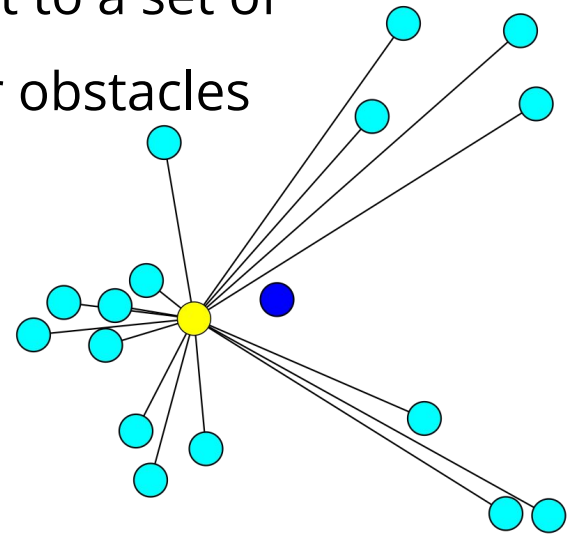
Star Topology Algorithms for Weighted Regions and Obstacles

Tyler King

June 24th, 2021

Overarching Goal

- Computing minimum paths from one point to a set of external points across weighted regions or obstacles
  - Solved for empty space
    - Weiszfeld's algorithm



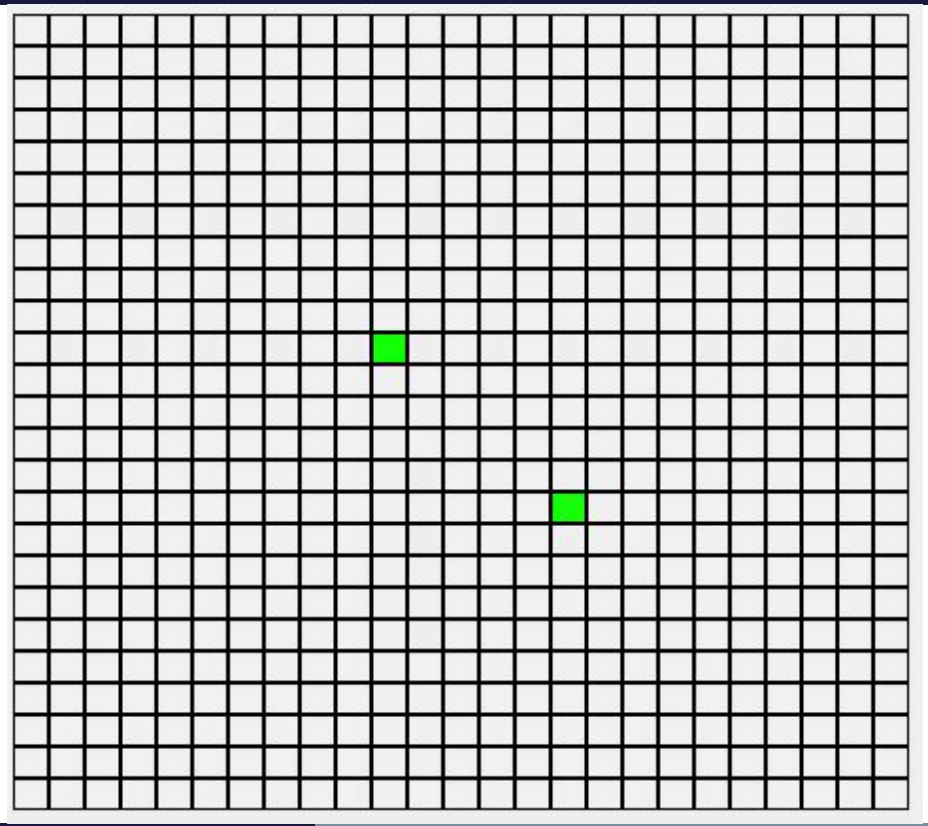
$$y_{i+1} = \left(\sum_{j=1}^{m} \frac{x_j}{\|x_j - y_i\|} \right) \Big/ \left(\sum_{j=1}^{m} \frac{1}{\|x_j - y_i\|} \right)$$

