Path Planning (A*, RRT)

Dr. Gaurav Pandey

Electrical Engineering

IIT Kanpur

Occupancy Grid Map: Quick Recap

- In occupancy grid map we divide the world into grid cells and estimate the occupancy of each cell (a binary probability).
- Mapping problem posed as an optimization of the posterior as shown below, assuming pose and measurements are known:

$$m^* = \arg\max_m P(m|x_{1:t}, z_{1:t})$$

- Problem with this:
  - High dimensionality of map. Consider a map of 100x100 (=10000) grid cells with each covering 25cm x 25cm area, the total area covered is only 25m x 25m. For this map what is the total number of possible maps that you can get ??

2^{10000}

Calculating the posterior is therefore intractable

How to make the occupancy grid mapping tractable ?

- Simplify the problem by assuming independence of grid cells. Thus the problem of estimating the full posterior breaks down to estimation of occupancy of each grid cell

$$P(m|x_{1:t}, z_{1:t}) = \prod_{i=1}^{i=n} P(m_i|z_{1:t}, x_{1:t})$$

- The estimation of the occupancy probability for each grid cell can be implemented as a **Binary Bayes Filter.** So we only need to find:

$$P(m_i|z_{1:t}, x_{1:t})$$

Occupancy of a grid cell

- Probability that ith cell is occupied can be written as

$$P(m_i|z_{1:t}, x_{1:t}) = \frac{P(z_t|m_i, z_{1:t-1}, x_{1:t})P(m_i|z_{1:t-1}, x_{1:t})}{P(z_t|z_{1:t-1}, x_{1:t})}$$

Bayes

$$= \frac{P(z_t|m_i, x_t)P(m_i|z_{1:t-1}, x_{1:t})}{P(z_t|x_t)}$$

Markov

$$P(z_t|m_i, x_t) = \frac{P(m_i|z_t, x_t)P(z_t|x_t)}{P(m_i|x_t)}$$

Bayes

$$P(m_i|x_t) = P(m_i)$$

State is assumed to be static

Occupancy of grid cell

- Probability that ith cell is occupied is given by

$$P(m_i|z_{1:t}, x_{1:t}) = \frac{P(m_i|z_t, x_t)P(m_i|z_{1:t-1}, x_{1:t-1})}{P(m_i)}$$

- Similarly probability that ith cell is free is given by

$$P(\tilde{m_i}|z_{1:t}, x_{1:t}) = \frac{P(\tilde{m_i}|z_t, x_t)P(\tilde{m_i}|z_{1:t-1}, x_{1:t-1})}{P(\tilde{m_i})}$$

$$P(\tilde{m_i}|z_{1:t}, x_{1:t}) = 1 - P(m_i|z_{1:t}, x_{1:t})$$

Binary Random Variable

Odds: Ratio of occupied vs free

- Ratio of probabilities (Odds ratio)

$$\frac{P(m_i|z_{1:t},x_{1:t})}{P(\tilde{m}_i|z_{1:t},x_{1:t})} = \underbrace{\frac{P(m_i|z_t,x_t)}{1-P(m_i|z_t,x_t)}}_{\text{Measurement Term}} \cdot \underbrace{\frac{P(m_i|z_{1:t-1},x_{1:t-1})}{1-P(m_i|z_{1:t-1},x_{1:t-1})}}_{\text{Recursive Term}} \cdot \underbrace{\frac{1-P(m_i)}{P(m_i)}}_{\text{Prior}}$$

- Log-odds ratio: Take the log of above equation

$$\boxed{l(m_i|z_{1:t},x_{1:t}) = l(m_i|z_t,x_t) + l(m_i|z_{1:t-1},x_{1:t-1}) - l(m_i)}$$

$$l(x) = \log \frac{P(x)}{1-P(x)}$$

Complete Algorithm

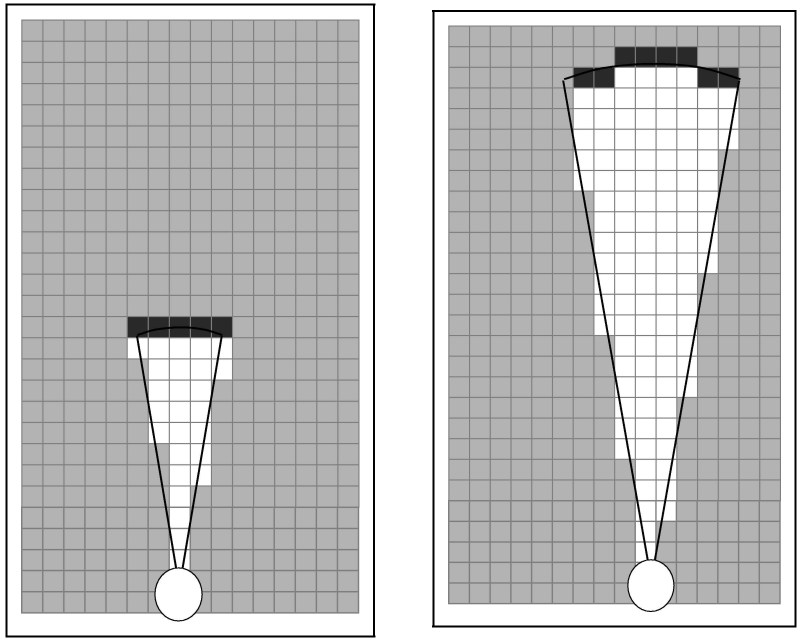
```
**Algorithm occupancy_grid_mapping**($\{l_{t-1,i}\}, x_t, z_t$):
    for all cells $\mathbf{m}_i$ do
        if $\mathbf{m}_i$ in perceptual field of $z_t$ then
            $l_{t,i} = l_{t-1,i} + \textbf{inverse\_sensor\_model}(\mathbf{m}_i, x_t, z_t) - l_0$
        else
            $l_{t,i} = l_{t-1,i}$
        endif
    endfor
    return $\{l_{t,i}\}$
```

Inverse Sensor Model

$$l(m_i|z_t, x_t) = \log \frac{P(m_i|z_t, x_t)}{1 - P(m_i|z_t, x_t)}$$

$$P(m_i|z_t, x_t) = \frac{P(z_t|m_i, x_t)P(m_i|x_t)}{P(z_t|x_t)}$$

$$P(m_i|z_t, x_t) = \eta P(z_t|m_i, x_t)P(m_i)$$



- Let x_i, y_i be the coordinates of ith cell then expected measurement

$$\hat{r} = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2} \qquad \hat{\phi} = tan^{-1}\frac{y_i - y_t}{x_i - x_t} - \theta_t$$

