



IJCAI-18 Tutorial: Recent Directions in Heuristic Search

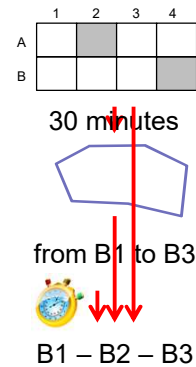
Sven Koenig
University of Southern California (USA)
idm-lab.org
skoenig@usc.edu

Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - H-Values
 - Search Algorithm

Preprocessing-Based Search

- The Grid-Based Path Planning Competition (GPPC) since 2012 (movingai.com/GPPC).
- Offline (= before search)
 - Map is given (without start and goal locations)
 - Preprocessing is performed for 30 minutes
- Online (= during the search)
 - Start and goal locations are given
 - Path-planning problem is solved
- Main evaluation criteria
 - How close to optimal is the path?
 - How fast is the path found?
 - How much memory is used?

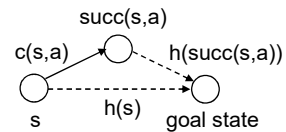


Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - **H-Values**
 - Pattern databases
 - Differential heuristics
 - FastMap
 - Search algorithm

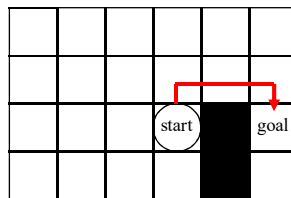
H-Values

- A* uses user-supplied h-values to focus its search
- The h-values approximate the goal distances
- We want the h-values to be consistent!
- The h-values $h(s)$ are consistent if they satisfy the triangle inequality:
 $h(s) = 0$ if s is the goal state
 $h(s) \leq c(s,a) + h(\text{succ}(s,a))$ otherwise
- Consistent h-values are admissible
- The h-values $h(s)$ are admissible if they do not overestimate the goal distances

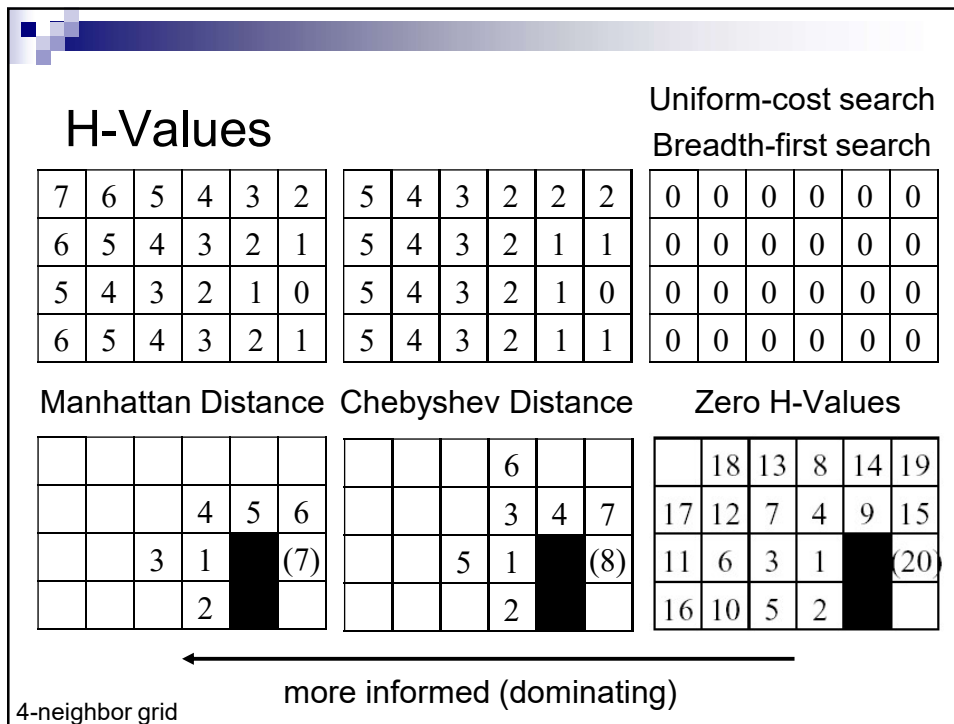


H-Values

- Search problem with uniform cost



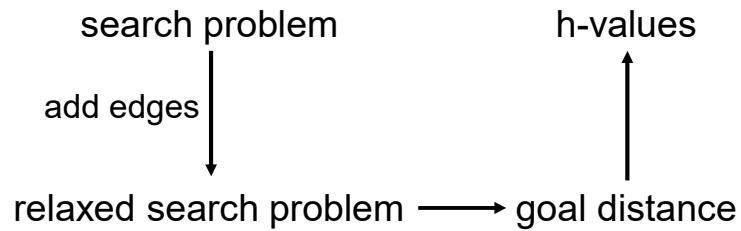
4-neighbor grid



H-Values without Preprocessing

- Obtain a new search problem by relaxing constraints of the original search problem, which can add edges (actions) to the state space
- Use the goal distance of a state for the new search problem as the h-value of the state for the original search problem
- Typically, this is done in a way so that the goal distances for the new search problem can be computed without search
- The resulting h-values are consistent and thus also admissible

H-Values without Preprocessing



H-Values without Preprocessing

- Manhattan-distance heuristic (7 for the example below)

- $h(\text{tile configuration}) = \text{the sum of the x- and y-displacements of each tile from its correct place}$

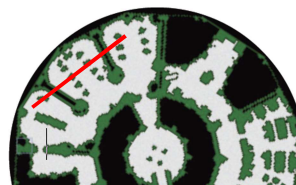
1	3	2
5	6	
7	8	4

1	2	3
4	5	6
7	8	

current configuration goal configuration

- Straight-line heuristic

- $h(\text{location}) = \text{the straight-line distance (ignoring obstacles) from the location to the goal location}$



Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - H-Values
 - Pattern databases (assumes goal is known)
 - Differential heuristics
 - FastMap
 - Search algorithm

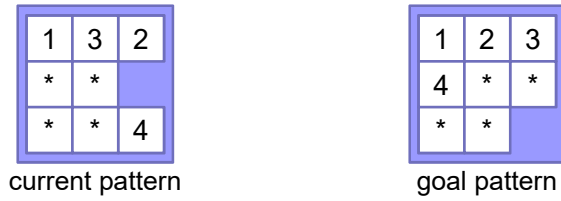
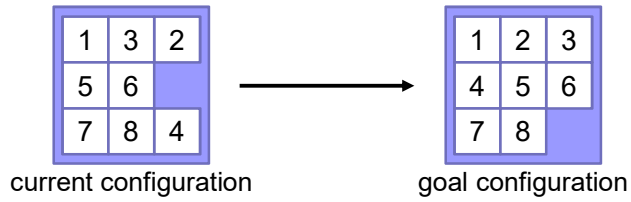
- Culberson and Schaeffer; Searching with Pattern Databases; Canadian Conference on AI; 1996

Pattern Databases

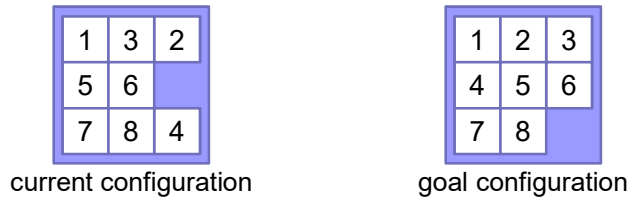
Current Configuration	Distance to goal configuration									
$9!/2 = 181,440$ <div style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td style="background-color: #ccccff;"></td></tr> </table> </div>	1	2	3	4	5	6	7	8		0
1	2	3								
4	5	6								
7	8									
<div style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td style="background-color: #ccccff;"></td></tr> <tr><td>7</td><td>8</td><td>6</td></tr> </table> </div>	1	2	3	4	5		7	8	6	1
1	2	3								
4	5									
7	8	6								
...	...									

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

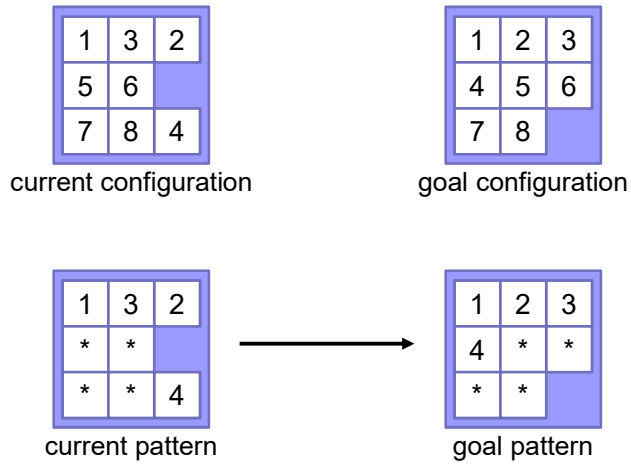
Pattern Databases



Pattern Databases



Pattern Databases

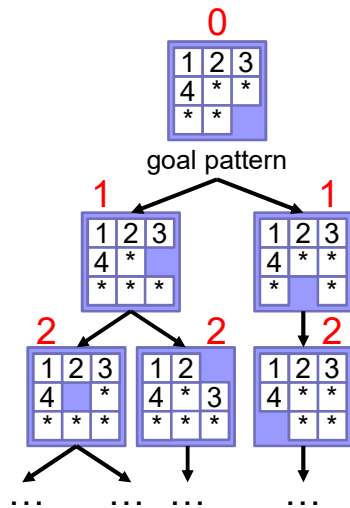


Pattern Databases

Current Pattern	Distance to goal pattern
$9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 = 15,120$...	 0 1 ...

■ These h-values are consistent and thus also admissible.

