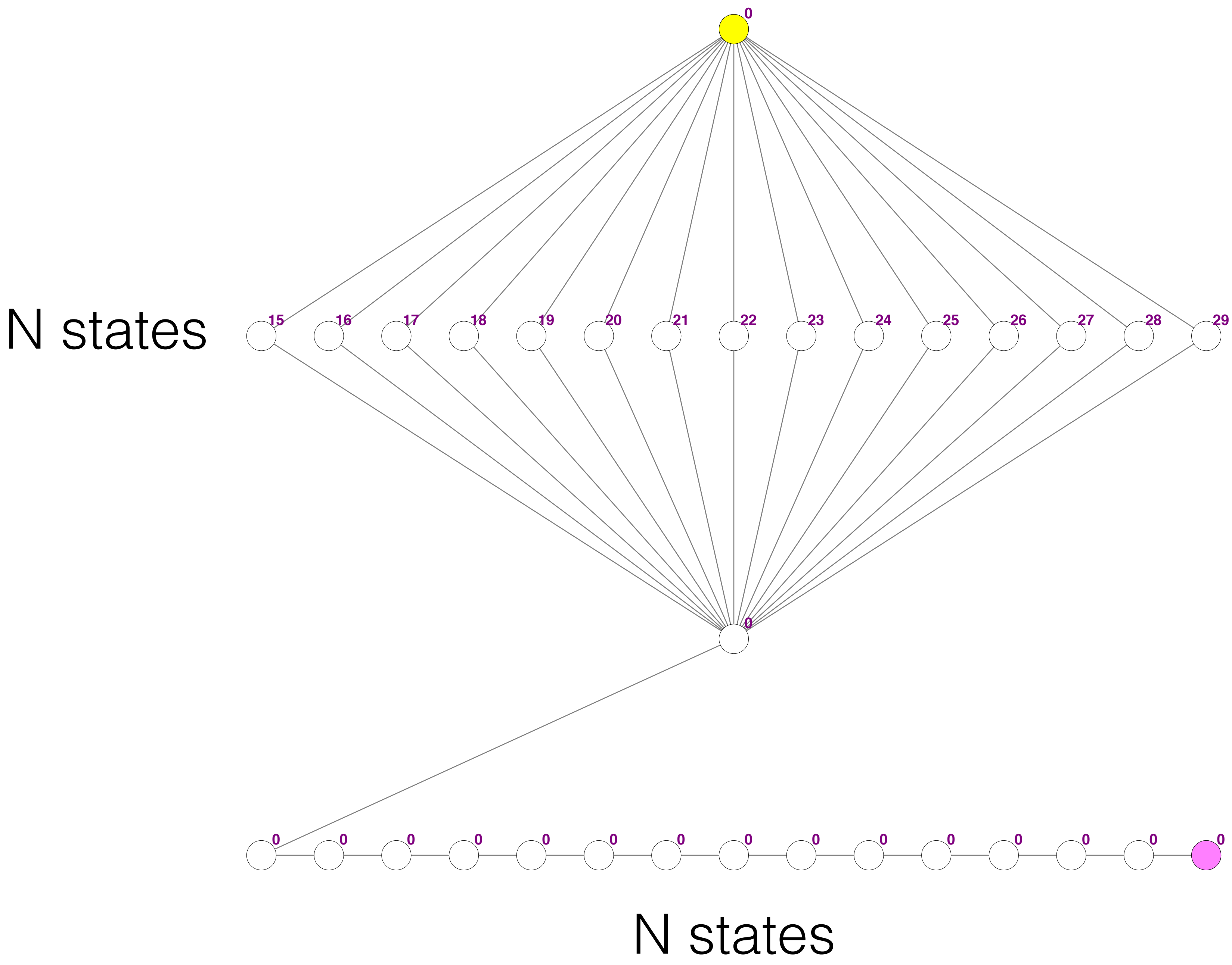
Inconsistent Heuristics

Compressed Differential Heuristic without BPMX

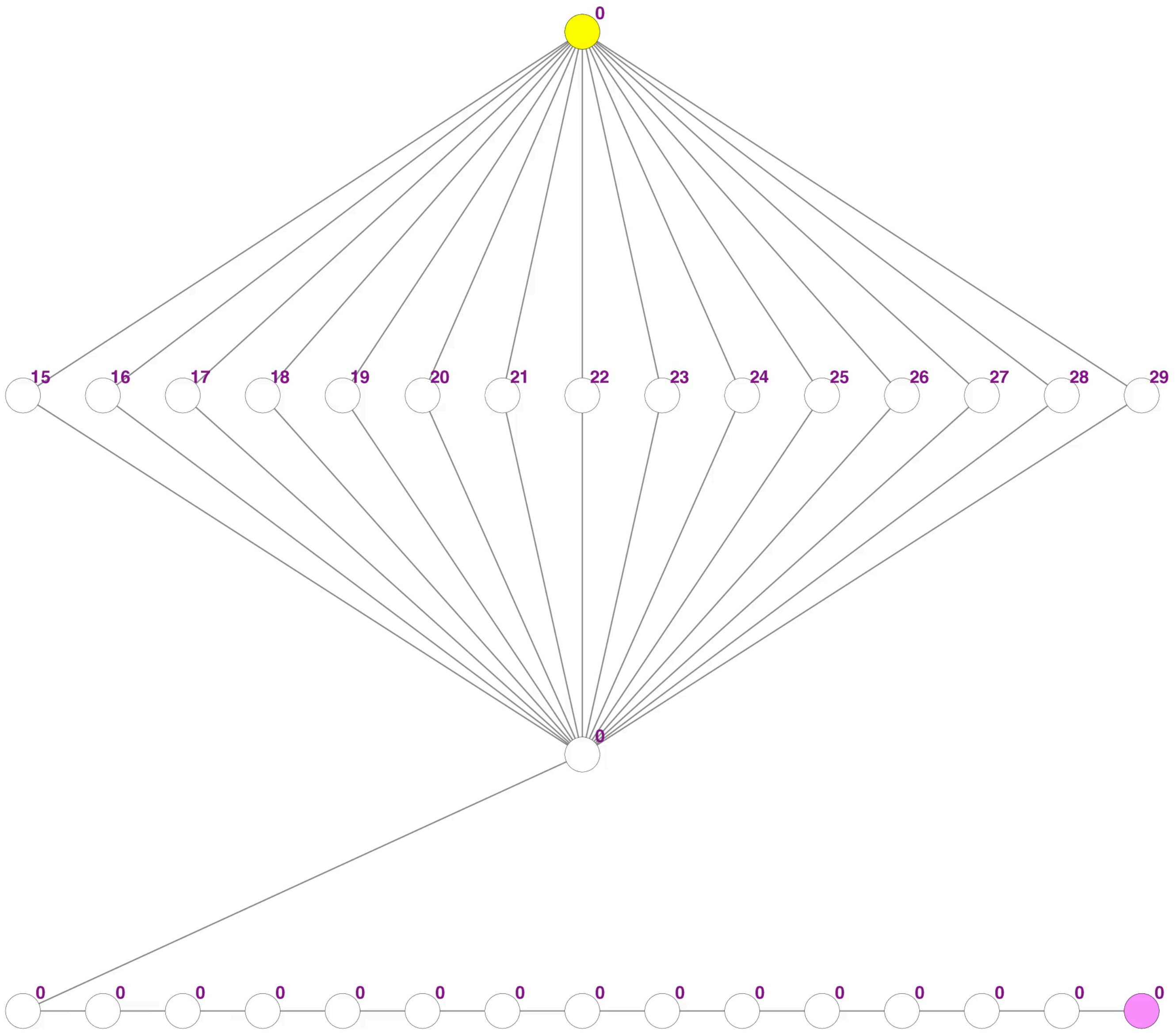


Compressed Differential Heuristic with BPMX





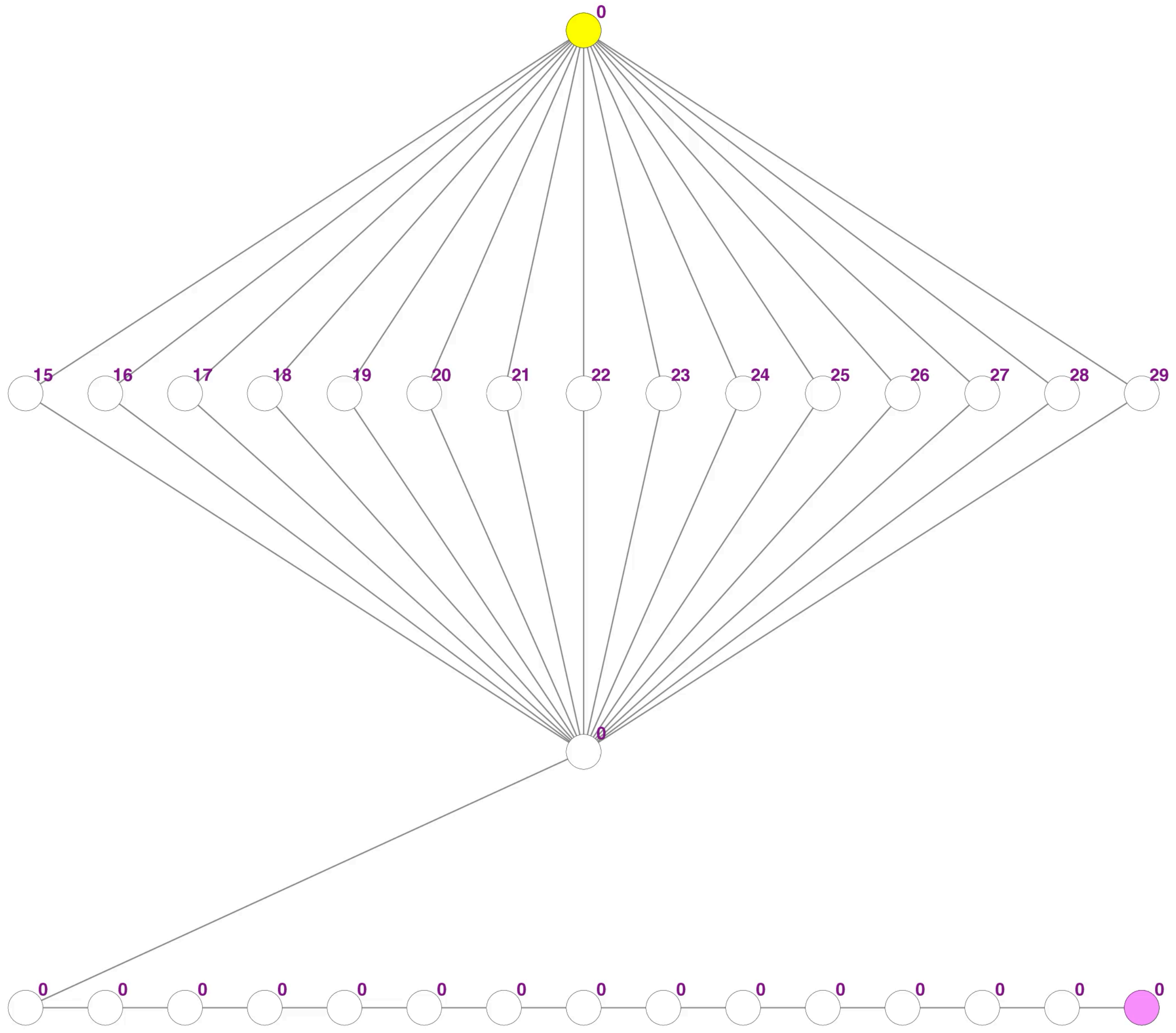
0
N states
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
N states


0
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Previous Work

- Inconsistent Heuristics
    - B/B’ (Martelli 1977; Mero, 1984)
    - Delay (Sturtevant et al, 2008)
    - BPMX (Felner et al, 2011)

IBEX/BGS



Weighted A*

- Weighted A* uses a different f-cost:
  - $f(n) = g(n) + w \cdot h(n)$

- Inflate the heuristic by a factor of *w*
- Results in *w*-optimal solutions
- No re-expansions required

Suboptimal Search

- Large body of algorithms build on Weighted A*
  - Anytime Search
  - Bounded-Cost Search
  - Suboptimal Search

- Many approaches require node re-expansions to prove bounds
  - Many *Focal List* approaches
  - Dynamic Potential Search

Re-expansions

- Re-expansions can have dire consequences
  - Assume N unique nodes are expanded
  - **May require N^2 expansions!**

- General worst case example

Revisiting Suboptimal Search
Jingwei Chen, Nathan R. Sturtevant, William Doyle, Wheeler Ruml
Symposium on Combinatorial Search, **2019**