Applicable Pre-existing Algorithms

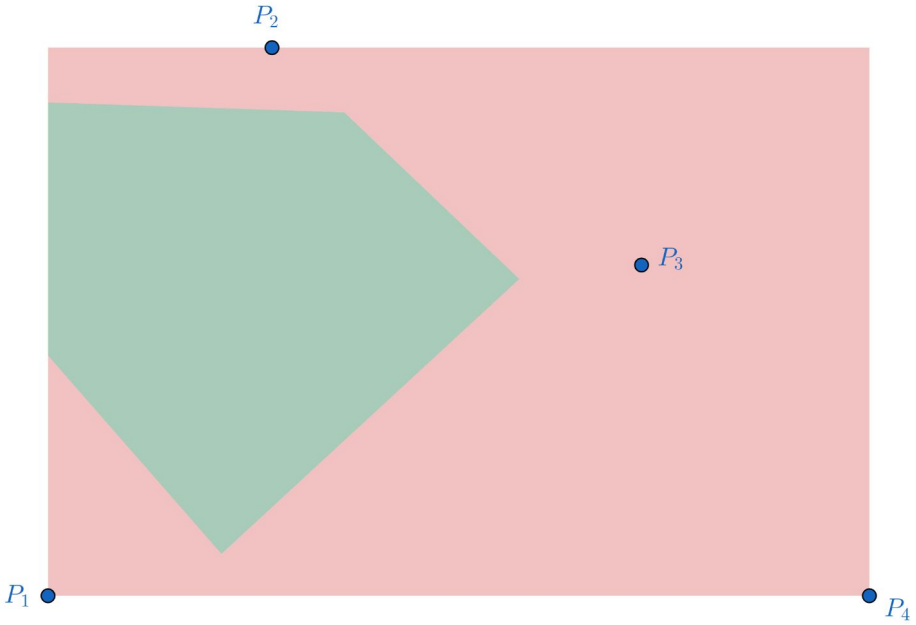
- Weiszfeld's algorithm
- A* Pathfinding
  - Heuristic Pathfinding algorithm for discrete space
- Continuous Dijkstra paradigm
  - Pathfinding algorithm for (continuous) Euclidean space

A* Pathfinding

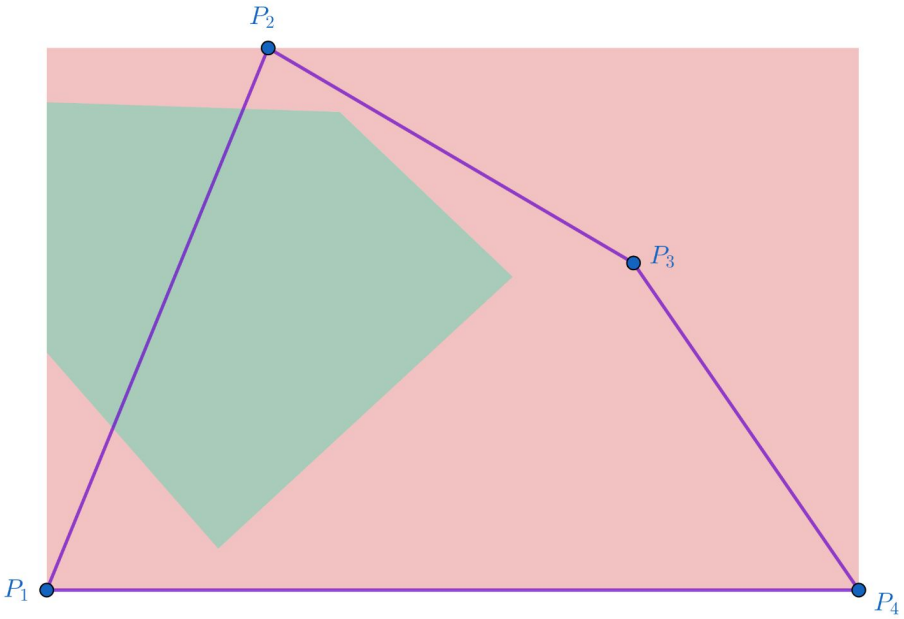
- Monotone heuristic utilized
  - Admissible
  - Will always find shortest path inside convex hull



