# New Ideas for Any-Angle Pathfinding

Daniel D. Harabor

MONASH University

GDC 18th March 2019

# Bona Fiedes

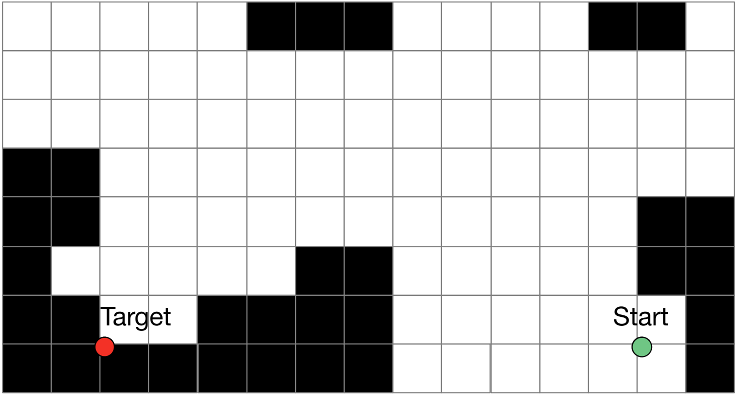
- Senior Research Fellow
- Faculty of Information Technology
- Monash University (Australia)

- Research focus: **pathfinding search**
    - Single agent and multi-agent problems.
    - On grids and navigation meshes.
    - On roads and in public transportation networks.
    - Subject to **constraints**.

http://harabor.net/daniel

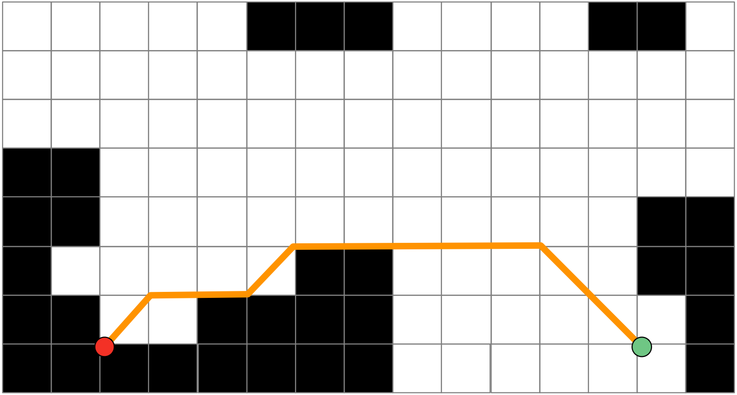
# Any-angle Pathfinding

Find a **Euclidean-path** of minimum cost between two traversable points, on a **grid** or on a **navigation mesh**.



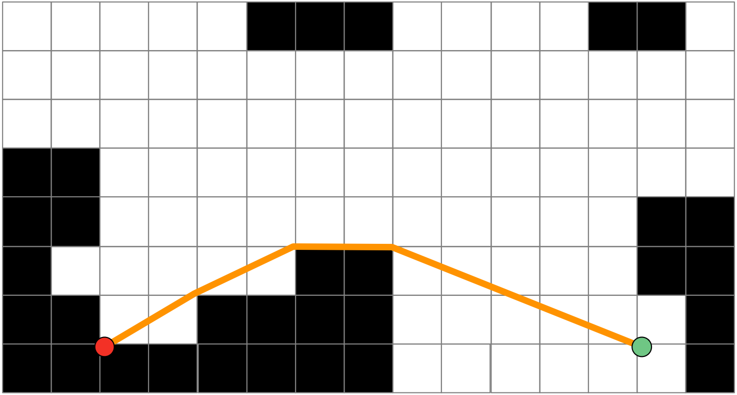
# Any-angle Pathfinding

Find a **Euclidean-path** of minimum cost between two traversable points, on a **grid** or on a **navigation mesh**.



# Any-angle Pathfinding

Find a **Euclidean-path** of minimum cost between two traversable points, on a **grid** or on a **navigation mesh**.



# Any-angle Pathfinding (2)

## Problem

Find a **Euclidean-path** of minimum cost between two traversable points, on a **grid** or on a **navigation mesh**.

## Simplifying assumptions

- Single-size agents
- Two terrain types: traversable and non-traversable.

## Desirable algorithmic properties

- Paths should be short (no detours)
- Paths should be smooth (no unnecessary turns)
- Paths should be computed fast (microseconds, not milliseconds).
- For static and dynamically changing maps
  (i.e. no large precomputes)

# Established ideas for Any-angle Pathfinding

# Established idea #1: String Pulling

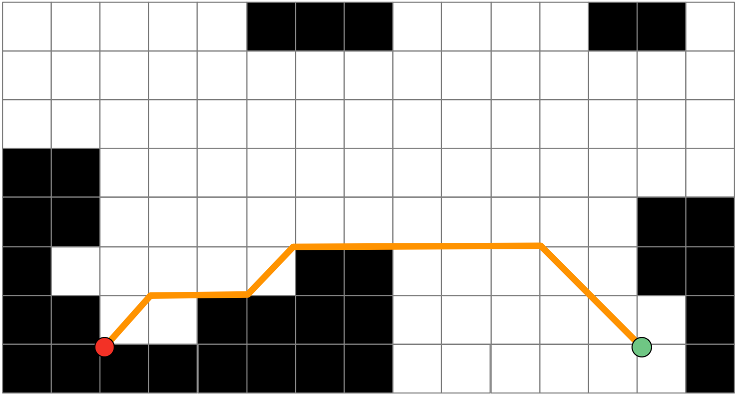
M. Pinter. **Toward More Realistic Pathfinding**. In Game Developer Magazine, 2001.

Compute a grid-optimal path (e.g. using A*) then “smooth” the path to remove unnecessary turns.

# Established idea #1: String Pulling

M. Pinter. **Toward More Realistic Pathfinding**. In Game Developer Magazine, 2001.

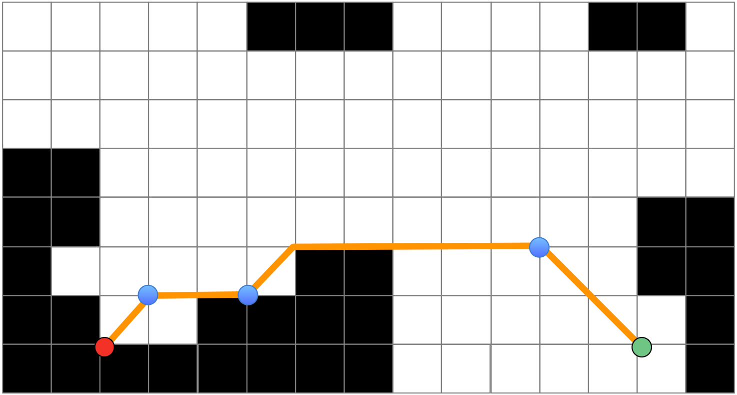
Compute a grid-optimal path (e.g. using A*) then "smooth" the path to remove unnecessary turns.



# Established idea #1: String Pulling

M. Pinter. **Toward More Realistic Pathfinding**. In Game Developer Magazine, 2001.

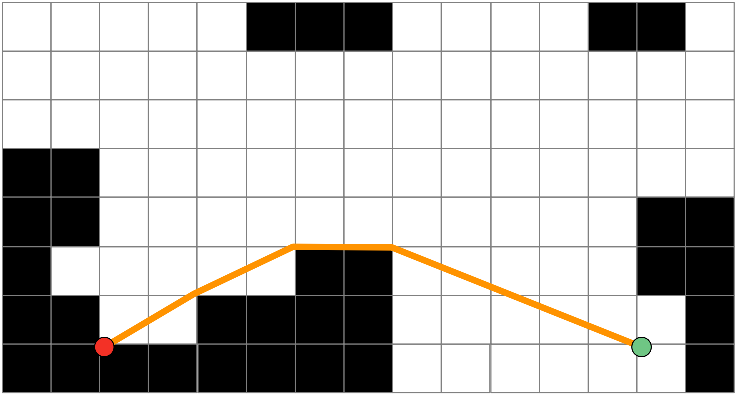
Compute a grid-optimal path (e.g. using A*) then "smooth" the path to remove unnecessary turns.



# Established idea #1: String Pulling

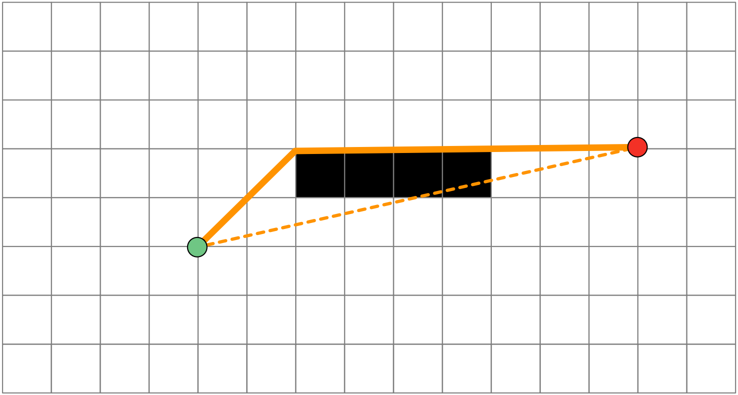
M. Pinter. **Toward More Realistic Pathfinding**. In Game Developer Magazine, 2001.

Compute a grid-optimal path (e.g. using A*) then “smooth” the path to remove unnecessary turns.