Inverse Sensor Model

- The measurement term for the occupancy grid map algorithm:

$$P(z_t = [r, \phi] | m_i = occupied, x_t) = \eta \exp(-\frac{1}{2}[r - \hat{r}, \phi - \hat{\phi}] \Sigma^{-1} \begin{bmatrix} r - \hat{r} \\ \phi - \hat{\phi} \end{bmatrix})$$

```
**Algorithm occupancy_grid_mapping**($\{l_{t-1,i}\}, x_t, z_t$):
    for all cells $\mathbf{m}_i$ do
        if $\mathbf{m}_i$ in perceptual field of $z_t$ then
            $l_{t,i} = l_{t-1,i} + \textbf{inverse\_sensor\_model}(\mathbf{m}_i, x_t, z_t) - l_0$
        else
            $l_{t,i} = l_{t-1,i}$
        endif
    endfor
    return $\{l_{t,i}\}$
```

Sample Occupancy Grid Maps

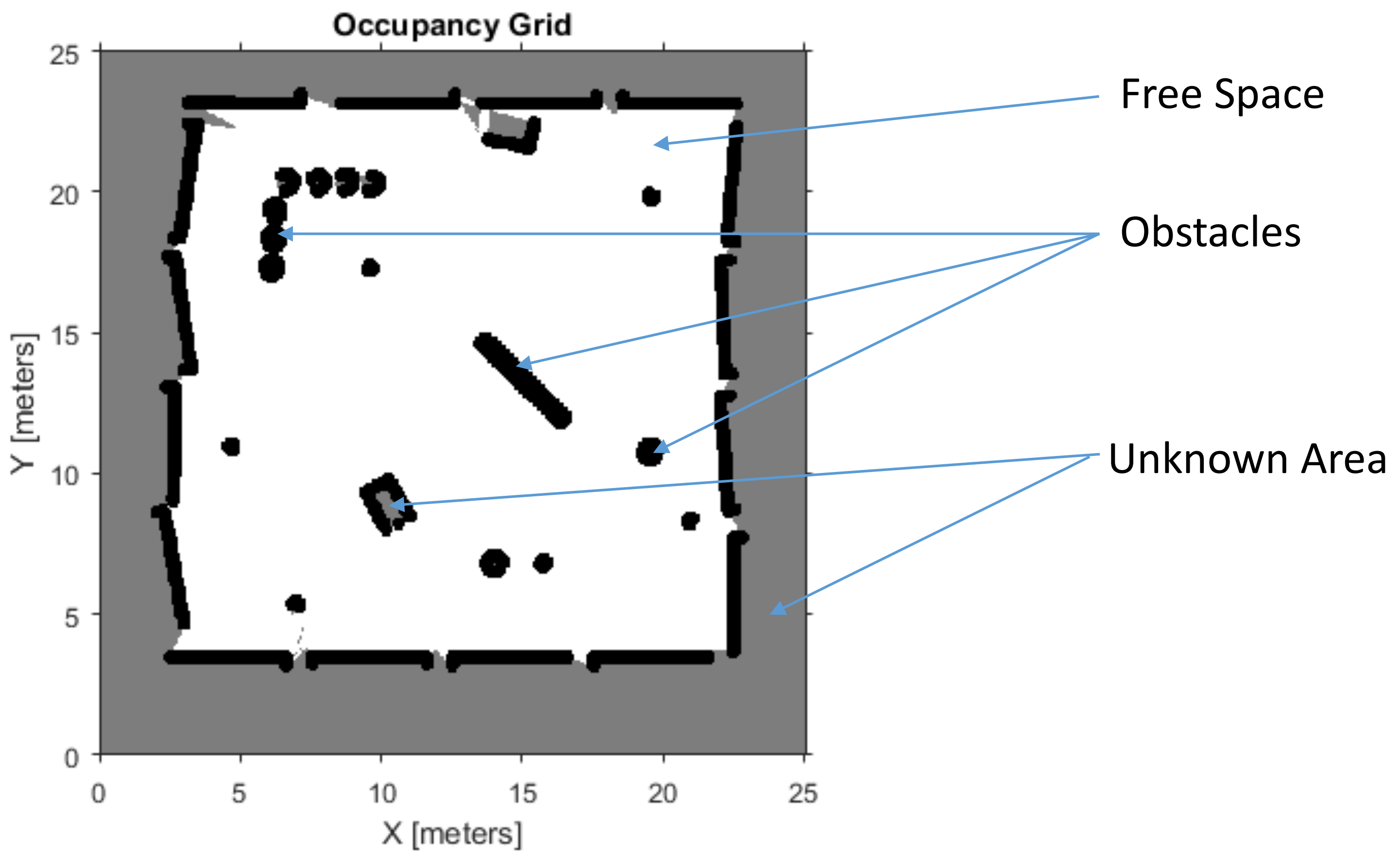
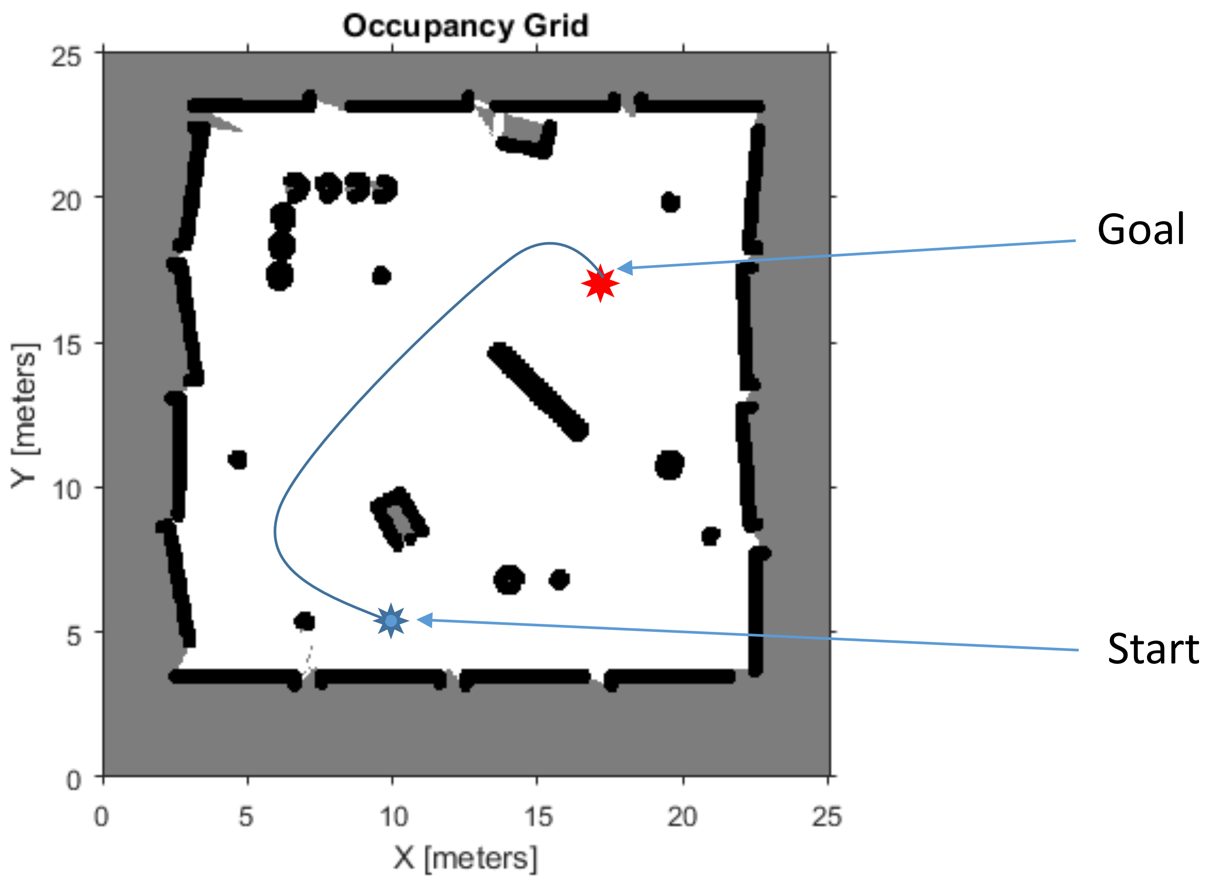
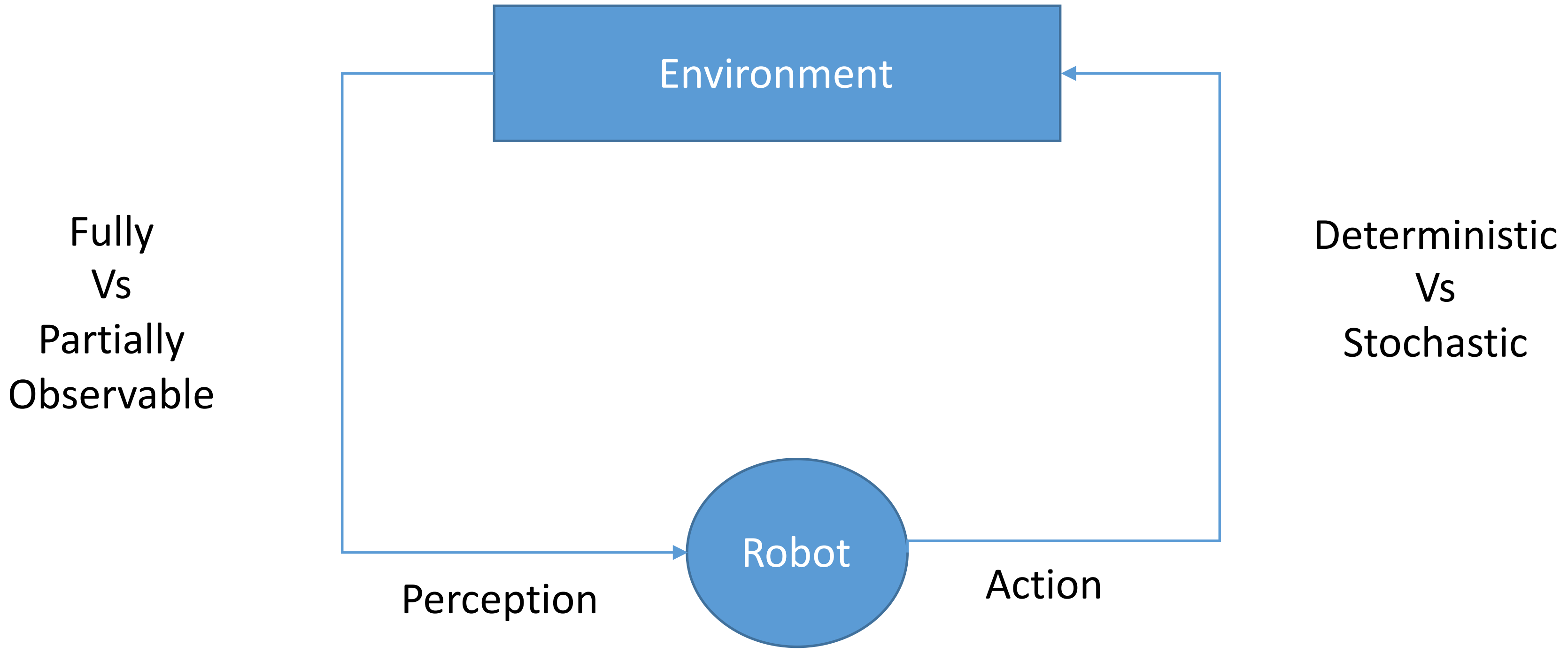