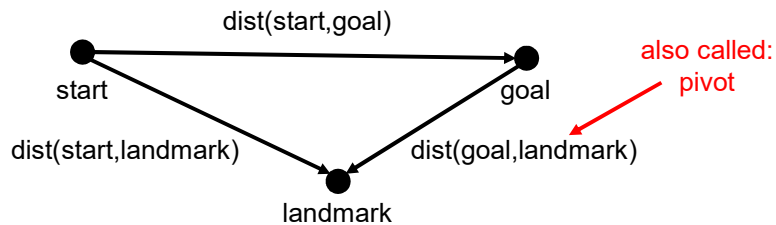
Pattern Databases



Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - **H-Values**
 - Pattern databases (assumes goal is known)
 - **Differential heuristics**
 - FastMap
 - Search algorithm
- Goldberg and Harrelson; Computing the Shortest Path: A* Search Meets Graph Theory; Proceedings of the Symposium on Discrete Algorithms; 2005
- Ng and Zhang; Predicting Internet Network Distance with Coordinates-Based Approaches; Proceedings of the International Conference on Computer Communications; 2002.

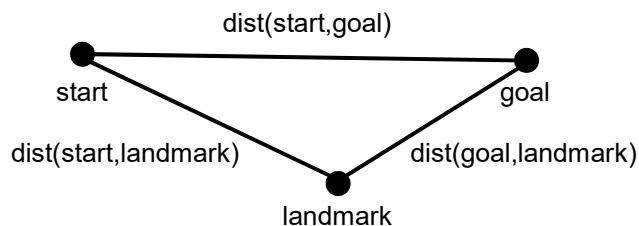
Differential Heuristics



- $\text{dist}(\text{start}, \text{landmark}) \leq \text{dist}(\text{start}, \text{goal}) + \text{dist}(\text{goal}, \text{landmark})$
- $\text{dist}(\text{start}, \text{landmark}) - \text{dist}(\text{goal}, \text{landmark}) \leq \text{dist}(\text{start}, \text{goal})$
- $\max_{\text{landmark}} (\text{dist}(\text{start}, \text{landmark}) - \text{dist}(\text{goal}, \text{landmark})) \leq \text{dist}(\text{start}, \text{goal})$
- These h-values are consistent and thus also admissible.

The informedness of the h-values increases monotonically with the number of landmarks.

Differential Heuristics



- $\text{dist}(\text{start}, \text{landmark}) \leq \text{dist}(\text{start}, \text{goal}) + \text{dist}(\text{goal}, \text{landmark})$ and $\text{dist}(\text{goal}, \text{landmark}) \leq \text{dist}(\text{start}, \text{goal}) + \text{dist}(\text{start}, \text{landmark})$
- $\text{dist}(\text{start}, \text{landmark}) - \text{dist}(\text{goal}, \text{landmark}) \leq \text{dist}(\text{start}, \text{goal})$ and $\text{dist}(\text{goal}, \text{landmark}) - \text{dist}(\text{start}, \text{landmark}) \leq \text{dist}(\text{start}, \text{goal})$
- $|\text{dist}(\text{start}, \text{landmark}) - \text{dist}(\text{goal}, \text{landmark})| \leq \text{dist}(\text{start}, \text{goal})$
- $\max_{\text{landmark}} |\text{dist}(\text{start}, \text{landmark}) - \text{dist}(\text{goal}, \text{landmark})| \leq \text{dist}(\text{start}, \text{goal})$
- These h-values are consistent and thus also admissible.

Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - H-Values
 - Pattern Databases (assumes goal is known)
 - Differential Heuristics
 - FastMap
 - Search algorithm

- Cohen, Uras, Jahangiri, Arunasalam, Koenig, and Kumar; The FastMap Algorithm for Shortest Path Computations; Proceedings of the International Joint Conference on Artificial Intelligence; 2018.

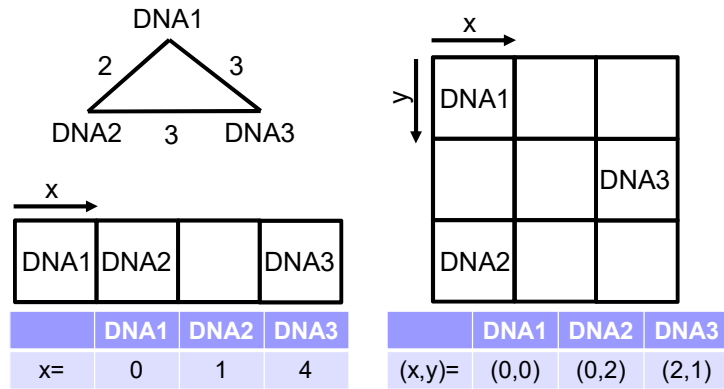
FastMap

- 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

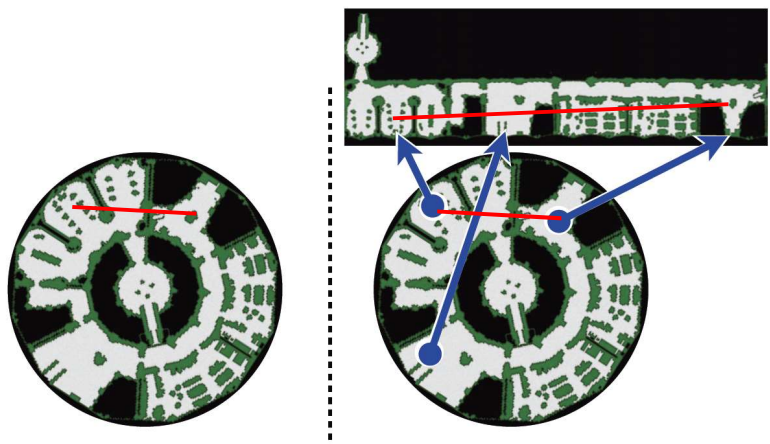
- Faloutsos and Lin; Fastmap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets; Proceedings of the International Conference on Management of Data; 1995

FastMap

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

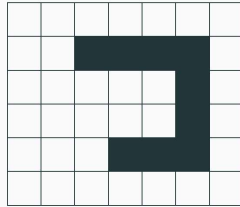


FastMap

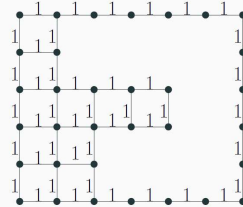


- Rayner, Bowling, and Sturtevant; Euclidean Heuristic Optimization; Proceedings of the AAAI Conference on Artificial Intelligence; 2011

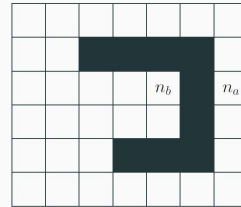
FastMap



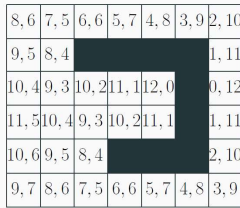
(a)



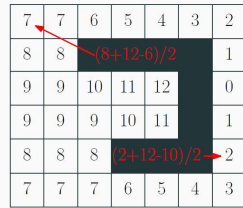
(b)



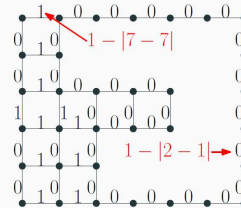
(c)



(d)



(e)

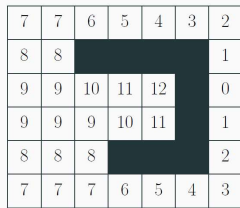


(f)

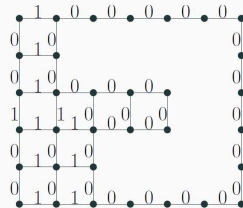
4-neighbor grid

4

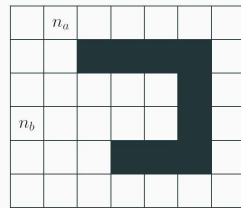
FastMap



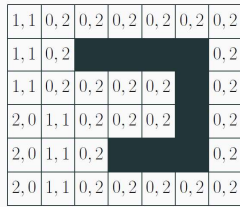
(e)



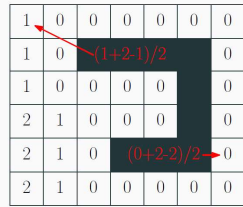
(f)



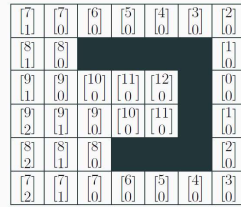
(g)



(h)



(i)



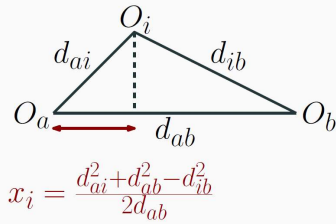
(j)