# The problem with String Pulling

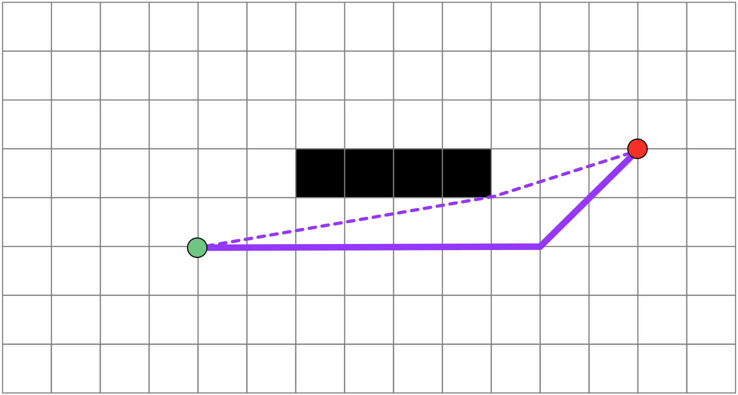
Sometimes the string gets pulled “the wrong way” around an obstacle.



This path is grid optimal and cannot be improved.

# The problem with String Pulling

Sometimes the string gets pulled “the wrong way” around an obstacle.



This path is also grid optimal but can be improved.

# Established idea #2: Theta*

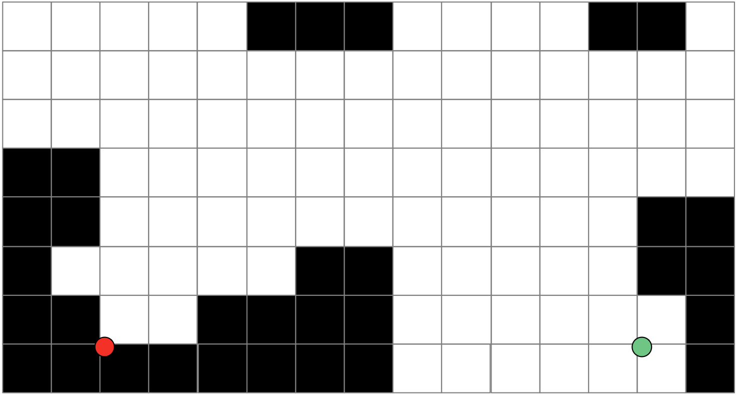
A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

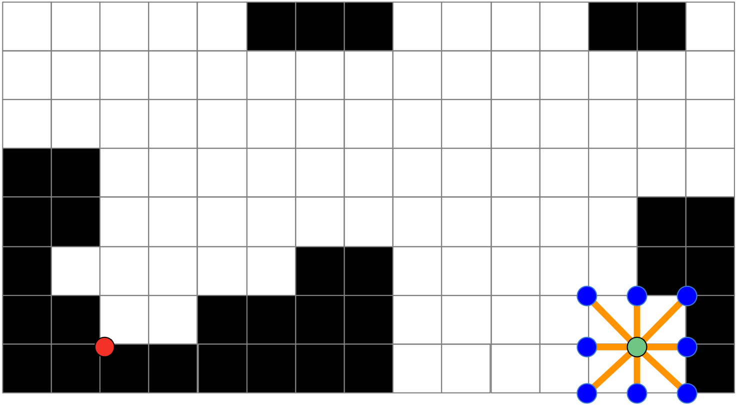


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

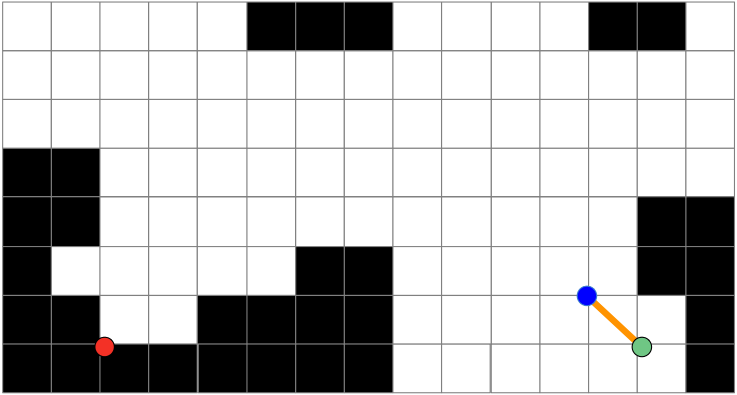


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

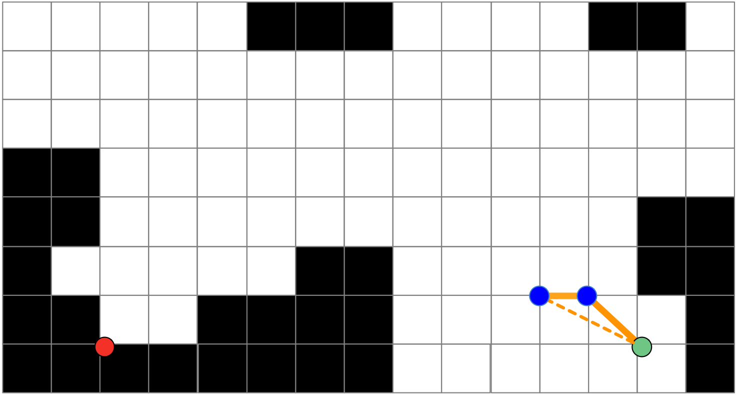


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

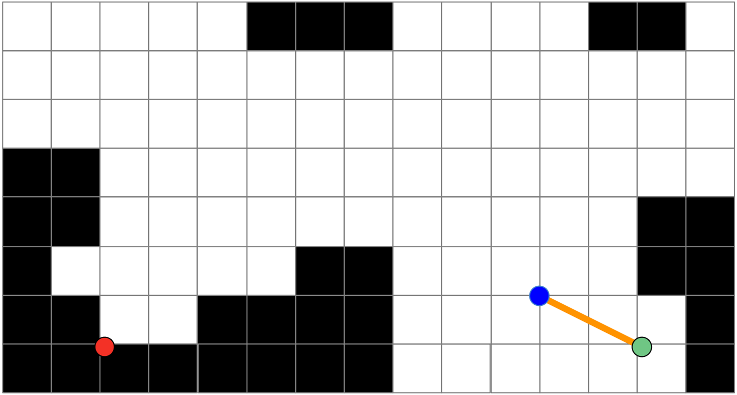


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

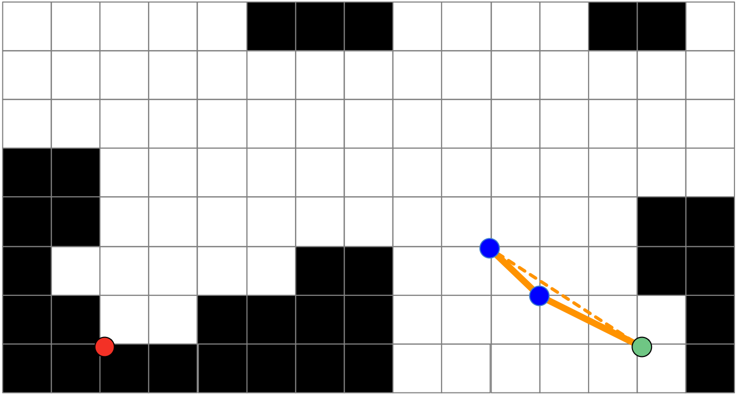


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

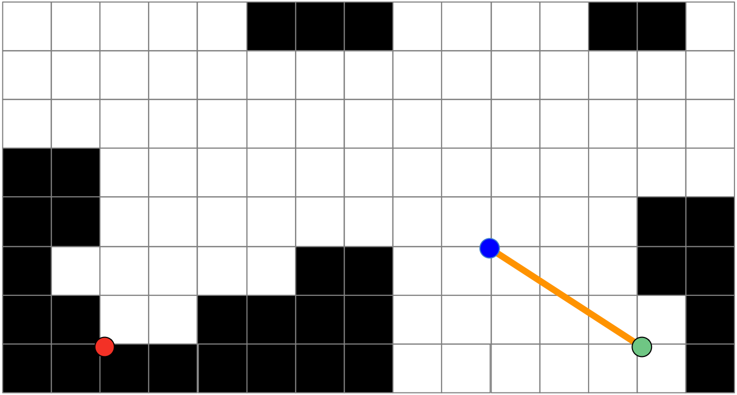


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

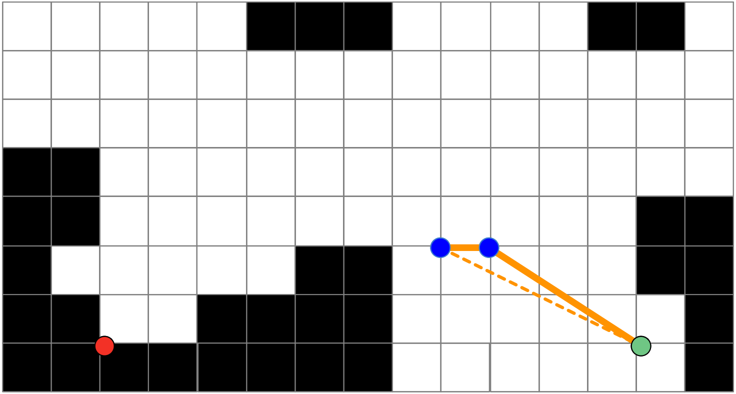


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

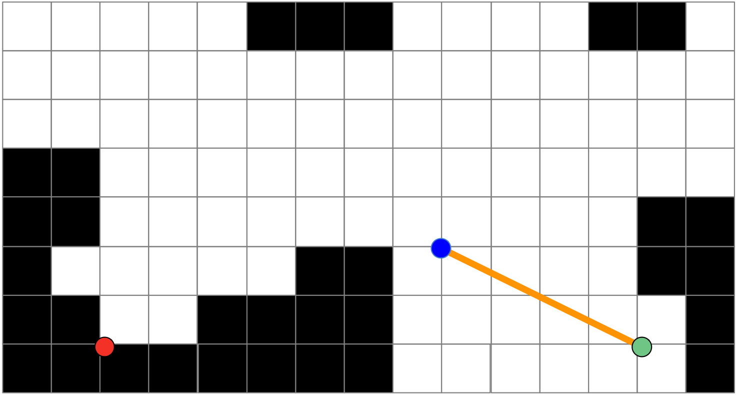


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

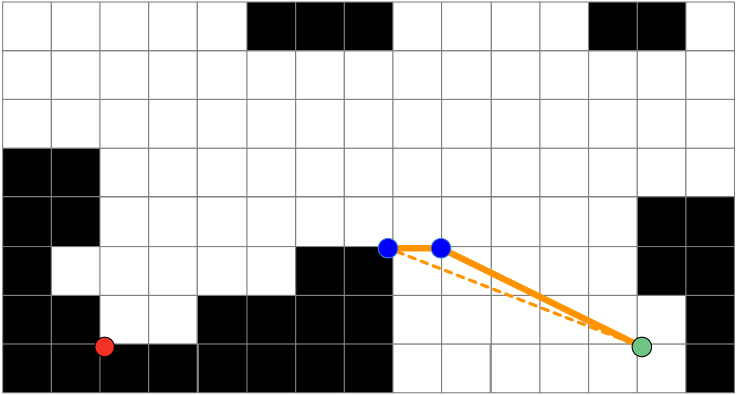


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).

