



# Path Planning w/ Uncertainty for Robotics Navigation

Audrey L, Amy P, Shashank S

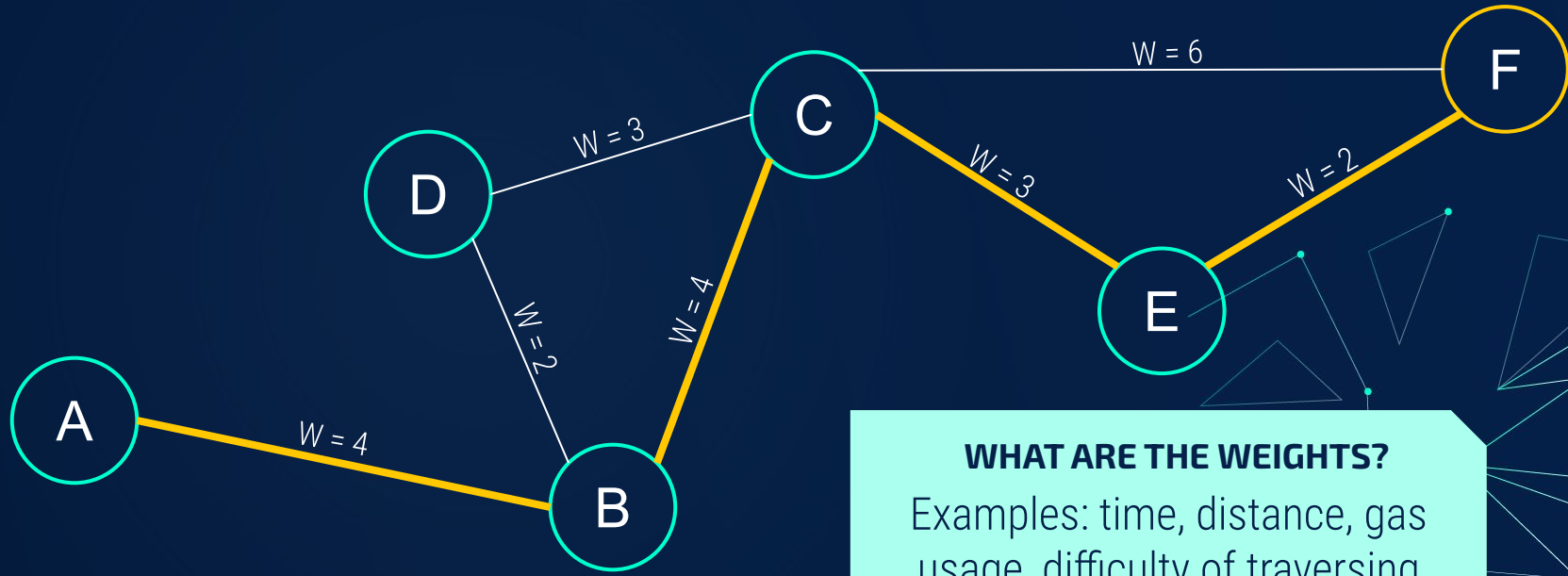
# Motivation

When would we want to find the shortest path?

- Public Transportation Networks
  - Finding a route between two places using public transit
- Traffic
  - Best route to avoid high-traffic areas
- Games
  - Moving around non-player characters



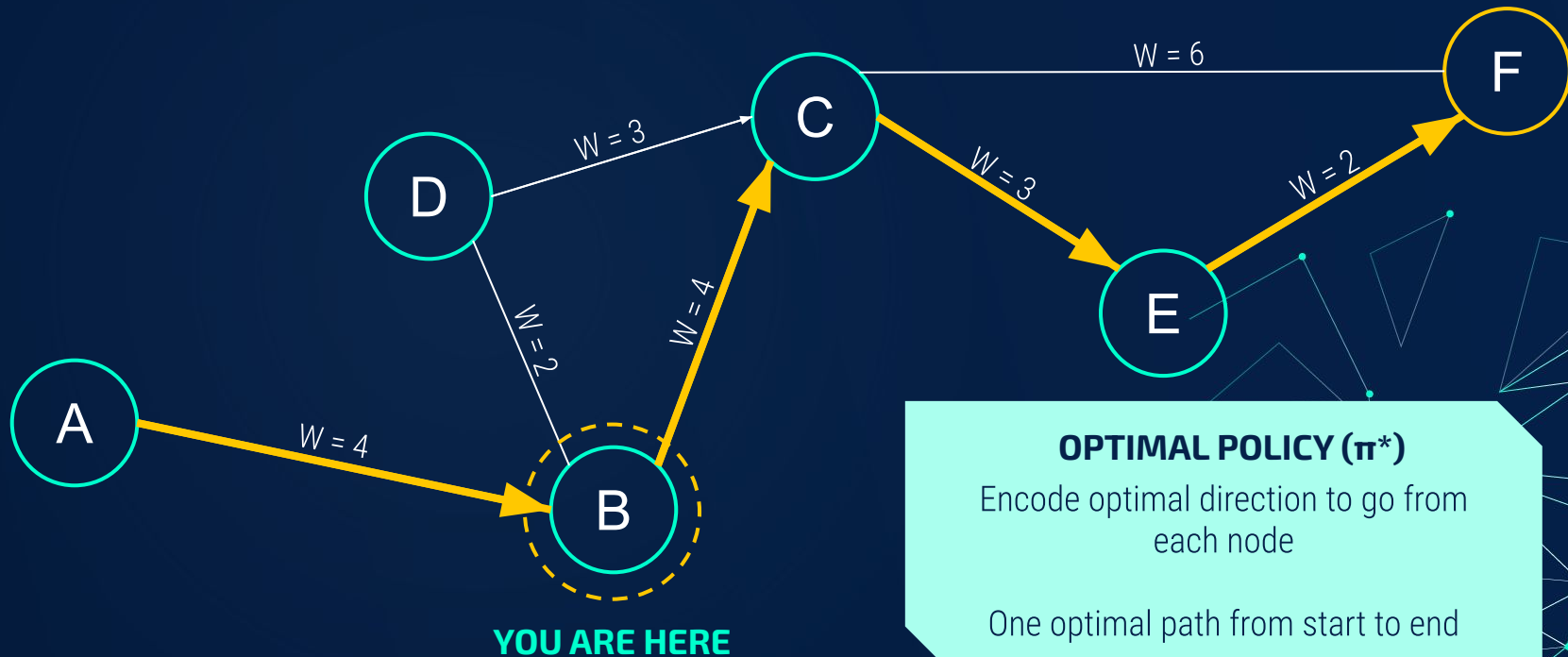
# Path Planning: Deterministic Graphs



## WHAT ARE THE WEIGHTS?

Examples: time, distance, gas usage, difficulty of traversing

# Path Planning: Deterministic Graphs



# Motivation

What happens if the edges aren't always there?