Demo(s)

http://www.movingai.com/SAS/WAB

Avoiding Re-expansions

- General classes of priority functions that avoid re-expansions can be developed

$$f(n) = \Phi(h(n), g(n))$$

$$\Phi_{WA*}(h, g) = \frac{1}{w}g + h$$

- Φ is an estimate of the optimal solution cost

Avoiding Re-expansions

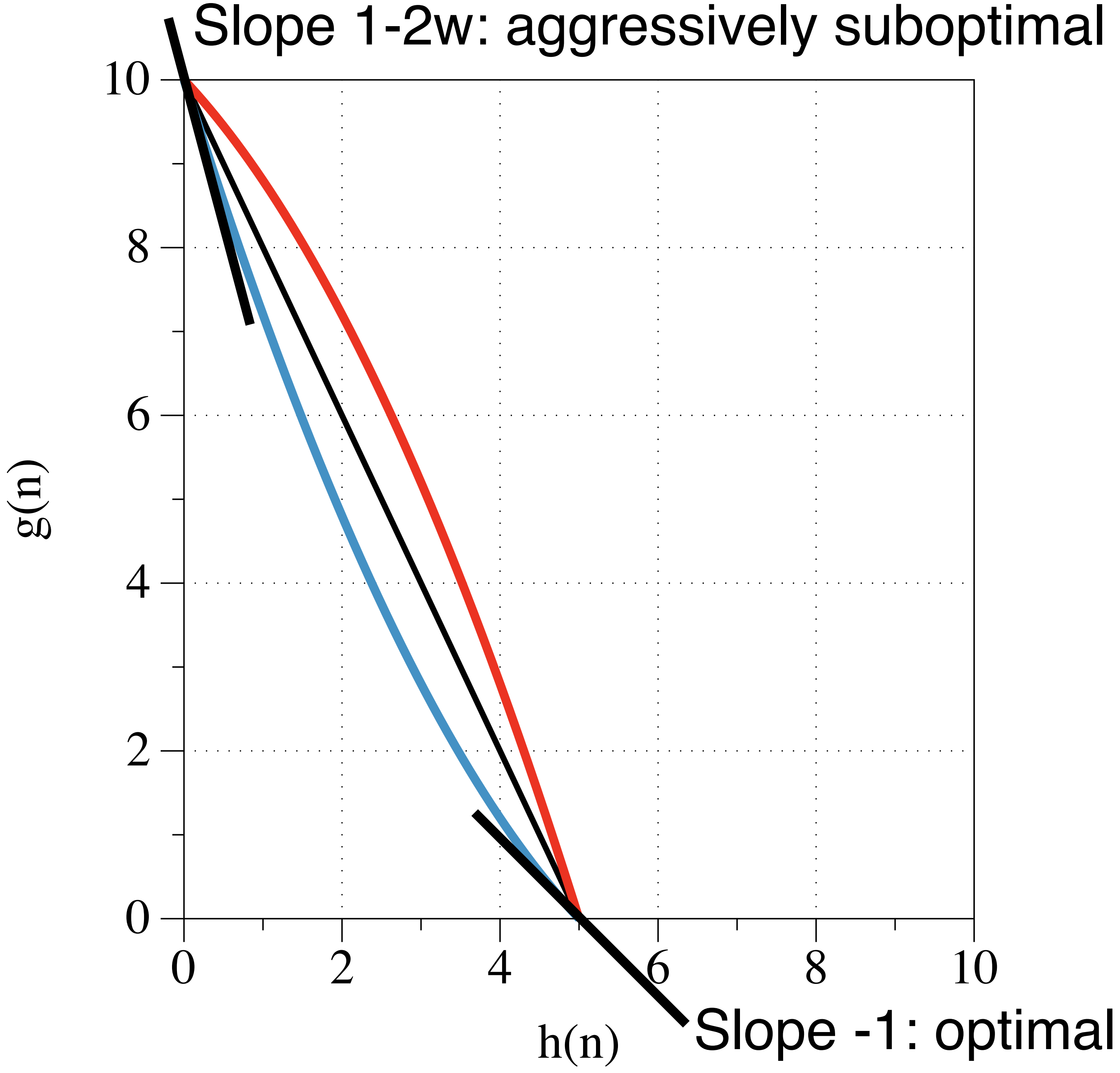
- General classes of priority functions that avoid re-expansions can be developed

$$f(n) = \Phi(h(n), g(n))$$

$$\Phi_{WA*}(h, g) = \frac{1}{w}g + h$$

$$\Phi_{XUP}(h, g) = \frac{1}{2w}[g + h + \sqrt{(g + h)^2 + 4w(w - 1)h^2}]$$

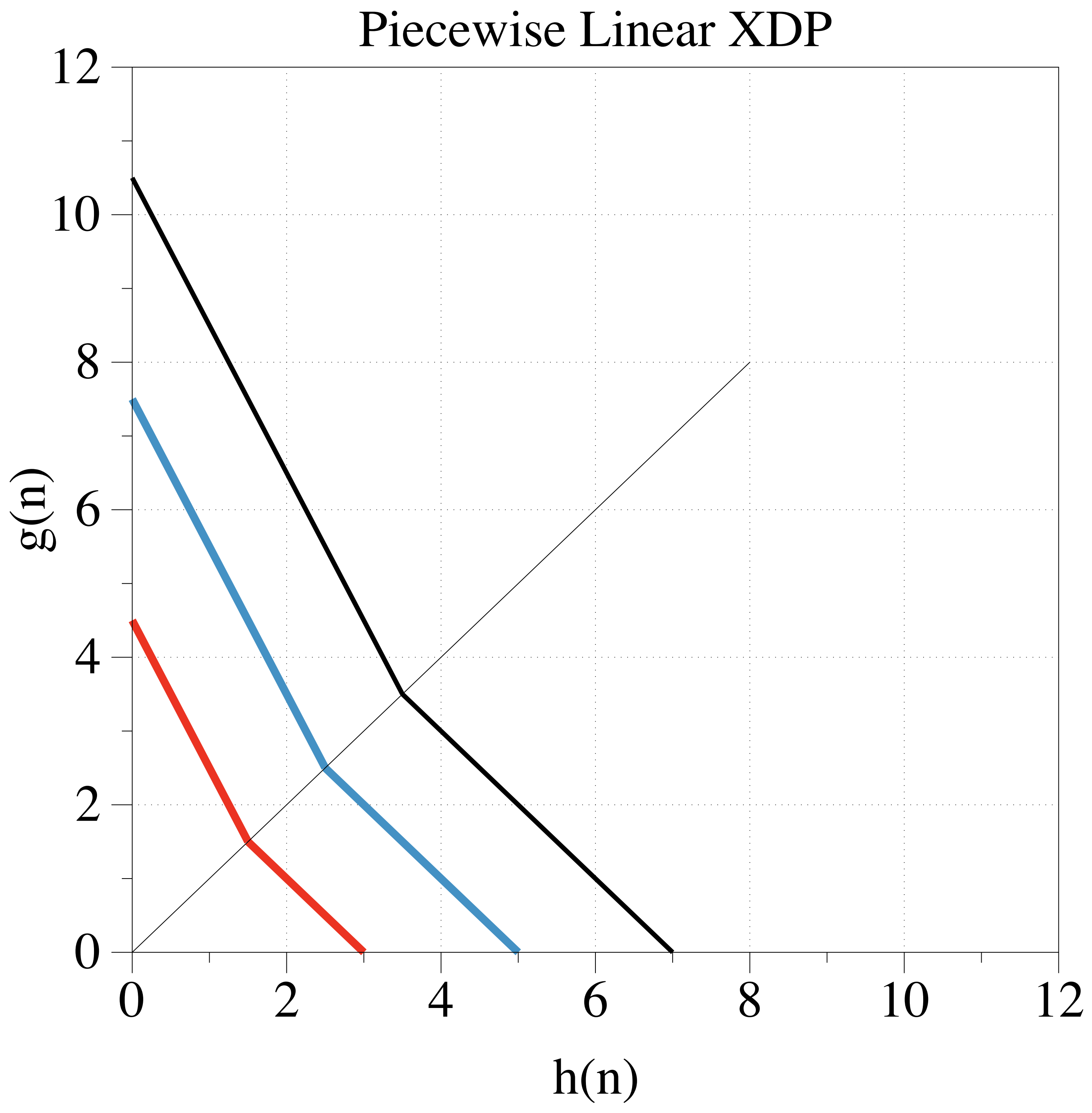
Conditions for Avoiding Node Re-expansions in Bounded Suboptimal Search
Jingwei Chen, Nathan R. Sturtevant
IJCAI, **2019**


Slope 1-2w: aggressively suboptimal
Slope -1: optimal
g(n)
h(n)
0
2
4
6
8
10

Avoiding Re-expansions

- Simpler function:

$$\Phi_{PWXDP} = \begin{cases} h > g : g + h \\ h \le g : \dfrac{(g + (2w - 1)h)}{w} \end{cases}$$


Piecewise Linear XDP
g(n)
h(n)
0
2
4
6
8
10
12
0
2
4
6
8
10
12

Demo(s)

http://www.movingai.com/SAS/SUB

Collapse and Restore

Heuristic Overview

- Part I problem definition
  - Implicit state space
  - Only given heuristic function
- Part II
  - Build new heuristics
  - Heuristics from explicit state space in memory