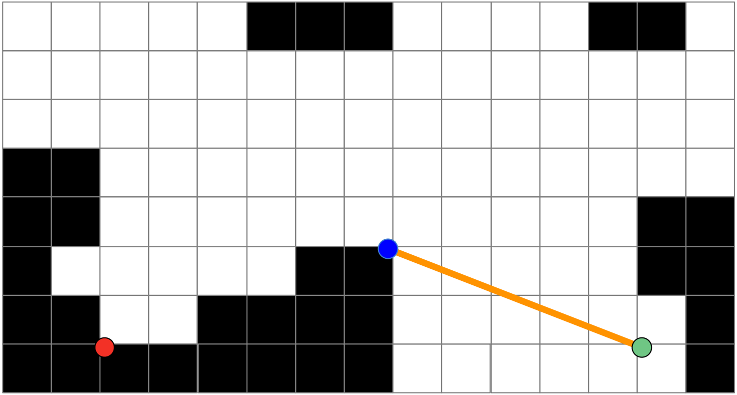


Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).



Only showing selected node expansions

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

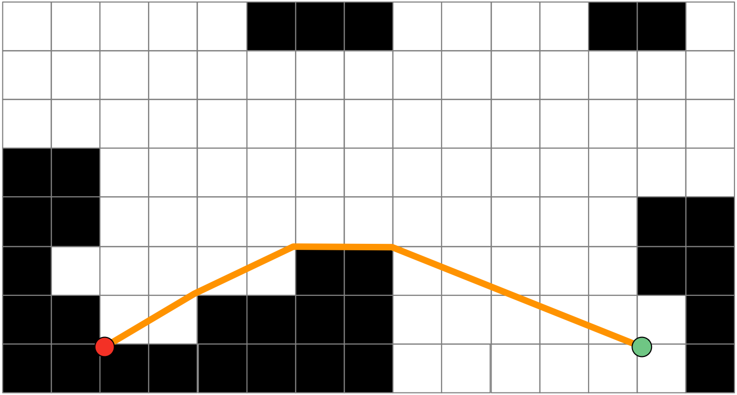
Pull the string during search (instead of post-processing the path).

(and so on, until the target)

# Established idea #2: Theta*

A. Nash *et. al.* **Theta*: Any-Angle Path Planning on Grids**. In AAAI, 2007

Pull the string during search (instead of post-processing the path).



The path returned by Theta*

# The problem with Theta*

## Problem #1

Constantly checking for visibility slows pathfinding search.

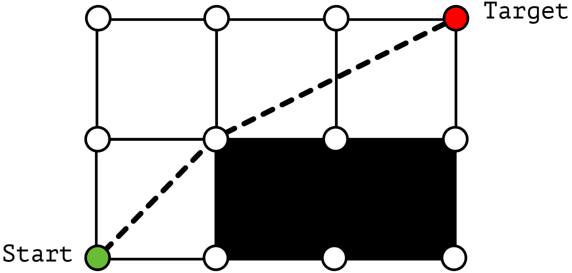
# The problem with Theta*

### Problem #1

Constantly checking for visibility slows pathfinding search.

### Problem #2

Theta* expands nodes out of order and is suboptimal in general.



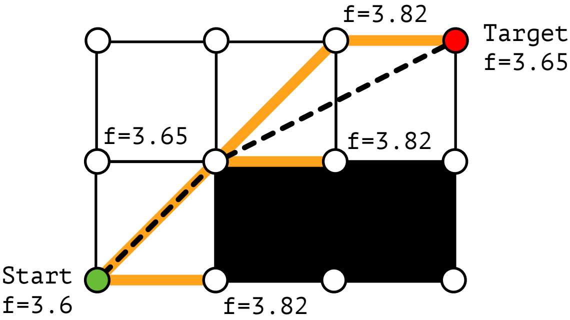
# The problem with Theta*

## Problem #1

Constantly checking for visibility slows pathfinding search.

## Problem #2

Theta* expands nodes out of order and is suboptimal in general.



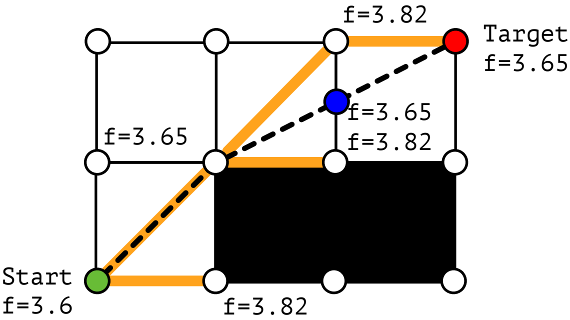
# The problem with Theta*

### Problem #1

Constantly checking for visibility slows pathfinding search.

### Problem #2

Theta* expands nodes out of order and is suboptimal in general.



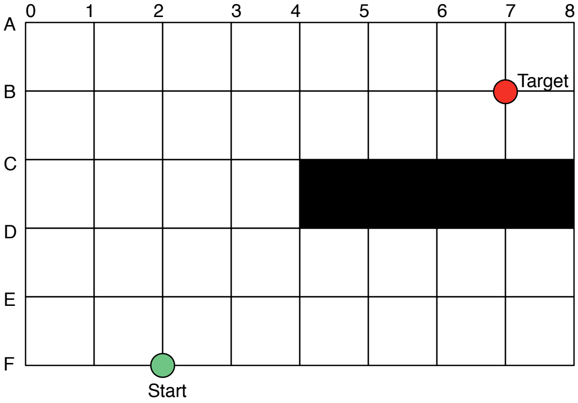
# New Idea #1: Anya

Daniel D. Harabor and Alban Grastien. **An Optimal Any-Angle Pathfinding Algorithm**. In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2013.

Daniel D. Harabor, Alban Grastien, Dindar Öz , and Vural Aksakalli. **Optimal Any-angle Pathfinding in Practice**. Journal of Artificial Intelligence Research, Vol 56 Issue 1, pp89–118, May 2016.