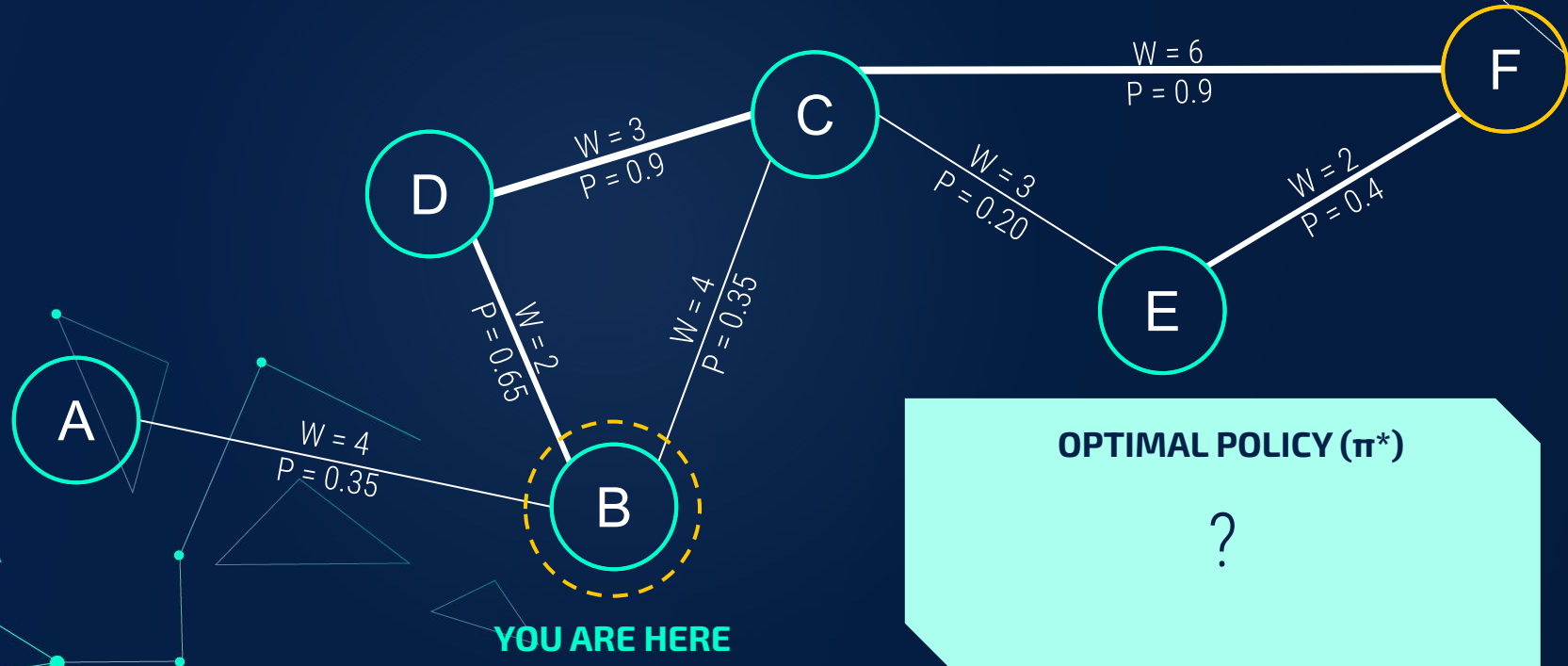
- Public Transportation Networks
  - What if a bus arrives  $\pm 5$  mins it's scheduled time?
  - How likely is it that a train gets delayed?
- Traffic
  - What if the traffic changes, or roads get blocked?
- Games
  - What if the player blocks the path?

**Assign each edge a probability!**

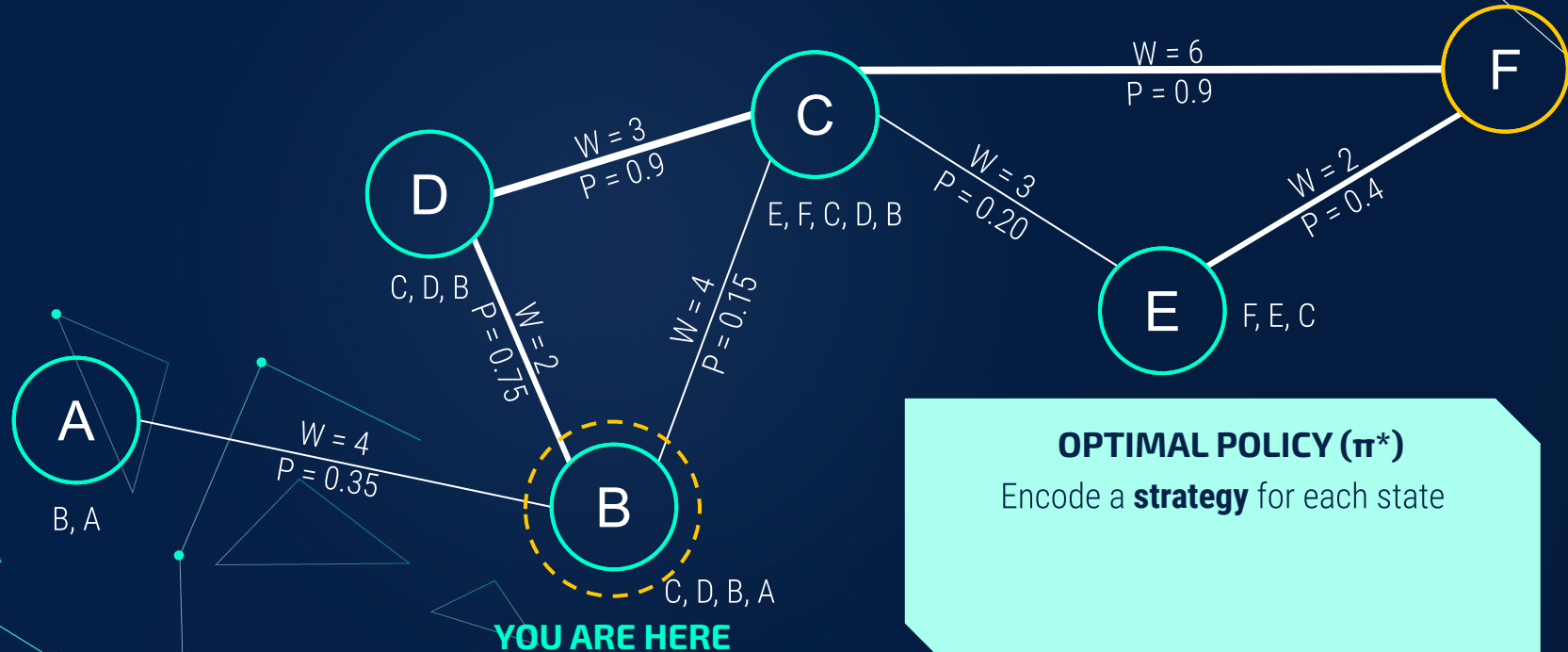
(aka create a **stochastic graph**)