4-neighbor grid

5

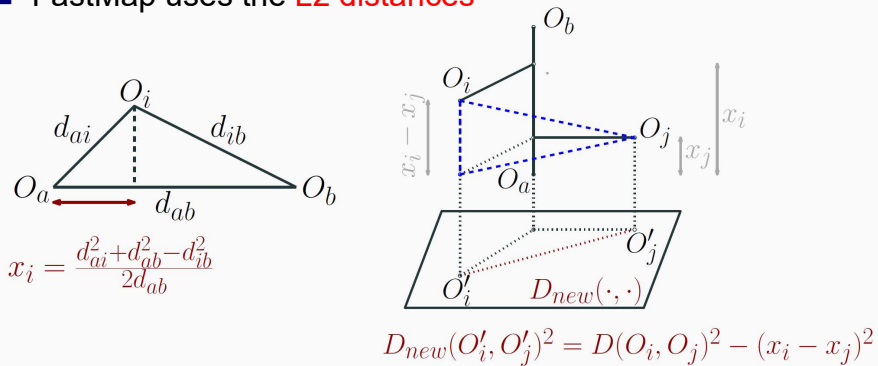
FastMap

- FastMap uses the **L2 distances**



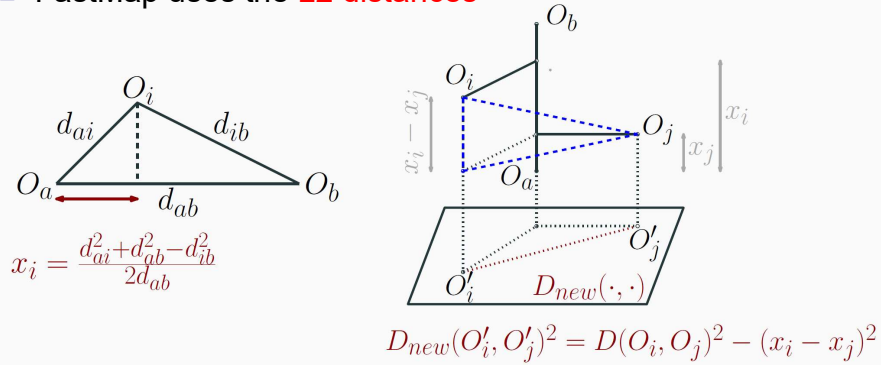
FastMap

- FastMap uses the **L2 distances**



FastMap

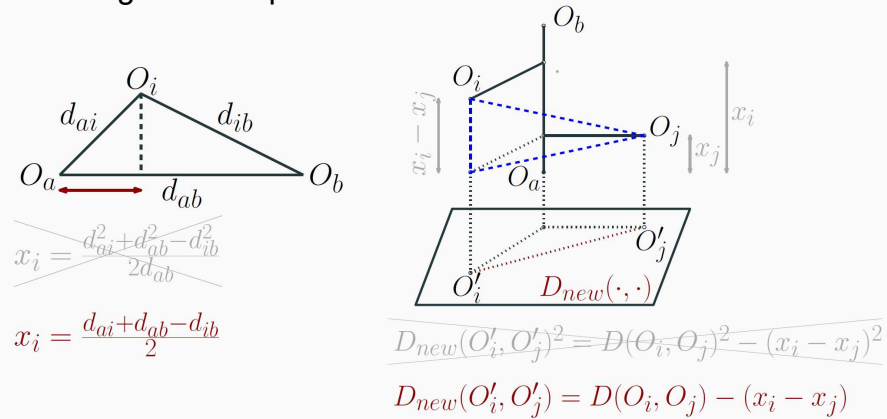
- FastMap uses the **L2 distances**



- The h-values are **not** consistent.

FastMap

- Change FastMap to use the **L1 distances**



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

FastMap

The informedness of the h-values increases monotonically with the number of dimensions K_{max} .

Input: $G = (V, E, w)$, K_{max} and ϵ .

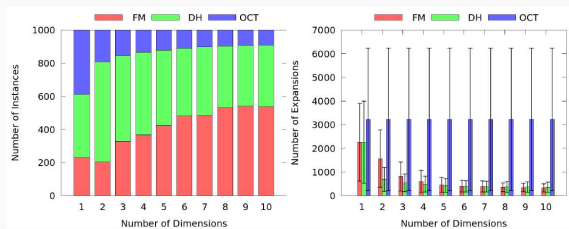
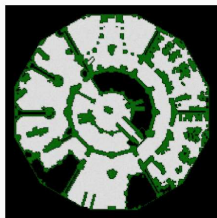
Output: K and $p_i \in \mathbb{R}^K$ for all $v_i \in V$.

```

1  $w' = w$ ;  $K = 1$ ;
2 while  $K_{max} > 0$  do
3   Let  $G' = (V, E, w')$ ;
4    $(n_a, n_b) \leftarrow \text{GetFarthestPair}(G')$ ;
5   Compute shortest path trees rooted at  $n_a$  and  $n_b$ 
   on  $G'$  to obtain  $d_{ab}$ ,  $d_{a_i}$  and  $d_{i_b}$  for all  $v_i \in V$ ;
6   if  $d_{ab} < \epsilon$  then
7     Break;
8   for each  $v \in V$  do
9      $[p_v]_K = (d_{a_v} + d_{ab} - d_{v_b})/2$ 
10  for each edge  $(u, v) \in E$  do
11     $w'(u, v) = w'(u, v) - |[p_u]_K - [p_v]_K|$ ;
12   $K = K + 1$ ;  $K_{max} = K_{max} - 1$ ;
  
```

- FastMap runs fast: its runtime is $O(|E| + |V| \log |V|)$.

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; 2018.
- Talk: Tue 17, 16:40-18:20, Heuristic Search and Optimization (IJCAI)

Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - H-Values
 - Search algorithm
 - Compressed Path Databases (see the slides by Daniel Harabor)
 - Contraction Hierarchies
 - Subgoal Graphs

Preprocessing-Based Search

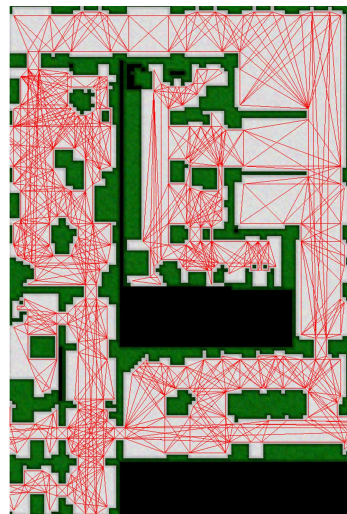
- Spend effort before the search to accelerate it
 - H-Values
 - Search algorithm
 - Compressed Path Databases
 - Contraction Hierarchies (see the slides by Daniel Harabor)
 - Subgoal Graphs

Preprocessing-Based Search

- Spend effort before the search to accelerate it
 - H-Values
 - Search algorithm
 - Compressed Path Databases
 - Contraction Hierarchies
 - Subgoal Graphs

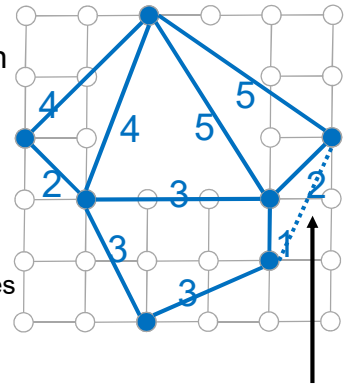
Subgoal Graphs

- Use overlay graph constructed during preprocessing
- Customizable via reachability relation
- Provides optimality guarantee



Subgoal Graphs

- **Idea 1:** Search the blue overlay graph (= **subgoal graph**) rather than the black original graph
- During **preprocessing**
 - Pick a set of subgoals $S \subseteq V$
 - Add edges between **direct-reachable** subgoals (= no shortest path passes through a subgoal) so that the distances on the subgoal graph equal the distances on the original graph

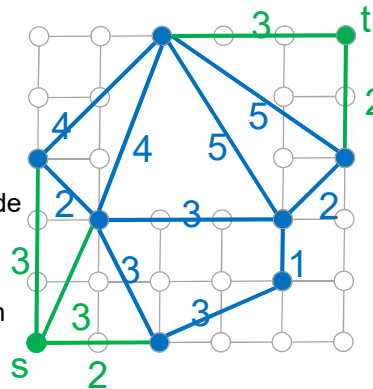


don't add this edge

- + search ignores non-subgoal nodes

Subgoal Graphs

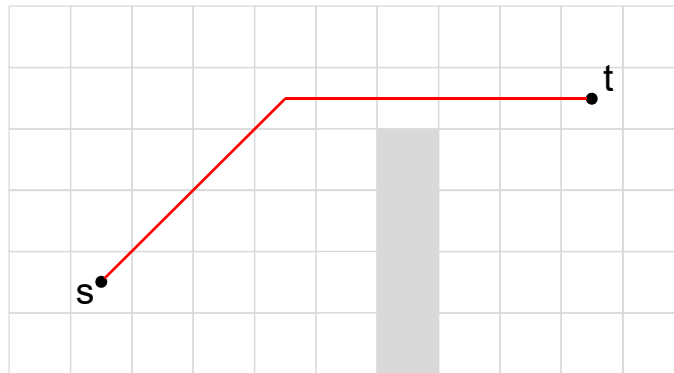
- **Idea 1:** Search an overlay graph (= **subgoal graph**) rather than the original graph
- During **(shortest path) query**
 - **Connect** the start node s and goal node t to the subgoal graph
 - **Search** the query subgoal graph
 - **Refine** the subgoal path into a path on the original graph



- - one has to connect and refine

Reachability Relation

- Freespace reachability
= at least one freespace path is unblocked



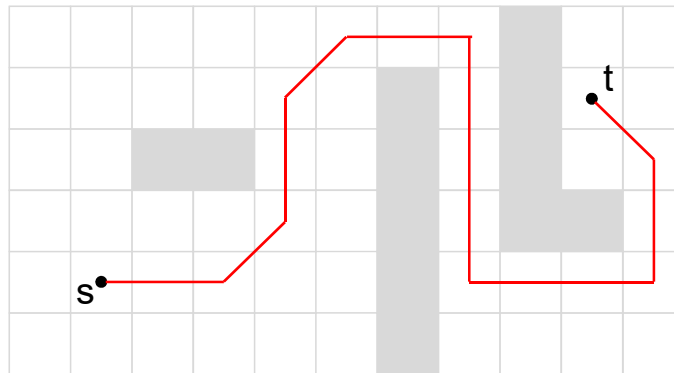
8-neighbor grid

Subgoal Graph

- A set of subgoals S is an **R-shortest path cover (R-SPC)**
= For any two nodes s and t ,
 - $(s,t) \in R$ or
 - at least one shortest s - t path has to be covered by at least one subgoal (= one or more subgoals split at least one shortest s - t path into segments)

- Uras and Koenig; Feasibility Study: Subgoal Graphs on State Lattices; Proceedings of the Symposium on Combinatorial Search; 2017.