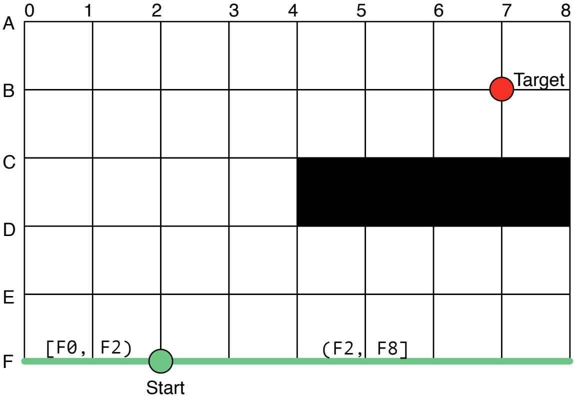
# Anya in broad strokes

Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



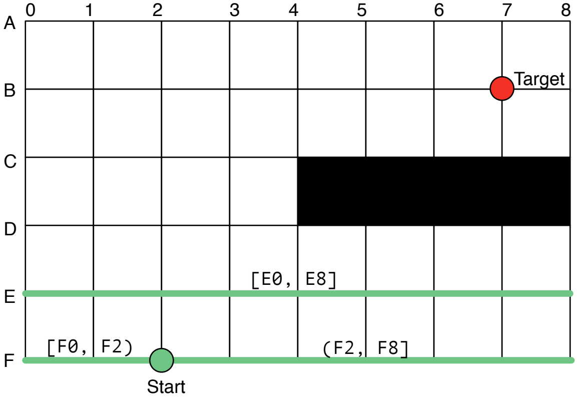
# Anya in broad strokes

Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



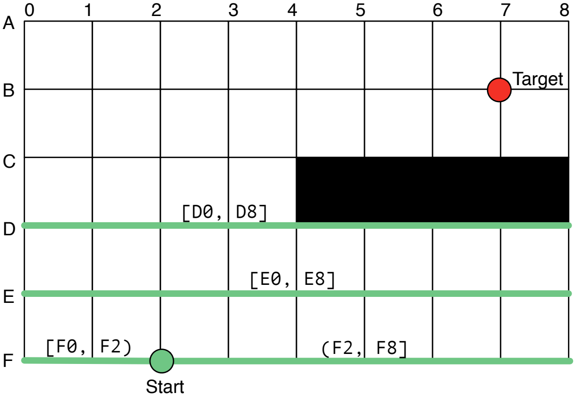
# Anya in broad strokes

Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



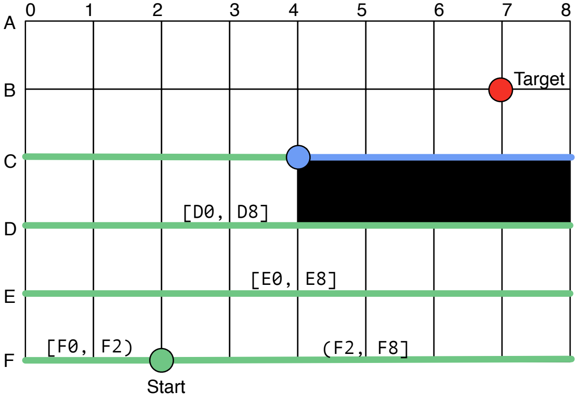
# Anya in broad strokes

Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



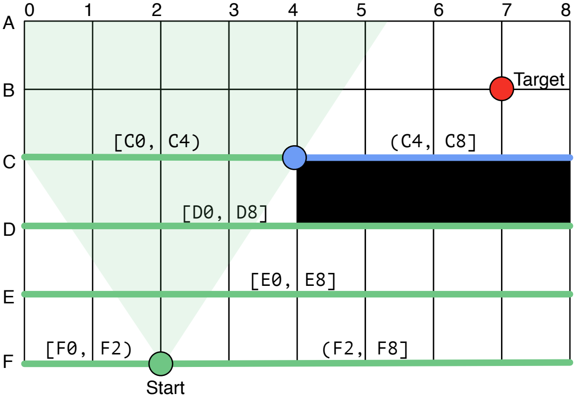
# Anya in broad strokes

Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



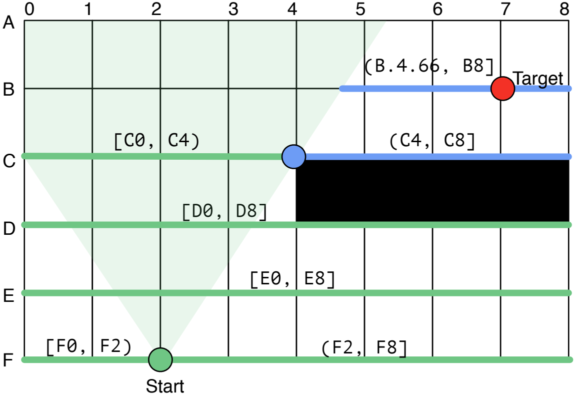
# Anya in broad strokes

Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



# Anya in broad strokes

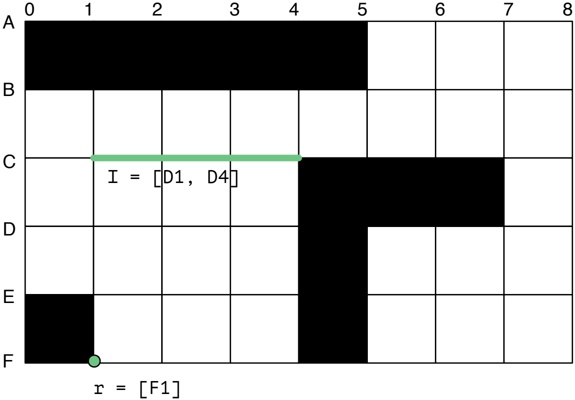
Anya is a **fast**, **optimal** and **online** algorithm for any-angle pathfinding on a grid. It works by expanding *sets* of nodes together at one time.



# Definition #1: Search Nodes

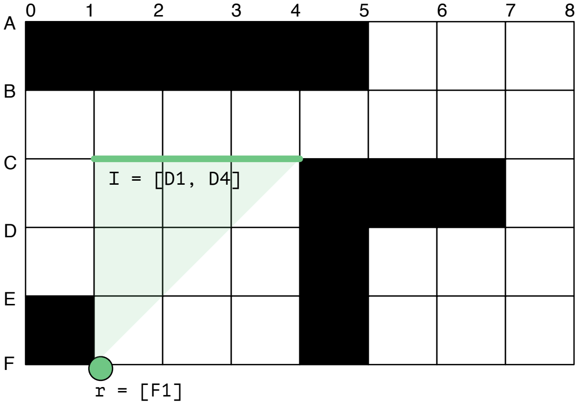
Every node is a tuple $(I, r)$ where:

- $r$ is a *root*; the most recent turning point.
- $I$ is an interval of contiguous points, all visible from $r$.
- The *start node* has a point interval and a root "off the grid"



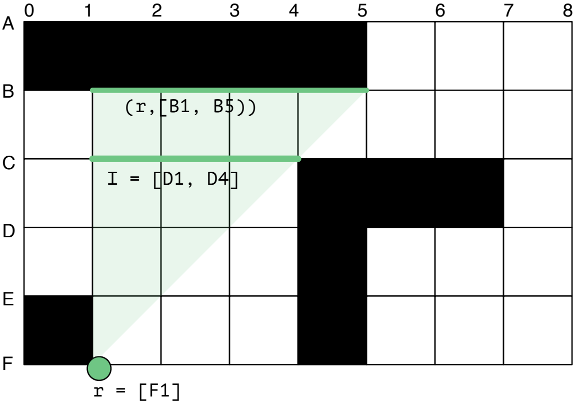
# Definition #2: Successors

- Successors of node $(I, r)$ are found by travelling from $r$ and through $I$ along a locally taut path.
- Two kinds of successors: *observable* and *non-observable*



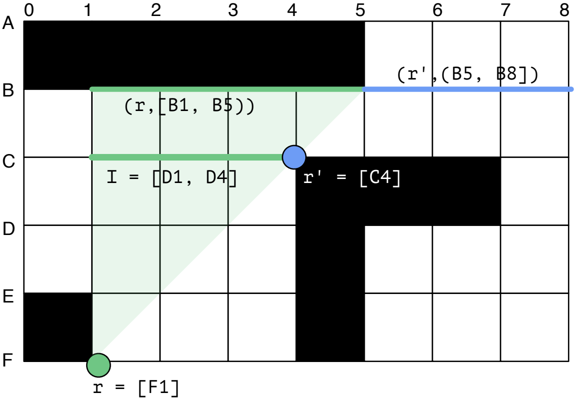
# Definition #2: Successors

- Successors of node $(I, r)$ are found by travelling from $r$ and through $I$ along a locally taut path.
- Two kinds of successors: *observable* and *non-observable*



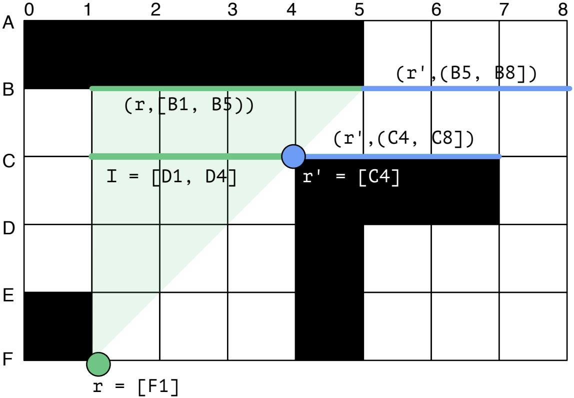
# Definition #2: Successors

- Successors of node $(I, r)$ are found by travelling from $r$ and through $I$ along a locally taut path.
- Two kinds of successors: *observable* and *non-observable*



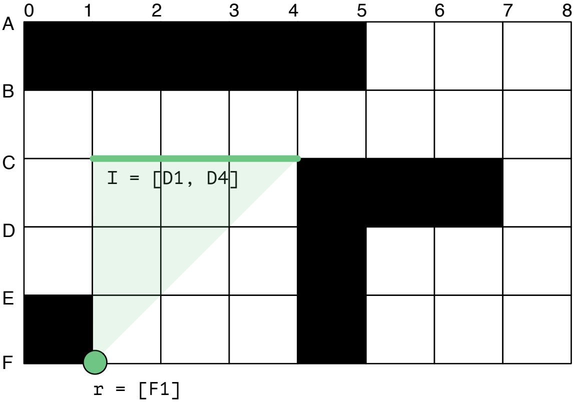
# Definition #2: Successors

- Successors of node $(I, r)$ are found by travelling from $r$ and through $I$ along a locally taut path.
- Two kinds of successors: *observable* and *non-observable*



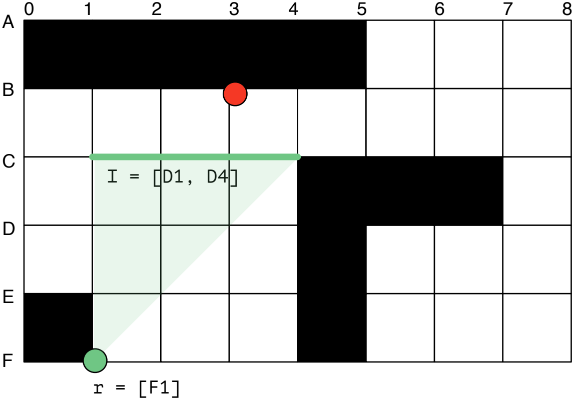
# Evaluation Function

- From each interval $I$ we choose a single point $p$ which minimises the cost-to-go (i.e. the $f$-value of the node).



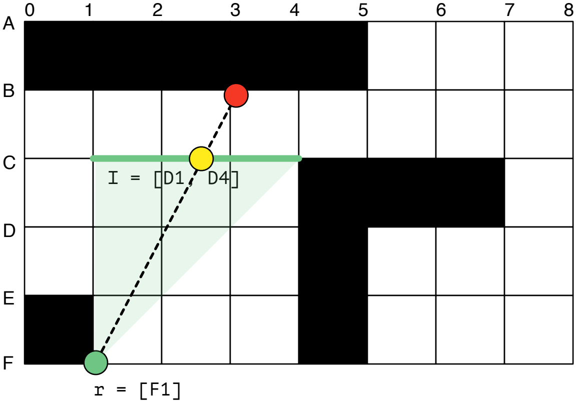
# Evaluation Function

- From each interval *I* we choose a single point *p* which minimises the cost-to-go (i.e. the *f*-value of the node).