Image Credit: Mathworks

Path Planning



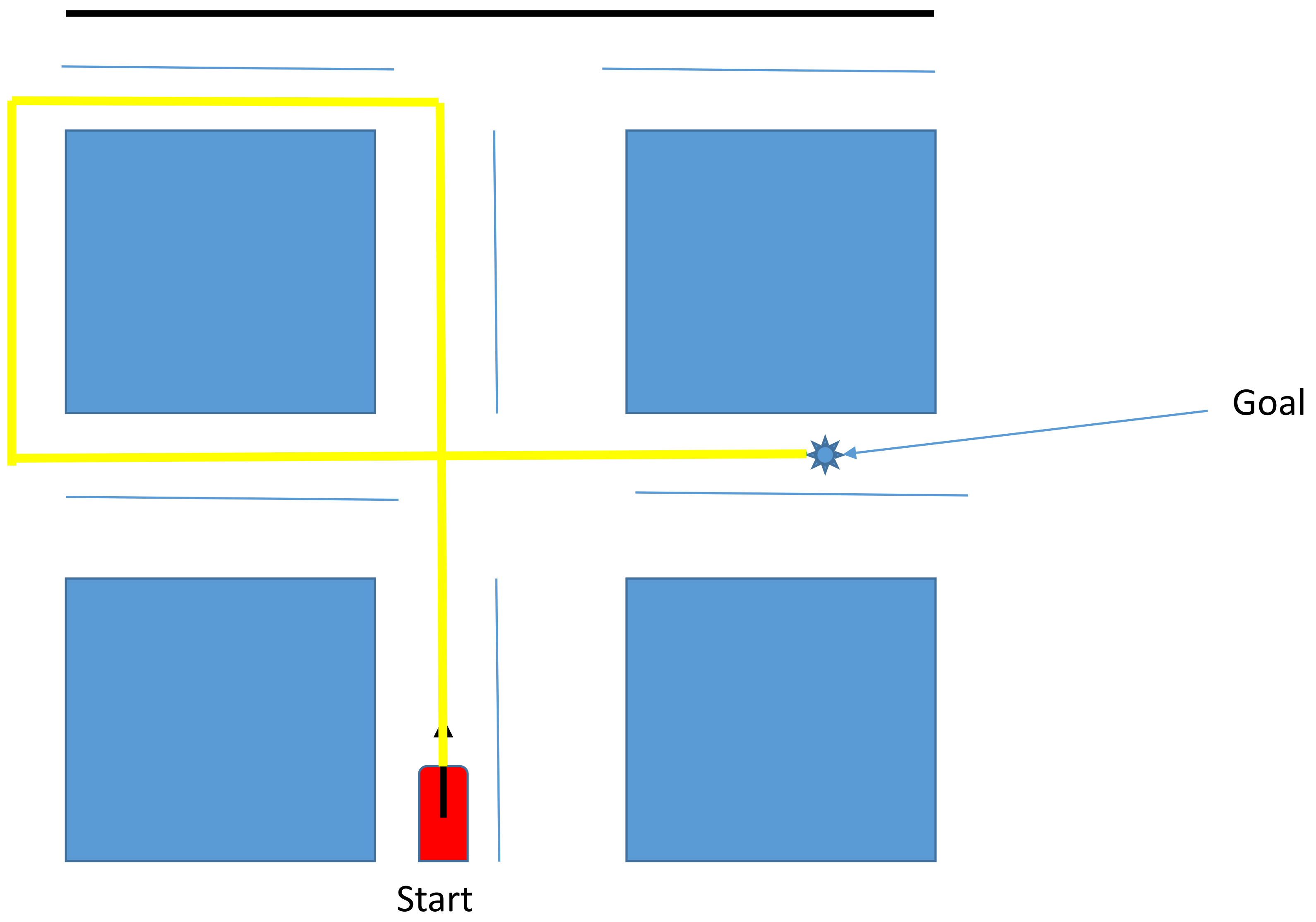
Path Planning



Classical Deterministic Planning

- Given:
    - Map (Complete map, static environment, fully observable)
    - Starting location
    - Goal location
    - Actions (Deterministic)
    - Cost (e.g. time taken to reach goal)

