

Theta*

Efficient Any-Angle Path Planning

Kenny Daniel and Alex Nash

March 26, 2007

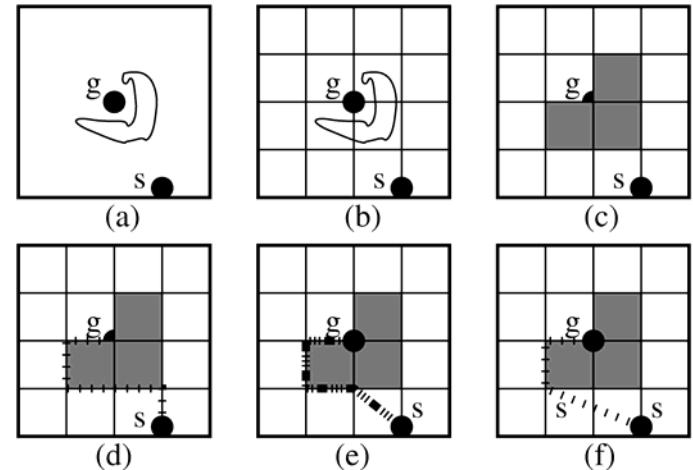
Path Planning Overview

- Uses for Path Planning
 - Navigation
 - Games
 - Mars Rovers
- Existing Algorithms
 - Dijkstra
 - A*
 - D* lite
 - Field D*

Discretization

- Problem: Environments are not necessarily discrete grids
- Solution: Discretize a continuous terrain into square cell

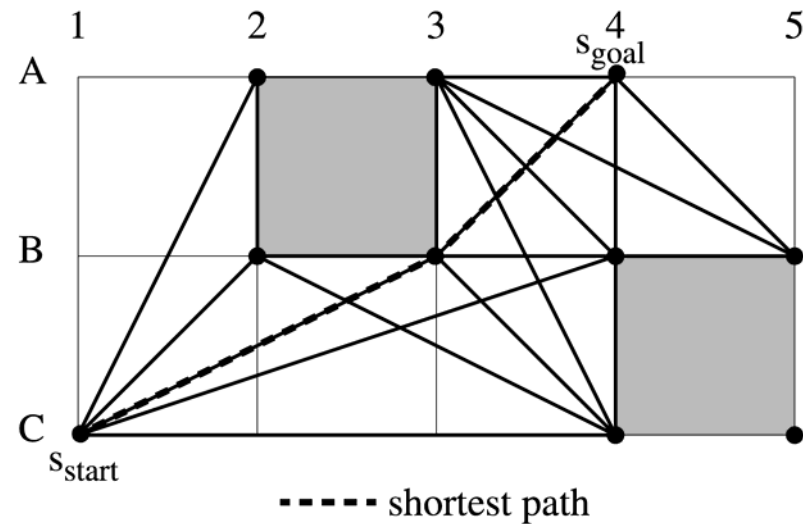
- Cells can be blocked or free space
- Goal is to find a short and realistic looking path from a start location to the goal location
- For our problem cells are anchored at the corners of cells.



Graph Construction

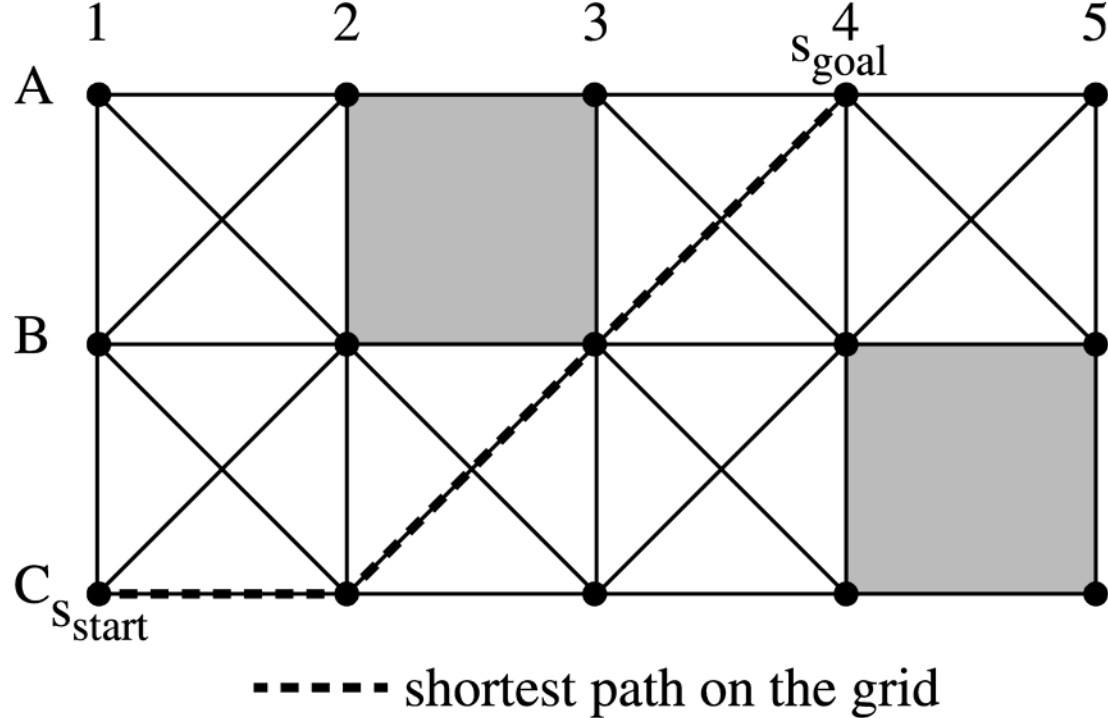
- Given a discrete map of the environment, we must then construct a graph on which to run our search algorithms
 - An optimal solution can be found by using Dijkstra's algorithm on a graph with complete connectivity (i.e. each cell has every other cell as a neighbor)
 - While this returns the optimal path, its runtime is very bad (worse than quadratic).
 - Two other approaches that attempt to more effectively balance the runtime path quality trade-off:
 - Visibility Graphs
 - Local Grids

Visibility Graphs



- Visibility graphs reduce the branching factor by observing that the optimal solution contains only the corners of blocked cells, the start node, and the goal node
 - Two nodes in the graph are connected iff there are no blocked cells obstructing the line-of-sight between the nodes.
 - Solution is optimal but the runtime is still quadratic.