Exponential Heuristics

UNIVERSITY OF ALBERTA

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

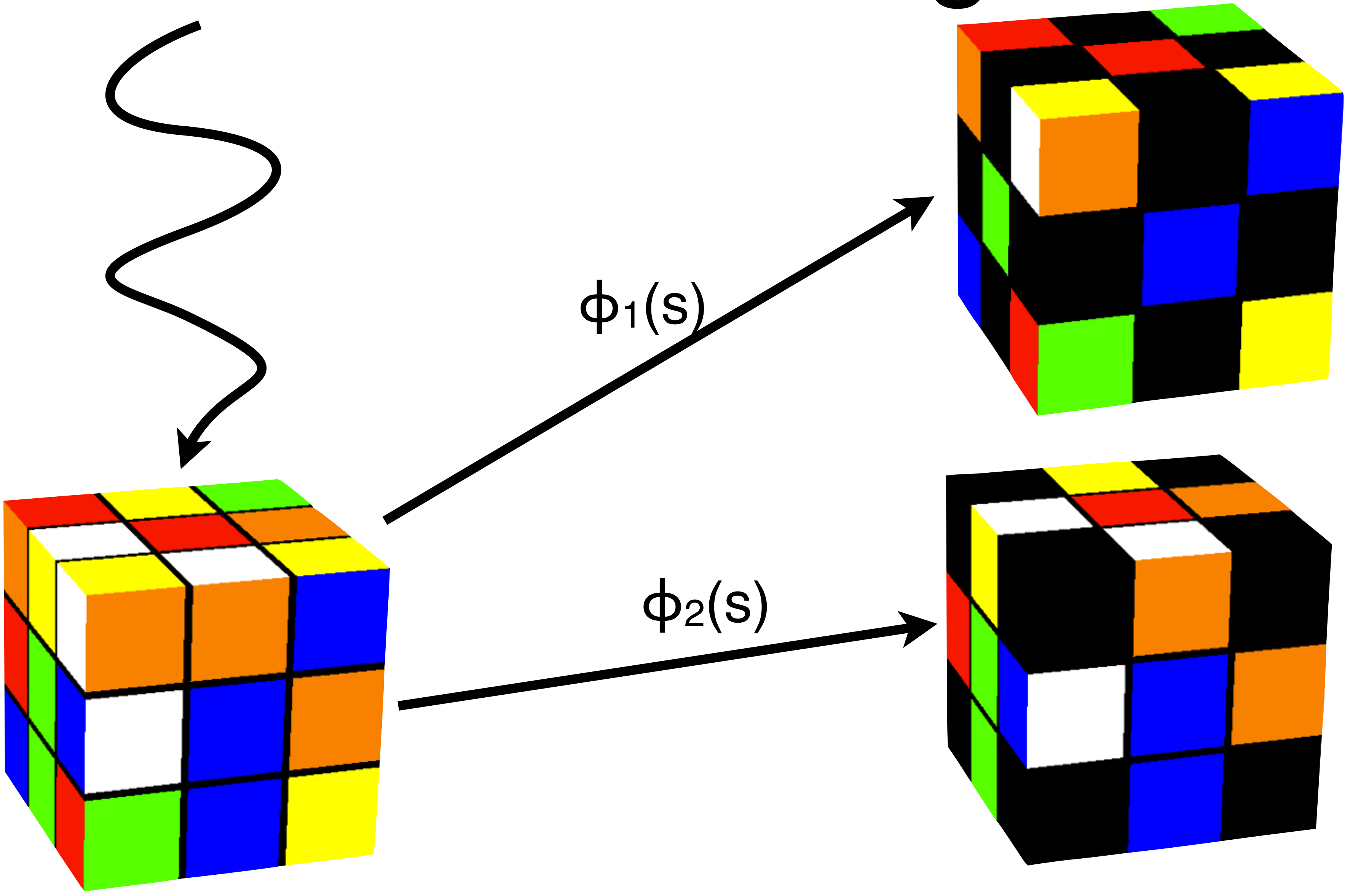
Domain Abstraction

- 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



Demo

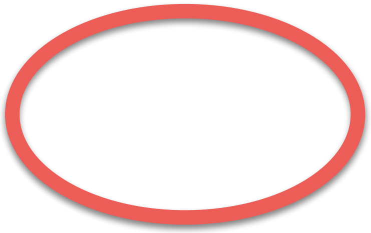
http://www.movingai.com/SAS/PDB

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

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

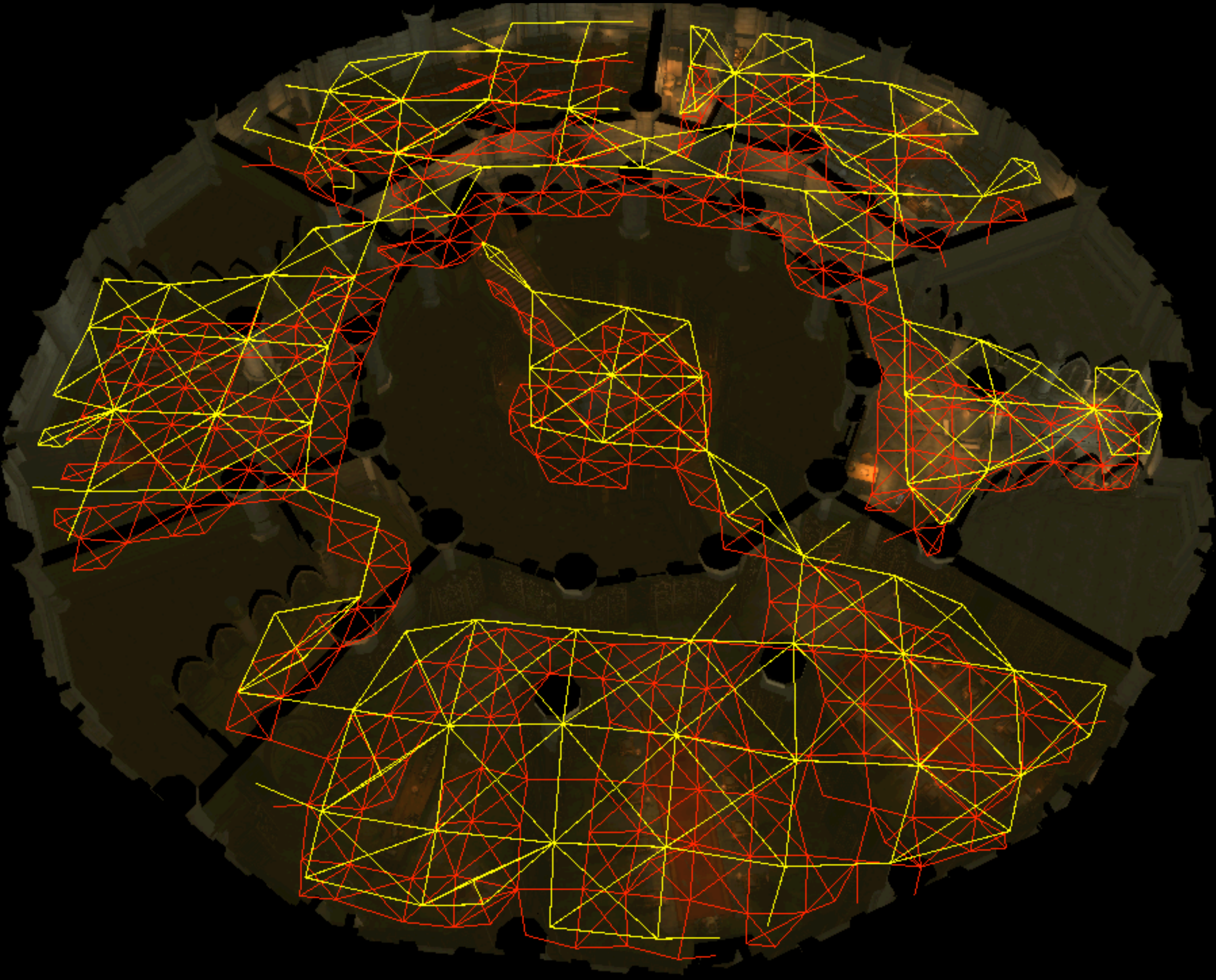


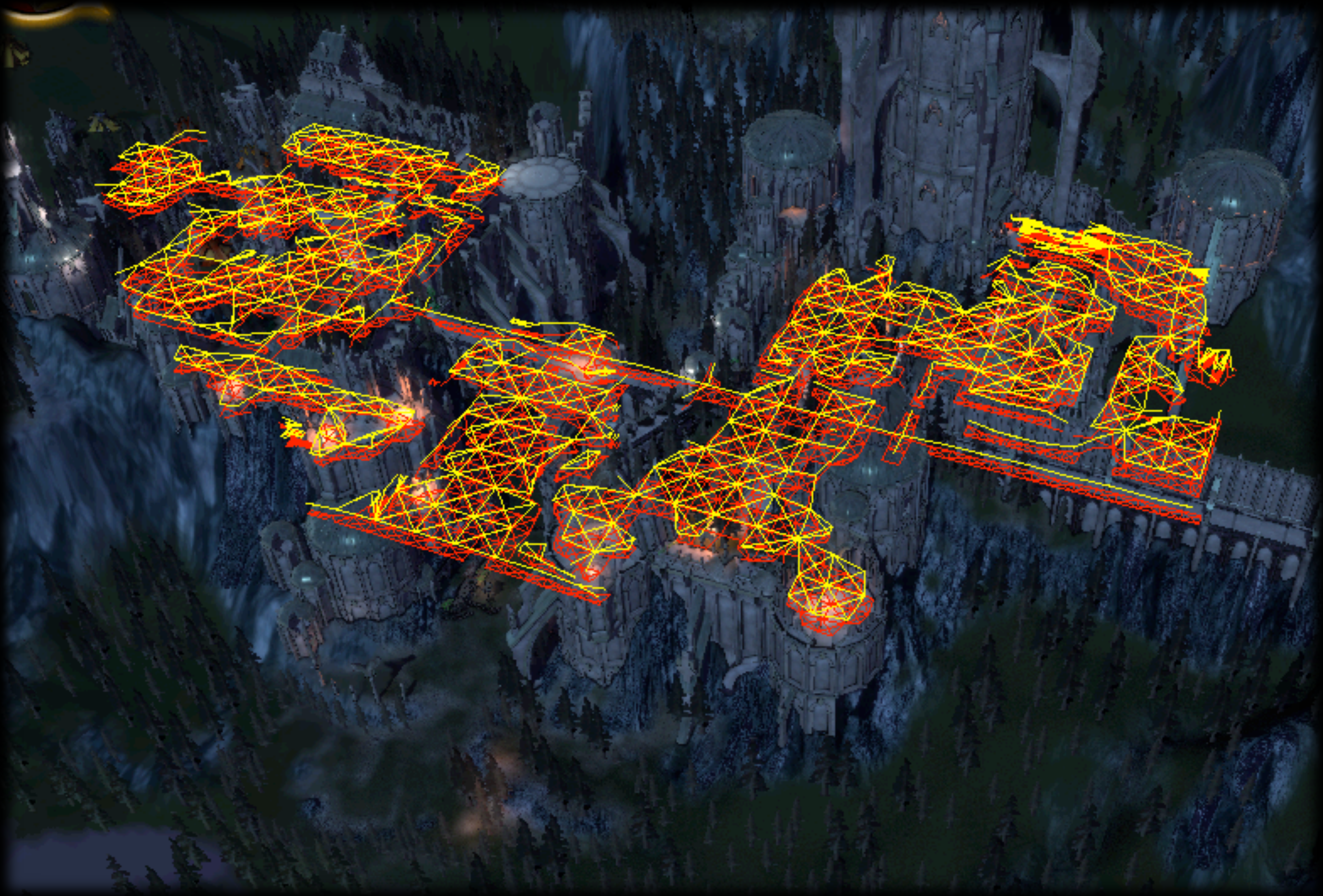
Polynomial Heuristics

UNIVERSITY OF ALBERTA

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

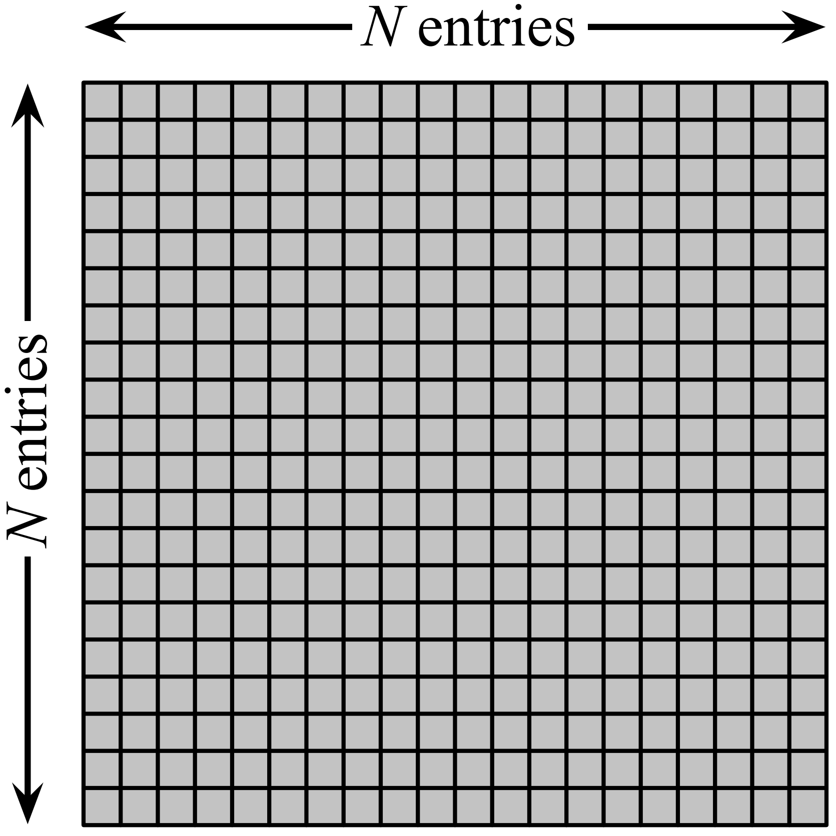




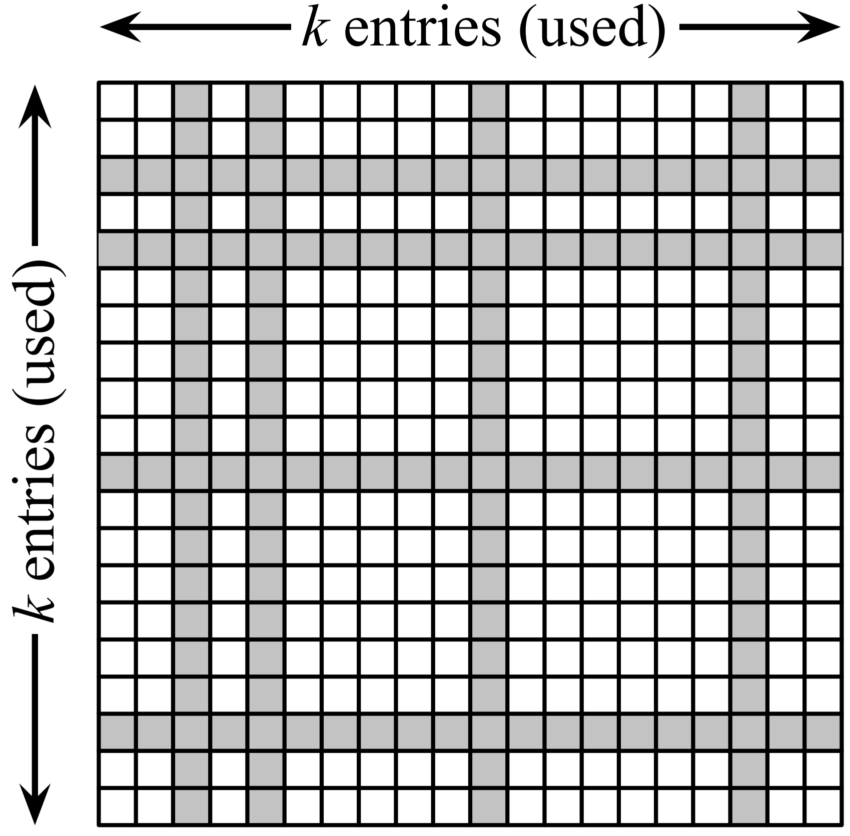
Polynomial State Spaces

- Often explicit:
  - Road networks
  - Game maps
  - Social network
- Can often afford O(n) memory with *n* nodes in graph
- Might be able to compute an oracle

Compressed Oracle



All-Pairs Shortest Path



Differential Heuristic (DH)

Differential Heuristics / Euclidean Embeddings

Heuristic Approaches

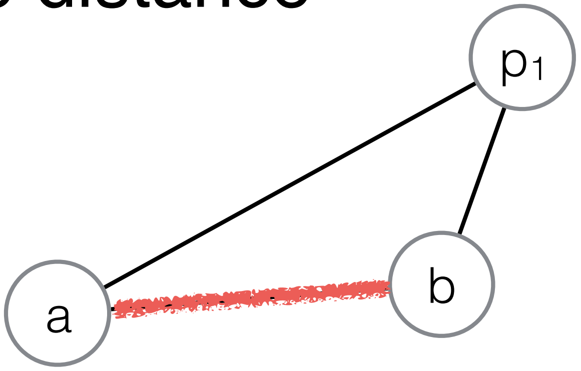
- Different heuristics in polynomial/exponential domains
  - Global Network Positioning (Ng & Zhang, 2001)
  - ALT (Goldberg & Harrelson, 2005)
  - True-Distance Heuristics; **Differential Heuristic** (Sturtevant, et al, 2009)
  - 1-Dimensional Euclidean Heuristic (Rayner, et al, 2011)
  - Compressed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