# Path Planning: Stochastic Graphs



# Optimal Policy in Stochastic Graphs






---

# How does this relate to robotics?

---





# Robotics Navigation



# Robotics Navigation



# Robotics Navigation



**Nodes:** Landmarks

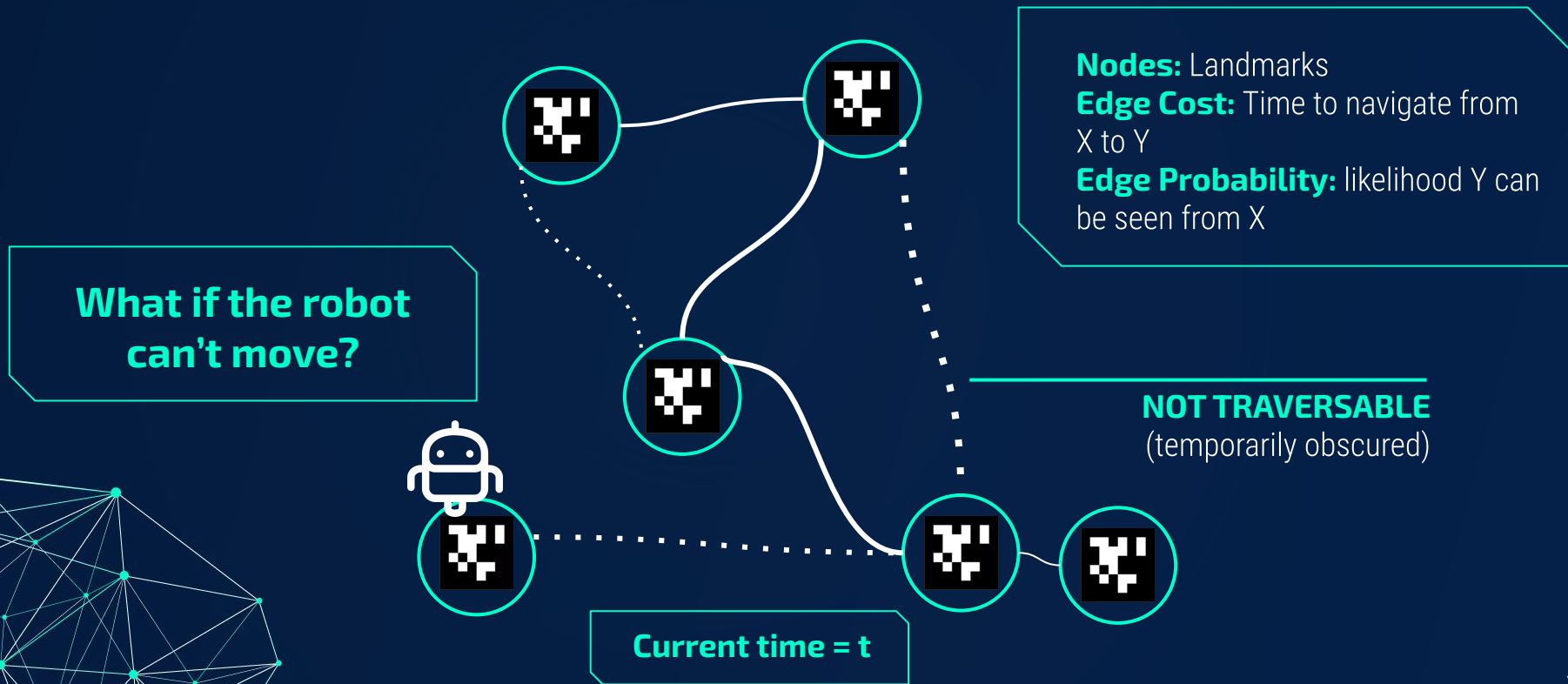
**Edge Cost:** Time to navigate from X to Y

**Edge Probability:** likelihood Y can be seen from X

**Generalized Form**



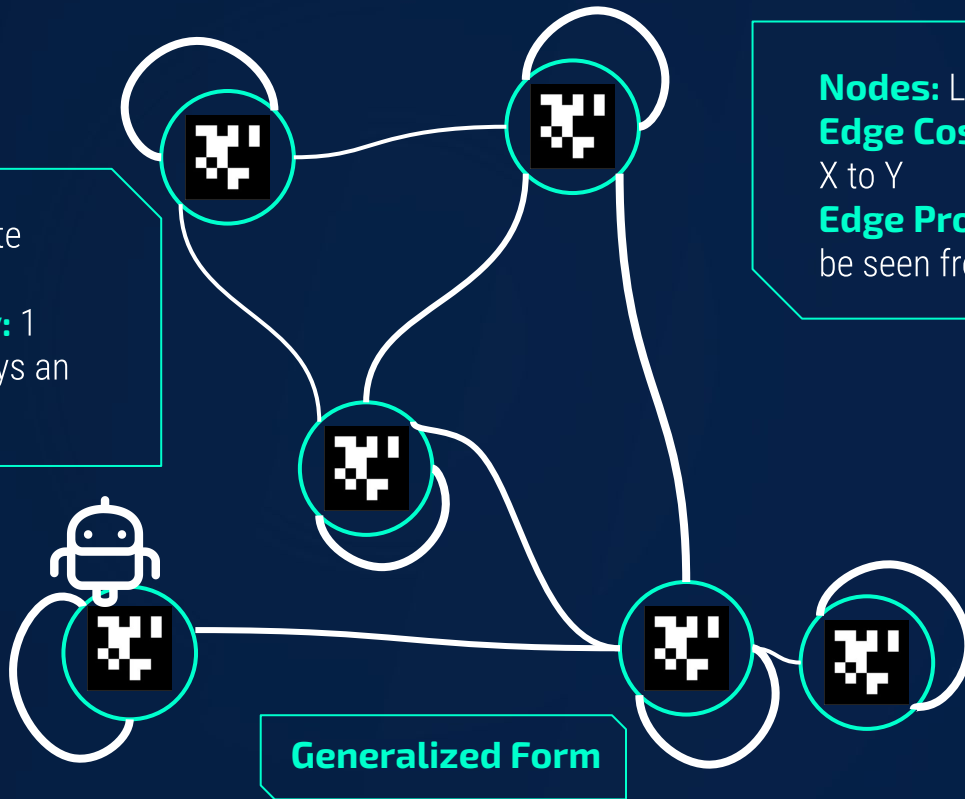
# Robotics Navigation



# Robotics Navigation

**Self-Edge Cost:** Discrete timestep of our problem  
**Self-Edge Probability:** 1  
(staying at a node is always an option)

**Nodes:** Landmarks  
**Edge Cost:** Time to navigate from X to Y  
**Edge Probability:** likelihood Y can be seen from X



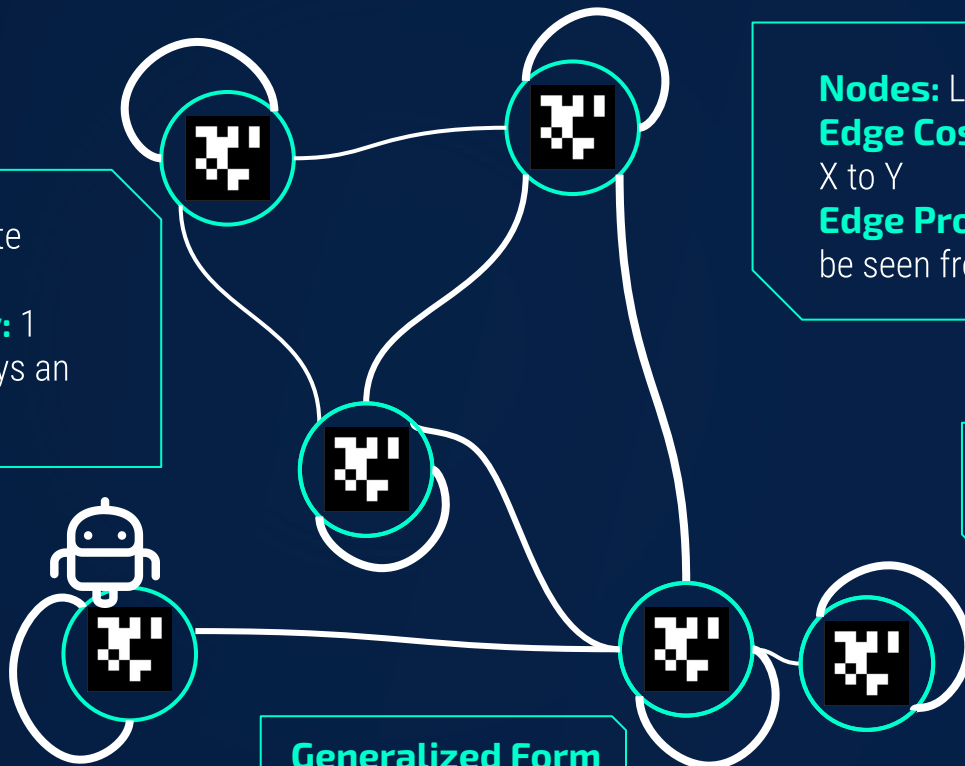
# Robotics Navigation


**Self-Edge Cost:** Discrete timestep of our problem  
**Self-Edge Probability:** 1  
(staying at a node is always an option)

**Nodes:** Landmarks  
**Edge Cost:** Time to navigate from X to Y  
**Edge Probability:** likelihood Y can be seen from X


Looks like a stochastic graph!