Grid



- Grids are similar in that the graphs contain all cells, but each cell has only its local neighbors (usually 4 or 8)
- Solutions are now found in time linear in the number of cells, but solution quality is reduced
 - Paths formed by grid edges are sub-optimal in length, and contain many unnecessary kinks.

A*

- A* is the standard heuristic search algorithm
 - A* expands significantly fewer nodes than Dijkstra, by using a heuristic $h(s)$
 - Fringe nodes are then expanded based on the smallest f value where
 - $f = g + h$
 - When $h = 0$, it degenerates to standard Dijkstra.
- A* maintains two values for every vertex s
 - The g -value $g(s)$, which is the length of the shortest path from the start vertex to s found so far
 - The parent $\text{parent}(s)$, that is used to extract the path after the search halts.
- A* maintains two global data structures
 - The open list, a priority queue that contains vertices considered for expansion
 - The closed list, which contains vertices that have already been expanded, and ensures that each vertex is expanded only once
- A* updates the g -value and parent (**Path 1**)
 - Vertex s has an unexpanded vertex s' .
 - The path from the start vertex to s [$=g(s)$] and from s to s' in a straight line [$=c(s,s')$], resulting in $g(s) + c(s,s')$.
 - If this new path is shorter than the existing path then the parent and g -value are updated.

Theta*

- We propose Theta*, a variant of A* which compromises between these two extremes
 - Information is propagated locally along grid edges (to achieve a short runtime), but
 - Paths are not constrained to grid edges.

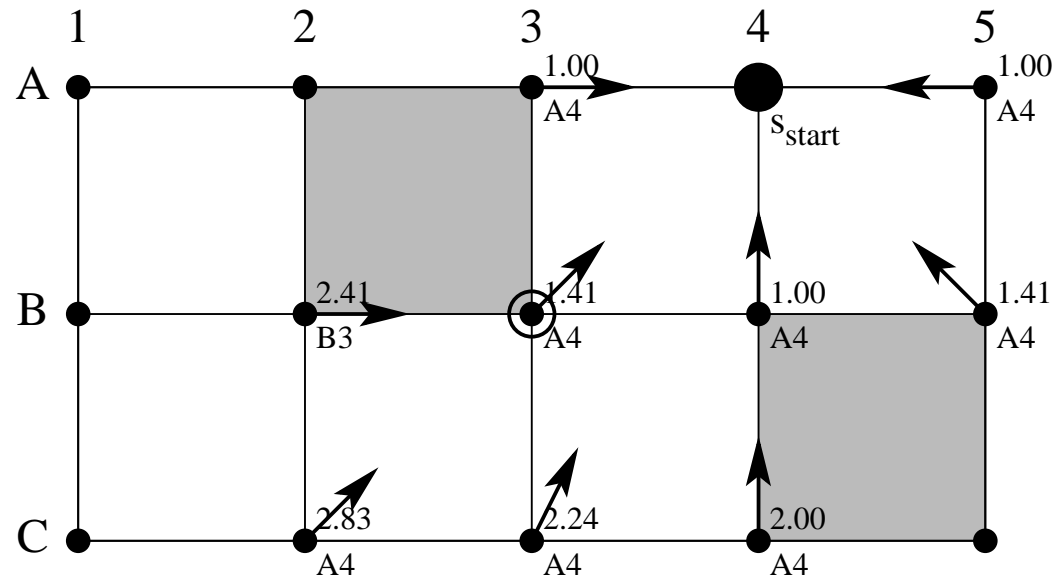
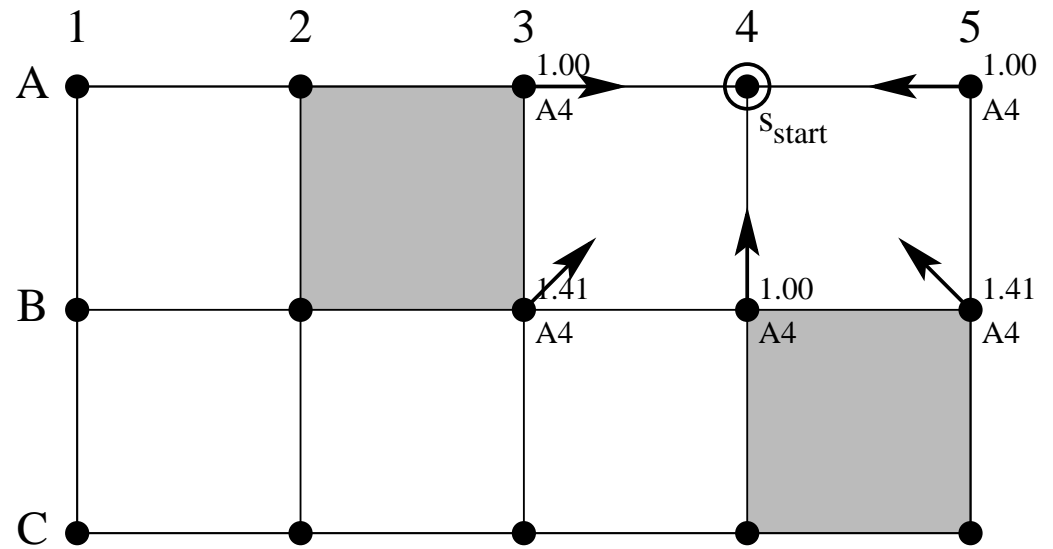
Basic Theta*