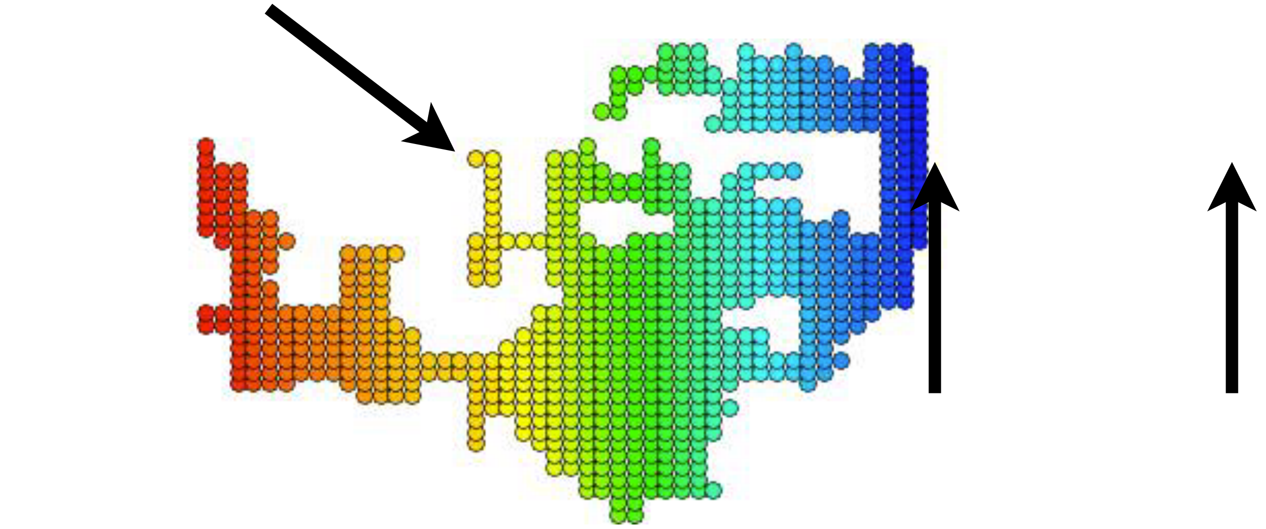
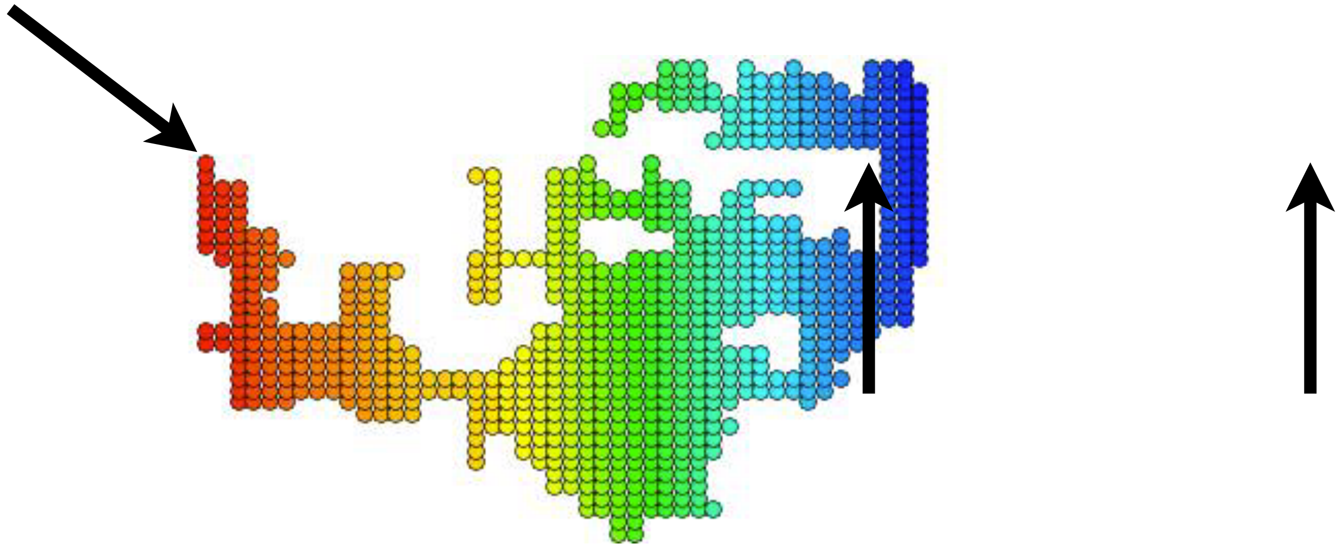
Distance Approximation

- 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?
  - Triangle inequality:

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

  - Key question:
    - Where to place p_i



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

http://www.movingai.com/SAS/DHP

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?

FastMap

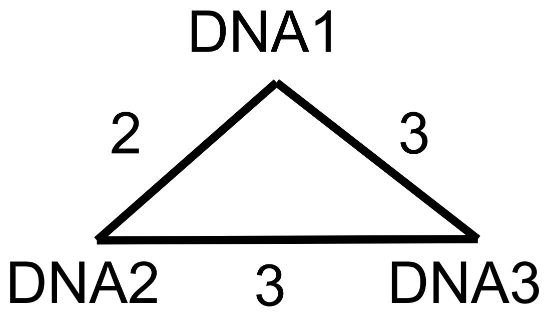
FastMap

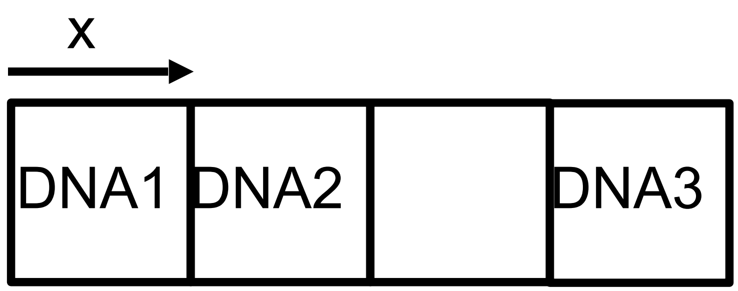
- FastMap is an algorithm from data mining for automatically generating embeddings of abstract objects into a high-dimensional Euclidean space so that their Euclidean distance is very similar to a given distance

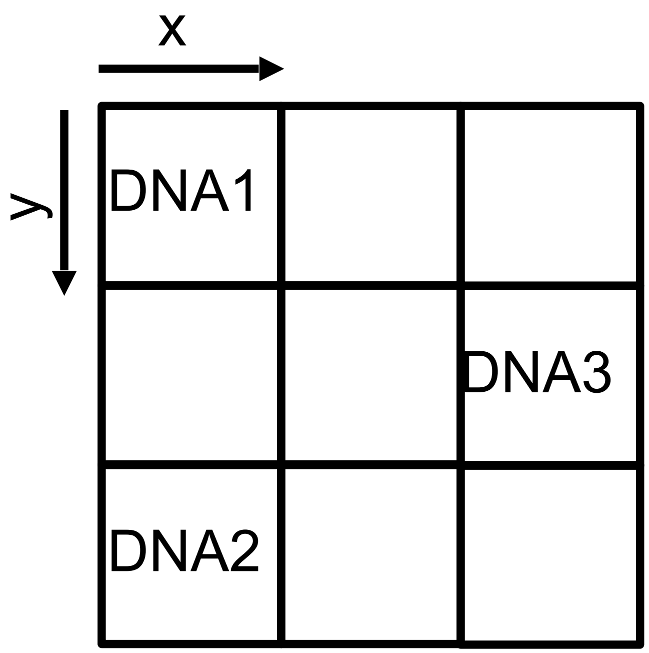
- Faloutsos and Lin; Fastmap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets; Proceedings of the International Conference on Management of Data; 1995

FastMap

- Example: DNA strings (given distance: edit distance)
  - Embed into Euclidean space for visualization or clustering
  - Here: We use the L1 distances



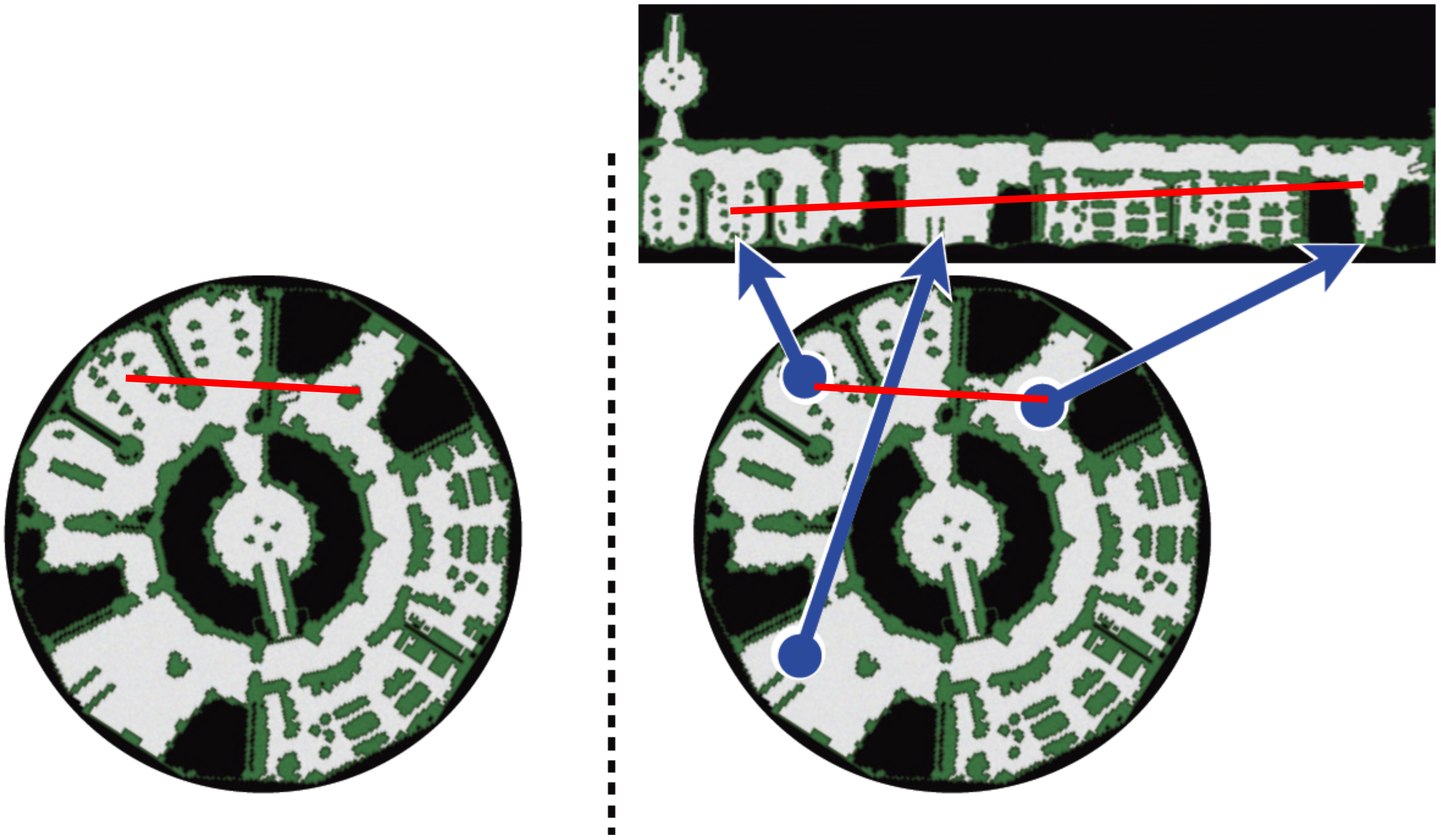




	DNA1	DNA2	DNA3
x=	0	1	3

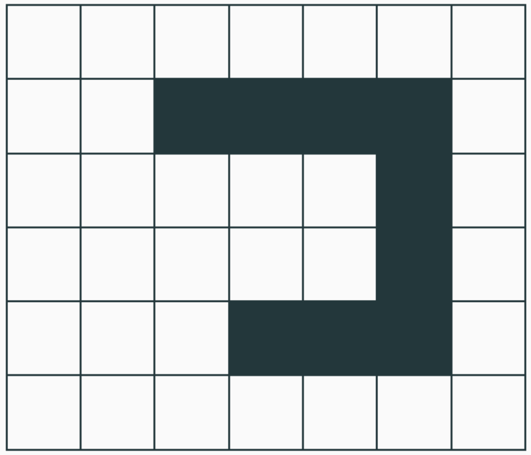
	DNA1	DNA2	DNA3
(x,y)=	(0,0)	(0,2)	(2,1)

FastMap

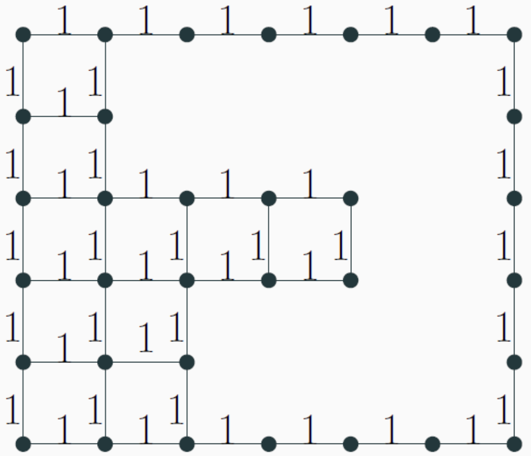


- Rayner, Bowling, and Sturtevant; Euclidean Heuristic Optimization; Proceedings of the AAAI Conference on Artificial Intelligence; 2011

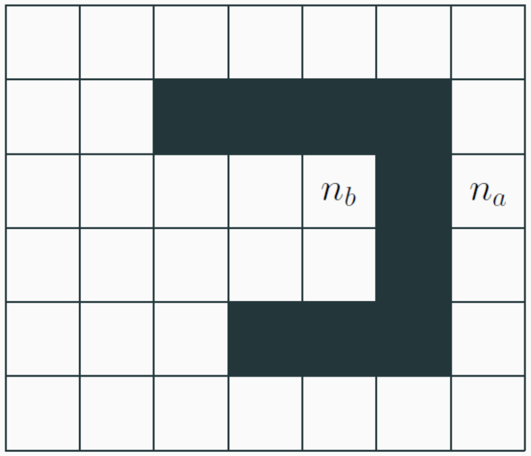
FastMap



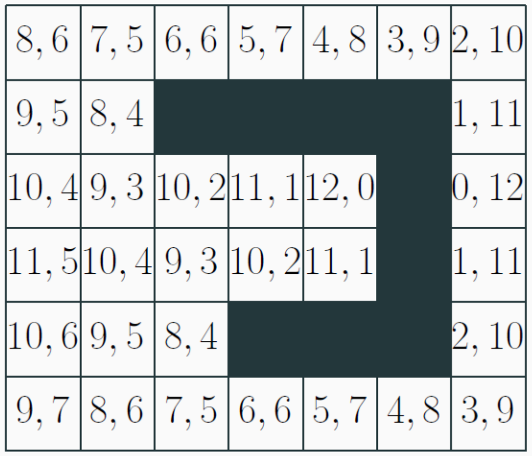
(a)



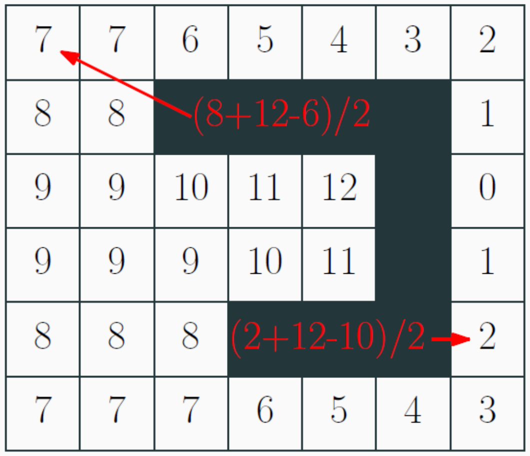
(b)