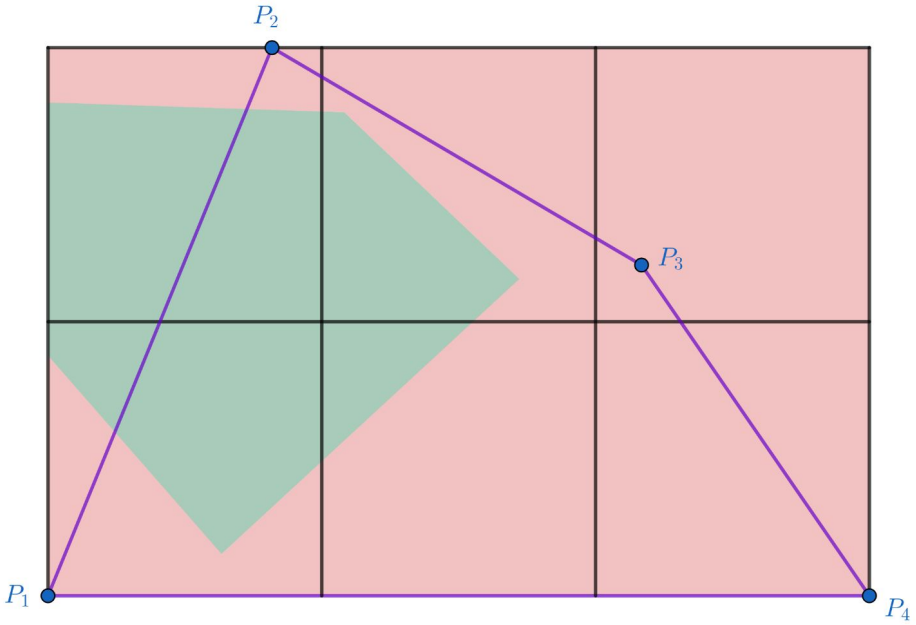
Weighted Regions Algorithm



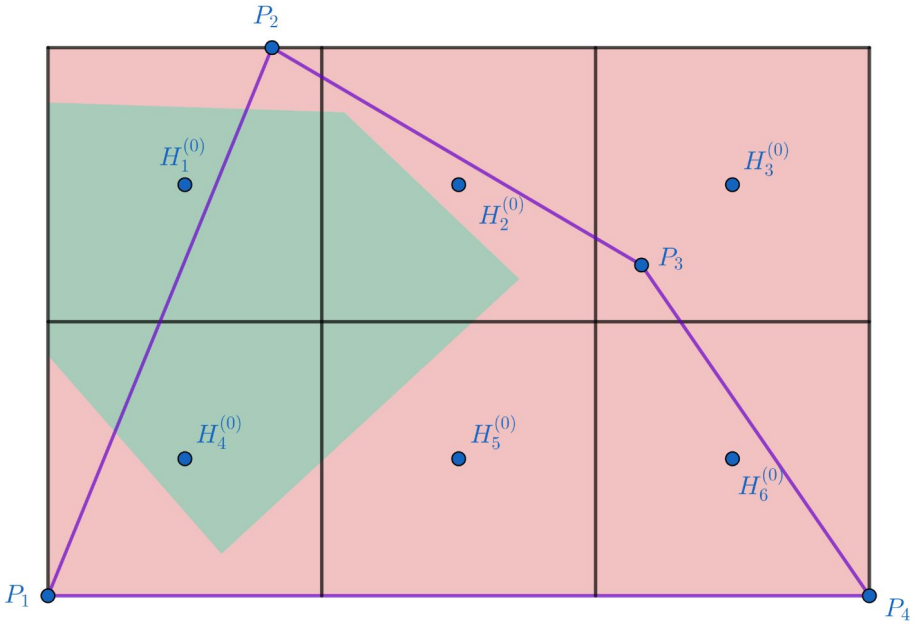
Weighted Regions Algorithm



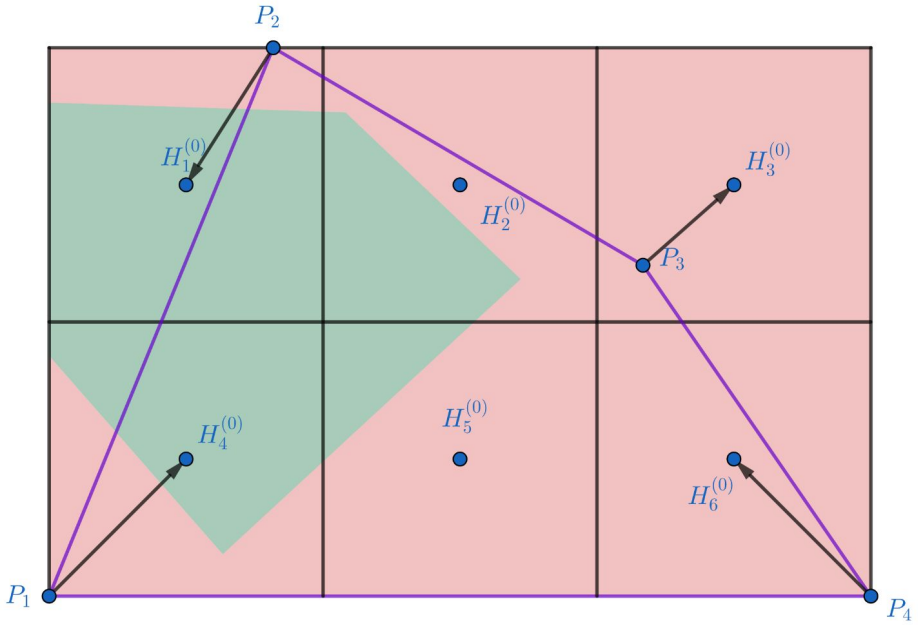
Weighted Regions Algorithm



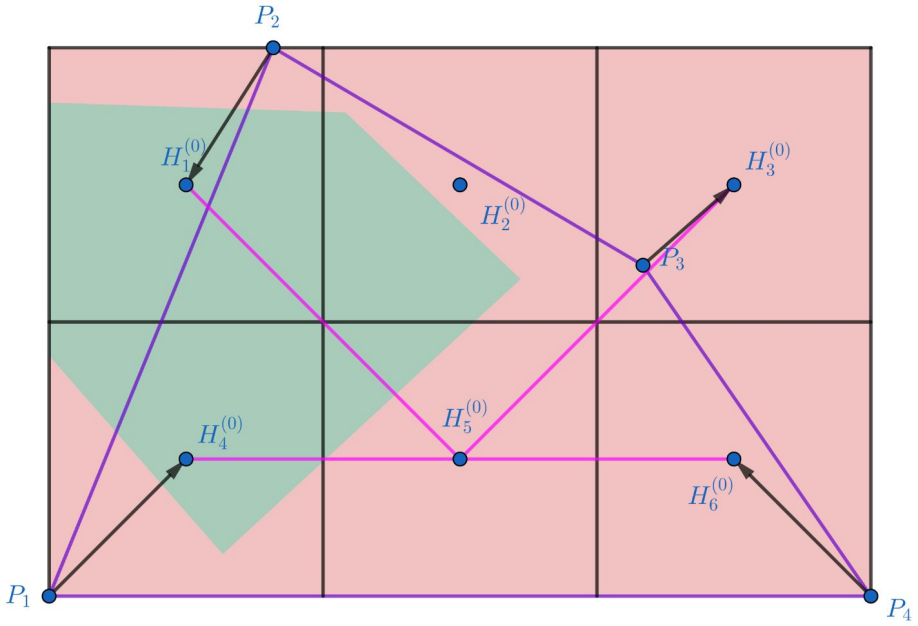
Weighted Regions Algorithm



Weighted Regions Algorithm

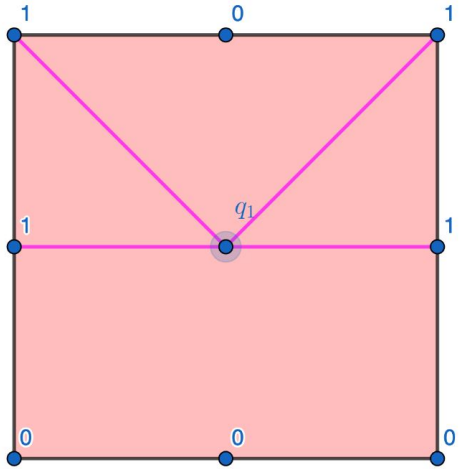


Weighted Regions Algorithm



Weiszfeld's Algorithm Extension

- Constant Weighting in Square:
  - Applying Weiszfeld's algorithm



Lemma 5.2. *Call some initial point inside square* s_i q_1. *Continvously iterating through*