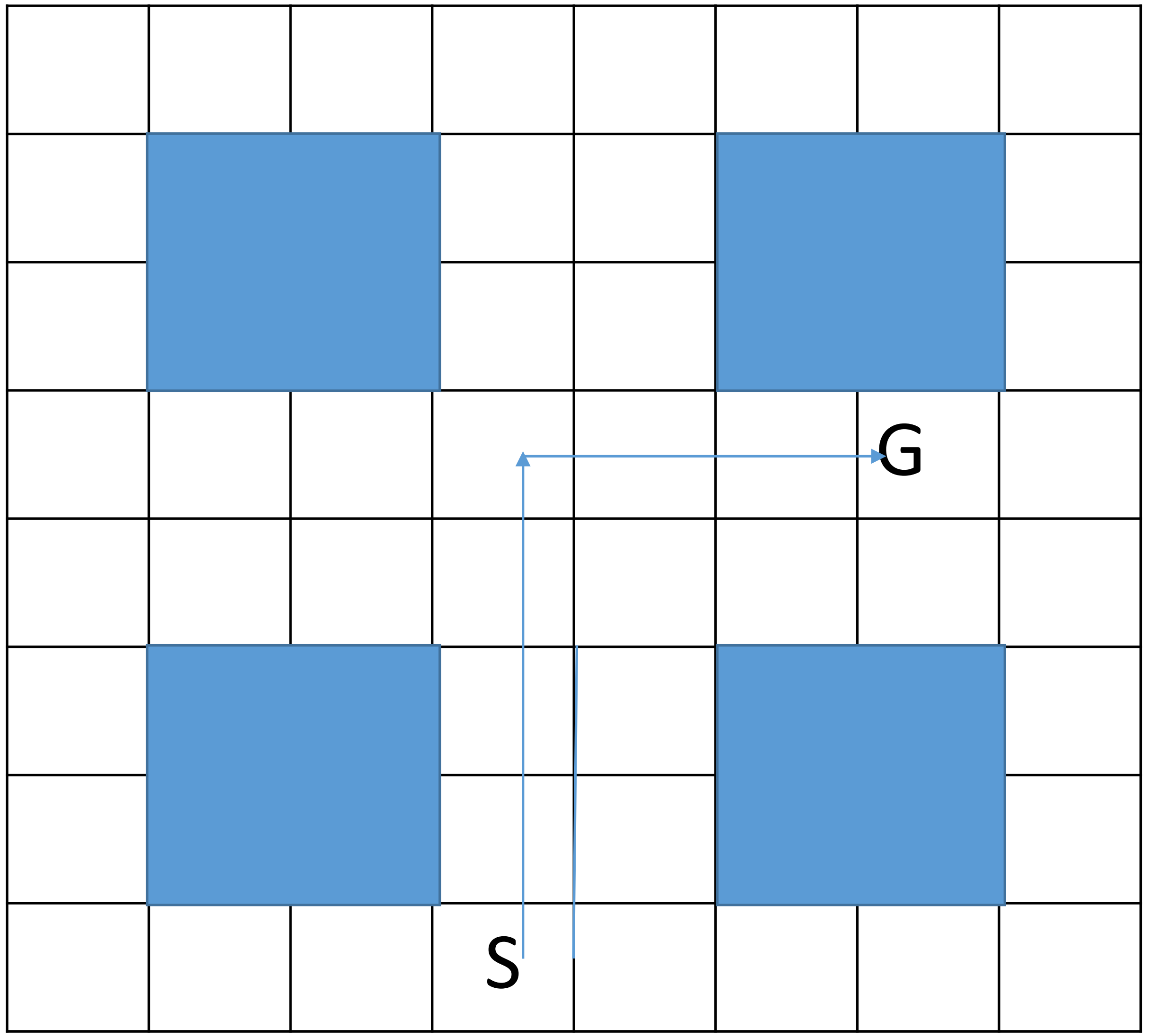
Example



Planning Example

- Lets discretize the world
- At each time step the vehicle moves forward or turns left or right

Case 1: Every move costs exactly one unit



Q: What is the minimum cost to reach "G" from "S" ?

A: 7 unit

Planning Example

- Case 2: Suppose turning right is more expensive (generally the case when the oncoming traffic is heavy). Lets say the cost of turning right is 10 units. What is the minimal cost and path now ? Does it change ?

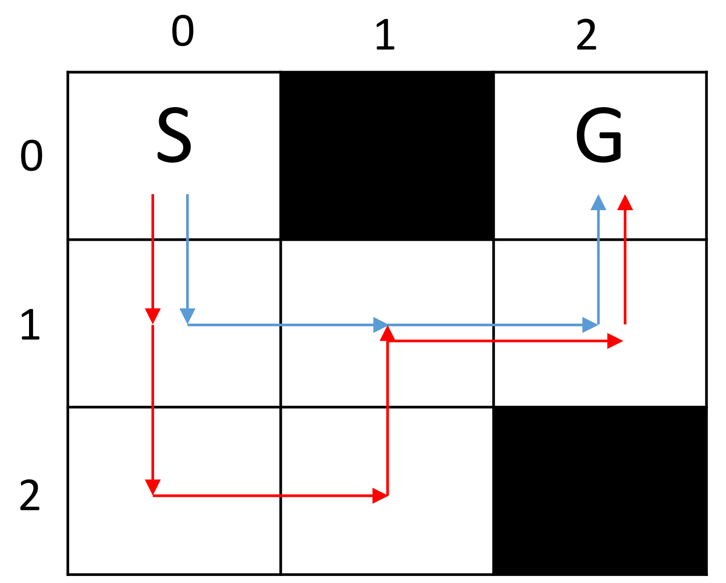
- Case 3: What happens when the cost of turning right is 20 units ? What is the best path ?

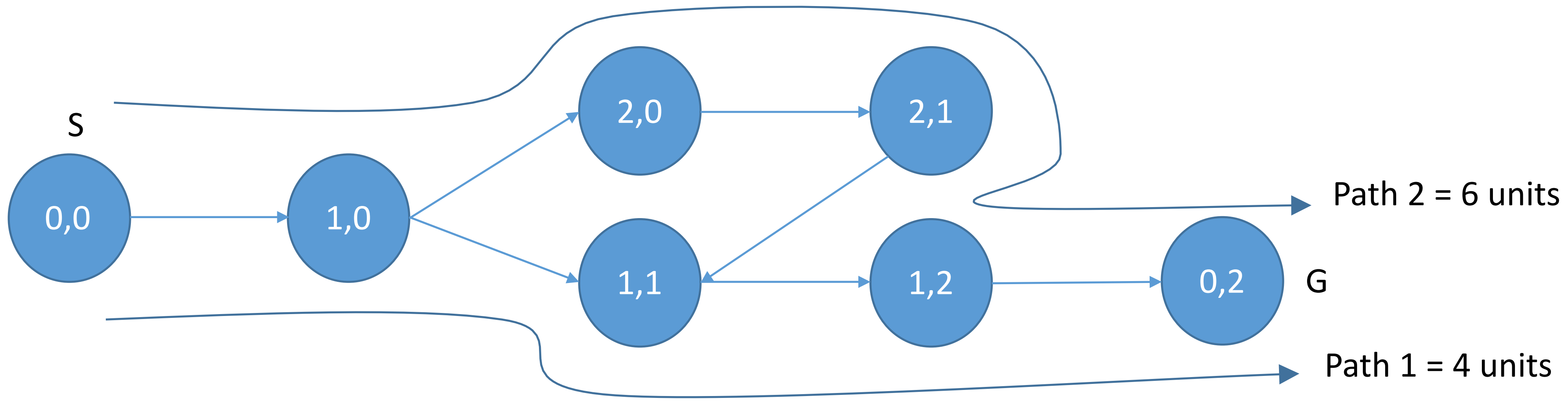
Deterministic Path Planning

- Under the assumption that robot takes every action with certainty, the path planning problem is equivalent to finding a sequence of actions that leads the robot to its goal with minimum cost !

- Therefore deterministic path planning can be viewed as a graph search problem where each node corresponds to the state of the robot (grid cell in our previous example) and the edges represents the cost of moving from one node to another

Path planning as graph search





Path planning graph search

- Basically you can expand each node and perform an exhaustive search to find the path with minimal cost.
- You continue expanding nodes until you reach the goal.
- Exhaustive search is computationally very expensive.
- How to reduce the search time ?

Heuristic A* search

- First described by Peter Hart, Nils Nelson, Bertran Raphael in 1968
- This is still a searching algorithm but this takes into account the **“right direction”** towards the goal.
- In exhaustive search we were just considering the cost of getting to any node.