Generalized Form






# How do we find the solution?



---

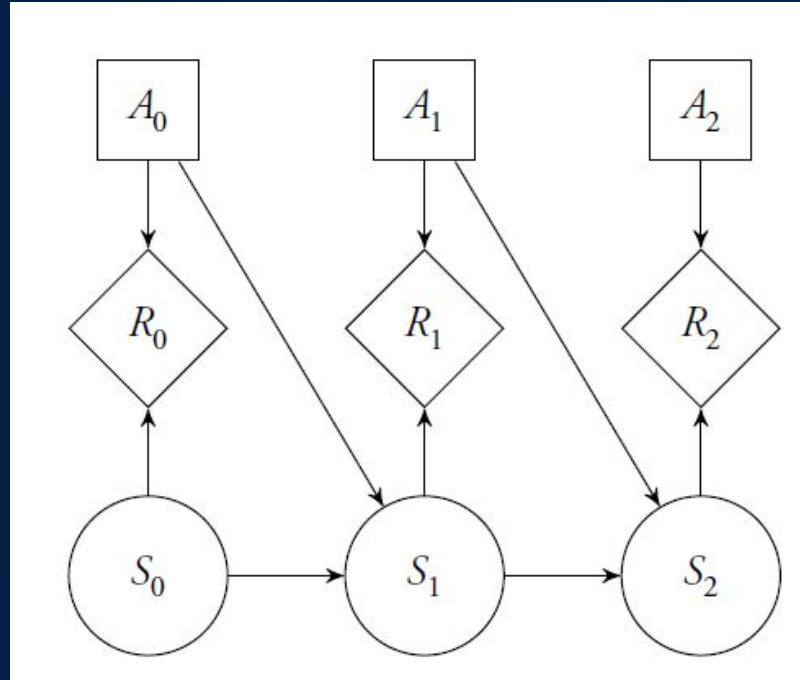
**We can use the Markov  
Decision Process (MDP)  
framing to help solve it!**

---





# Markov Decision Process (MDP) Overview



(Decision Making Under Uncertainty, Mykel J Kochenderfer, 2015)

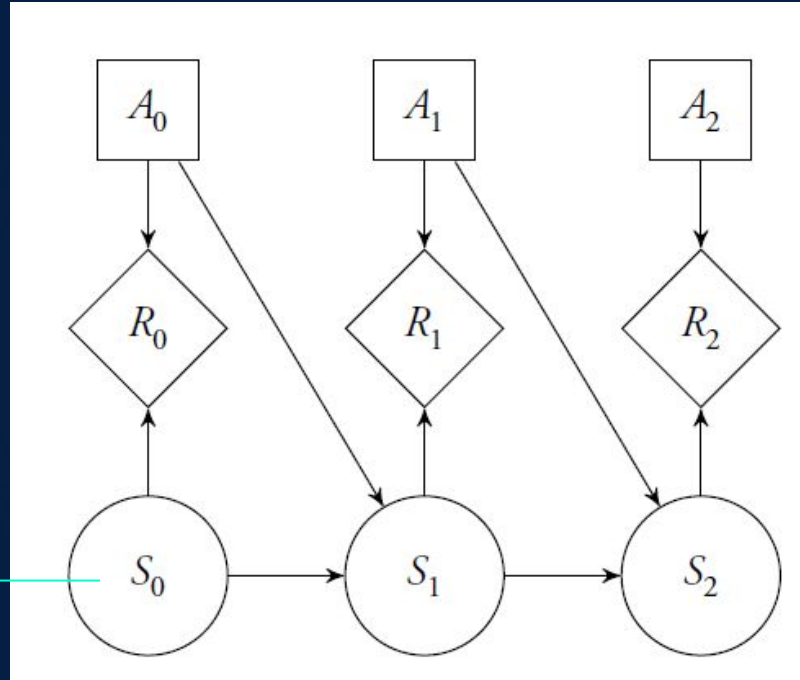


# Framing the MDP for the system

What are the ...

**STATES**

What are all the possible states of the agent?



# Framing the MDP for the system

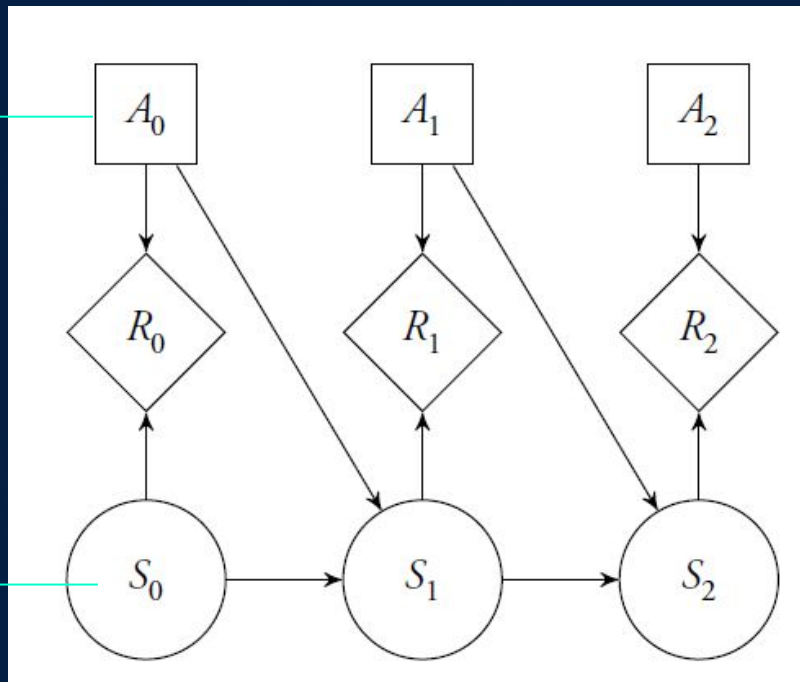
## ACTIONS

What are all the possible actions that the agent can take?

What are the ...

## STATES

What are all the possible states of the agent?



# Framing the MDP for the system

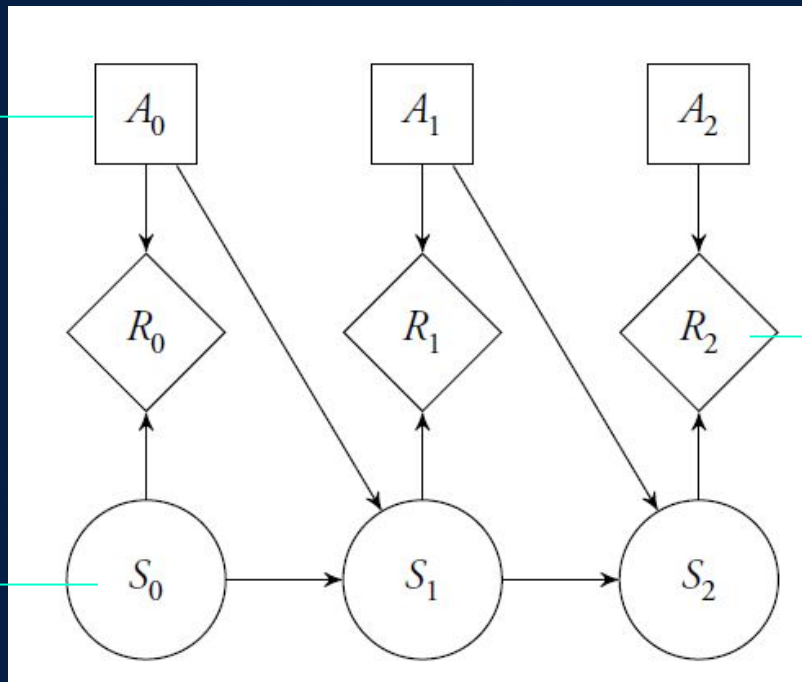
## ACTIONS

What are all the possible actions that the agent can take?