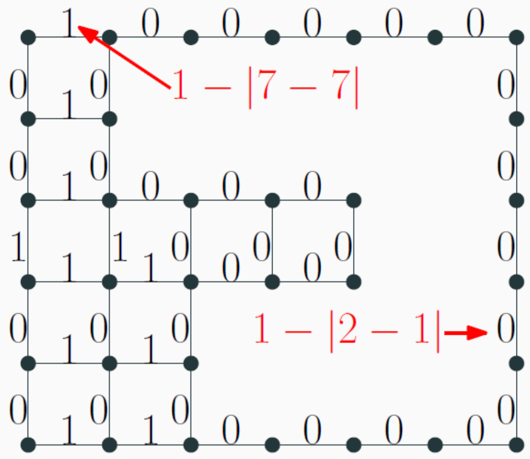
(c)



(d)



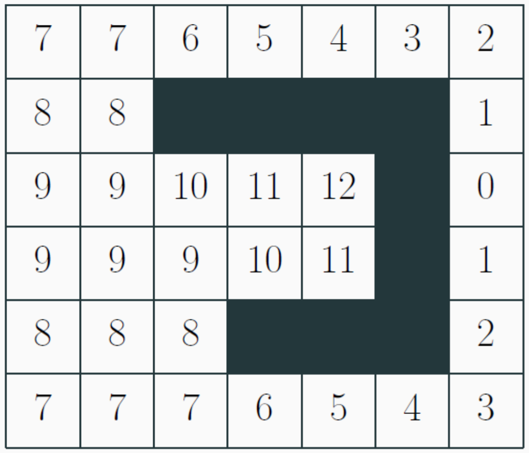
(e)



(f)

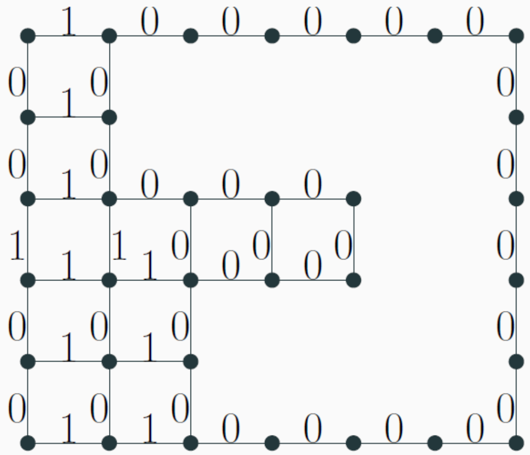
4-neighbor grid

FastMap

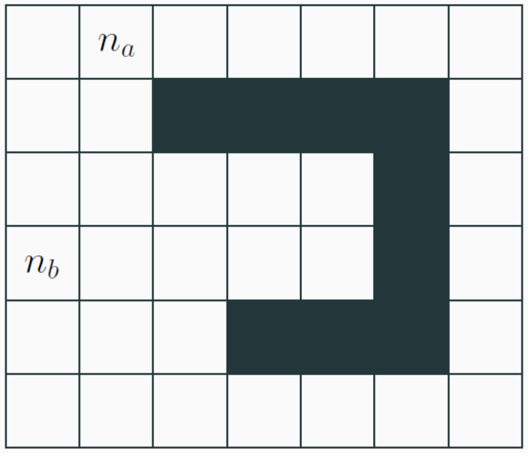


7	7	6	5	4	3	2
8	8	■	■	■	■	1
9	9	10	11	12	■	0
9	9	9	10	11	■	1
8	8	8	■	■	■	2
7	7	7	6	5	4	3

(e)

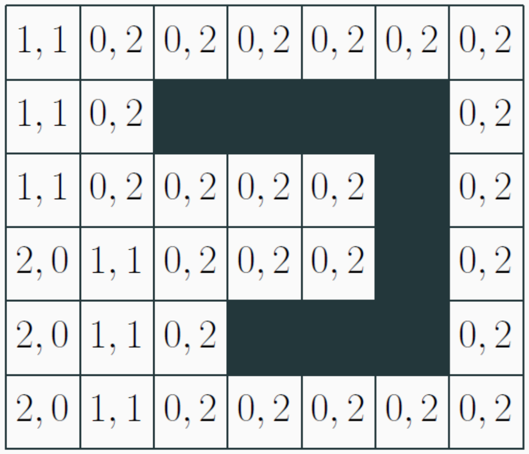


(f)



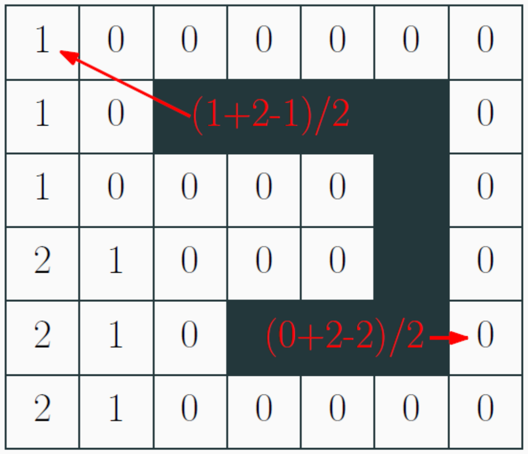
	n_a					
		■	■	■	■	
					■	
n_b					■	
			■	■	■	

(g)



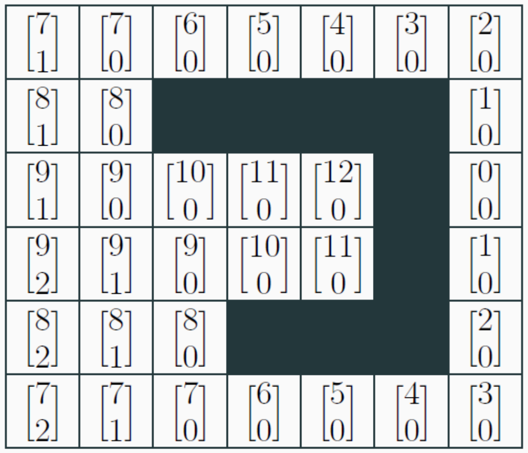
1, 1	0, 2	0, 2	0, 2	0, 2	0, 2	0, 2
1, 1	0, 2	■	■	■	■	0, 2
1, 1	0, 2	0, 2	0, 2	0, 2	■	0, 2
2, 0	1, 1	0, 2	0, 2	0, 2	■	0, 2
2, 0	1, 1	0, 2	■	■	■	0, 2
2, 0	1, 1	0, 2	0, 2	0, 2	0, 2	0, 2

(h)



1	0	0	0	0	0	0
1	0	■	■	■	■	0
1	0	0	0	0	■	0
2	1	0	0	0	■	0
2	1	0	■	■	■	0
2	1	0	0	0	0	0

(i)



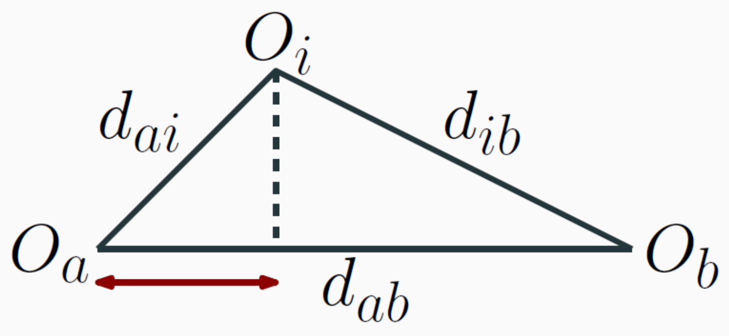
$\begin{bmatrix}7\\1\end{bmatrix}$	$\begin{bmatrix}7\\0\end{bmatrix}$	$\begin{bmatrix}6\\0\end{bmatrix}$	$\begin{bmatrix}5\\0\end{bmatrix}$	$\begin{bmatrix}4\\0\end{bmatrix}$	$\begin{bmatrix}3\\0\end{bmatrix}$	$\begin{bmatrix}2\\0\end{bmatrix}$
$\begin{bmatrix}8\\1\end{bmatrix}$	$\begin{bmatrix}8\\0\end{bmatrix}$	■	■	■	■	$\begin{bmatrix}1\\0\end{bmatrix}$
$\begin{bmatrix}9\\1\end{bmatrix}$	$\begin{bmatrix}9\\0\end{bmatrix}$	$\begin{bmatrix}10\\0\end{bmatrix}$	$\begin{bmatrix}11\\0\end{bmatrix}$	$\begin{bmatrix}12\\0\end{bmatrix}$	■	$\begin{bmatrix}0\\0\end{bmatrix}$
$\begin{bmatrix}9\\2\end{bmatrix}$	$\begin{bmatrix}9\\1\end{bmatrix}$	$\begin{bmatrix}9\\0\end{bmatrix}$	$\begin{bmatrix}10\\0\end{bmatrix}$	$\begin{bmatrix}11\\0\end{bmatrix}$	■	$\begin{bmatrix}1\\0\end{bmatrix}$
$\begin{bmatrix}8\\2\end{bmatrix}$	$\begin{bmatrix}8\\1\end{bmatrix}$	$\begin{bmatrix}8\\0\end{bmatrix}$	■	■	■	$\begin{bmatrix}2\\0\end{bmatrix}$
$\begin{bmatrix}7\\2\end{bmatrix}$	$\begin{bmatrix}7\\1\end{bmatrix}$	$\begin{bmatrix}7\\0\end{bmatrix}$	$\begin{bmatrix}6\\0\end{bmatrix}$	$\begin{bmatrix}5\\0\end{bmatrix}$	$\begin{bmatrix}4\\0\end{bmatrix}$	$\begin{bmatrix}3\\0\end{bmatrix}$

(j)

4-neighbor grid

FastMap

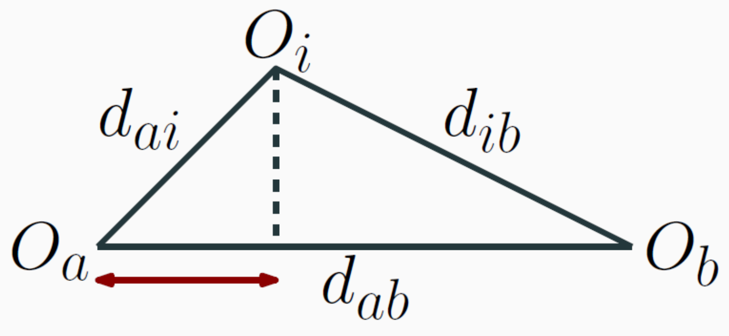
- FastMap uses the L2 distances



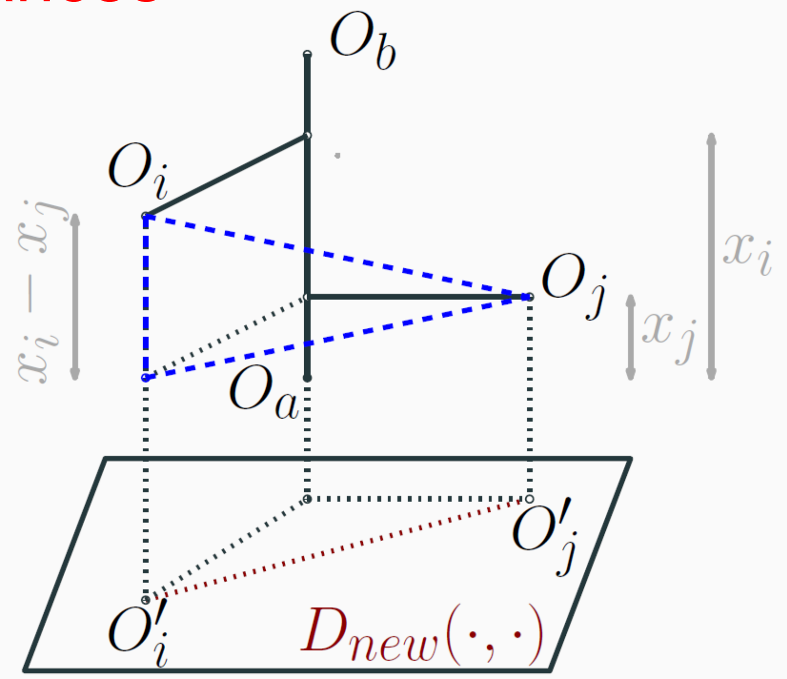
$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{ib}^2}{2d_{ab}}$$

FastMap

- FastMap uses the L2 distances



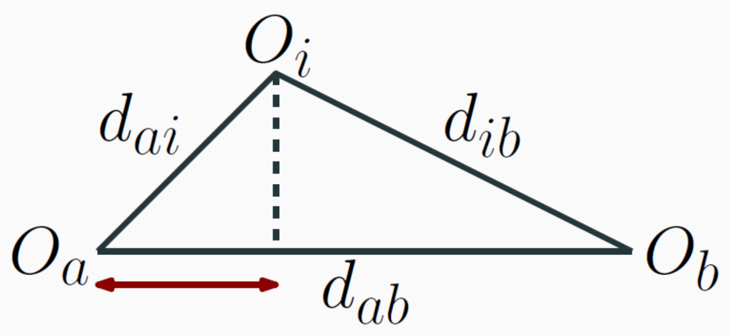
$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{ib}^2}{2d_{ab}}$$