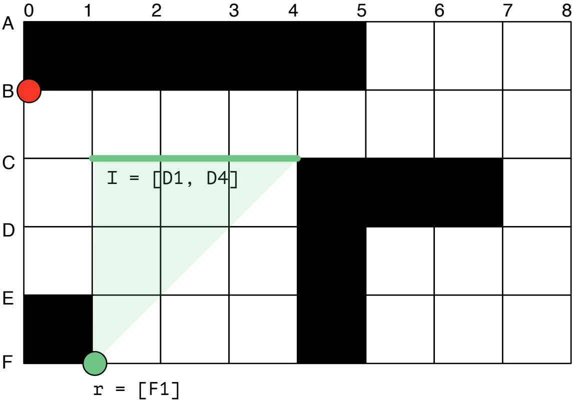
# Evaluation Function

- From each interval *I* we choose a single point *p* which minimises the cost-to-go (i.e. the *f*-value of the node).



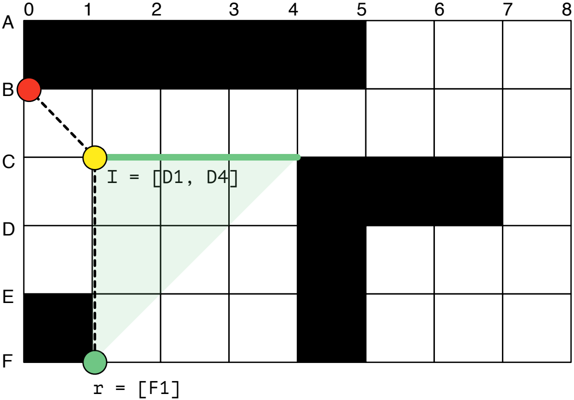
# Evaluation Function

- From each interval $I$ we choose a single point $p$ which minimises the cost-to-go (i.e. the $f$-value of the node).



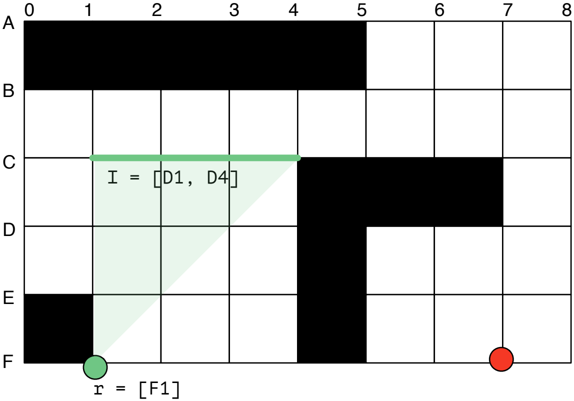
# Evaluation Function

- From each interval $I$ we choose a single point $p$ which minimises the cost-to-go (i.e. the $f$-value of the node).



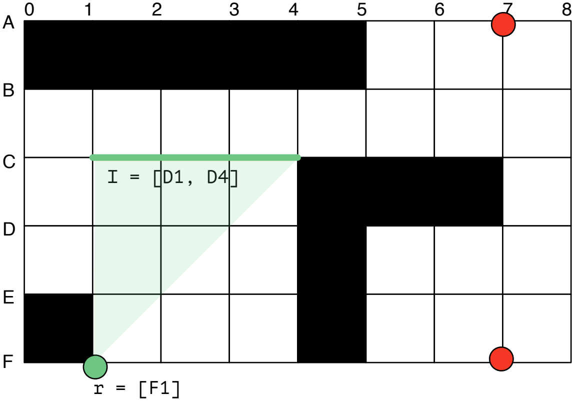
# Evaluation Function

- From each interval $I$ we choose a single point $p$ which minimises the cost-to-go (i.e. the $f$-value of the node).



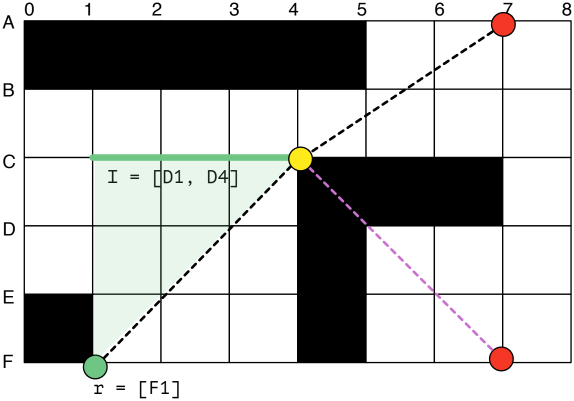
# Evaluation Function

- From each interval $I$ we choose a single point $p$ which minimises the cost-to-go (i.e. the $f$-value of the node).



# Evaluation Function

- From each interval *I* we choose a single point *p* which minimises the cost-to-go (i.e. the *f*-value of the node).



# Theoretical properties

## Completeness (Sketch)

- Every point is a corner or belongs to an interval.
- Every interval is visible from some predecessor.

## Optimality (Sketch)

- Each representative point has a minimum $f$-value.
- The $f$-value of each successor is monotonically increasing.
- A node whose interval contains the target is eventually expanded.

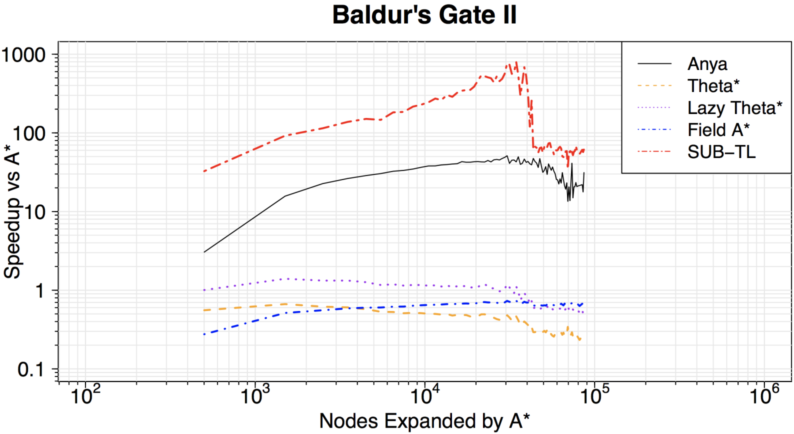
## Online

Each search is performed entirely online and without reference to any pre-computed data structures or heuristics.

Full technical details in the 2016 JAIR paper!

# Results on Games Maps

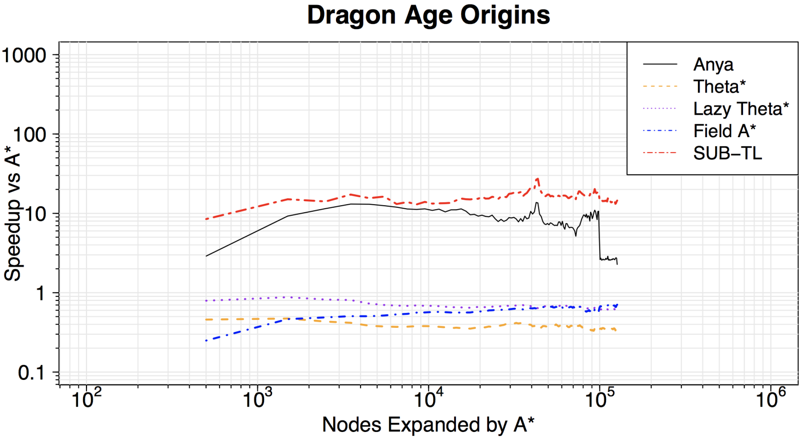
Speedup (time) vs grid A* on a range of game benchmarks appearing in Nathan Sturtevant's repository at `http://movingai.com`.



75 maps, 93,160 problem instances.

# Results on Games Maps

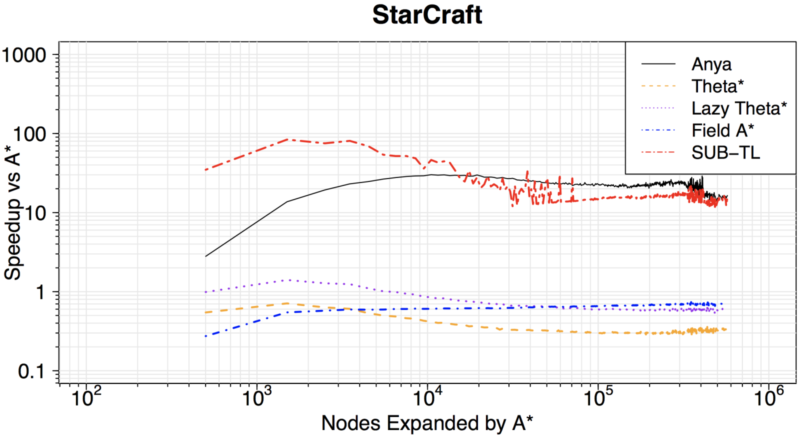
Speedup (time) vs grid A* on a range of game benchmarks appearing in Nathan Sturtevant's repository at `http://movingai.com`.



156 maps, 159,465 problem instances.

# Results on Games Maps

Speedup (time) vs grid A* on a range of game benchmarks appearing in Nathan Sturtevant's repository at `http://movingai.com`.