What are the ...

## STATES

What are all the possible states of the agent?



## REWARDS

How does the reward depend on agent's state and action?

# Framing the MDP for the system

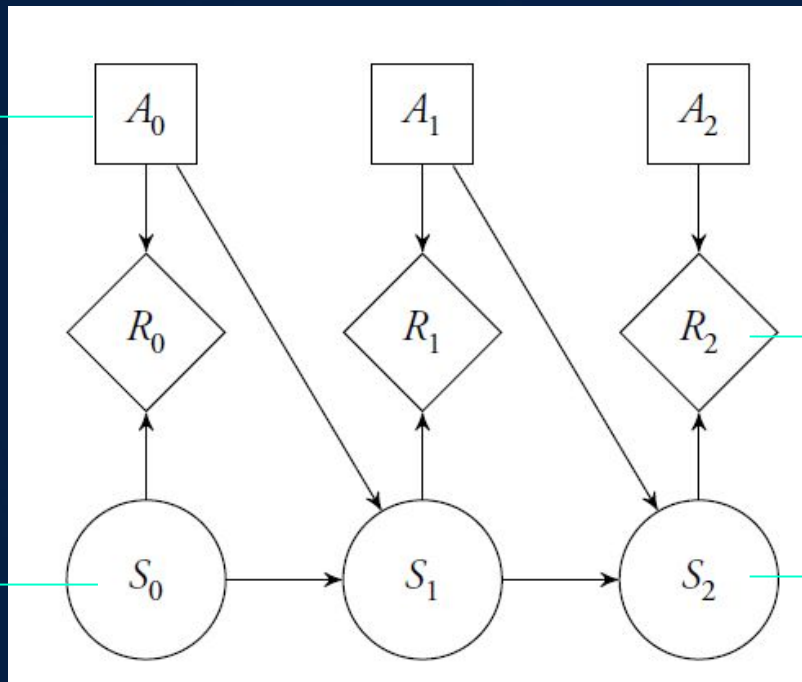
## ACTIONS

What are all the possible actions that the agent can take?

What are the ...

## STATES

What are all the possible states of the agent?



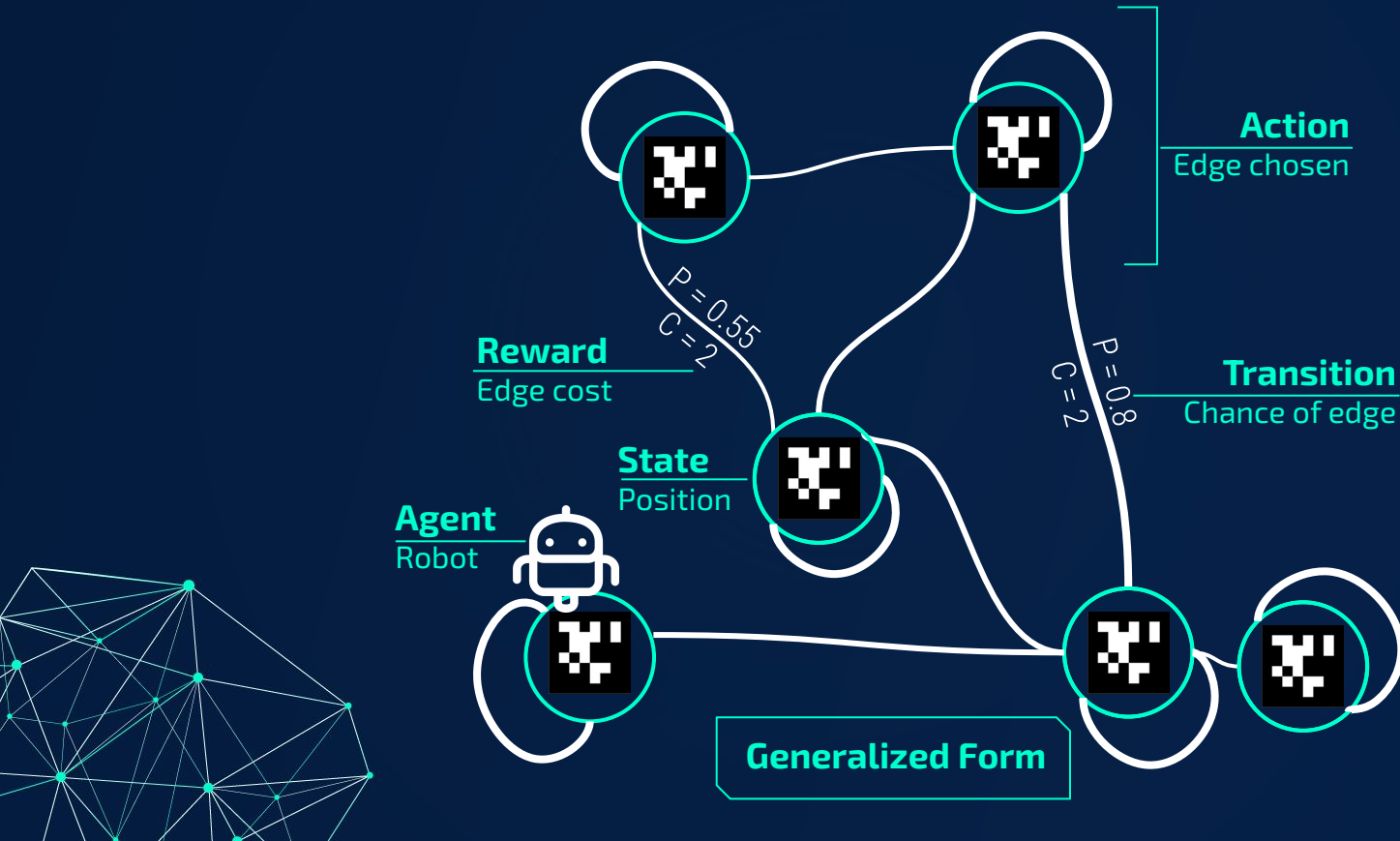
## REWARDS

How does the reward depend on agent's state and action?

## TRANSITIONS

How does the agent move to the next state, based on the current state and action?

# Robotics Navigation & MDP



# Rewards & Utility

## Reward

The expected benefit of taking an action at a state

The expected benefit of taking a series of actions along a path

- The sum of rewards gained along the path

## Utility

# Optimal Policy, Maximum Utility

## Rationality


An agent always takes the action that maximizes its reward

## Utility and Policy

Utility is the sum of rewards gained by following a policy for some time

## Optimal Policy

Under the optimal policy, the utility of following the policy is at the maximum



**Optimal Policy: The set of actions that maximizes utility**



# Utility & Time Horizons



## GREEDY

Just considers the reward for the next step.