- The key different between Theta* and A*:
 - A*: parent vertex must be a local neighbor
 - Theta*: parent vertex can be any vertex
- Basic Theta* is identical to A* except that Basic Theta* updates the g-value and parent by considering 2 paths
 - Path 1 from A*
 - Path 2: Basic Theta* also considers the path from the start vertex to parent(s) [=g(parent(s))] and from parent(s) to s' in a straight line [=c(parent(s), s')] resulting in g(parent(s)) + c(parent(s), s'). If s to s' is unobstructed.
- The idea is that Path 2 is shorter than Path 1 due to the triangle inequality.

- Example Trace

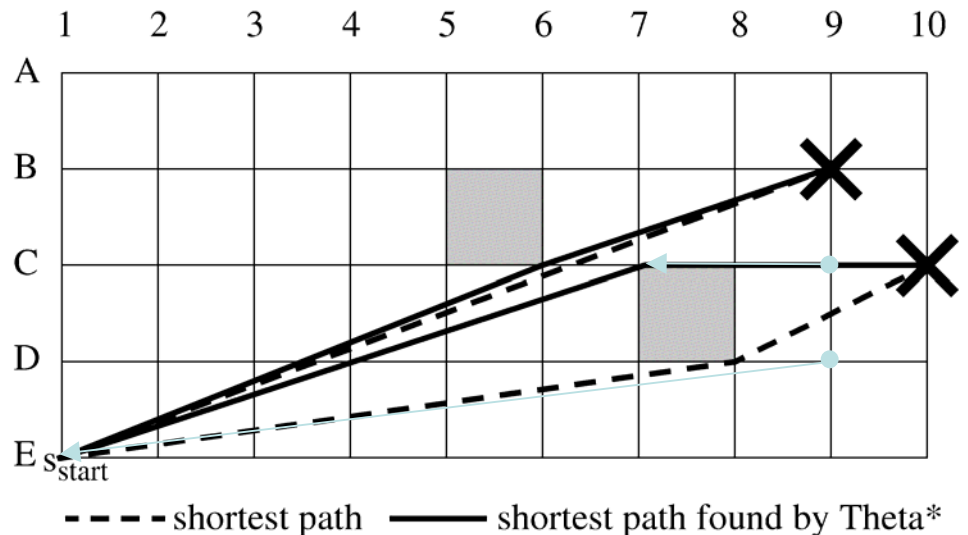
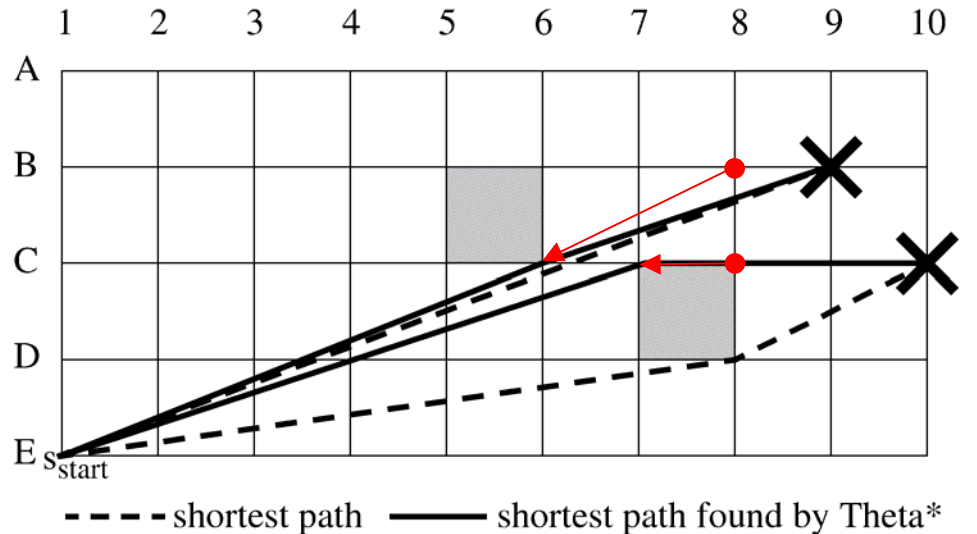
- Top: B3 (with parent A4) gets expanded. B2 is an unexpanded neighbor of B3 which doesn't have line-of sight to A4 and thus Path 1 is applied