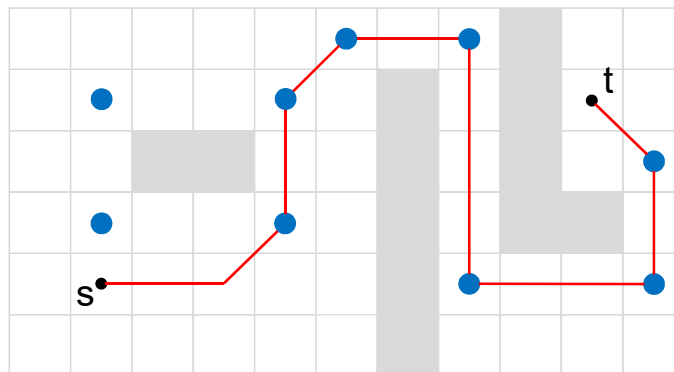
Subgoal Graph



8-neighbor grid

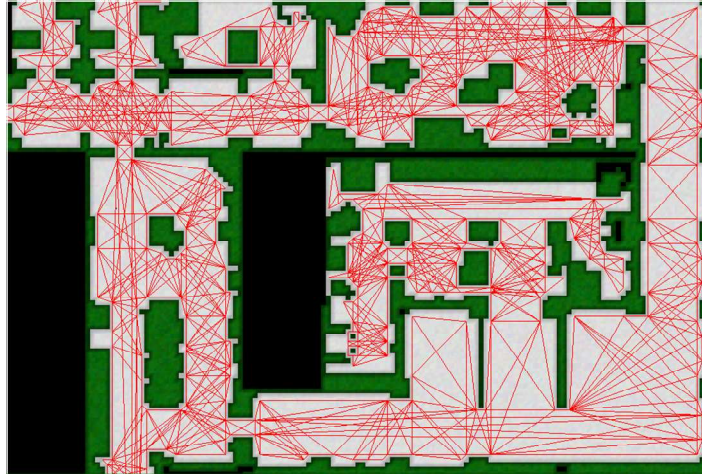
Subgoals

- Place subgoals at the “convex corners” of obstacles



8-neighbor grid

Subgoal Graphs

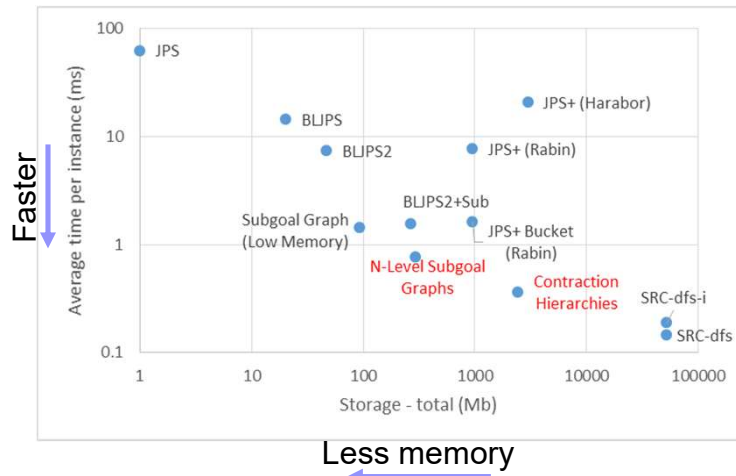


Experimental Results

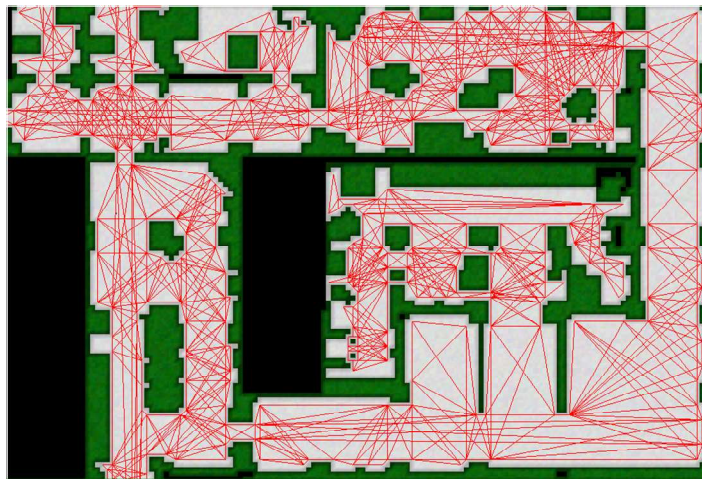
	Preprocessing time (ms)	Storage (MB)	Query time (μ s)			Speed up over A*
			Connect	Search	Refine	
game	8.91	1.24	5.20	82.13	2.10	36.63
maze	11.62	1.11	2.24	518.06	12.31	29.91
random	64.64	2.27	1.85	1635.98	7.25	2.64
room	7.30	1.06	2.55	83.36	2.83	78.46
all	23.12	1.42	2.96	579.88	6.12	12.95

Experimental Results: GPPC 2014

- Optimal algorithms

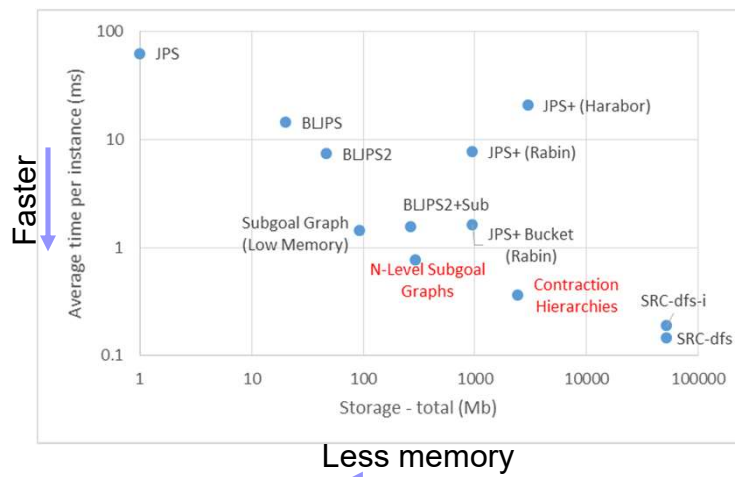


Subgoal Graphs



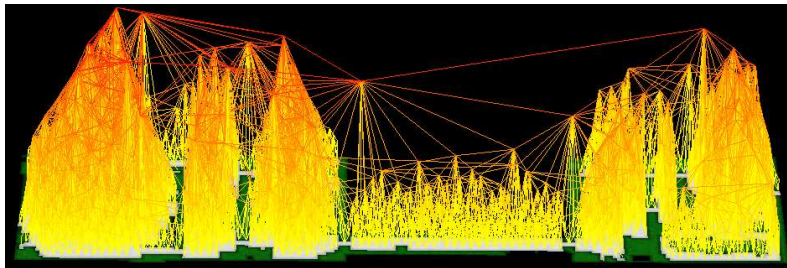
Experimental Results: GPPC 2014

- Optimal algorithms

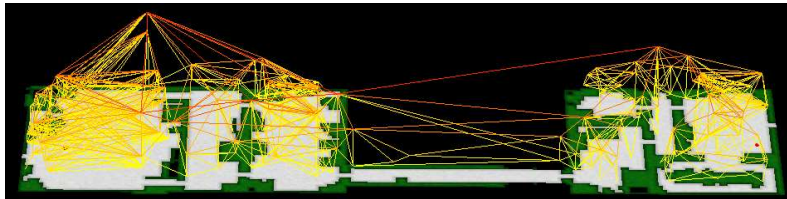


Contraction Hierarchies on Subgoal Gs

- Contraction hierarchies: 4,543 nodes and 32,552 edges

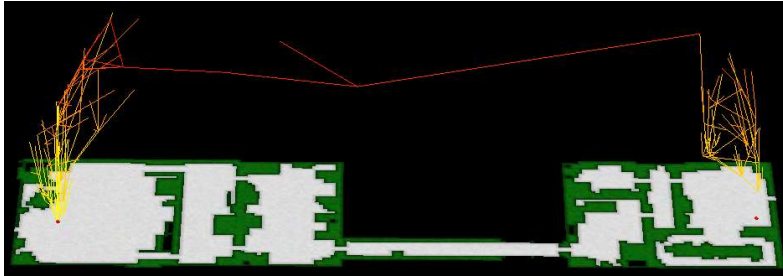


- Contraction hierarchies on subgoal graphs: 183 nodes and 763 edges

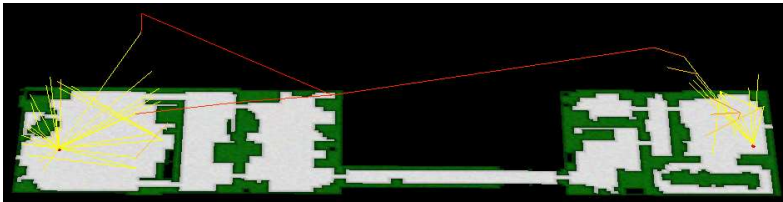


Contraction Hierarchies on Subgoal Gs

- Contraction hierarchies: 4,543 nodes and 32,552 edges



- Contraction hierarchies on subgoal graphs: 183 nodes and 763 edges



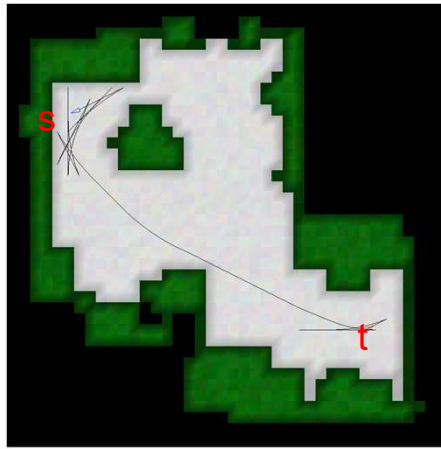
Contraction Hierarchies on Subgoal Gs

	% discarded over CH		Factor of improvement over CH				
	Nodes	Edges	Search	Refine	Connect +	Connect	Memory
					Search	+ Search	
game	97.98	97.75	5.18	2.22	3.67	3.39	10.85
maze	94.44	98.10	10.39	1.68	6.19	2.69	12.83
random	60.40	53.37	1.07	0.79	1.03	0.96	1.84
room	98.18	98.83	4.61	1.82	3.59	3.01	17.78
all	87.70	89.76	2.76	1.42	2.35	2.01	6.16