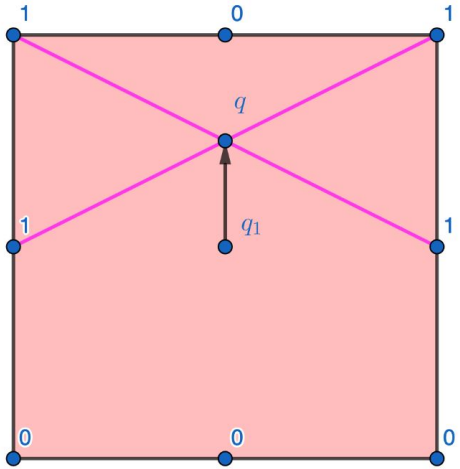
$$q_{i+1} = \left(\sum_{n=1}^{8} \frac{w(n) \cdot c(\theta_n)}{\|q_i - c(\theta_n)\|} \right) \Bigg/ \left(\sum_{n=1}^{8} \frac{w(n)}{\|q_i - c(\theta_n)\|} \right)$$

will approach the geometric median q*:*

$$\lim_{i \to \infty} q_i = q.$$

Weiszfeld's Algorithm Extension

- Constant Weighting in Square:
  - Applying Weiszfeld's algorithm



Lemma 5.2. *Call some initial point inside square* s_i q_1. *Continvously iterating through*

$$q_{i+1} = \left(\sum_{n=1}^{8} \frac{w(n) \cdot c(\theta_n)}{\|q_i - c(\theta_n)\|} \right) \Bigg/ \left(\sum_{n=1}^{8} \frac{w(n)}{\|q_i - c(\theta_n)\|} \right)$$

will approach the geometric median q:

$$\lim_{i \to \infty} q_i = q.$$

Weiszfeld's Algorithm Extension

- Manifold produced by Weiszfeld's algorithm

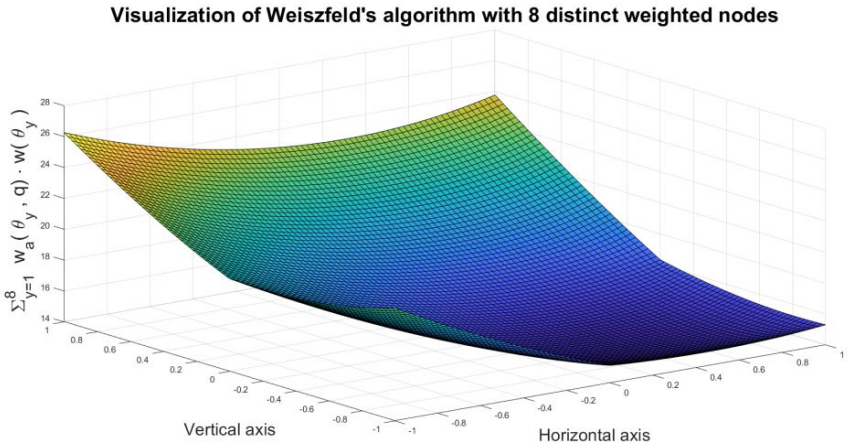


Figure 7: Manifold produced by calculating the distance to each $c(\theta_y)$ for $y \in [\![1, 8]\!]$. From $(-1, 1)$ and going clockwise, the weightings are as follows: $1, 0, 1, 2, 4, 2, 1, 2$. Weiszfeld's algorithm would approximate the minimum along this manifold