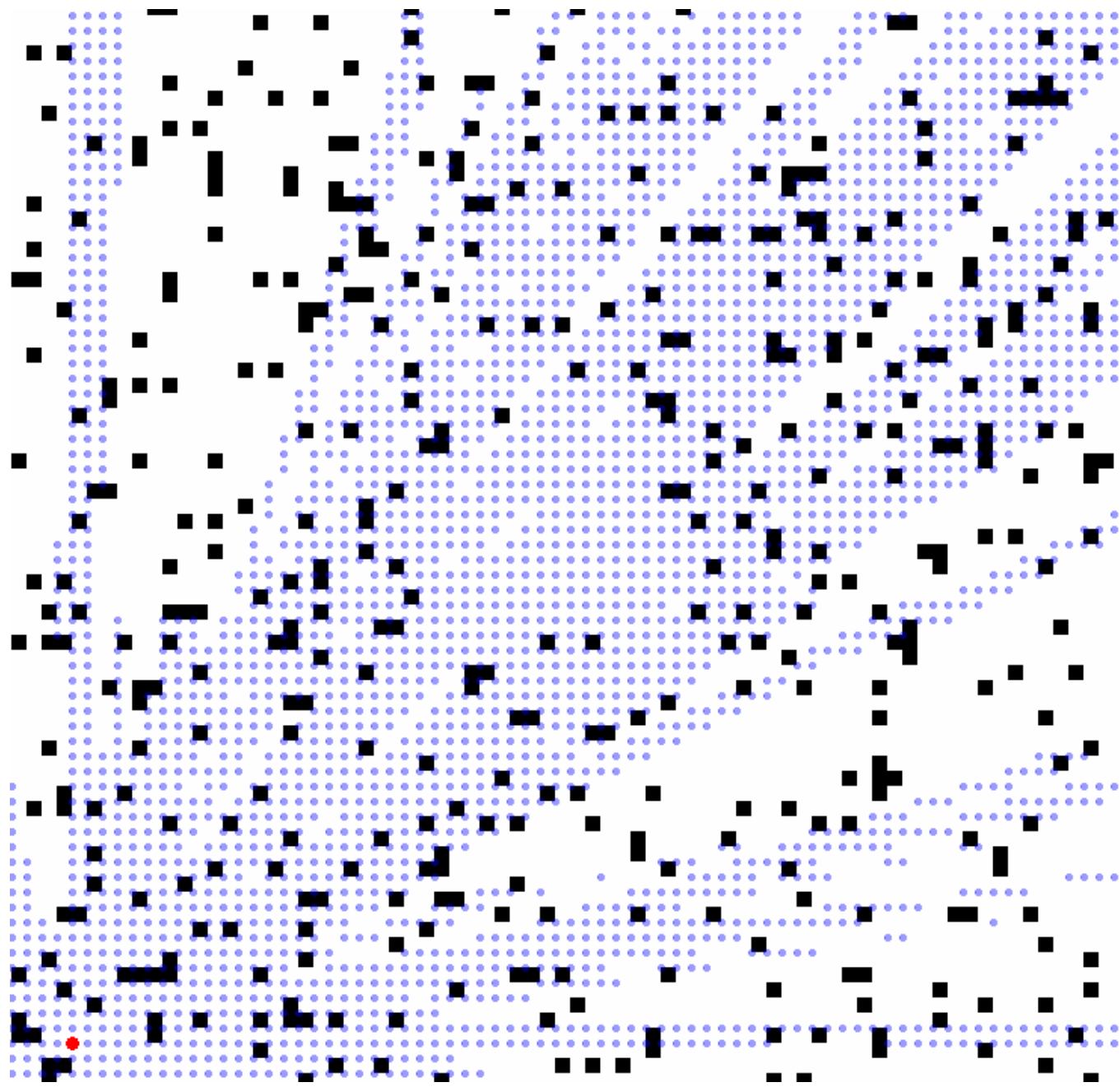
- Bottom: C3 is an unexpanded neighbor of B3 which does have line-of-sight to A4 and thus Path2 is applied.



Basic Theta* (suboptimality)

- Basic Theta* is simple and fast but is not guaranteed to find optimal solutions.
- The problem is as follows:
 - Vertex p can only be the parent of another vertex s if either p is a neighbor of s (Path1) or p is the parent of a neighbor of s (Path 2)
 - Vertex D8 should be the parent of vertex C10 since this results in a shortest path from the start vertex E1 to C10. However, none of the neighbors of C10 have D8 as a parent since the shortest paths from the start vertex E1 to them move around the blocked cell in different ways. For example, C7 is correctly the parent of C9, and E1 is correctly the parent of D9.
 - Similarly, E1 should be the parent of B9 but none of the neighbors of B9 have it as a parent because none of them have line-of-sight to E1 through the small gap formed by the two blocked cells.
- In these two instances the paths produced by Basic Theta* are less than 0.2 percent longer than the minimal