## TIME HORIZONS

Utility is defined as the sum of the rewards over  $T$  timesteps.

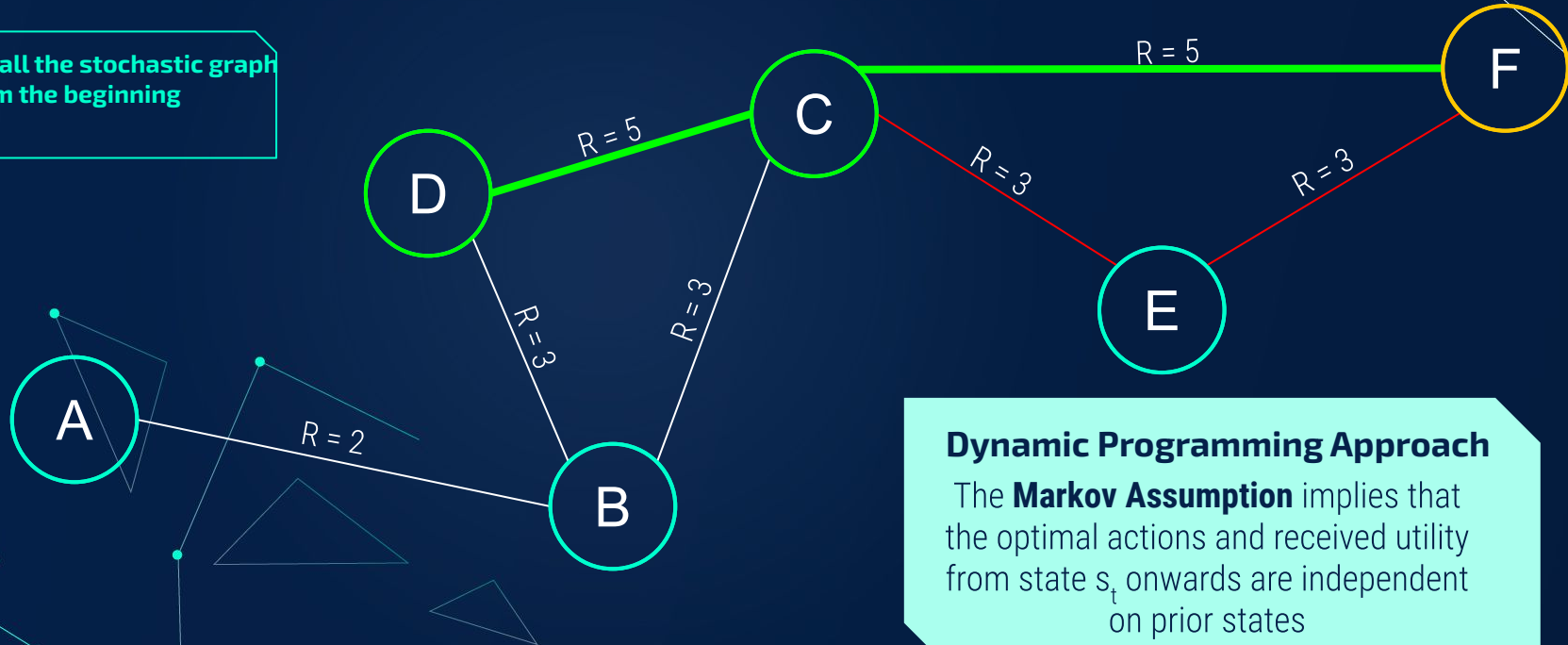


## INFINITE

Considers the reward for an infinite future.

# Dynamic Programming (DP) & MDP

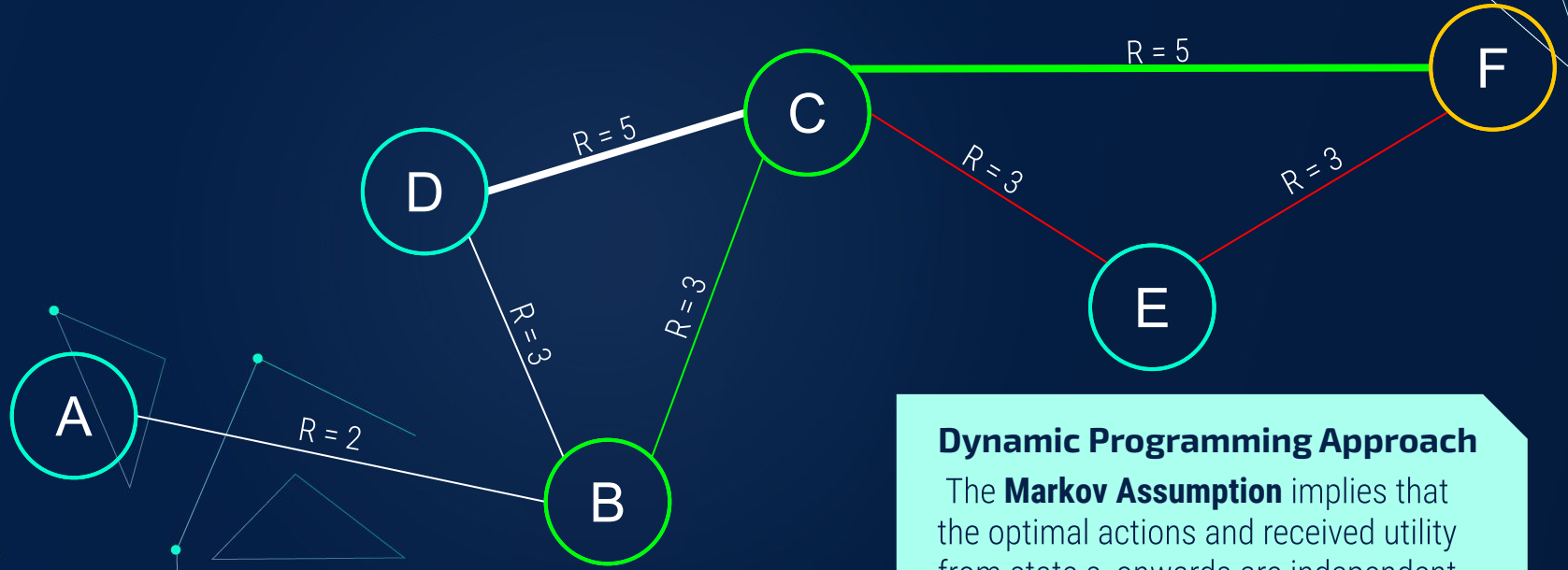
Recall the stochastic graph from the beginning



## Dynamic Programming Approach

The **Markov Assumption** implies that the optimal actions and received utility from state  $s_t$  onwards are independent on prior states