Weighted Regions Algorithm: Accuracy

- Computing to some level of accuracy

Lemma 5.3. *The number of iterations needed to calculate the target accuracy (represented by some variable $\epsilon \propto n_1$) can be modeled by*

$$\left\lceil \frac{3\log 2 - 2\log \epsilon_c}{2\log\left[\sqrt{\left\lceil \frac{l}{n_1} \right\rceil \cdot \left\lceil \frac{m}{n_1} \right\rceil}\right] - 2\log 2} \right\rceil + 1$$

where $\epsilon_c = \frac{\epsilon}{n_1}$, ϵ *is the target accuracy, and* $\left[\sqrt{\left\lceil \frac{l}{n_1} \right\rceil \cdot \left\lceil \frac{m}{n_1} \right\rceil}\right]$ *represents the number of squares after iteration* $y+1$ *in* s_y $\forall y$ *(this technique does not take into account the possible utilization of Weiszfeld's algorithm).*

Weighted Regions Algorithm: Overview

Algorithm 1 Star Topology Weigheted Regions Algorithm

Input: points q, p_1, p_2,..., $p_N \in \mathbb{R}^2$
Input: set of polygonal regions $\mathcal{P} = \{P_1, P_2, P_3, ..., P_k\} \in \mathbb{R}^2$
Input: desired accuracy $\epsilon \in (0, 2n_1\sqrt{2}]$

1: Create convex hull C such that $\forall p_i \in \mathbb{R}^2, p_i \in C$
2: Place a rectangle with dimensions l, m such that C is inscribed
3: Position a two-dimensional grid equally spaced some n_1 distance apart (and is able to extend over edges l, m) where all squares have a node $h_i^{(0)}$ at its center
4: $y, k = 0$
5: **while** $k \; != i = \left\lceil \frac{3\log 2 - 2\log \epsilon_c}{2\log\left[\sqrt{\left\lceil \frac{l}{n_1}\right\rceil \cdot \left\lceil \frac{m}{n_1}\right\rceil}\right] - 2\log 2} \right\rceil + 1$ **do**
6: | **for** $i \in \left[\!\left[1, \left[\sqrt{\left\lceil \frac{l}{n_i}\right\rceil \cdot \left\lceil \frac{m}{n_i}\right\rceil}\right]^2\right]\!\right]$ **do**
7: | | calculate $\sum_{j=1}^{N} w_a(h_i^{(y)}, p_j)$
8: | **end for**
9: | $\forall h_i^{(y)}$, find $\min \sum_{j=1}^{N} w_a(h_i^{(y)}, p_j)$
10: | create square s_{y+1} around $\min h_i^{(y)}$ with the 8 surrounding $h_i^{(y)}$ nodes
11: | cut s_{y+1} into $\left[\sqrt{\left\lceil \frac{l}{n_i}\right\rceil \cdot \left\lceil \frac{m}{n_i}\right\rceil}\right]^2$ new squares
12: | place $h_i^{(y+1)}$ at the center of each new square
13: | $y \leftarrow y + 1$
14: | $k \leftarrow k + 1$
15: **end while**
16: **Output** $\min \sum_{j=1}^{N} w_a(h_i^{(y)}, p_j)$

- High level overview

Remark 7.1. *Note that on line 6 (in algorithm 1) for the first iteration,*

$$i \in \left[\!\left[1, \left\lceil \frac{l}{n_i}\right\rceil \cdot \left\lceil \frac{m}{n_i}\right\rceil\right]\!\right].$$

Remark 7.2. *For the sake of brevity we choose not to include the possible Weiszfeld's algorithm addition, which can serve as an alternative to the while loop once the region inside some square s_i contains the same weighting.*