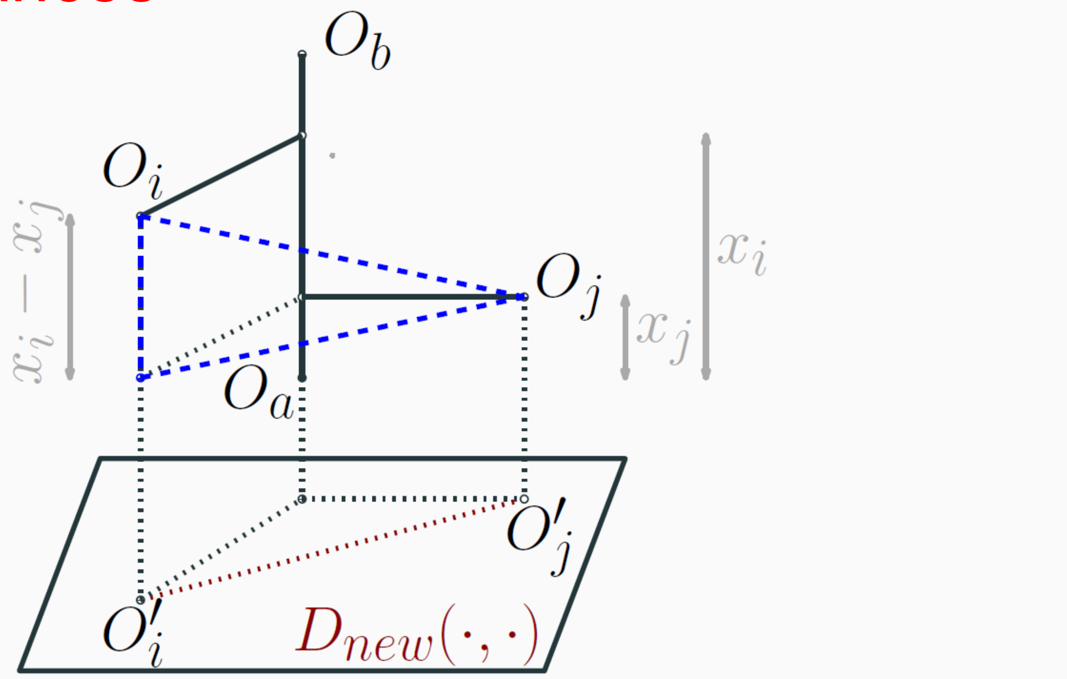
$$D_{new}(O'_i, O'_j)^2 = D(O_i, O_j)^2 - (x_i - x_j)^2$$

FastMap

- FastMap uses the L2 distances



$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{ib}^2}{2d_{ab}}$$

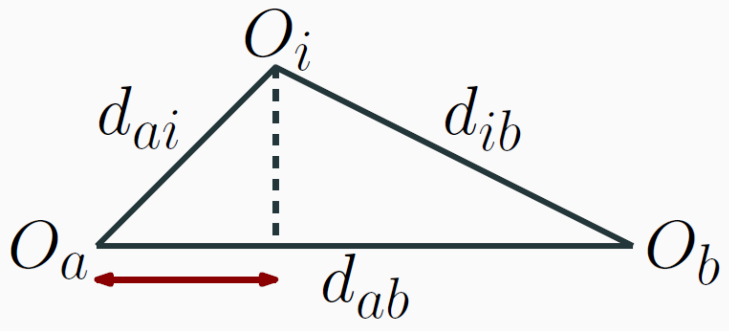


$$D_{new}(O'_i, O'_j)^2 = D(O_i, O_j)^2 - (x_i - x_j)^2$$

- The h-values are not necessarily consistent.

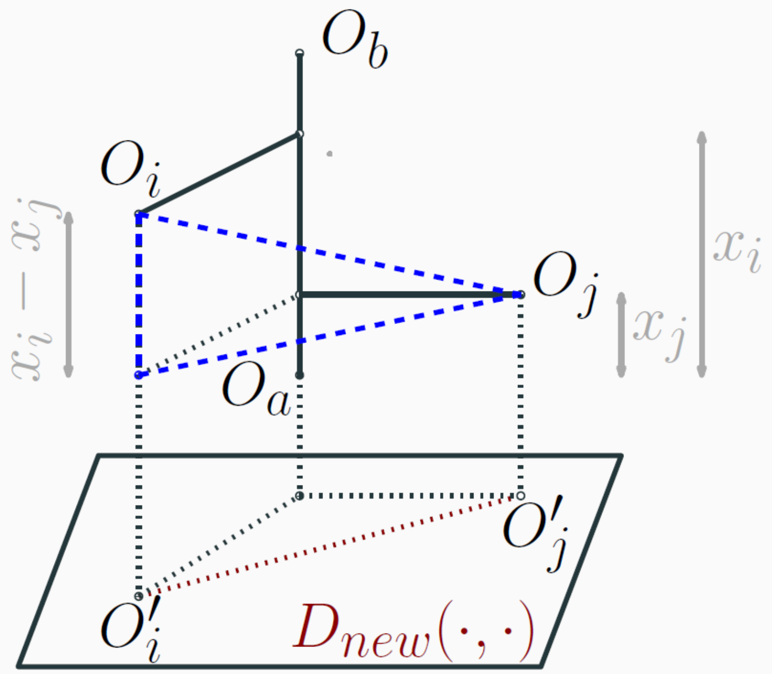
FastMap

- Change FastMap to use the L1 distances



$$\cancel{x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{ib}^2}{2 d_{ab}}}$$

$$x_i = \frac{d_{ai} + d_{ab} - d_{ib}}{2}$$



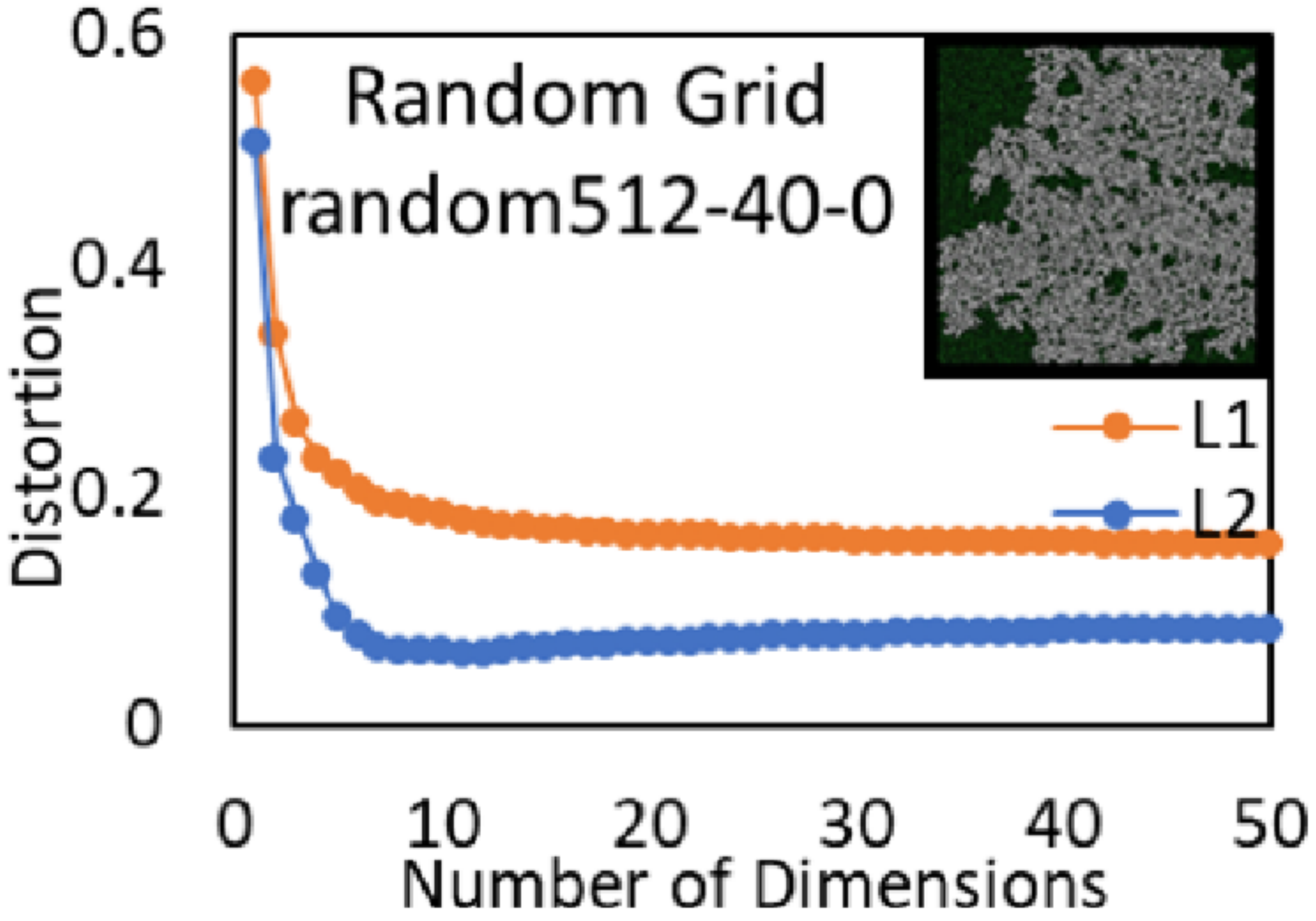
$$\cancel{D_{new}(O'_i, O'_j)^2 = D(O_i, O_j)^2 - (x_i - x_j)^2}$$

$$D_{new}(O'_i, O'_j) = D(O_i, O_j) - (x_i - x_j)$$

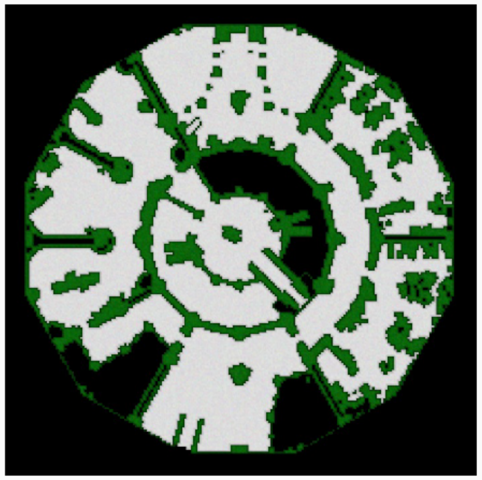
- The h-values are consistent and thus also admissible.

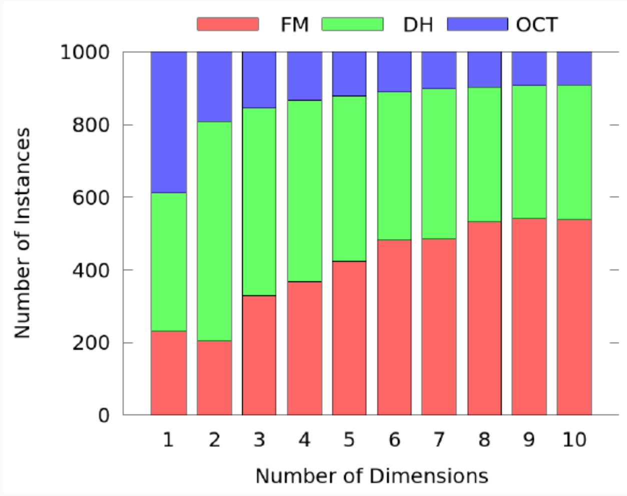
FastMap

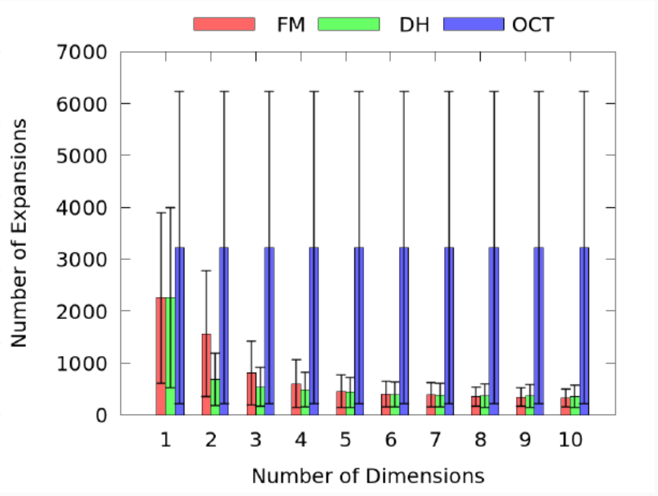
Normalized Root Mean Square Deviation



FastMap





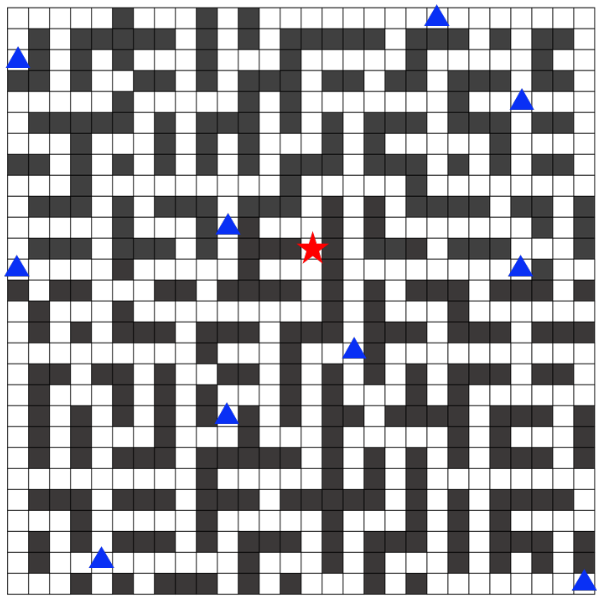


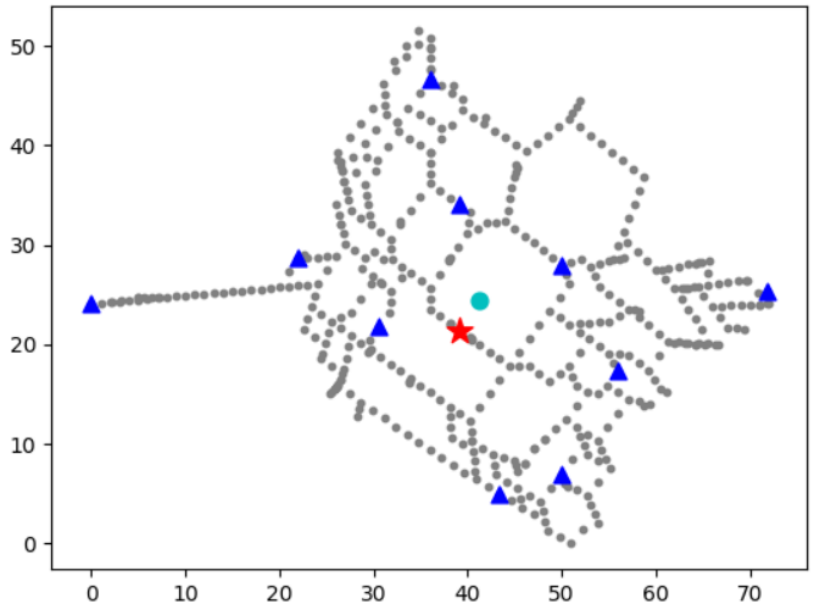
FM – FastMap, DH – Differential heuristic [Sturtevant et al., 2009], OCT – Octile distance.