- [Uras and Koenig; Understanding Subgoal Graphs by Augmenting Contraction Hierarchies; Proceedings of the International Joint Conference on Artificial Intelligence; 2018.](#)
- [Talk: Thu 19, 10:25-11:40, Heuristic Search Session \(IJCAI\)](#)

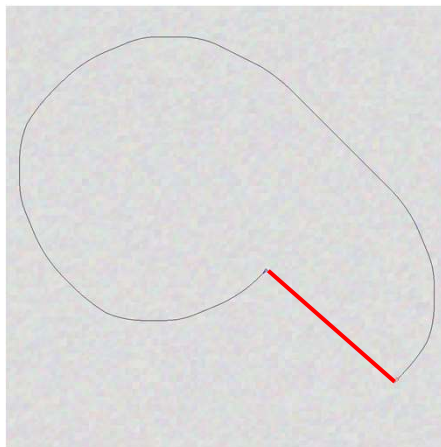
Subgoal Graphs on State Lattices

- State: (x,y,θ) with **location** and **orientation**
- Here: 16 orientations and 4 motion primitives per pose



Subgoal Graphs on State Lattices

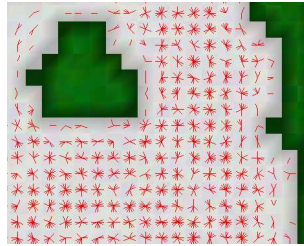
- Store freespace distances up to a bound B
- R-Connect: BFS – R-Refine: DFS



Experimental Results

- 209 x 281 arena2.map with 16 poses and 4 motion primitives per pose
- State lattice: 375,391 nodes and 1,262,812 edges
- Reachability relation: freespace reachability with bound 75

	Preprocessing time (s)	Size vs original graph		Query time (ms)			Speed up over A*
		Nodes	Edges	Connect	Search	Refine	
SUB-FR75	90	44%	149%	0.006	18.797	0.039	1.75



Experimental Results

- The problem persists even if
 - we change the reachability bound
 - we use bounded-distance reachability as the reachability relation
 - we use slow preprocessing to guarantee the minimality of subgoals
 - we use contraction hierarchies
- One solution: give up on the optimality guarantee

Experimental Results (Urban Primitives)

- 768 x 1024 map with 32 poses and 29-32 motion primitives per pose
- State lattice: 15,255,281 nodes and 333,465,922 edges

	Preprocessing time (s)	Query time (ms)			Speed up over A*	Suboptimality factor	
		Connect	Search	Refine		Average	Maximum
FR50	980	6.357	17.787	0.295	128.52	1.116	1.637
FR100	2521	47.353	9.117	0.747	54.89	1.087	1.374
FR150	3390	99.687	7.535	1.394	28.92	1.093	1.427
CR50	207	0.807	21.192	0.016	142.67	1.110	1.657
CR100	356	5.795	9.790	0.015	201.34	1.088	1.458
CR150	475	12.119	7.922	0.015	156.61	1.096	1.613

- [Uras and Koenig; Fast Near-Optimal Path Planning on State Lattices with Subgoal Graphs; Proceedings of the Symposium on Combinatorial Search; 2018.](#)
- [Talk: Sun 15, 14:20-14:40, A* and Path Planning Session \(SoCS\)](#)

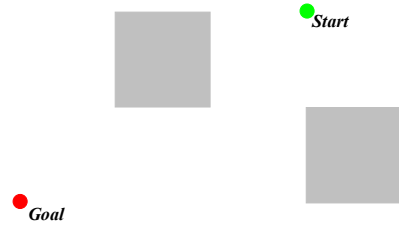
Any-Angle Search

- Do not restrict paths to a grid or an a-priori given graph to find optimal paths in continuous environments

Any-Angle Search



[from JPL]



- A. Nash and S. Koenig. Any-Angle Path Planning. Artificial Intelligence Magazine, 34(4), 85-107, 2013.

A* on Visibility Graphs

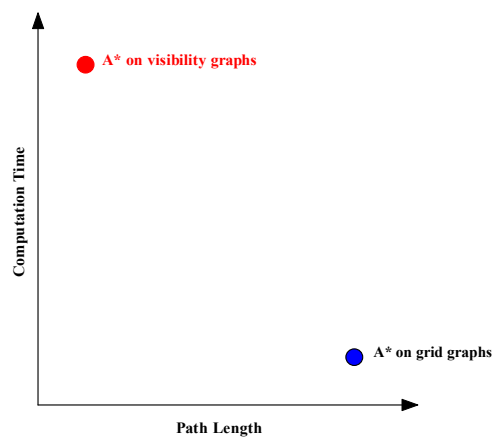
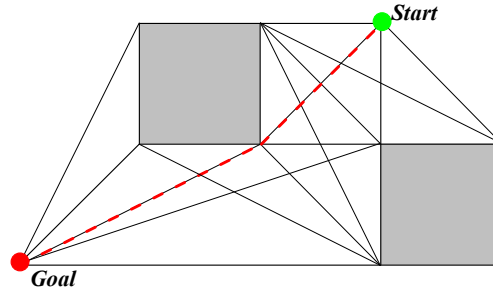


figure is notional

A* on Visibility Graphs

- A* on Visibility Graphs [Lozano-Perez et al.]
- Note: Sophisticated versions exist, e.g. [Shah and Gupta]



- Shortest path in 2D terrain
- Slow due to many edges and line-of-sight checks

A* on Grid Graphs

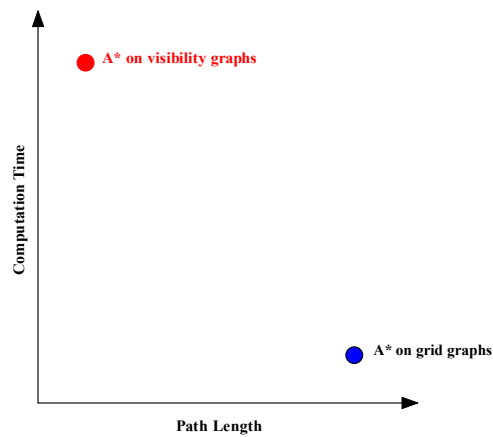
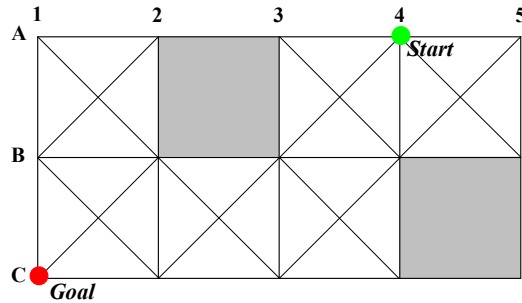


figure is notional

A* on Grid Graphs

A* on grid graphs

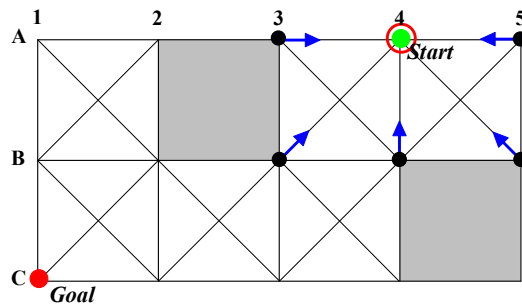


- A* assigns two values to every vertex s
 - $g(s)$: the length of the shortest path from the start vertex to s found so far
 - parent(s): the parent pointer used to extract the path after termination
 - Following the parents from s to the start vertex results in a path of length $g(s)$