A* search

- In A* the priority of expanding any node is given by the following cost
  f(n) = g(n) + h(n)

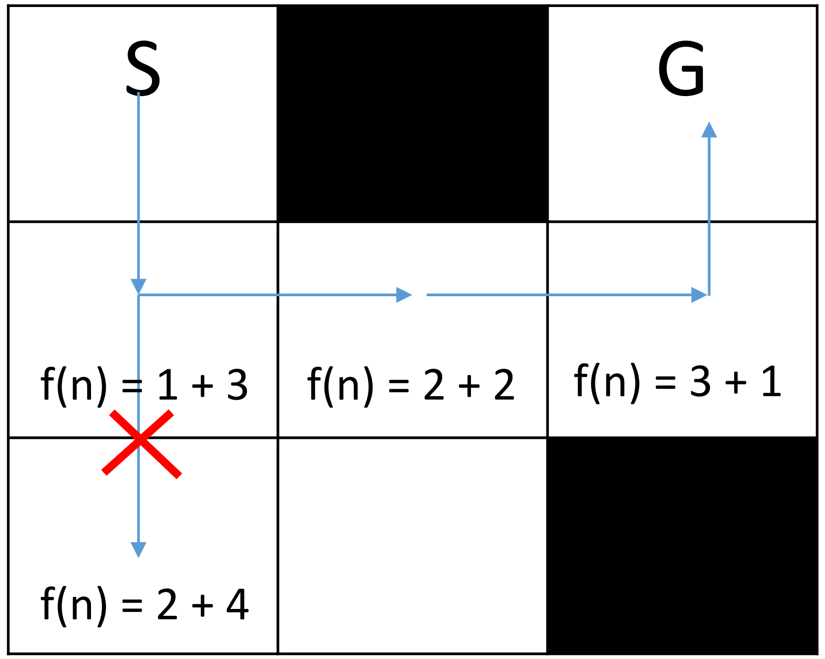
  where g(n) = Cost to get to the node “n”

  h(n) = Cost to get to **goal** from “n”

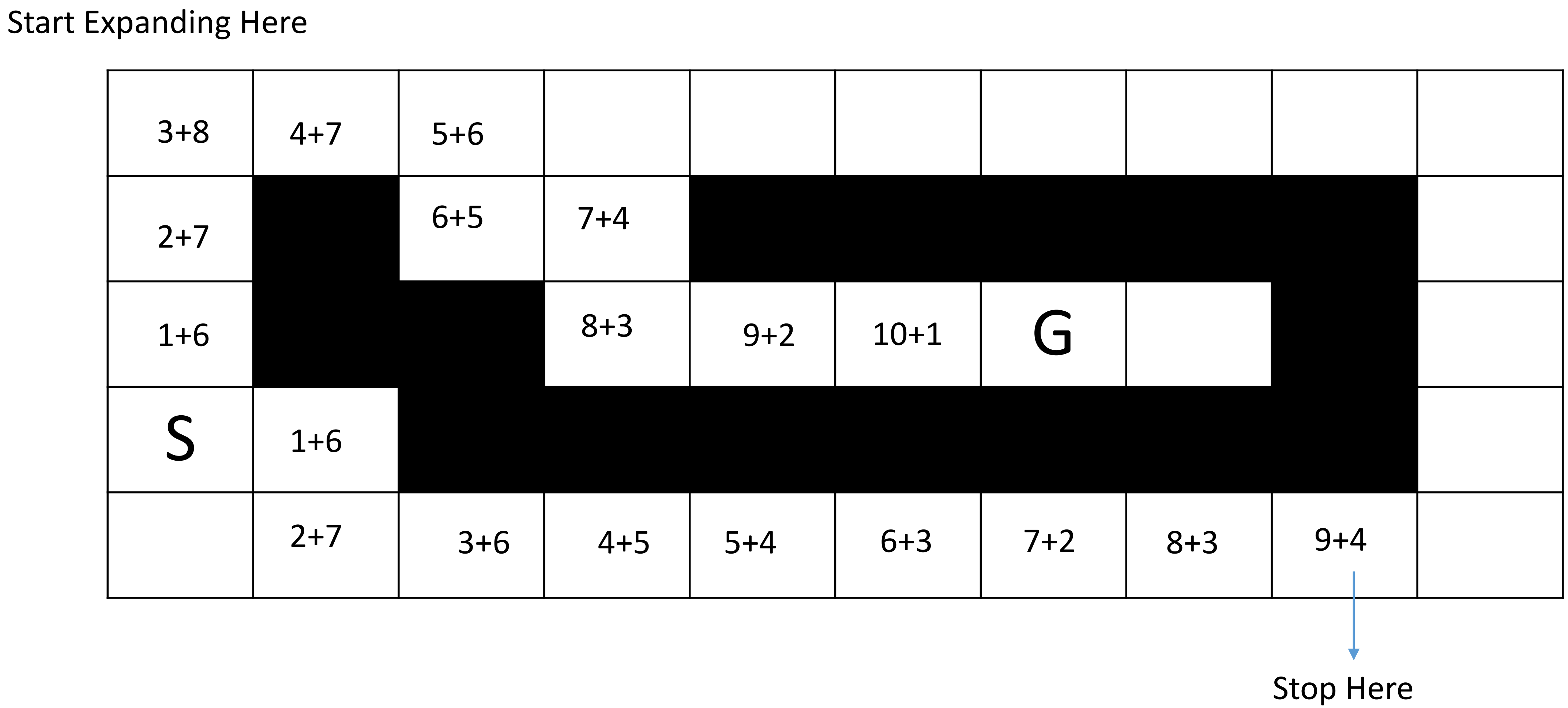
- **h(n)** is also called the heuristic function, it is just an estimate of the cost to go from node “n” to “goal”.

- A common heuristic is the euclidean or manhattan distance between the node “n” and the “goal” in the absence of any obstacles

Revisiting the example



Another example: A* gives the shortest path



Optimality of A*

- Theorem: If **h(n)** is an admissible heuristic function then A* search with **h(n)** is optimal

- Admissible: A heuristic function h is admissible if it never overestimates the true cost to get to the goal, i.e.

$$h(n) <= h^*(n)$$

where “n” is a node in the search graph, h(n) is the heuristic function evaluated at “n”, h*(n) is the actual cost of the least cost path from ”n” to goal

Proof of Optimality of A*

- Let n_g be the goal node and

$$f(n) = g(n) + h(n) \qquad \& \qquad f^*(n) = g(n) + h^*(n)$$

- Suppose h is an admissible heuristic and n_g is the first goal node to be expanded. Therefore