Continuous Dijkstra Paradigm

- Based on propagating wavelets

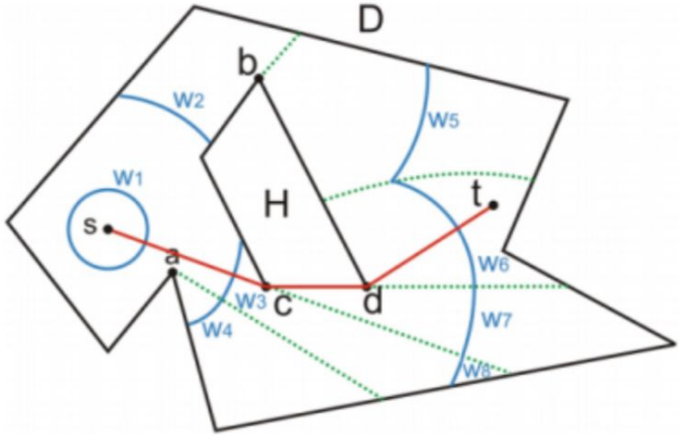
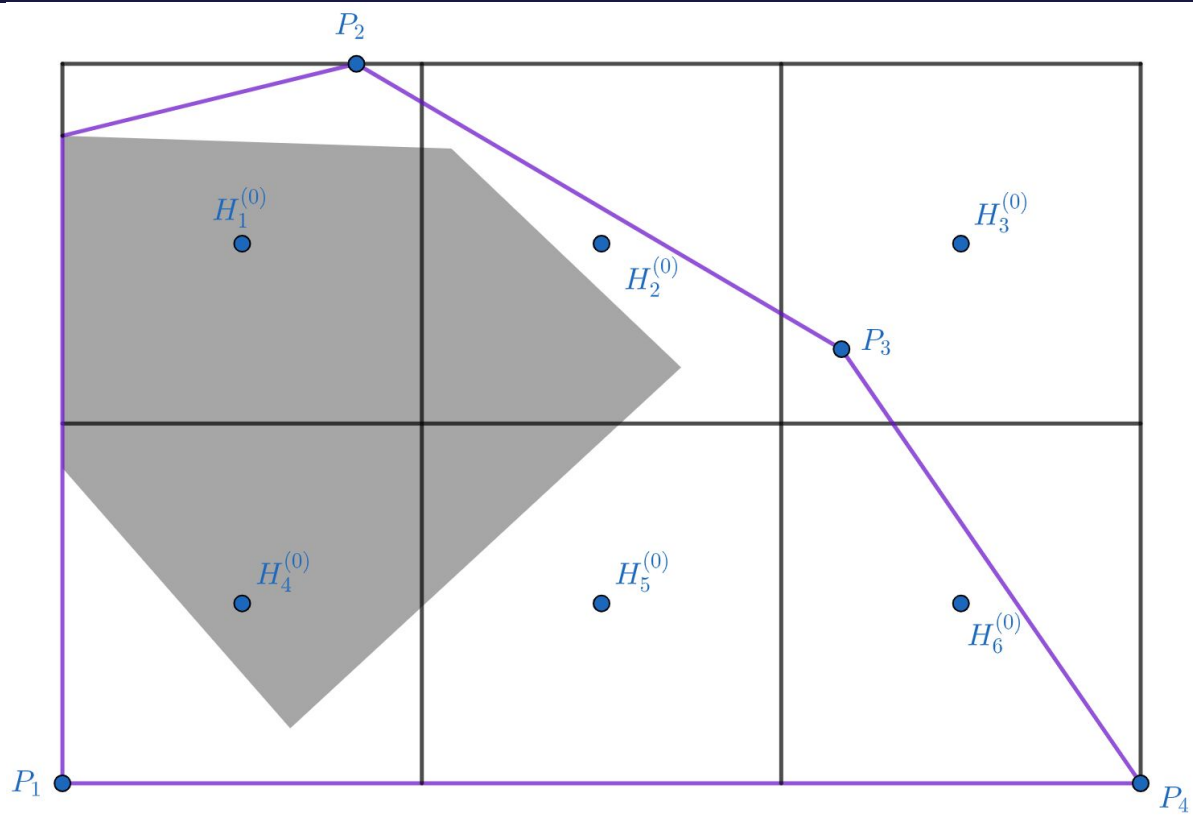


Figure 5: Continuous Dijkstra paradigm finding a minimum path from node s to node t (each W represents another stage of the propagation of the wavelet) around obstacle H [10]

Obstacles Algorithm



Obstacles Algorithm: Accuracy

- Computing algorithm to some level of accuracy

Lemma 6.1. *The number of iterations needed to calculate the accuracy of ϵ can be determined by*

$$\left\lceil \frac{3\log 2 - 2\log \epsilon_c}{2\log\left[\sqrt{r}\right] - 2\log 2} \right\rceil + 1$$

where $\epsilon_c = \frac{\epsilon}{n}$, ϵ is the target accuracy and $\left[\sqrt{r}\right]^2$ represents the number of squares after iteration $y+1$ in s_y $\forall y$.