8-neighbor grid

A* on Grid Graphs

A* on grid graphs

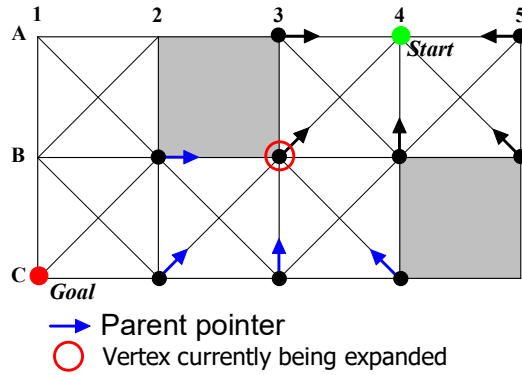


- Parent pointer
- Vertex currently being expanded

8-neighbor grid

A* on Grid Graphs

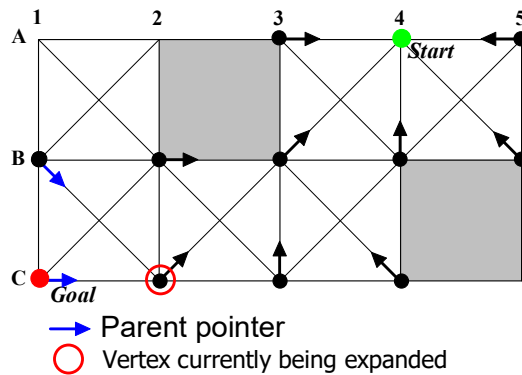
- A* on grid graphs



8-neighbor grid

A* on Grid Graphs

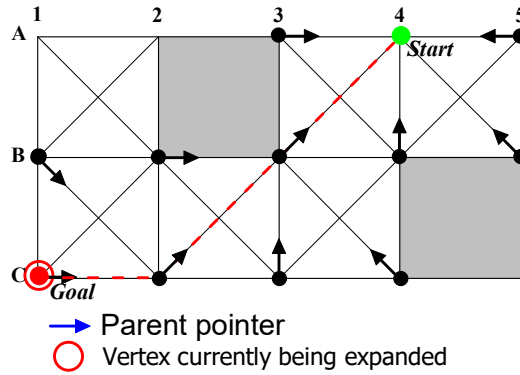
- A* on grid graphs



8-neighbor grid

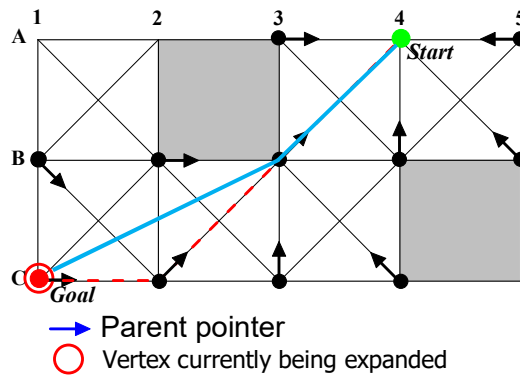
A* on Grid Graphs

- A* on grid graphs



A* on Grid Graphs

- A* on grid graphs

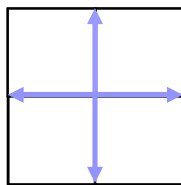


A* on Grid Graphs

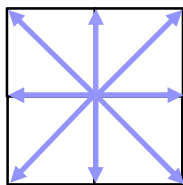
Dimension	Regular Grid	Neighbors	% Longer Than Shortest Path
2D	triangular grid corners	3-neighbor	≈ 100
		6-neighbor	≈ 15
	square grid corners	4-neighbor	≈ 41
		8-neighbor	≈ 8
	hexagonal grid centers	6-neighbor	at least ≈ 15
12-neighbor		at least ≈ 4	
3D	cubic grid corners	6-neighbor	at least ≈ 73
		26-neighbor	at least ≈ 13

Grids with Higher Degree Vertices

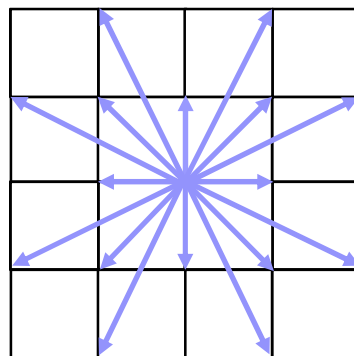
- Grid path finding on the 2^k neighborhoods [Rivera et al.]



$2^2=4$
neighborhood



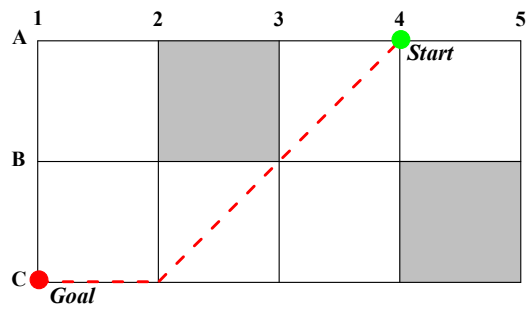
$2^3=8$
neighborhood



$2^4=16$
neighborhood

A* with Post Smoothing

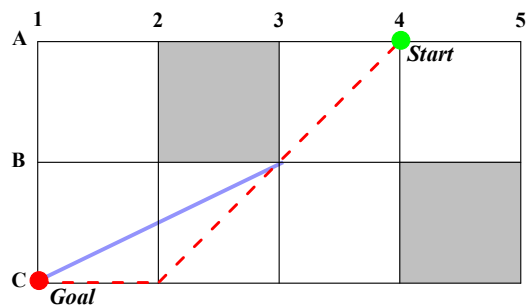
- A* with Post Smoothing [Thorpe; Botea et al.; Millington]



8-neighbor grid

A* with Post Smoothing

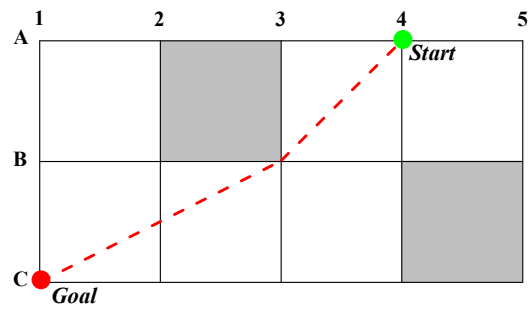
- A* with Post Smoothing



8-neighbor grid

A* with Post Smoothing

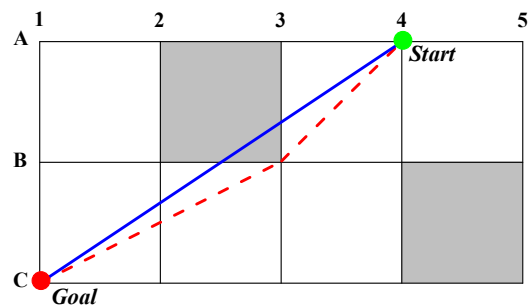
- A* with Post Smoothing



8-neighbor grid

A* with Post Smoothing

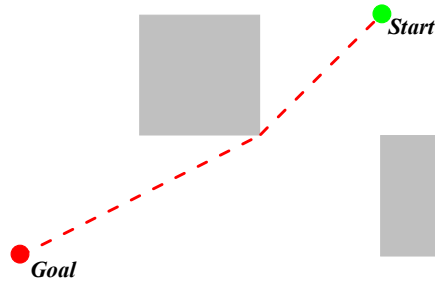
- A* with Post Smoothing



8-neighbor grid

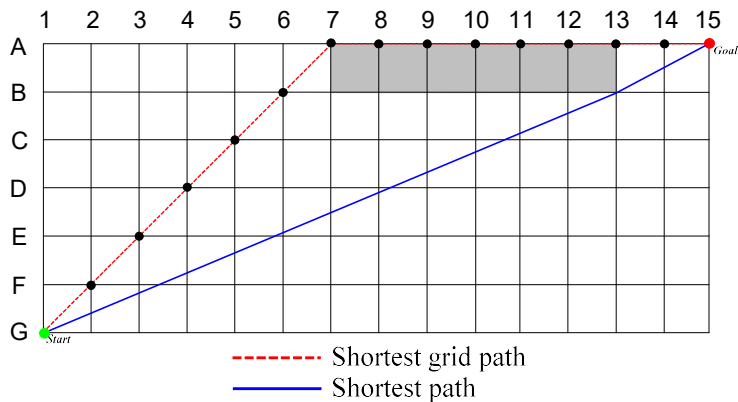
A* with Post Smoothing

- A* with Post Smoothing



A* with Post Smoothing

- A* with Post Smoothing



- Postprocessing often leaves path homotopy unchanged
- Better to interleave the search and the optimization

Suboptimal Any-Angle Search

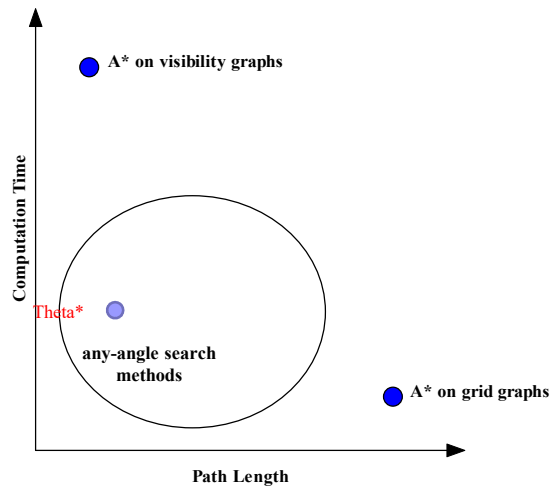


figure is notional

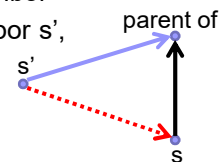
Suboptimal Theta*

■ A*

- The parent of a vertex has to be its neighbor in the graph.
- When expanding vertex s and generating its neighbor s' , A* considers
 - Making s the parent of s' (Path 1)

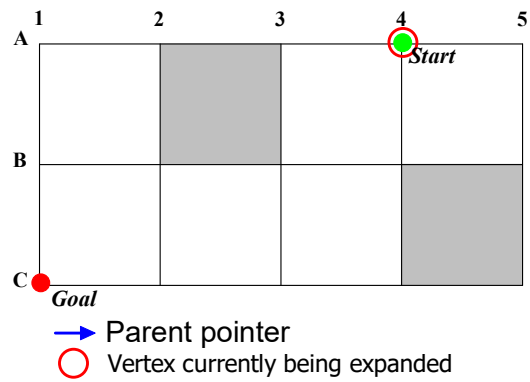
■ Theta*

- The parent of a vertex does not need to be its neighbor
- When expanding vertex s and generating its neighbor s' , Theta* considers
 - Making s the parent of s' (Path 1)
 - Making the parent of s the parent of s' (Path 2)
 - Note: Path 2 is no longer than Path 1 iff it is unblocked. The line-of-sight check can be performed with fast line-drawing algorithms from computer graphics.