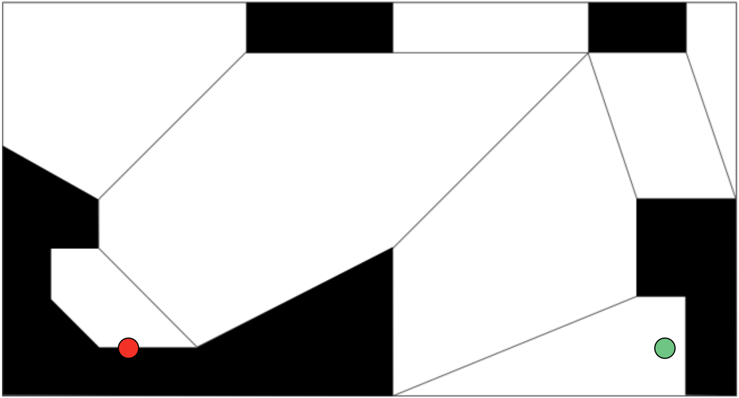
75 maps, 198,230 problem instances.

# New Idea #2: Polyanya

Michael L. Cui, Daniel D. Harabor, and Alban Grastien.
**Compromise-free Pathfinding on a Navigation Mesh**. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2017.

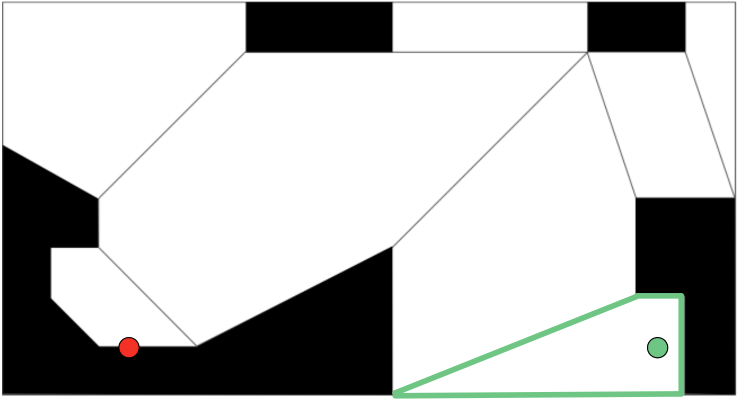
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



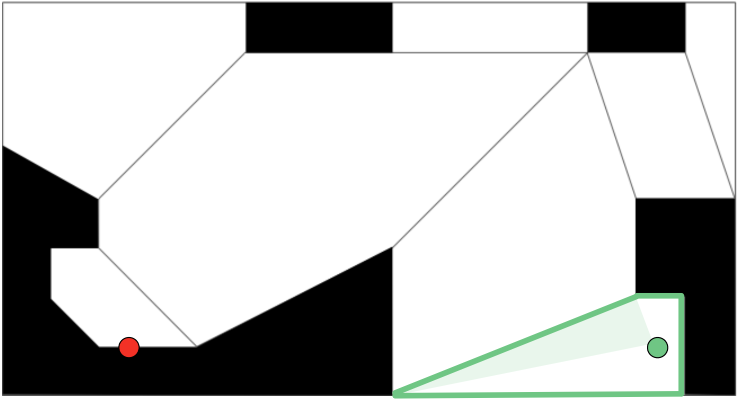
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



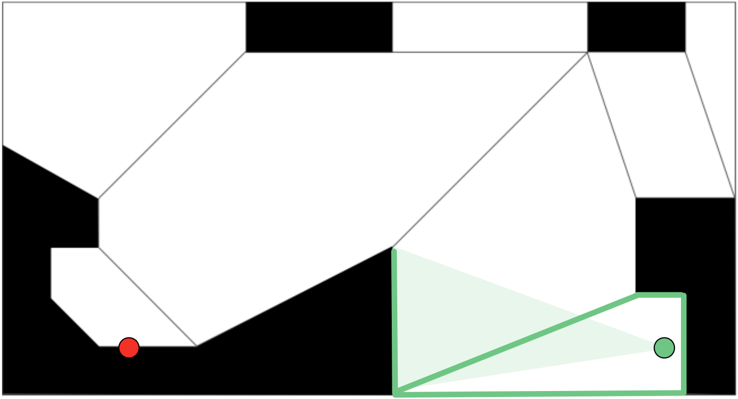
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



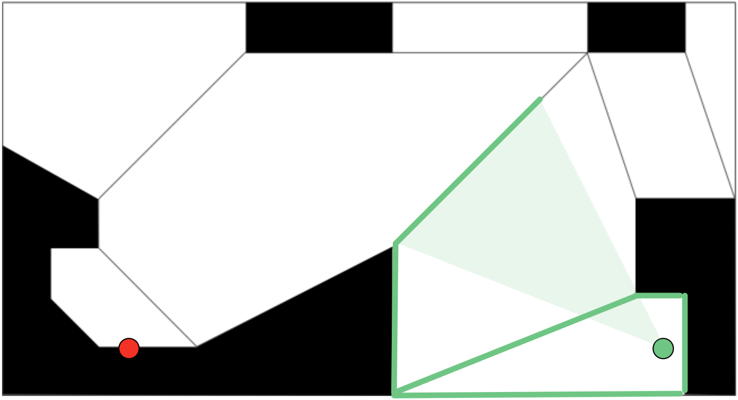
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



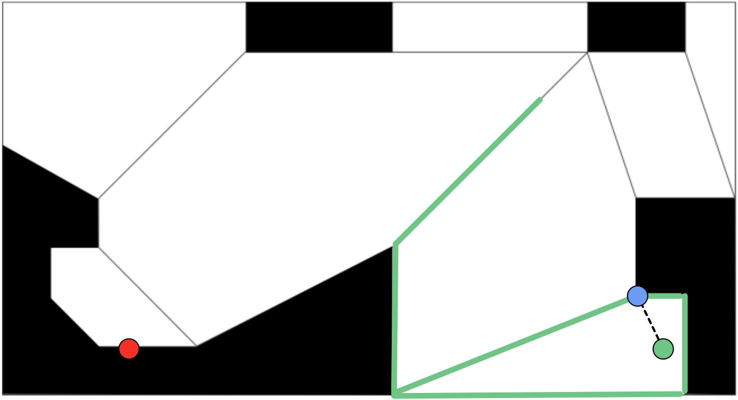
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



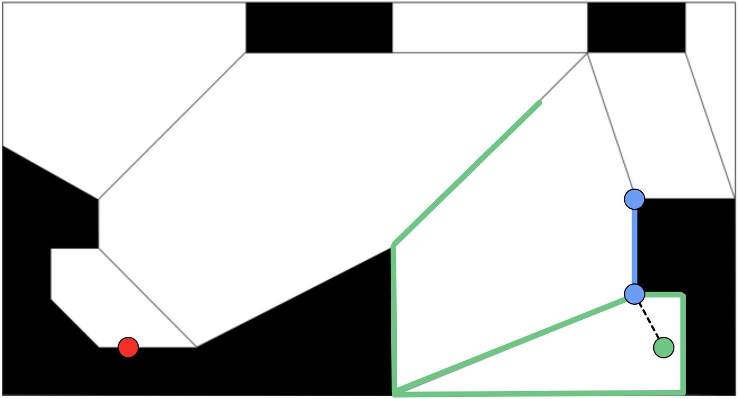
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



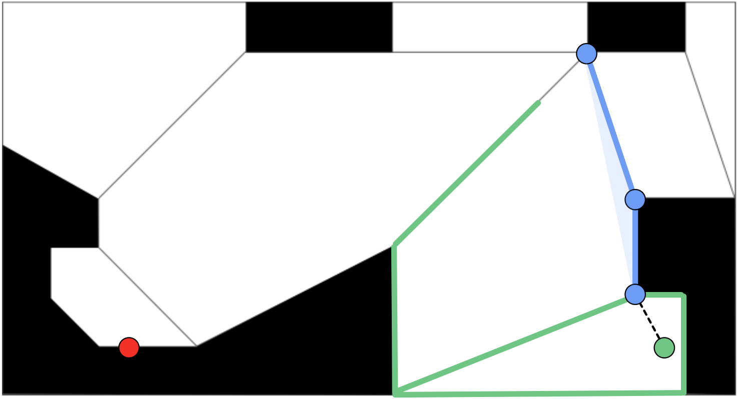
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



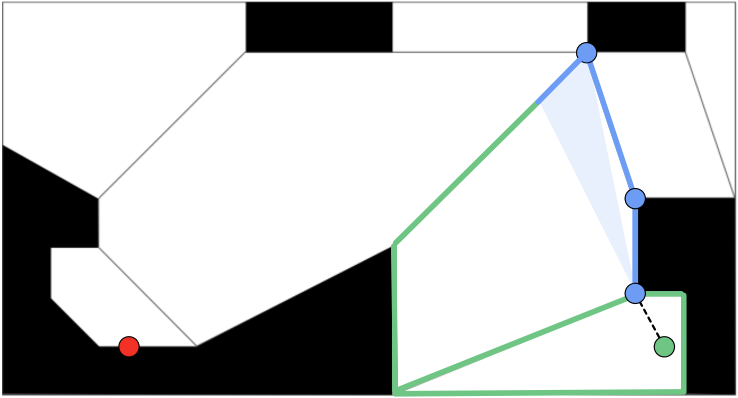
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



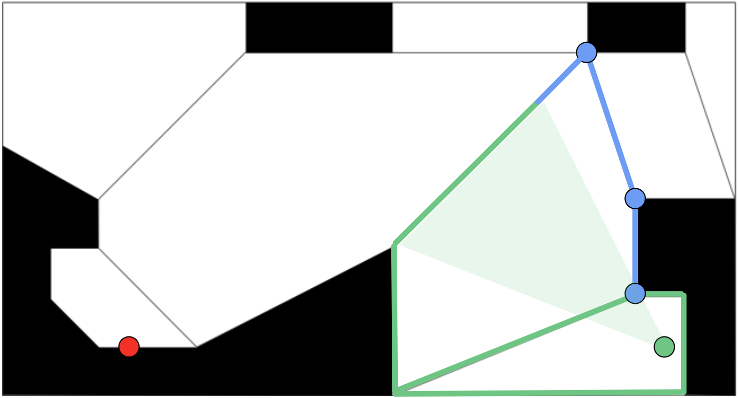
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