- Four dimensions are often sufficient.
- The preprocessing time is O(|E| + |V| log |V|) - often under one minute.
- Cohen, Uras, Jahangiri, Arunasalam, Koenig, and Kumar; The FastMap Algorithm for Shortest Path Computations; Proceedings of the International Joint Conference on Artificial Intelligence;

FastMap

- The meeting location problem

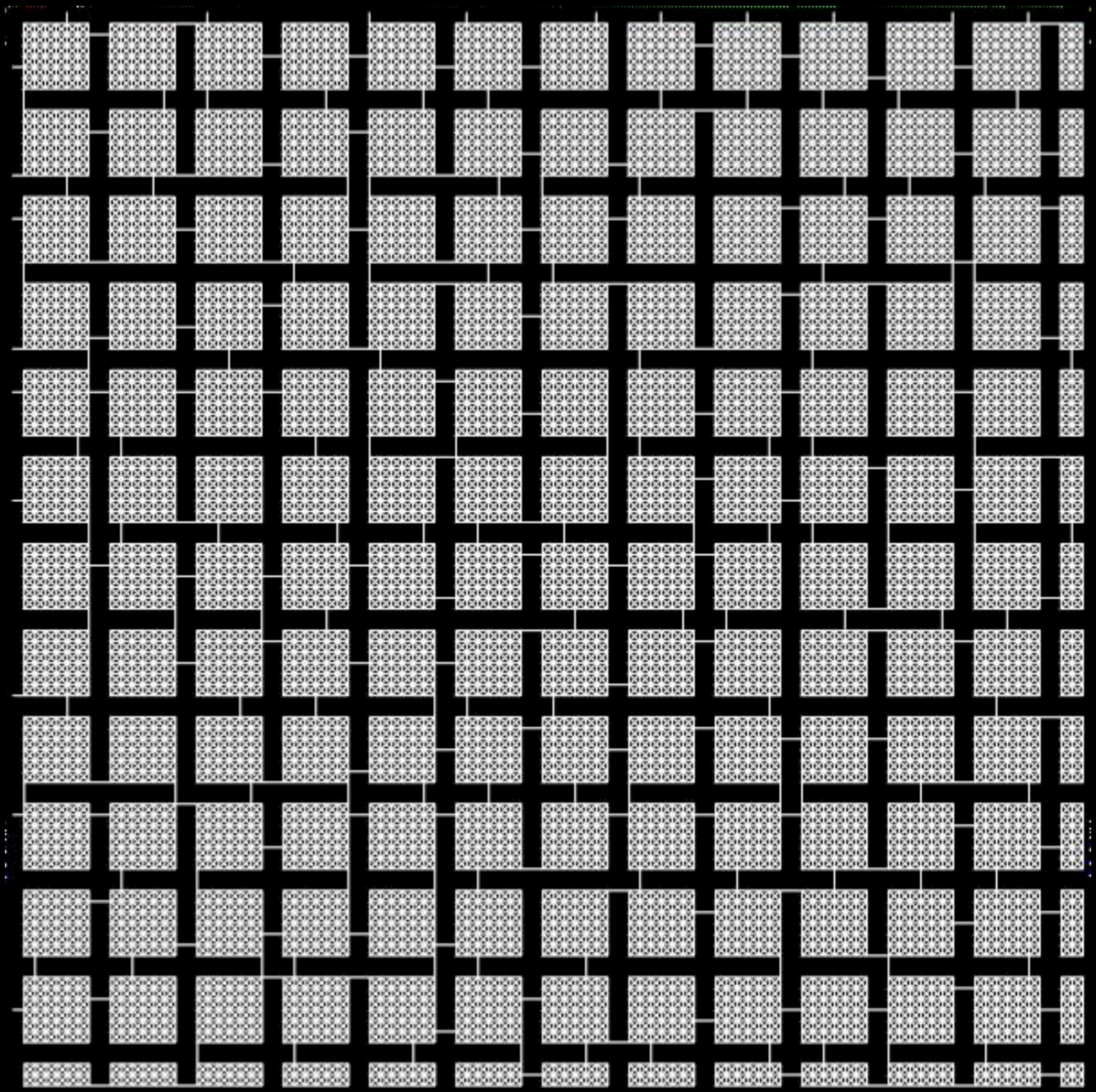


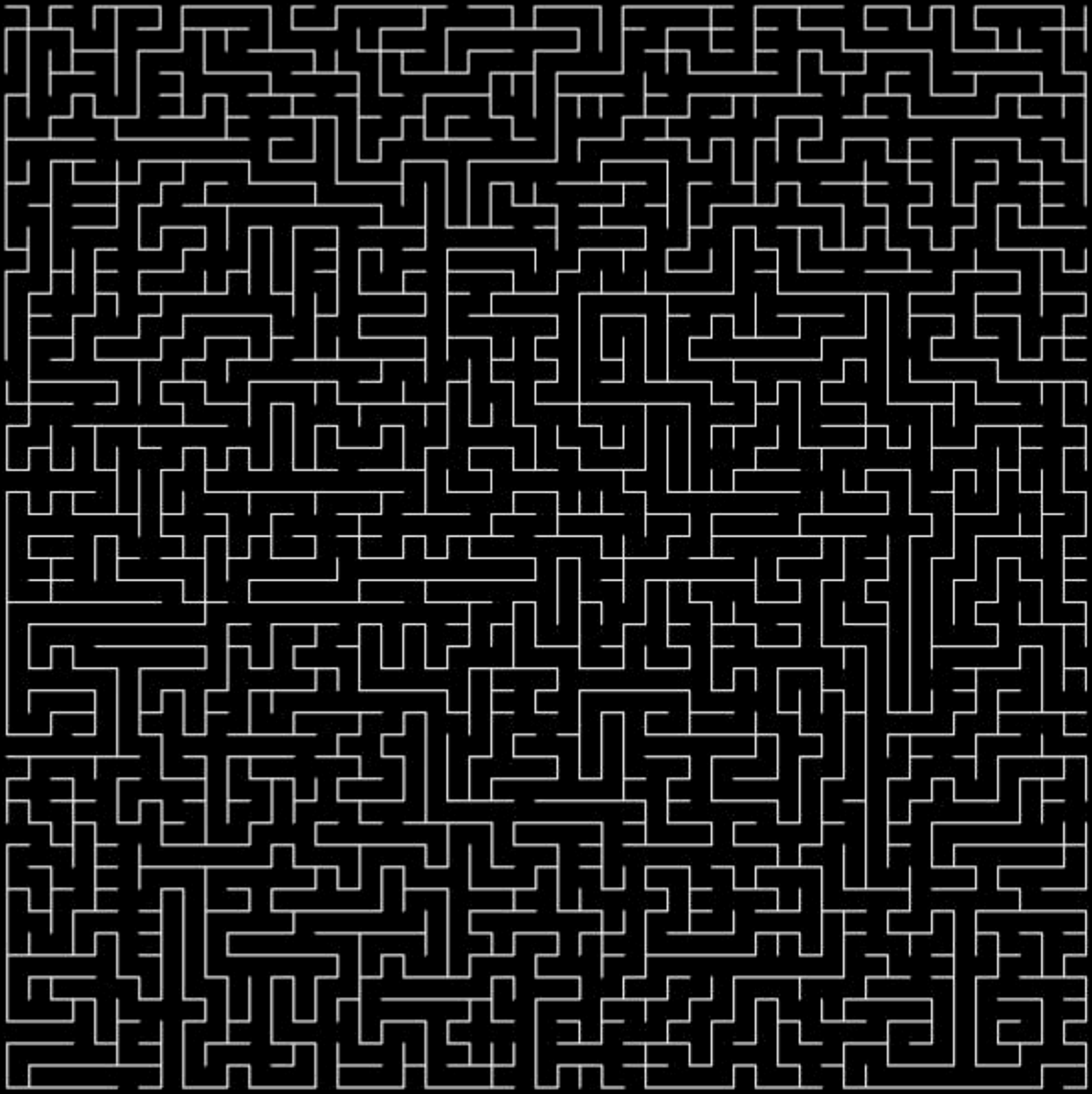


▲ Start locations ★ Meeting location ● Median point in the Euclidean space

Demo

http://www.movingai.com/SAS/FMP





Part III
Preprocessing & Constraints

UNIVERSITY OF ALBERTA

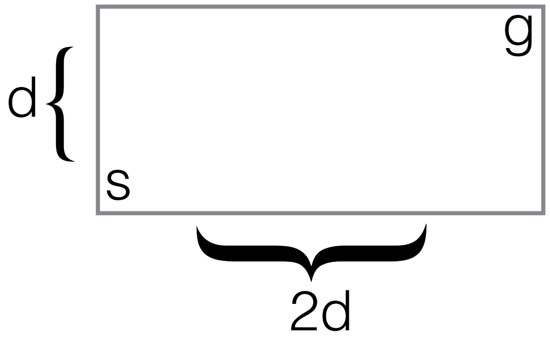
Overview

- Part I problem definition
  - Implicit state space
  - Only given heuristic function
- Part III
  - More than just heuristics

Canonical Ordering
Jump Point Search

How many duplicates?

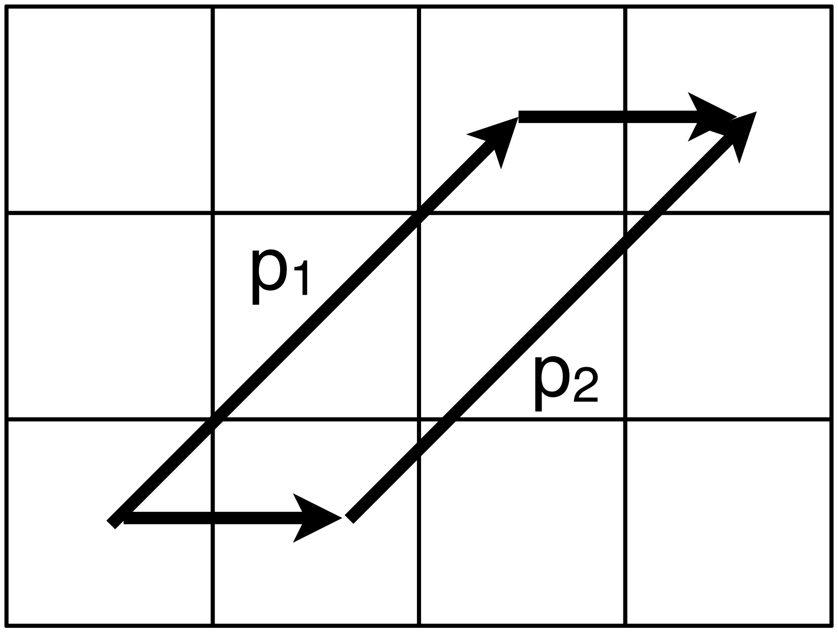
- How many unique paths between two states in an octile map?
  - Assume a 2d x d rectangle with start/goal in the corners