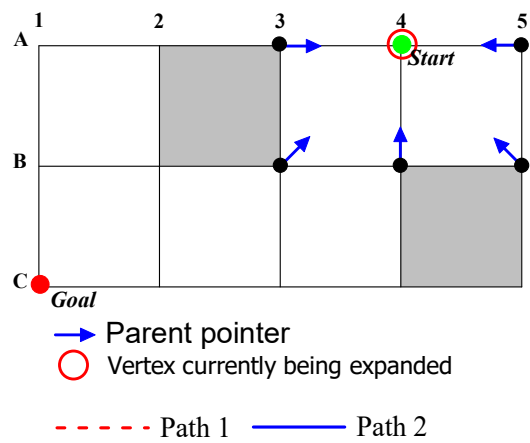
Suboptimal Theta*

■ Theta*



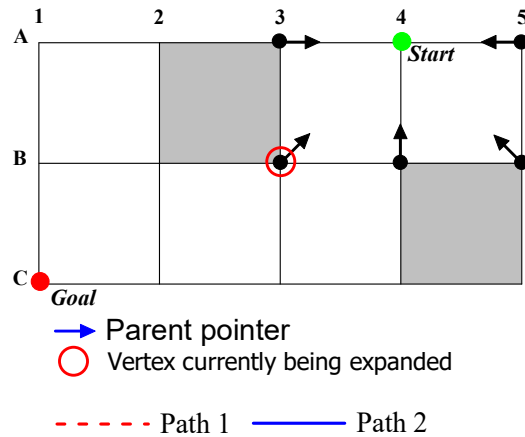
Suboptimal Theta*

■ Theta*



Suboptimal Theta*

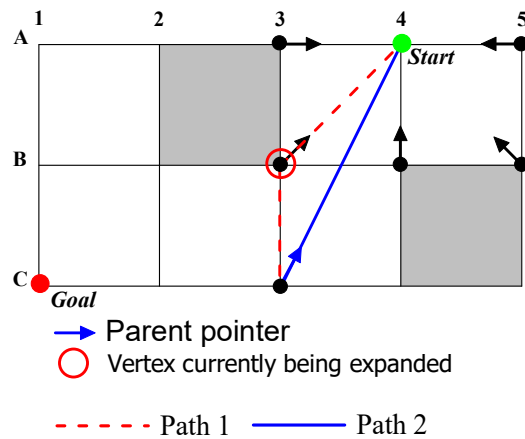
■ Theta*



8-neighbor grid

Suboptimal Theta*

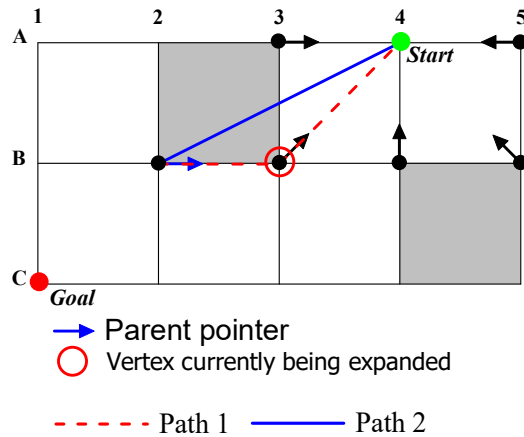
■ Theta*



8-neighbor grid

Suboptimal Theta*

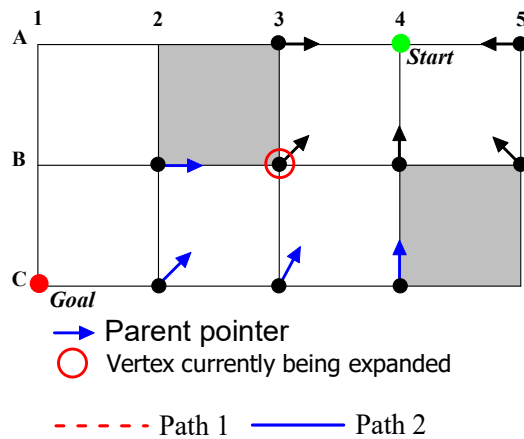
■ Theta*



8-neighbor grid

Suboptimal Theta*

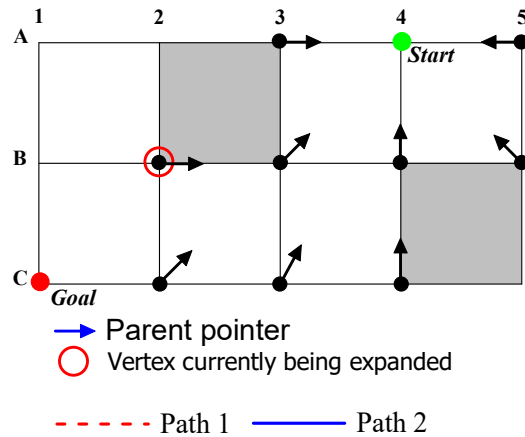
■ Theta*



8-neighbor grid

Suboptimal Theta*

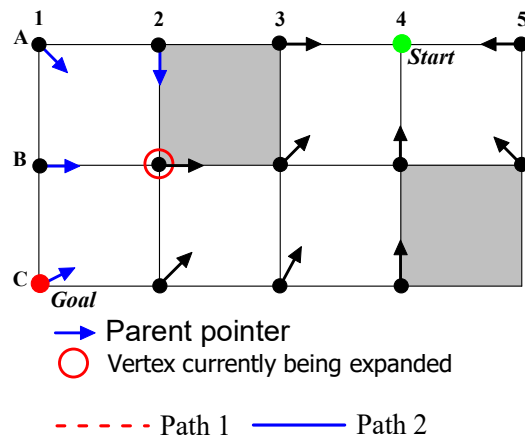
■ Theta*



8-neighbor grid

Suboptimal Theta*

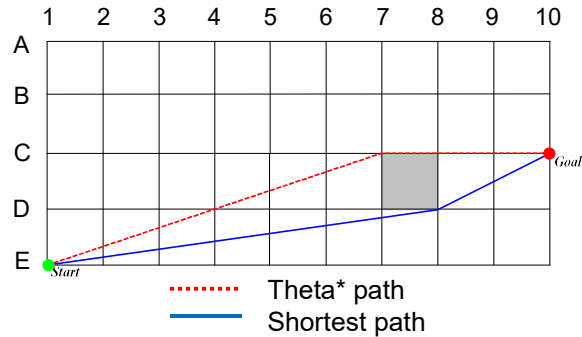
■ Theta*



8-neighbor grid

Suboptimal Theta*

- Theta* is not guaranteed to find shortest paths since the parent of a vertex can only be a neighbor of the vertex or the parent of a neighbor



- The length of the path is still within 0.2% of optimal

8-neighbor grid

Suboptimal Lazy Theta*

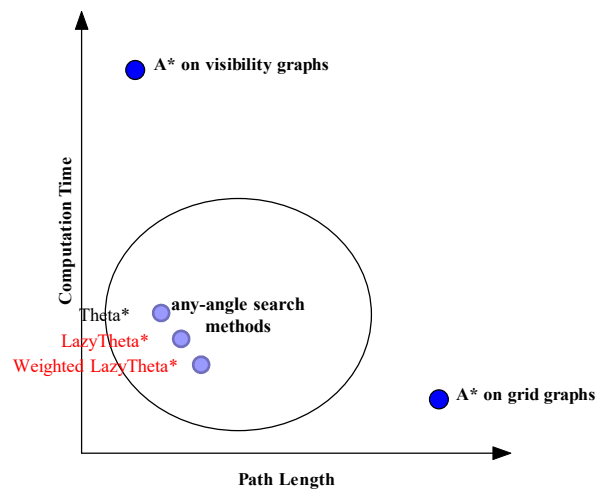


figure is notional

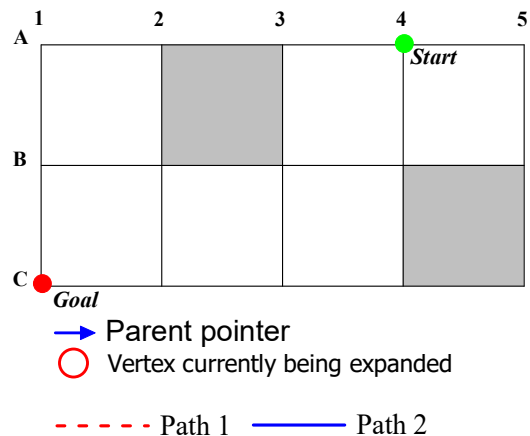
Suboptimal Lazy Theta*

■ Lazy Theta*

- When expanding vertex s and generating its neighbor s' , Lazy Theta* makes the parent of s the parent of s' (Path 2) without a line-of-sight check
- When expanding vertex s' and s' does not have line-of-sight to its parent, then Lazy Theta* makes the best neighbor of s' (= the one that minimizes the g -value of s') the parent of s' (Path 1).
- [Such a neighbor exists since s is one of them.]
- Thus, Lazy Theta* performs one line-of-sight check only for each expanded vertex while Theta* performs one line-of-sight check for each generated vertex

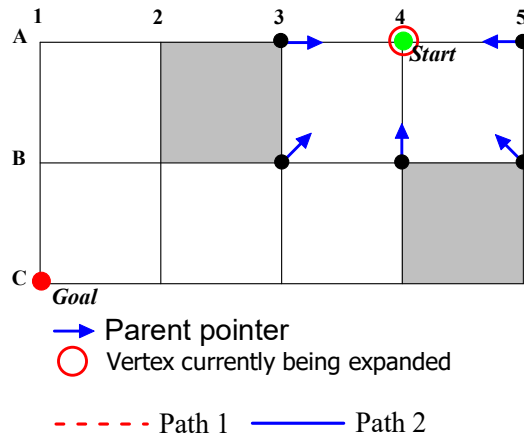
Suboptimal Lazy Theta*

■ Lazy Theta*



Suboptimal Lazy Theta*

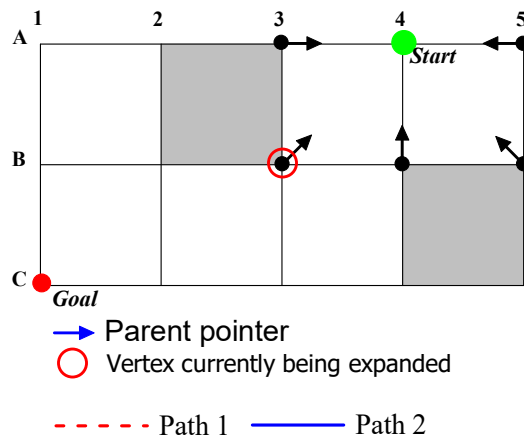
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