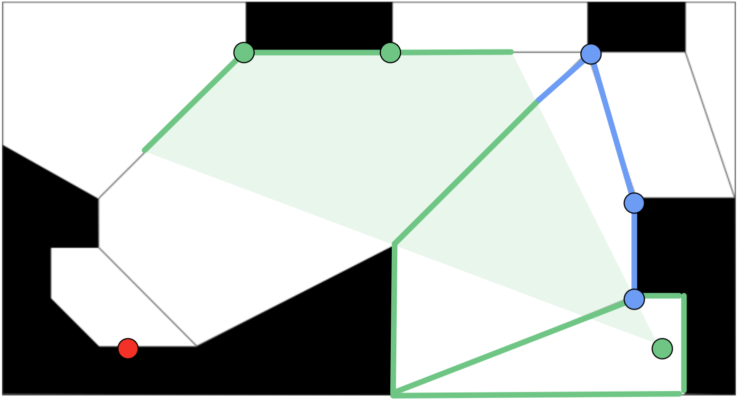
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



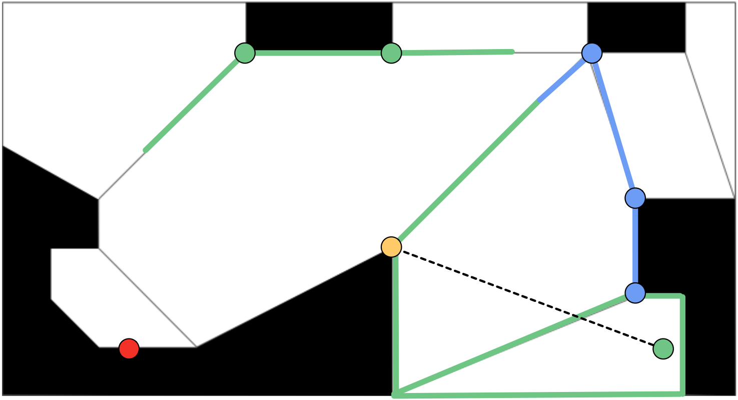
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



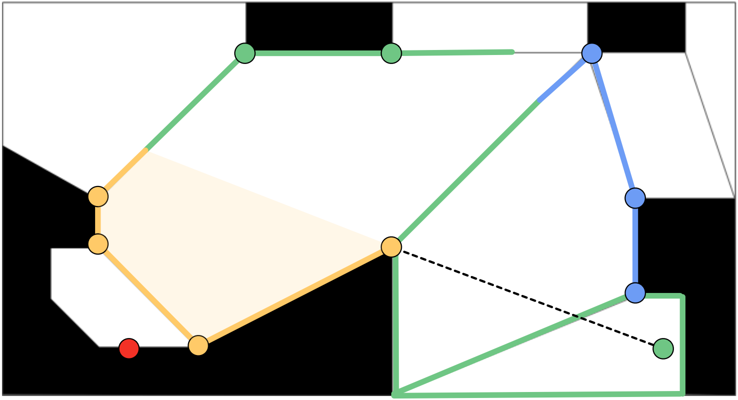
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



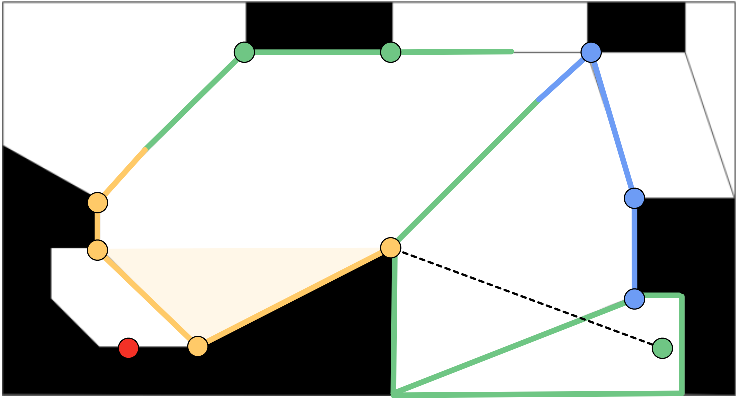
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



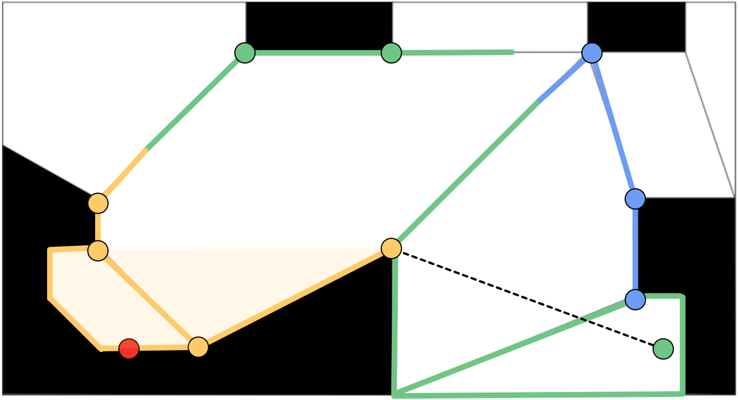
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



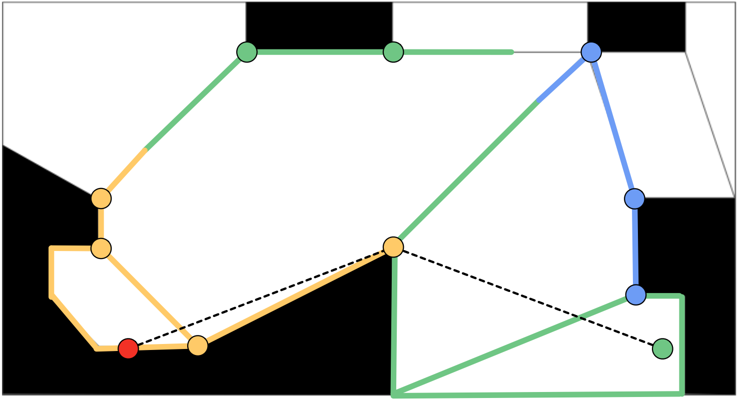
# Polyanya in broad strokes

Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



# Polyanya in broad strokes

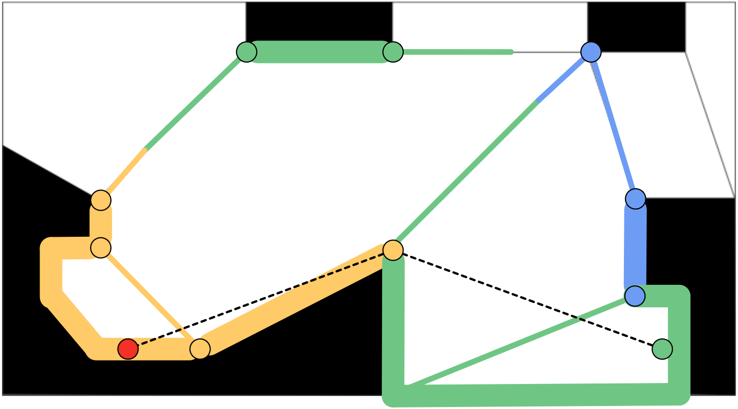
Polyanya is an optimal algorithm that extends and generalises Anya, from grids to navigation meshes.



# Further optimisations

## Dead-end pruning

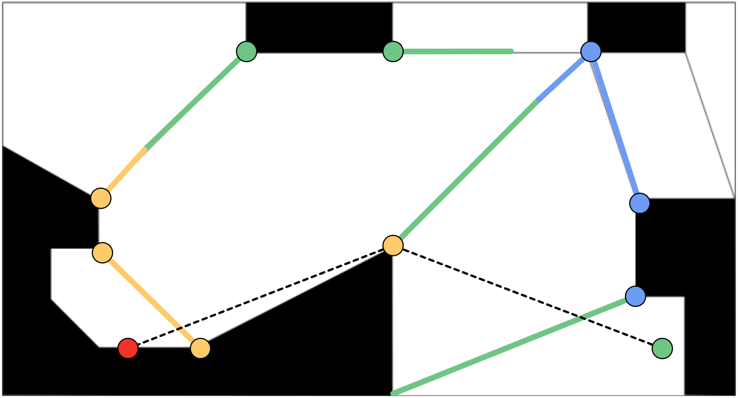
Prune all nodes that “push” into obstacles or into polygons that have only one entry edge.



# Further optimisations

## Dead-end pruning

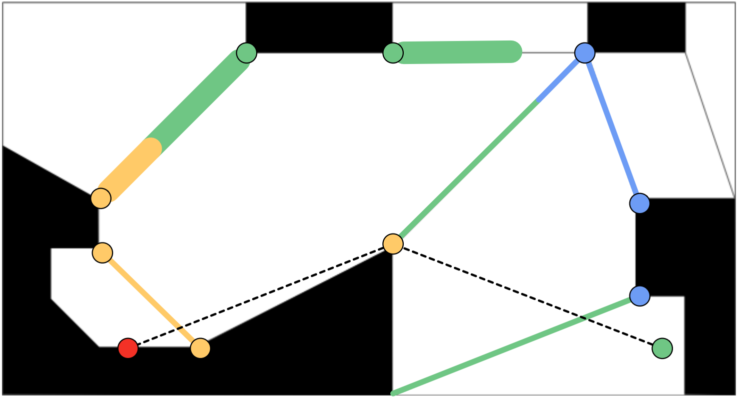
Prune all nodes that “push” into obstacles or into polygons that have only one entry edge.