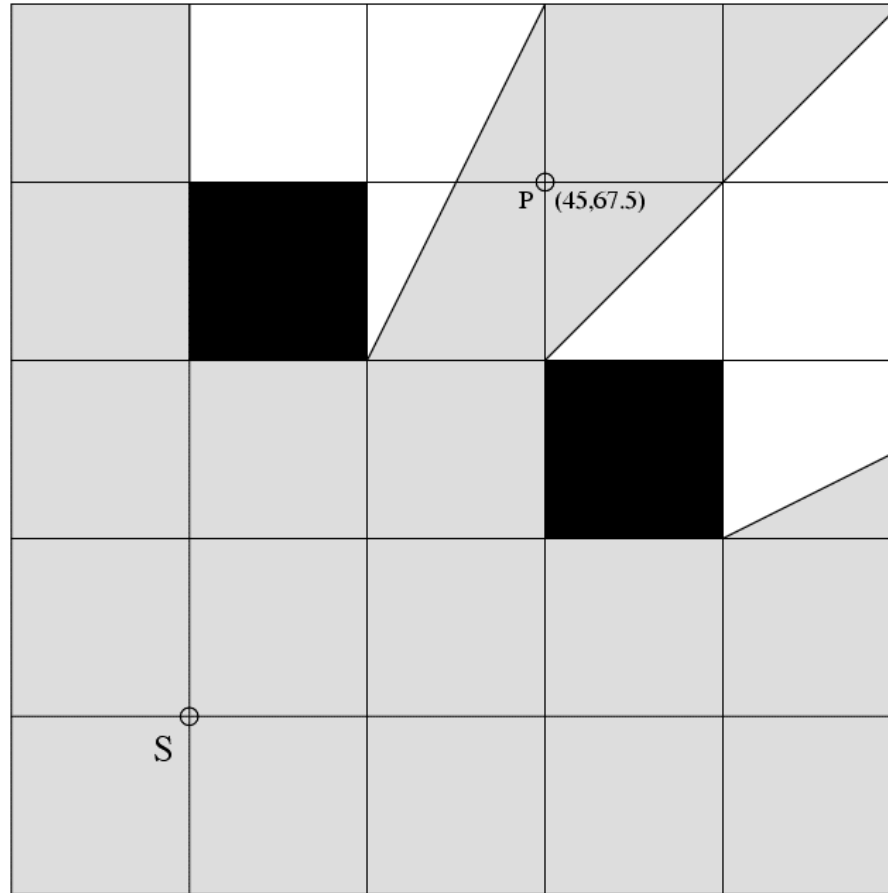




Angle-Propagation Theta*

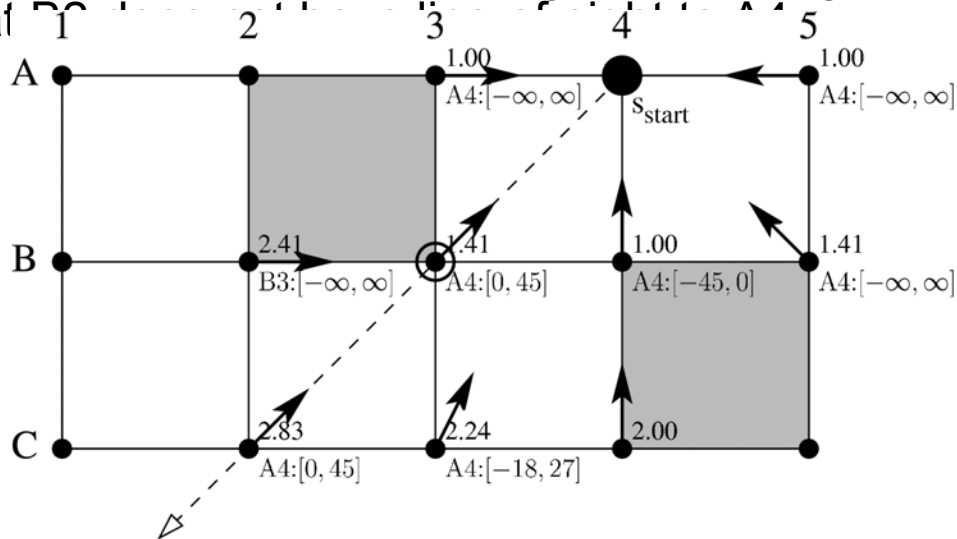
- Basic Theta* is simple and produces good solutions, but it performs many line-of-sight checks which are linear in the number of cells.
- Angle-Propagation Theta* performs the line-of-sight checks in constant time by calculating and propagating angle ranges incrementally.

Shadowing



Angle Ranges

- Angle-Propagation Theta* maintains two additional values for each vertex s : a lower angle bound $lb(s)$ and an upper angle bound $ub(s)$ that together form the angle range $[lb(s), ub(s)]$.
- To explain their meaning, we need to define $\angle(s, p, s')$, which gives Theta* its name. $\angle(s, p, s')$ denotes the angle $\angle(s, p, s')$ in the range $[-180, 180]$.
 - This angle is positive if the ray from p through s' is strictly counterclockwise of the ray from p through s .
- Now consider a vertex s with neighbor s' . By constraining the angle range of s appropriately, Angle-Propagation Theta* maintains the following invariant: If $lb(s) \leq \angle(s, \text{parent}(s), s') \leq ub(s)$ then s' is guaranteed to have line-of-sight to the parent of s .
- For example, (below), $lb(B3) = 0$ and $ub(B3) = 45$. $\angle(B3, A4, C3) = 18$ and thus $C3$ is guaranteed to have line-of-sight to $A4$. On the other hand, $\angle(B3, A4, B2) = -18$ and thus $B2$ is not guaranteed to have line-of-sight to $A4$. Angle-Propagation Theta* thus assumes that