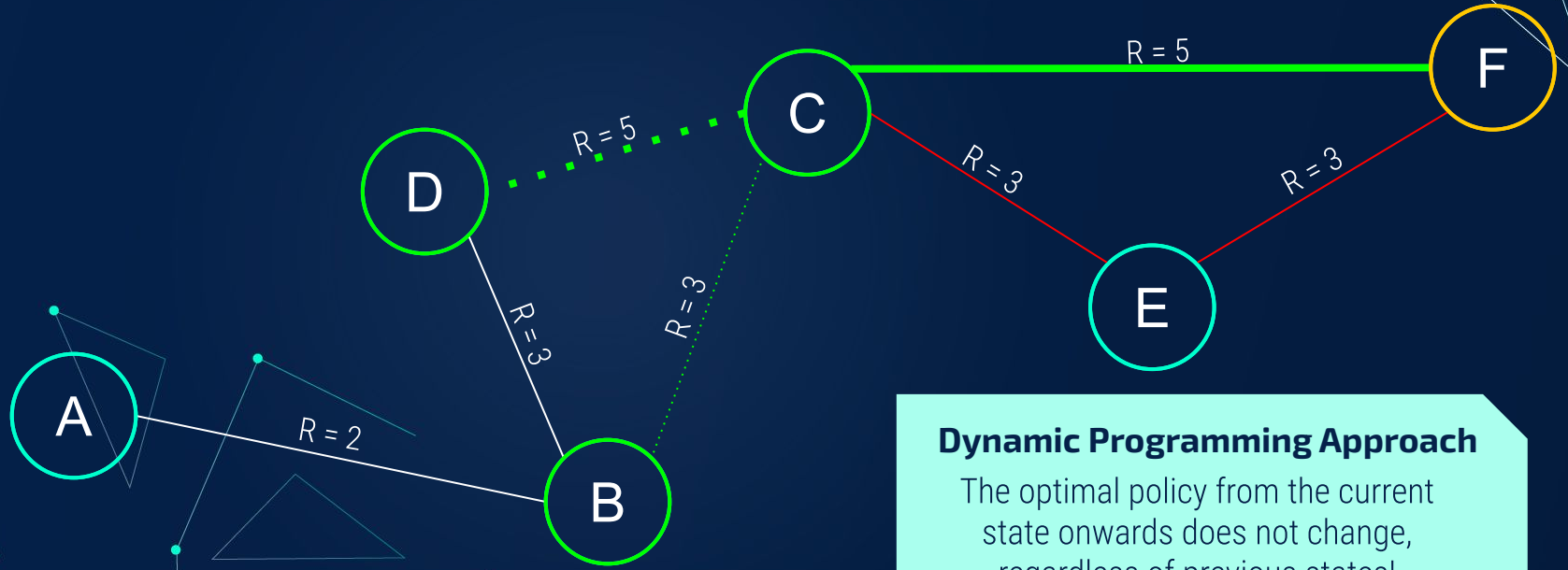
# Dynamic Programming (DP) & MDP



## Dynamic Programming Approach

The **Markov Assumption** implies that the optimal actions and received utility from state  $s_t$  onwards are independent on prior states

# Dynamic Programming (DP) & MDP



## Dynamic Programming Approach

The optimal policy from the current state onwards does not change, regardless of previous states!

# Solving the Problem

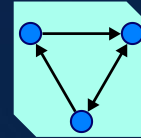
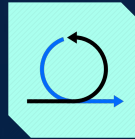


## Transition Probability Matrix

Finding the probability that these nodes are connected

Find the optimal value function to get the optimal policy  
OR  
Find the optimal policy

## Value Iteration or Policy Iteration?

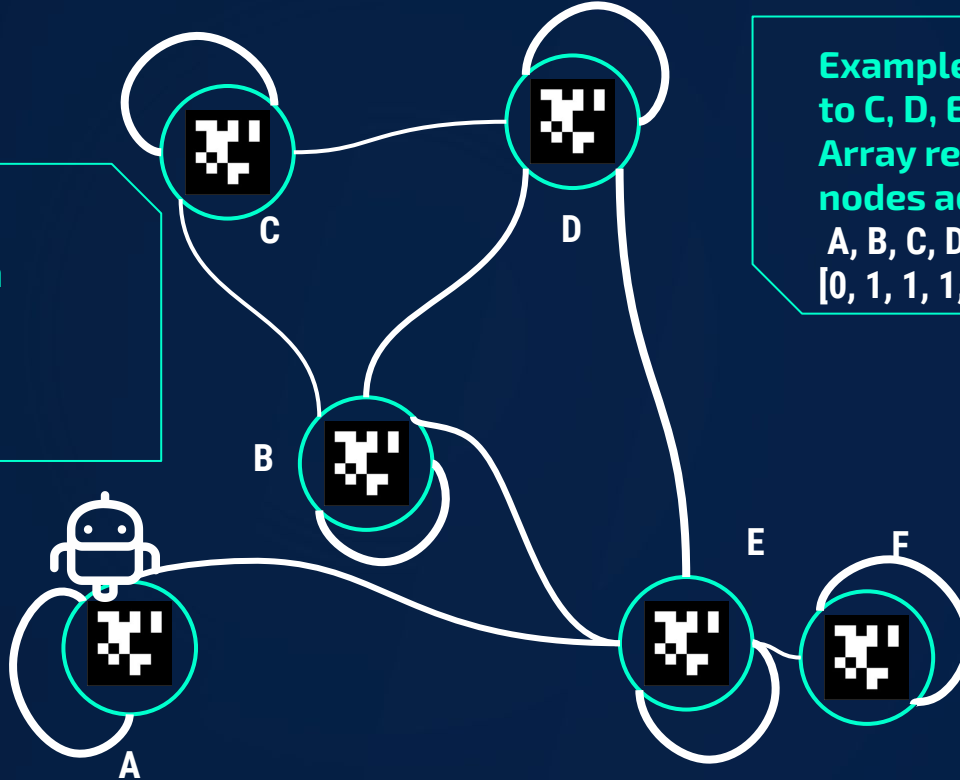


## Find the optimal "instruction"

With the optimal value function or policy, we can calculate how the robot chooses where to go based off its current position

# Solving - Adjacent Nodes

Can represent which nodes are adjacent in matrix format



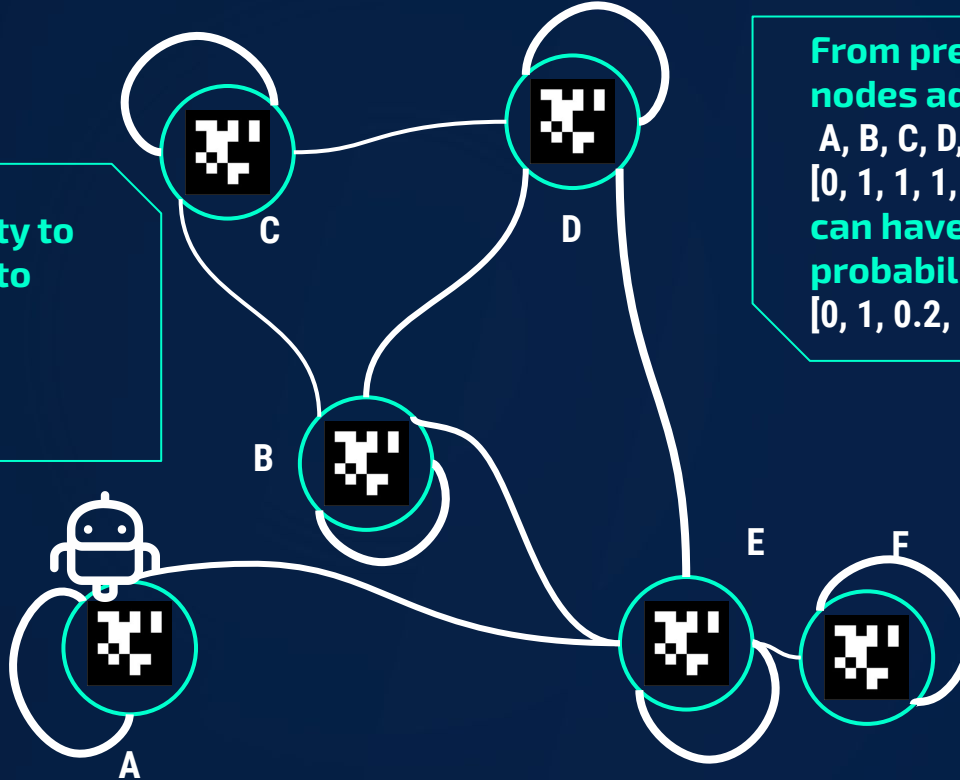
Example :Node B is adjacent to C, D, E and itself.