# Further optimisations

## Dead-end pruning

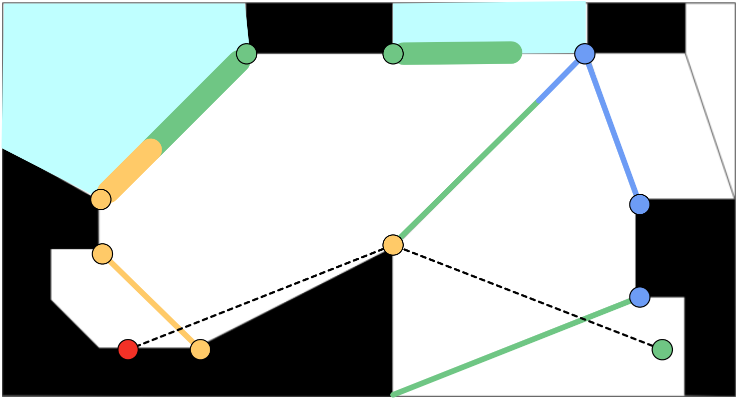
Prune all nodes that "push" into obstacles or into polygons that have only one entry edge.



# Further optimisations

## Dead-end pruning

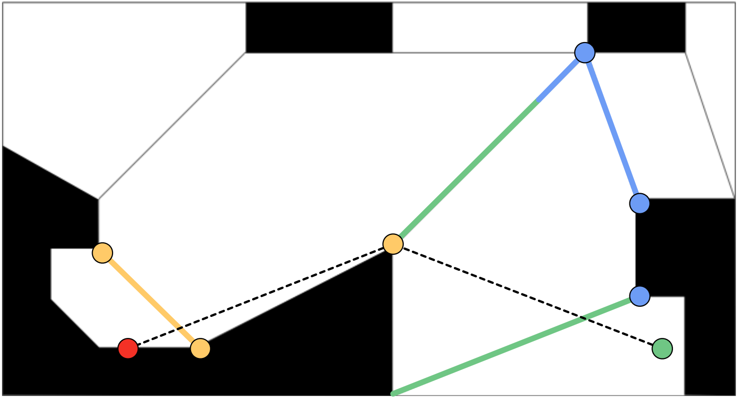
Prune all nodes that “push” into obstacles or into polygons that have only one entry edge.



# Further optimisations

## Dead-end pruning

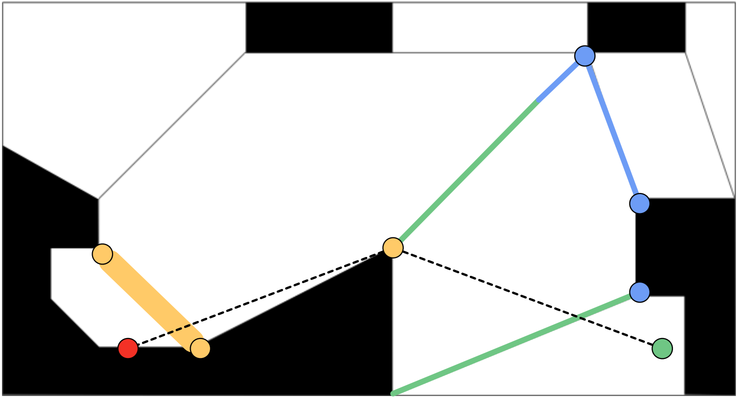
Prune all nodes that “push” into obstacles or into polygons that have only one entry edge.



# Further optimisations

## Intermediate Pruning

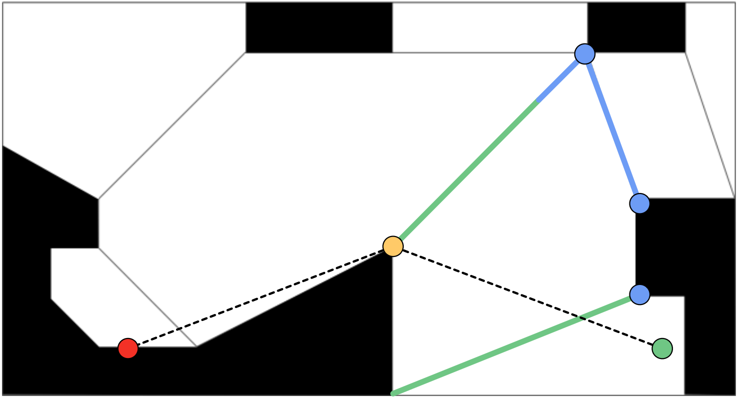
Immediately and recursively expand any node that has only a single successor.



# Further optimisations

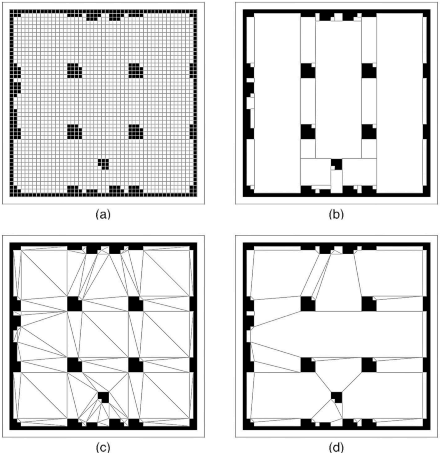
## Intermediate Pruning

Immediately and recursively expand any node that has only a single successor.

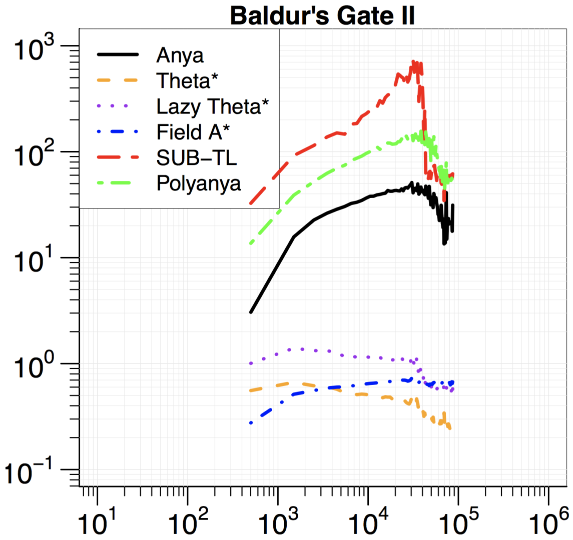


# Mesh selection

We generted a variety of meshes including: grids, rectangles, Constrained Delaunay Triangulations (CDT), and greedily merged CDTs. **Bigger polys means better performance**



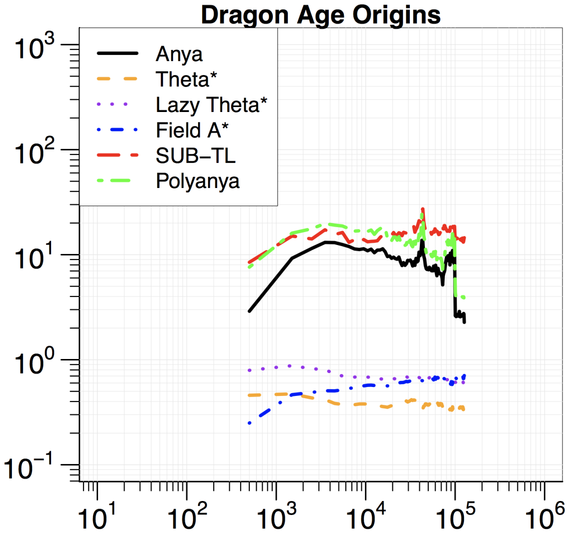
# Results on Game Maps



**y-axis:**
speedup (time) vs grid A*.

**x-axis:**
problem instances, ordered by difficulty (measured as node expansions required by grid A*).

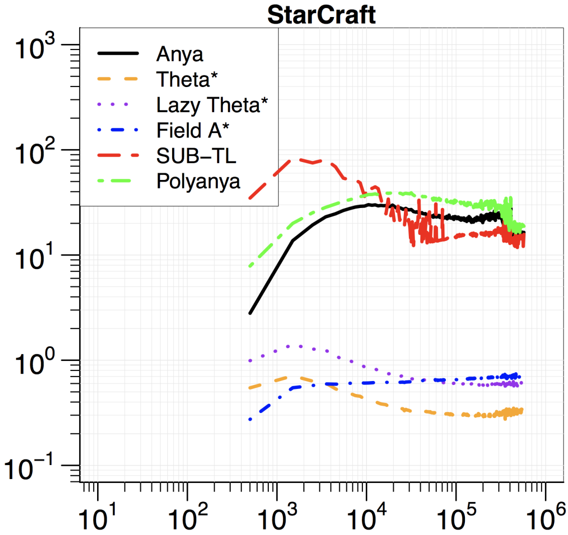
# Results on Game Maps



**y-axis:**
speedup (time) vs grid A*.

**x-axis:**
problem instances, ordered by difficulty (measured as node expansions required by grid A*).

# Results on Game Maps



**y-axis:**
speedup (time) vs grid A*.

**x-axis:**
problem instances, ordered by difficulty (measured as node expansions required by grid A*).

# Wrap Up

## Any-angle pathfinding has come a long way!

- Performance has increased dramatically.
- We're making fewer tradeoffs.
- We can now solve a much broader range of problems.

## But more work is needed!

- Kinematic constraints remain challenging.
- Weighted terrains remain challenging.
- 3D pathfinding and flying AI.

It's not yet clear to what degree new algorithms like Anya and Polyanya can help improve the state-of-the-art in these areas.

For more info (including papers and links to experimental source code) please visit my homepage at `http://harabor.net/daniel`.