$\rightarrow 0 <= h(n_g) <= h^*(n_g) = 0$

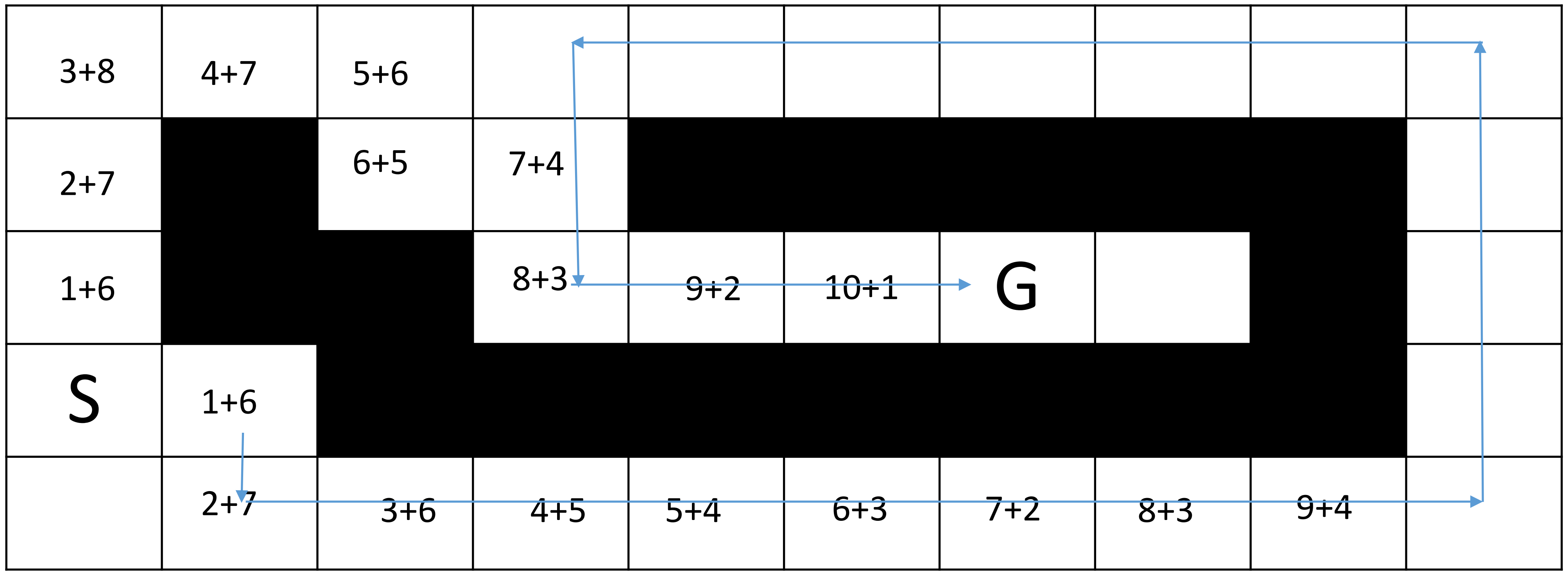
$\rightarrow h(n_g) = 0$

$\rightarrow f(n_g) = g(n_g) + h(n_g) = g(n_g)$

- For optimality of A* we need to show that the path to n_g (i.e. $g(n_g)$ or $f(n_g)$) is not longer than any other potential path through some other un-expanded node.

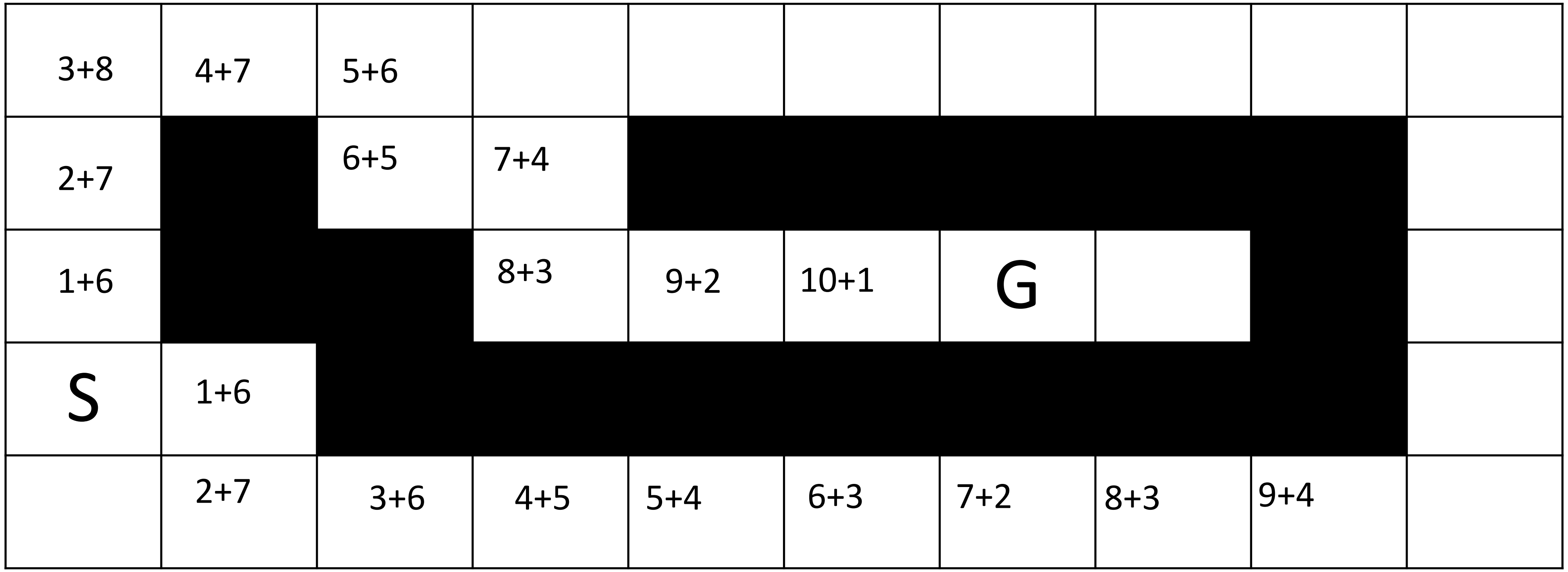
Optimality of A*

- Suppose that by expanding more nodes we can find another path to reach the goal !



Optimality of A*

- Let “n” be the first node on this path that has not been expanded yet. Note that n’s parent must have been expanded previously and therefore n is on the fringe of our search tree at the instance when the A* algorithm reaches the goal node.