Obstacles Algorithm: Convergence

- Graph of rate of convergence for different number of squares

Lemma 6.1. *The number of iterations needed to calculate the accuracy of ϵ can be determined by*

$$\left\lceil \frac{3\log 2 - 2\log \epsilon_c}{2\log\left\lceil\sqrt{r}\right\rceil - 2\log 2} \right\rceil + 1$$

where $\epsilon_c = \frac{\epsilon}{n}$, ϵ is the target accuracy and $\left\lceil\sqrt{r}\right\rceil^2$ represents the number of squares after iteration $y+1$ in s_y $\forall y$.

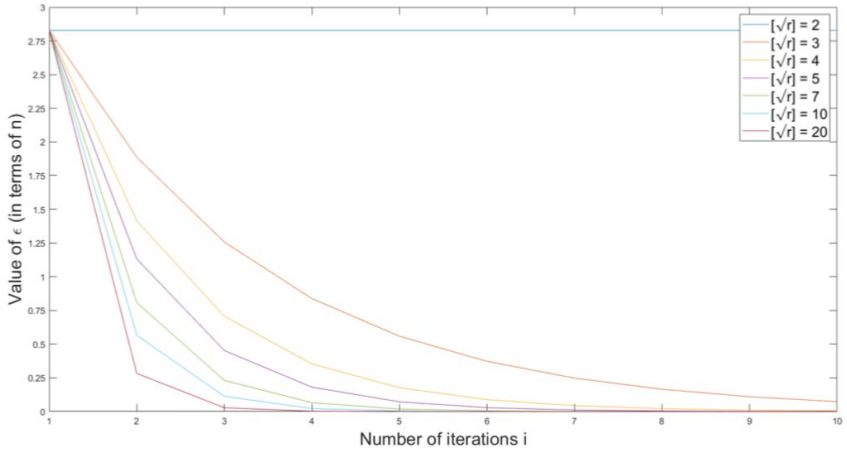


Figure 8: Rate of convergence for distinct values of $\left\lceil\sqrt{r}\right\rceil$. If $\left\lceil\sqrt{r}\right\rceil \leq 2$, iterations of the algorithm will not converge

Obstacles Algorithm: Overview

- High level overview

Algorithm 2 Star Topology Obstacles Algorithm

Input: points q, p_1, p_2,..., $p_N \in \mathbb{R}^2$
Input: set of polygonal obstacles $\mathcal{P} = \{P_1, P_2, P_3, ..., P_k\} \in \mathbb{R}^2$
Input: desired accuracy $\epsilon \in (0, 2n\sqrt{2}]$

Create C such that $\forall p_i, P_j \in \mathbb{R}^2, p_i, P_j \in C$
Position a two-dimensional grid equally spaced some n distance apart where all squares who have some part $\in C$ and some part $\notin \mathcal{P}$ have a node $h_i^{(0)}$ at its center
$y, k = 0$
while $k\ != i = \left\lceil \frac{3\log 2 - 2\log \epsilon_c}{2\log\left\lceil\sqrt{r}\right\rceil - 2\log 2} \right\rceil + 1$ **do**
  for $i \in \left[\!\left[1, \left\lceil\sqrt{r}\right\rceil^2\right]\!\right]$ **do**
    calculate $\sum_{j=1}^{N} d(h_i^{(y)}, p_j)$
  end for
  $\forall h_i^{(y)}$, find $\min \sum_{j=1}^{N} d(h_i^{(y)}, p_j)$
  create square s_{y+1} around $\min h_i^{(y)}$ with the 8 surrounding $h_i^{(y)}$ nodes
  cut s_{y+1} into $\left\lceil\sqrt{r}\right\rceil^2$ new squares
  place $h_i^{(y+1)}$ at the center of each new square
  $y \leftarrow y + 1$
  $k \leftarrow k + 1$
end while
Output $\min \sum_{j=1}^{N} d(h_i^{(y)}, p_j)$

Obstacles Algorithm: Runtime

Lemma C.2. *The Star Topology Obstacles algorithm given some ϵ target accuracy in $\mathbb{R}^2$ runs in $O(\lceil\sqrt{r}\rceil^2 n \log n \log \frac{1}{\epsilon_c} \log^{-1} \lceil\sqrt{r}\rceil)$ time*