Updating Angle Ranges

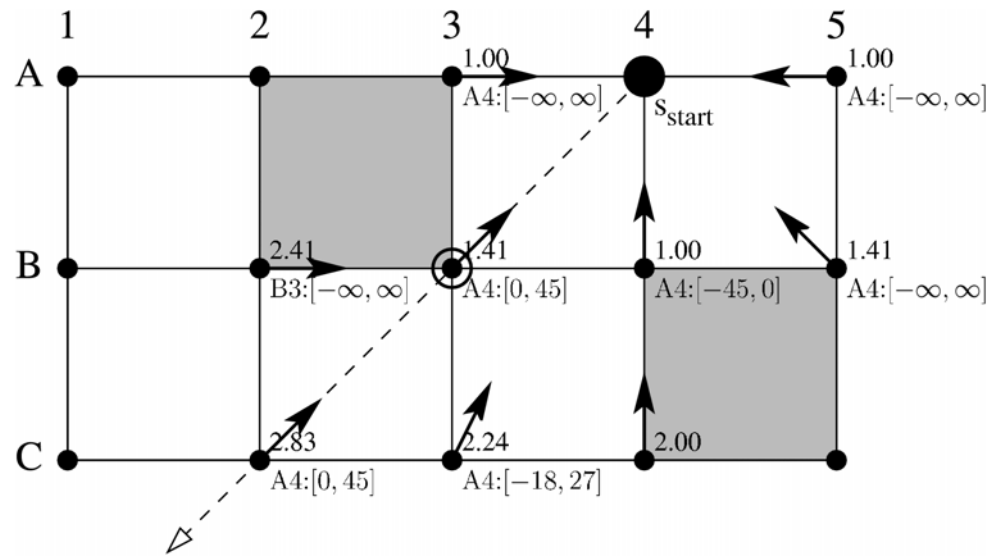
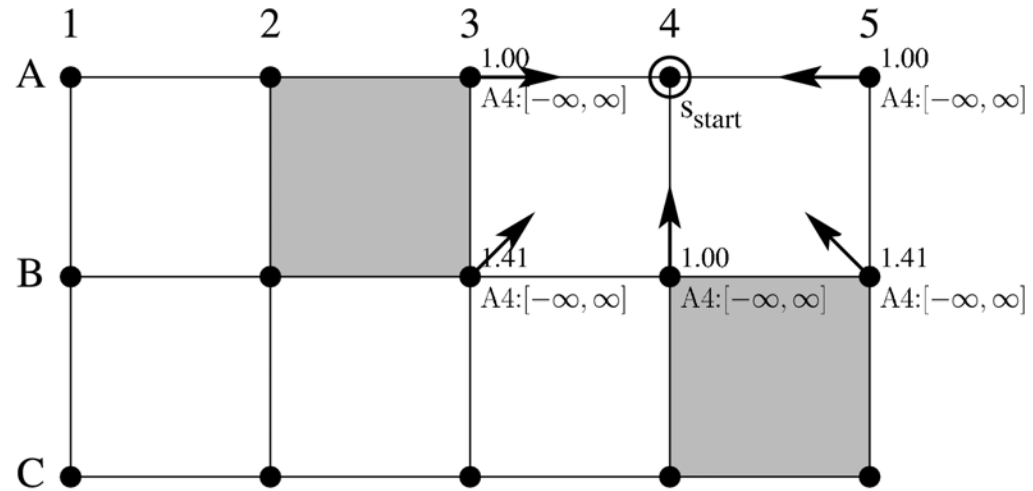
- Before Angle-Propagation Theta* expands a vertex s , it first constrains its angle range based on any blocked cells c of which s is a corner.
 - If moving clockwise from a vertex s on the corner of a cell would obscure line-of-sight, then we set s as a lower bound: $lb(s) = 0$. Likewise for an upperbound. The bound is set to 0, since angles are all relative to the vertices they are stored at.
- Angle-Propagation Theta* also tightens the angle range of vertex s by taking into account its neighbors s' that are either **(1)** unexpanded or **(2)** have parents other than $parent(s)$.
 - Such a neighboring vertex s' has insufficient or no information about line-of-sight to $parent(s)$, which is a problem if s' is closer to $parent(s)$ than s itself. Angle-Propagation Theta* is then forced to make the conservative assumption that there may be blocked cells that s' does not know about, and thus constrains the angle range of s to exclude s' .
 - The resulting angle range may be over-constrained, meaning that it prevents some possible paths which are actually clear of obstacles.

- Example

- The lower and upper angle bound of B3 have been constrained in the bottom figure.

- The lower bound is increased to 0 because of the obstacle above it.

- We decrease the upper bound to 45, because vertex B4 is unexpanded.