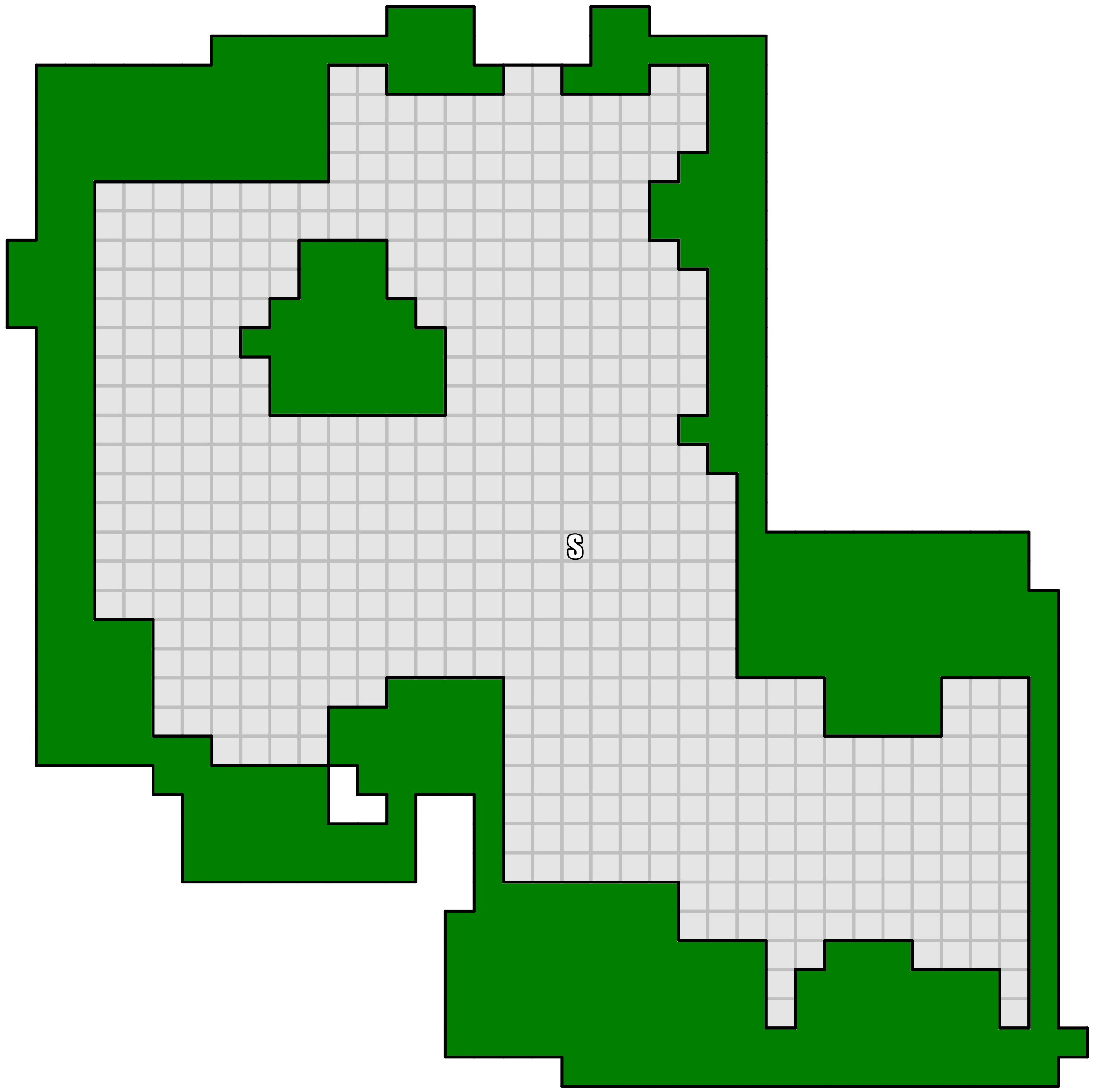
$$\binom{2d}{d} = \frac{(2d)!}{2(d!)}$$

Canonical Ordering of Paths

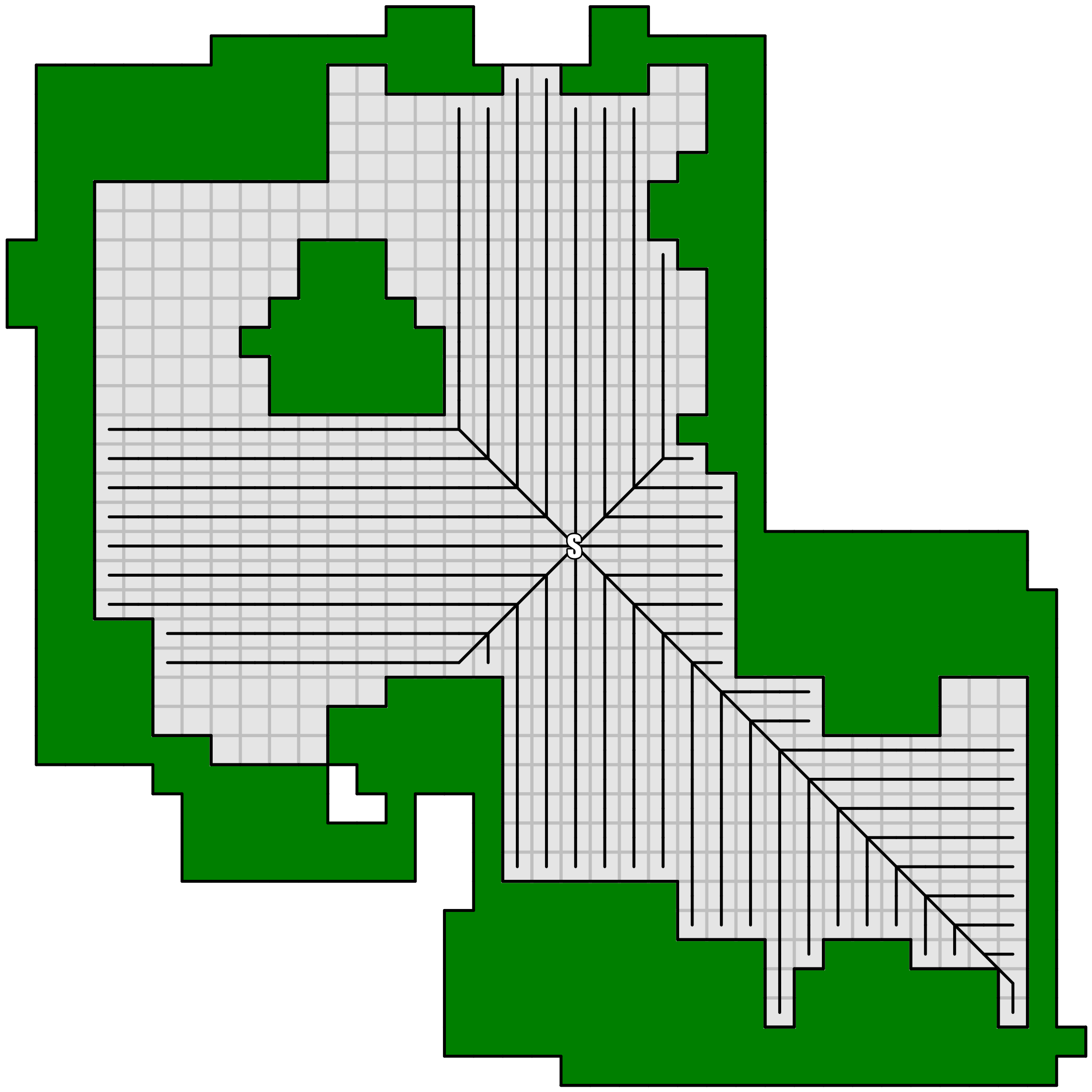
- Order all optimal paths:
  - Path p_1 is preferred over path p_2 if
    - p_1 has diagonal actions prior to cardinal actions



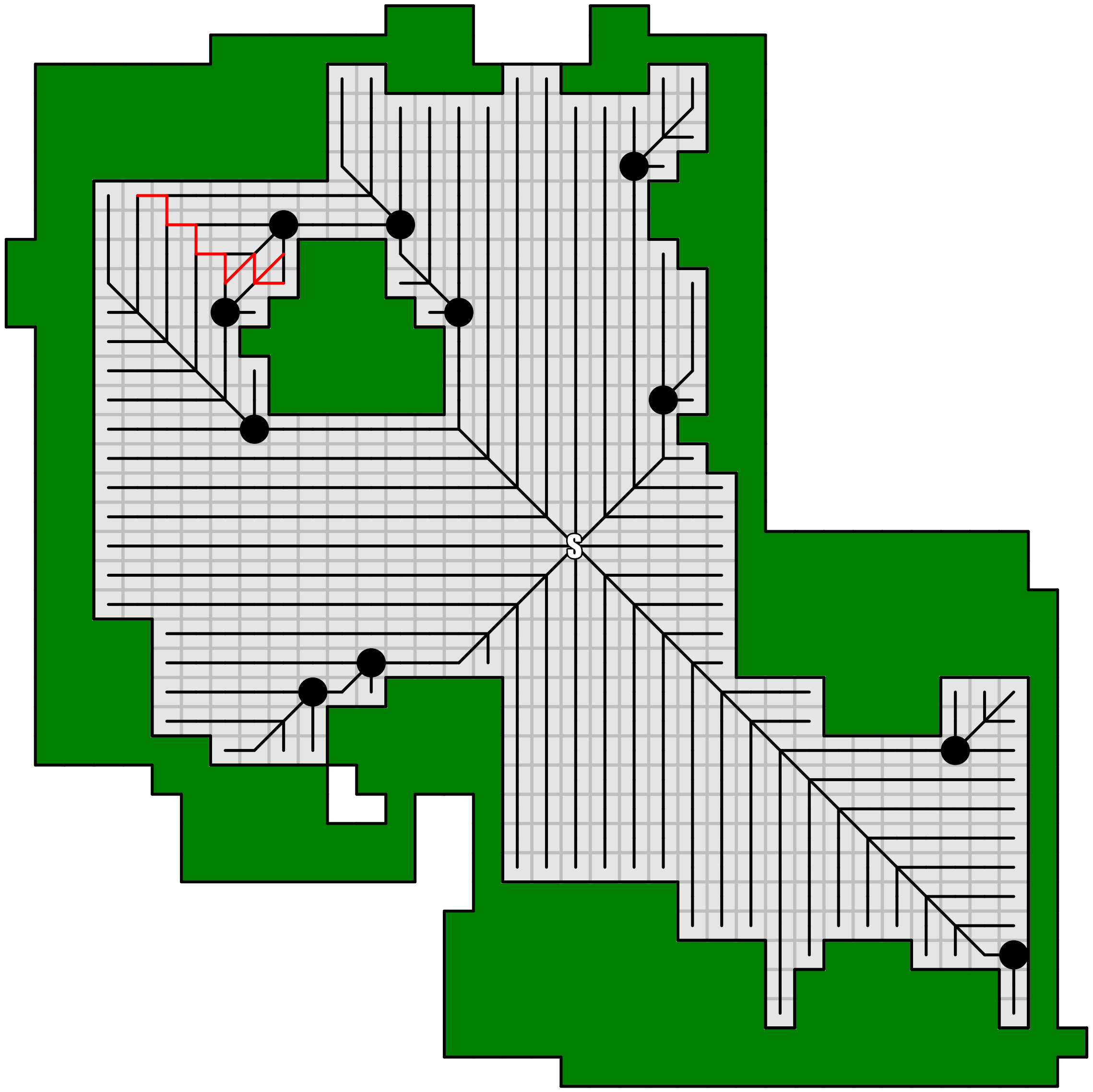
Basic Map



Simple Canonical Ordering



Canonical Ordering



UNIVERSITY OF ALBERTA

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
  - Could use other policies

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
  - Could use other policies

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

http://www.movingai.com/SAS/JPS

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?

Distance Constraints (Reach, Bounding Boxes)

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

UNIVERSITY OF ALBERTA

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)

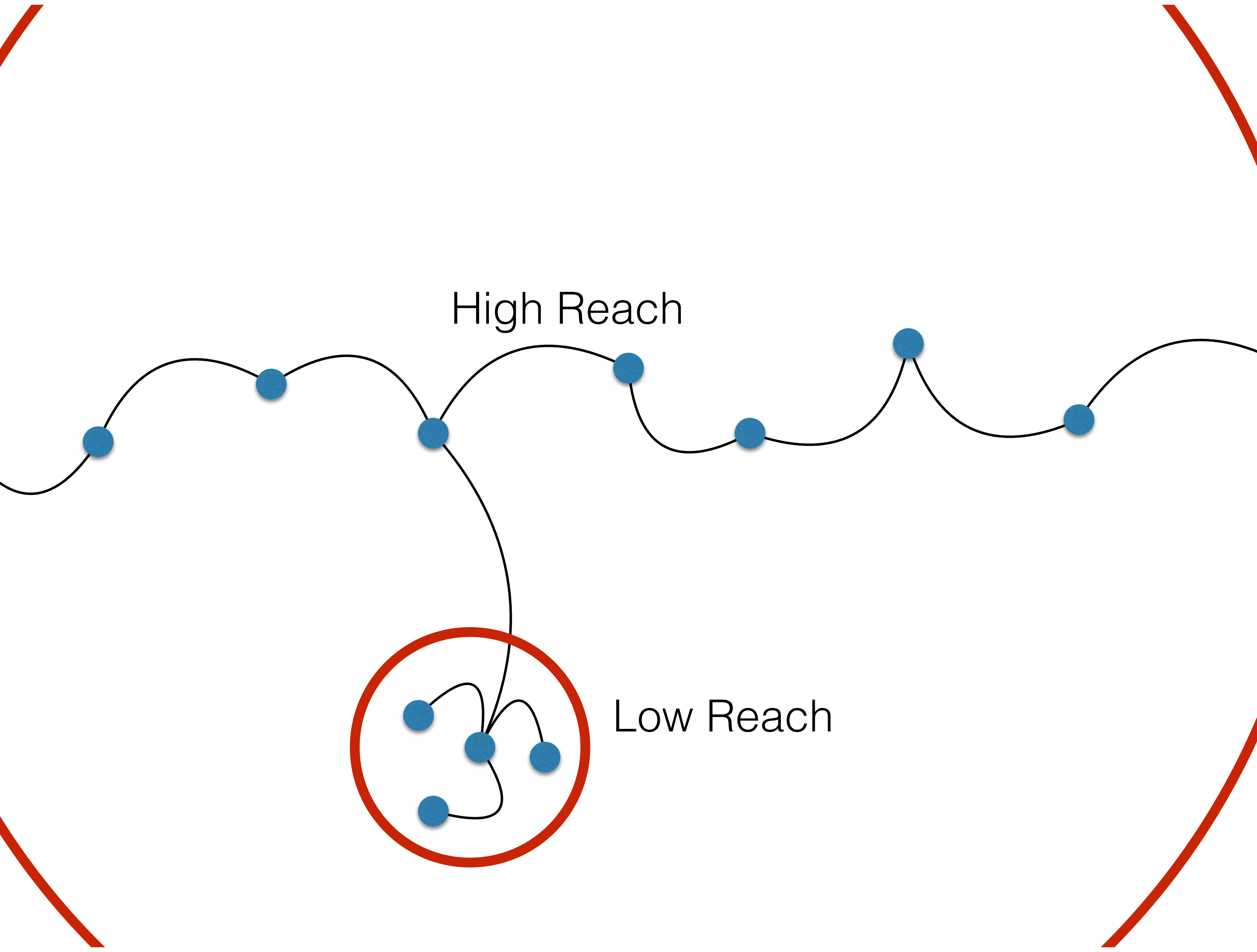
UNIVERSITY OF ALBERTA

(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)


High Reach
Low Reach

Demo

http://www.movingai.com/SAS/RCH

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

http://www.movingai.com/SAS/BBX

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?