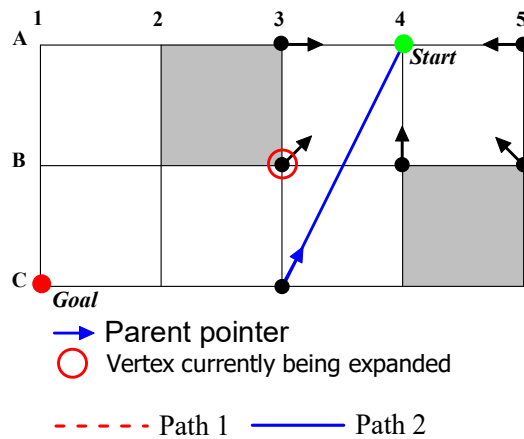
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

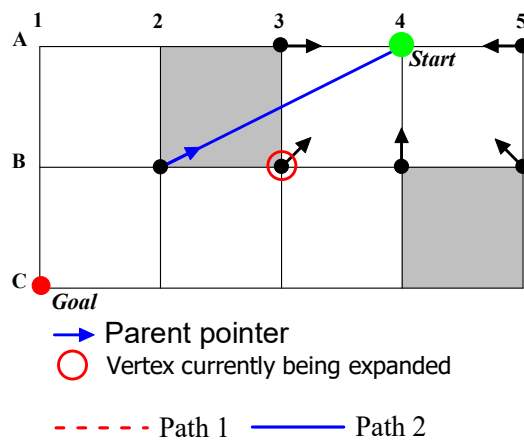
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

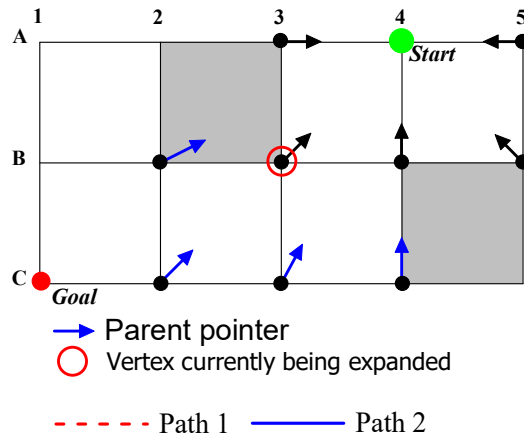
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

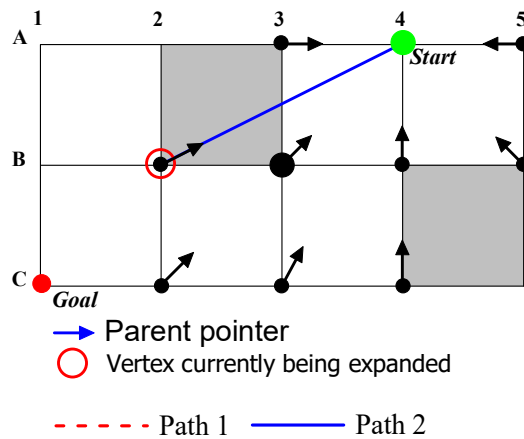
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

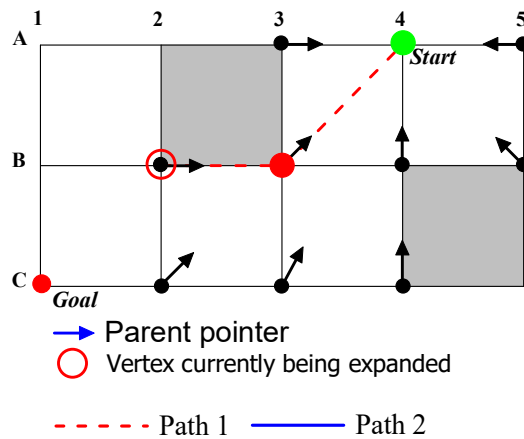
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

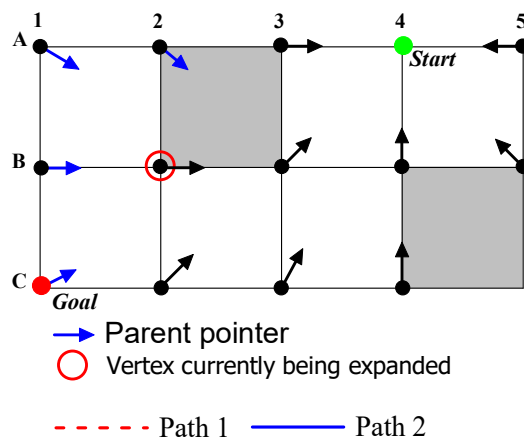
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

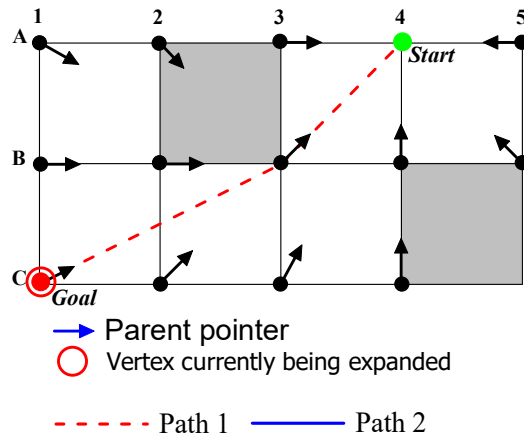
■ Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*

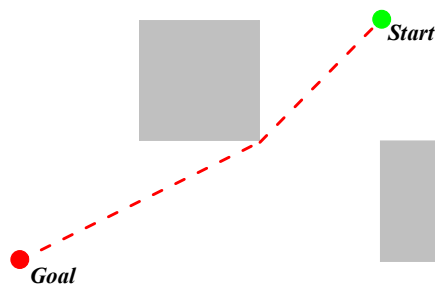
- Lazy Theta*



8-neighbor grid

Suboptimal Lazy Theta*


- Lazy Theta*



- Theta* performed 19 line-of-sight checks
- Lazy Theta* performs 4 line-of-sight checks

Suboptimal Alternatives to Theta*

- Other any-angle search algorithms

- Several versions of Theta*:
Lazy Theta*, Any-Angle Subgoal Graphs, ...
- Accelerated A* [Sislak et al.]
a sophisticated version of Theta*
- Field D* [Ferguson and Stentz] 
an any-angle version of D* (Lite) with interpolation [from JPL]
- Block A* [Yap et al.]
an any-angle version of A* that operates on blocks of cells

[work done by different research groups, not me]

Suboptimal Any-Angle Search

- Overview paper with lots of references
 - A. Nash and S. Koenig. Any-Angle Path Planning. *Artificial Intelligence Magazine*, 34(4), 85-107, 2013.
- Any-Angle Analysis
 - J. Bailey, C. Tovey, T. Uras, S. Koenig and A. Nash. Path Planning on Grids: The Effect of Vertex Placement on Path Length. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, pages 108-114, 2015.
- Theta* Dissertation
 - A. Nash. Any-Angle Path Planning. Dissertation. Computer Science Department, University of Southern California, 2012.

Suboptimal Any-Angle Search

■ Theta* Publications

- K. Daniel, A. Nash, S. Koenig and A. Felner. Theta*: Any-Angle Path Planning on Grids. *Journal of Artificial Intelligence Research*, 39, 533-579, 2010.
- A. Nash, S. Koenig and C. Tovey. Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2010.
- A. Nash, S. Koenig and M. Likhachev. Incremental Phi*: Incremental Any-Angle Path Planning on Grids. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1824-1830, 2009.
- S. Koenig, K. Daniel and A. Nash. A Project on Any-Angle Path Planning for Computer Games for 'Introduction to Artificial Intelligence' Classes. *Technical Report, Department of Computer Science, University of Southern California, Los Angeles (California)*, 2008.
- A. Nash, K. Daniel, S. Koenig and A. Felner. Theta*: Any-Angle Path Planning on Grids. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 1177-1183, 2007.

Suboptimal Any-Angle Search

■ Field D* and 3D Field D* Publications

- J. Carsten, A. Rankin, D. Ferguson and A. Stentz. Global Planning on the Mars Exploration Rovers: Software Integration and Surface Testing. *Journal of Field Robotics*, 26, 337-357, 2009.
- J. Carsten, D. Ferguson and A. Stentz. 3D Field D*: Improved Path Planning and Replanning in Three Dimensions. *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 3381-3386, 2006.
- D. Ferguson and A. Stentz. Using Interpolation to Improve Path Planning: The Field D* Algorithm. *Journal of Field Robotics*, 23(2), 79-101, 2006.
- A more general closed form linear interpolation equation that can be used on triangular meshes was introduced in L. Sapronov and A. Lacaze: Path Planning for Robotic Vehicles using Generalized Field D*. *Proceedings of the SPIE*, 6962, 2010.

■ 2^k Neighborhood Publications

- N. Rivera, C. Hernandez and J. Baier: Grid Pathfinding on the 2^k Neighborhoods. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 891-897, 2017.

Suboptimal Any-Angle Search

■ Block A* Publications

- P. Yap, N. Burch, R. Holte and J. Schaeffer: Block A*: Database-Driven Search with Applications in Any-Angle Path Planning. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2011.
- P. Yap, N. Burch, R. Holte and J. Schaeffer: Any-Angle Path Planning for Computer Games. *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2011.

■ Accelerated A* Publications

- D. Sislak, P. Volf and M. Pechoucek: Accelerated A* Trajectory Planning: Grid-Based Path Planning Comparison. *Proceedings of the ICAPS 2009 Workshop on Planning and Plan Execution for Real-World Systems*, 2009.
- D. Sislak, P. Volf and M. Pechoucek: Accelerated A* Path Planning. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2009.

Conclusions

- For more information on all projects, 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 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.