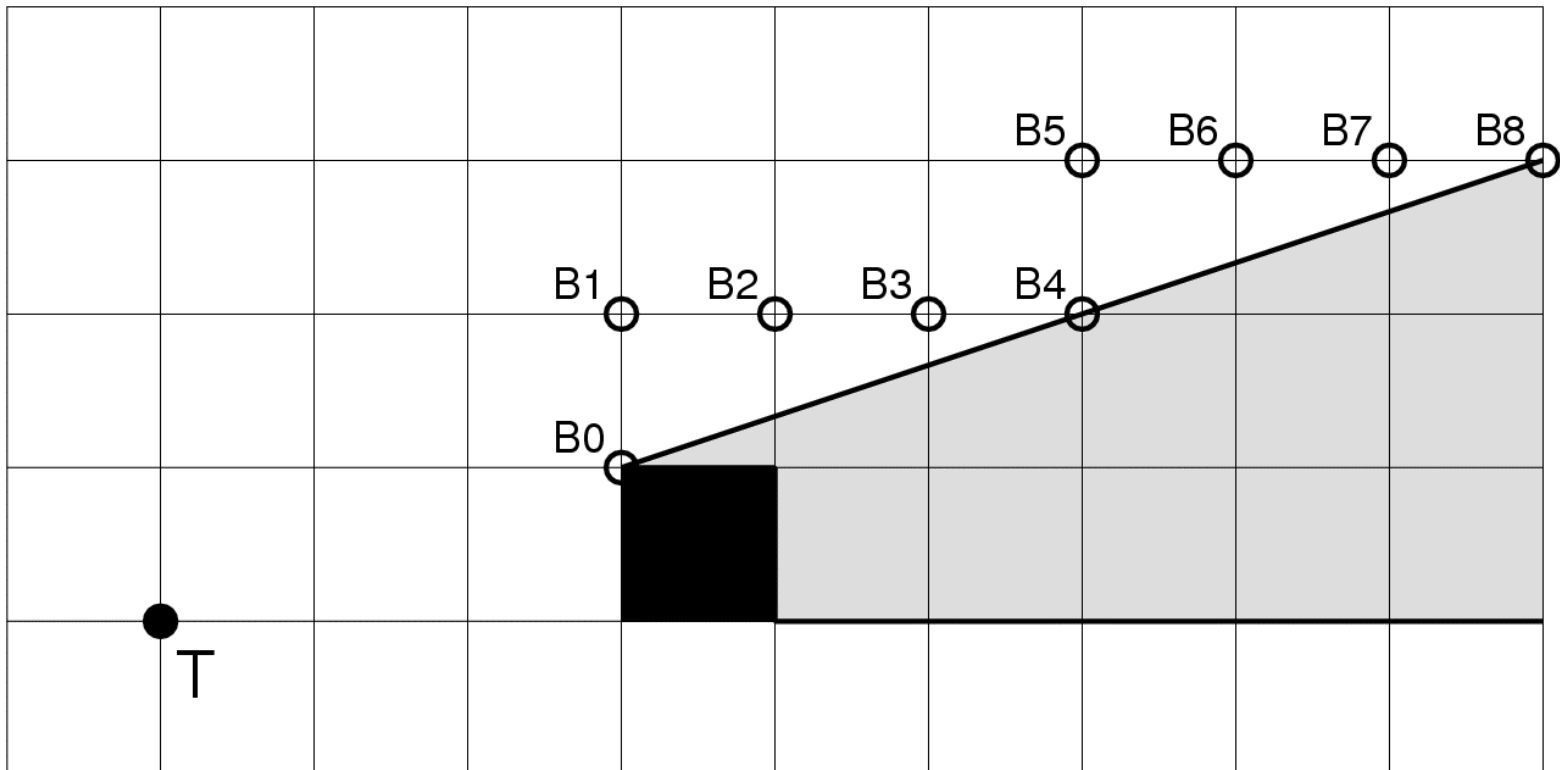


Propagating Angles

- Angle-Propagation Theta* then updates the g-value, parent and angle ranges of an unexpanded neighbor s' of vertex s according to the following three cases:
 - **Case 1:** If s' has the same parent as s , then it has already been set to Path 2, which is shorter than Path 1. Thus, Angle-Propagation Theta* does not need to update the g-value or parent of vertex s' . However, it does tighten the angle range of s' by intersecting it with the angle range of s .
 - **Case 2:** Otherwise, Angle-Propagation Theta* considers the path directly to the parent of s . Since s' inherits its parent from s , it also inherits the angle range of s .
 - **Case 3:** Finally, Angle-Propagation Theta* considers the path directly to vertex s . Since the parent of s' is set to s (a neighbor of s'), the lower and upper angle bounds of s' are initialized to be unbounded.

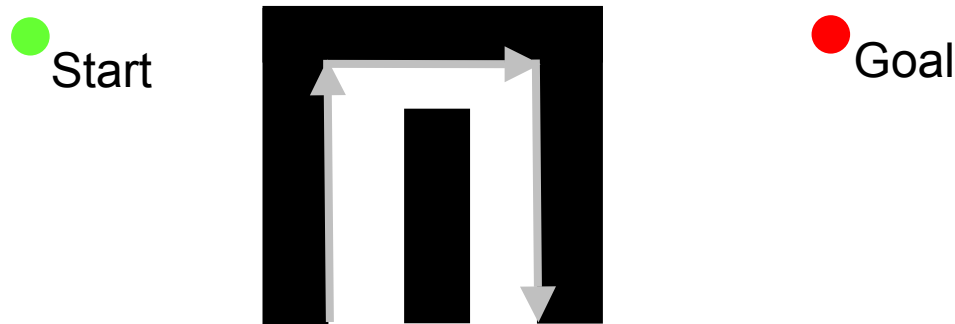
Properties

- We can prove that Angle-Propagation Theta* finds a path from the start vertex to the goal vertex, if such a path exists, simply because it considers all grid edges.
- Furthermore, it is guaranteed that the path that it finds indeed does not pass through any blocked cells.



Angle-Propagation v. Basic

- Angle-Propagation Theta* is faster than Basic Theta* due to its incremental line-of-sight checks, but tends to find slightly longer paths because it loses some line-of-sight information and thus over-constrains the angle ranges, which incorrectly rules out some paths.