Optimality of A*

- Let f*(n) denote the minimum cost of the path to goal that passes through the un-expanded node n

$$f^*(n) = g(n) + h^*(n)$$

- Since "n" was not expanded when A* chose to expand n_g

$\rightarrow f(n_g) <= f(n)$

$\rightarrow g(n_g) <= f(n)$

$<= g(n) + h(n)$

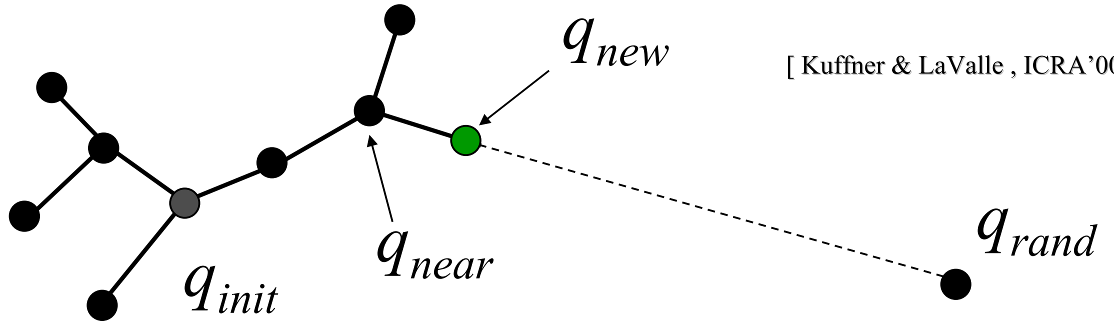
$<= g(n) + h^*(n)$

$<= f^*(n)$

- Actual cost of any other path is greater than the cost of the path chosen by A*. Hence proved that A* is optimal (i.e. it finds the minimum cost path)

Rapidly-exploring Random Tree (RRT)

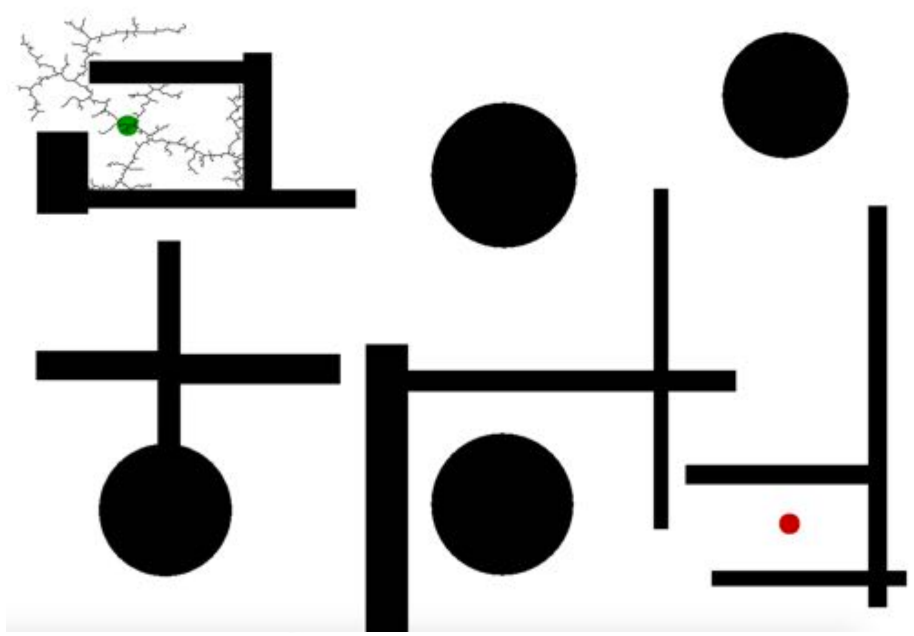
- **Algorithm**
- Input: Start node q_{init}, Map (M), Robot Motion Constrains (C)
- Output: RRT Path P
- *Initialize the graph at the start node G*.init(q_{init})
    - **While (Reached the goal)**
    - $q_{rand} \leftarrow$ RAND_CONF()
    - $q_{near} \leftarrow$ NEAREST_VERTEX(q_{rand}, G)
    - $q_{new} \leftarrow$ NEW_CONF(q_{near}, q_{rand}, M, C)
    - *G*.add_vertex(q_{new})
    - *G*.add_edge(q_{near}, q_{new})
    - End while
    - Get path P
    - **return** *P*

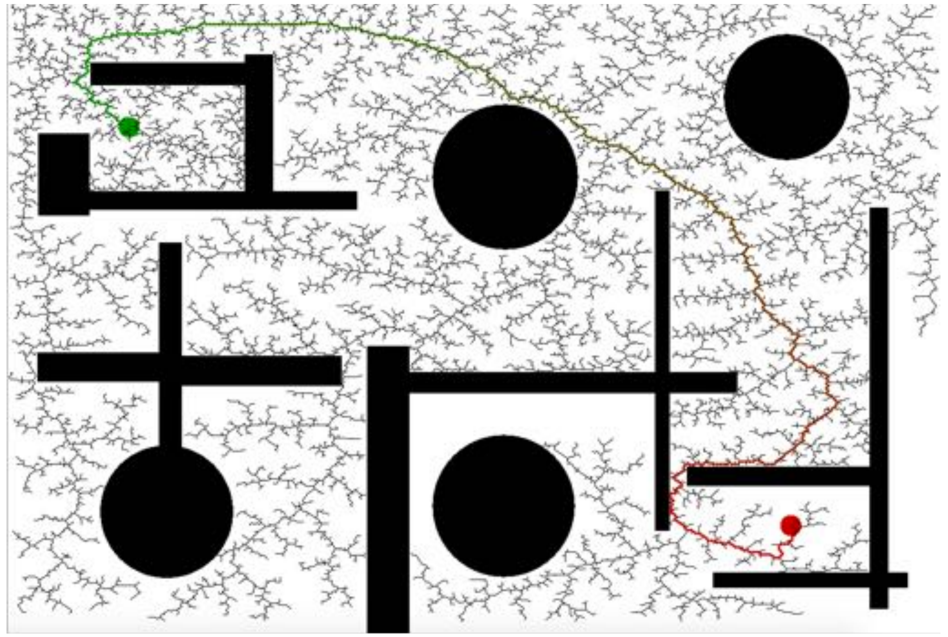


[Kuffner & LaValle , ICRA'00]

RRT example

- https://www.youtube.com/watch?v=Mezx29eBbNk





Obstacle Avoidance

- Inflate obstacles by robot size.