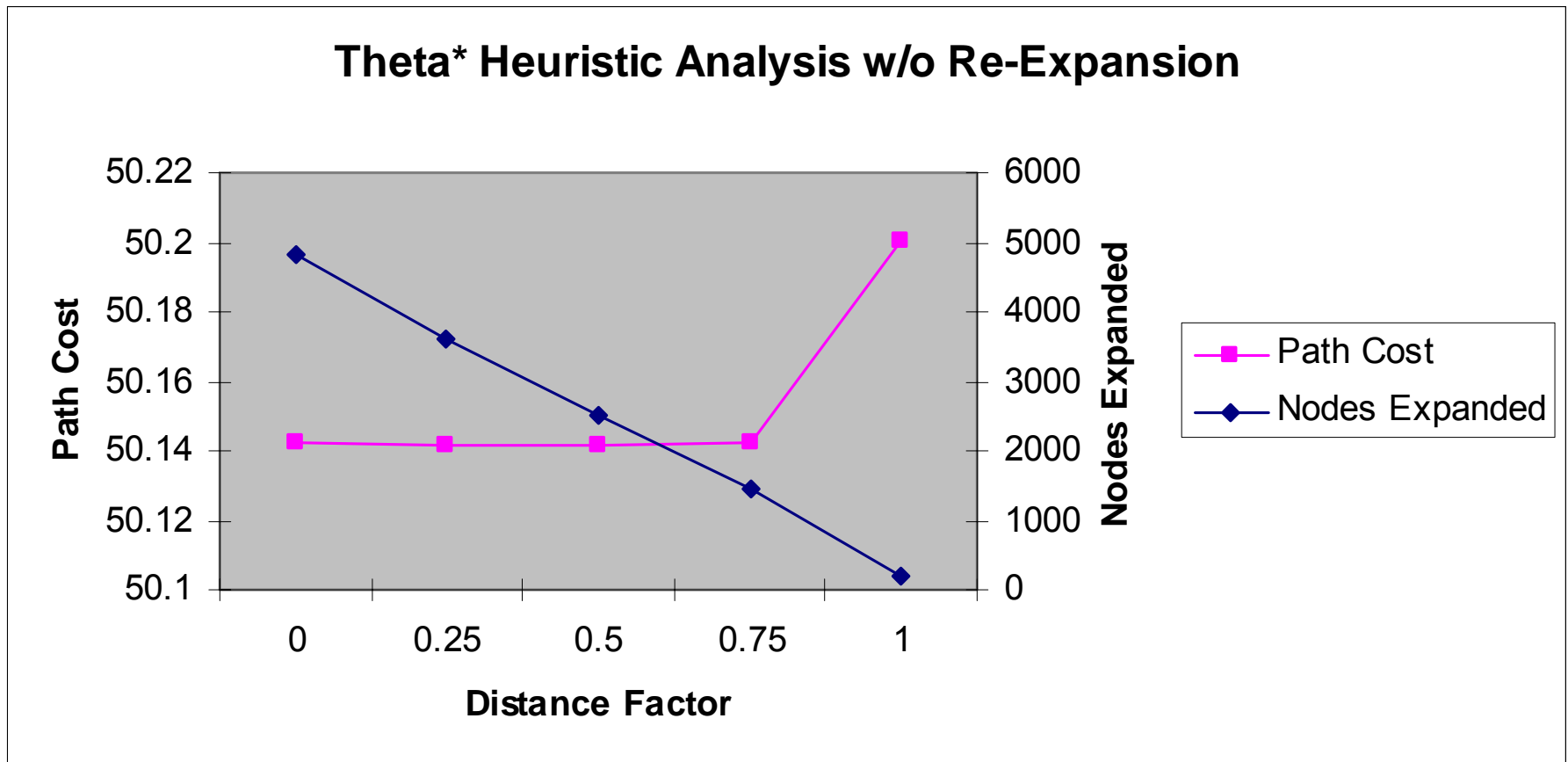


Heuristics

- For A* to perform optimally the $h(s)$ values must be both:
 - Admissible: never over estimates distance to the goal
 - Consistent: obeys triangle inequality
- Theta* heuristic is not consistent because:
 - when Theta* updates the g-value of an unexpanded neighbor s' of vertex s , the g-values of s and s' can be almost identical if the parent of s' is updated according to Path 2.
 - Which means that the f values are not guaranteed to be monotonically non-decreasing over time
- As a result when Theta* expands a vertex the path from the start vertex to that vertex is not guaranteed to be optimal.
 - This means that re-expanding vertices might improve path quality at a cost of increased run time.
- To mitigate this effect, we used Weighted A*
 - w : $h(s) = w \times c(s, s_{\text{goal}})$.
- Path quality is also effected because as w gets closer to 1 nodes are not expanded radially which means there can be incomplete information and thus theta* must over constrain angle ranges and assume cells are blocked

Heuristic Experiments

- A good compromise is achieved for $w = 0.75$.

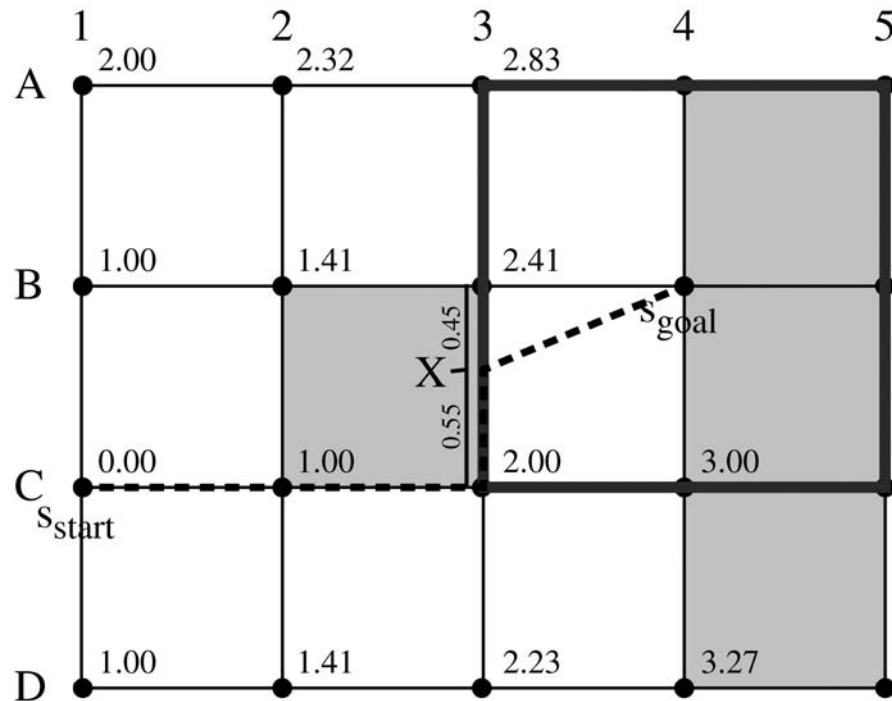


FieldD*

- FieldD* is the only other A* variant which propagated information on a grid without constraining paths to grid edges.

FieldD* errors

- For example, the perimeter of $s' = B4$ is formed by connecting all of the neighbors of $B4$ and shown in bold below. Consider point X on the perimeter, whose neighbors are $B3$ and $C3$. Since g -values are only stored for vertices, the g -value of X is linearly interpolated using $g(B3) = 2.41$ and $g(C3) = 2.00$ to get $g(X) = 0.55 \times 2.41 + 0.45 \times 2.00 = 2.23$. It turns out that X minimizes $g(X') + c(X', B4)$ among all points X' on the perimeter of $B4$ and thus becomes the parent of $B4$.



----- shortest path found by Field D*

Future Research

- Incremental Replanning
- Traversal Costs
- Improving Theta* solutions