Array representation for nodes adjacent to B:

A, B, C, D, E, F  
[0, 1, 1, 1, 1, 0]

# Solving - Probability

What is the probability to move from one node to another?




From previous example, the nodes adjacent to B

A, B, C, D, E, F


$[0, 1, 1, 1, 1, 0]$

can have a possible set of probabilities of


$[0, 1, 0.2, 0.5, 0.3, 0]$



# Why does the probability matrix matter?








---

**Because it will help  
calculate how good the  
current state is!**

**(and what might happen after taking some action)**

---

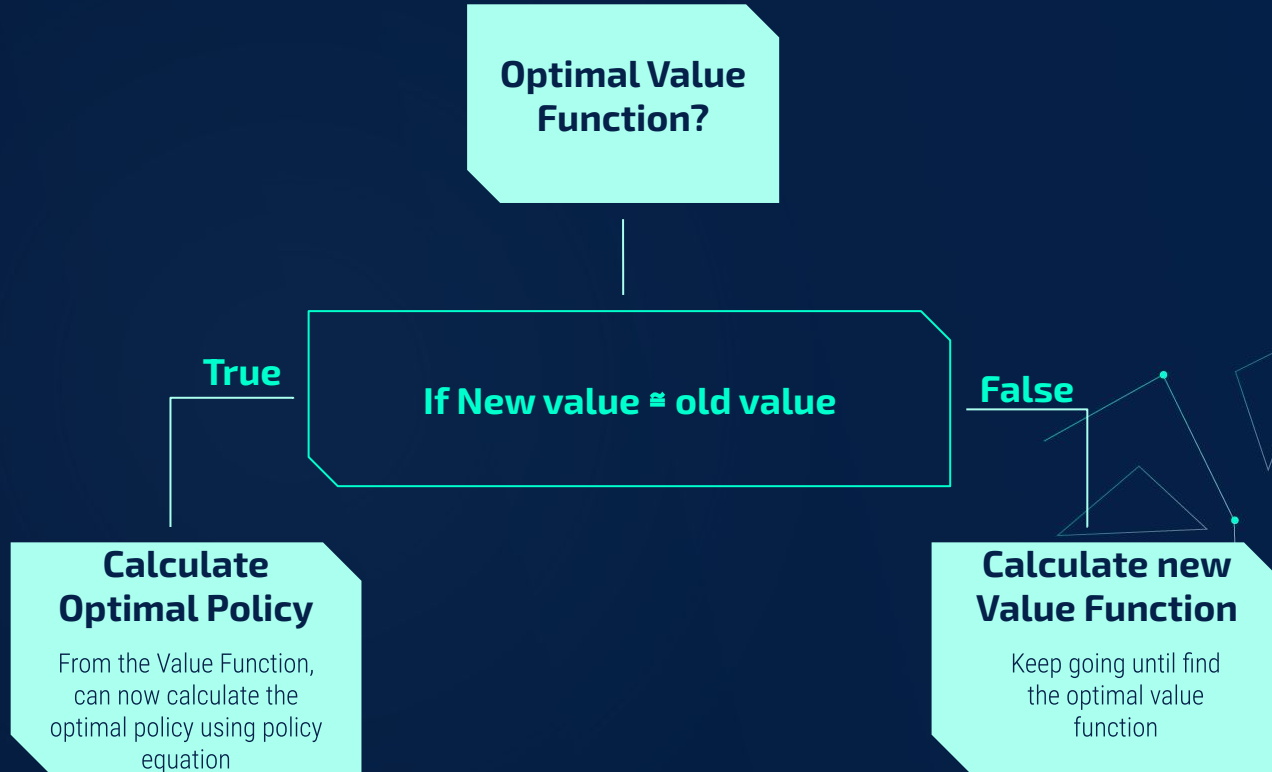




# Value Iteration

Now we actually start finding the optimal “instruction” for the robot!

# Solving - Value Iteration

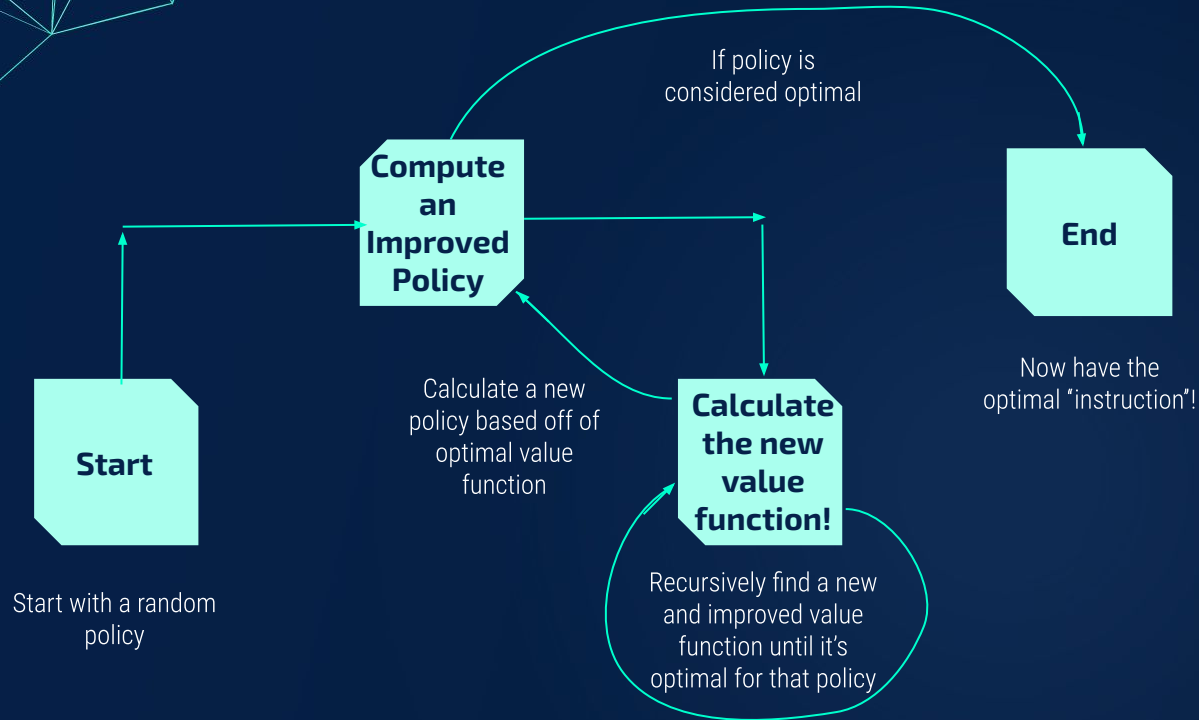




# Policy Iteration

Similar to Value Iteration EXCEPT...

# Solving- Policy Iteration



# Comparing Value & Policy Iteration

## Value Iteration

Finds optimal value function  
then computes optimal policy

Evaluates current policy, will  
compute a new policy from an  
optimal value function, then  
uses that function to find a  
new & improved policy

## Policy Iteration



# Comparing Value & Policy Iteration

## Value Iteration

Only has one loop

Has smaller time complexity

Has two loops

Has larger time complexity,  
but converges faster

## Policy Iteration





# Thanks for Listening!

Questions?

## Resources used

- **Robotics applications:** Briggs et al. "Expected Shortest Paths for Landmark-Based Robot Navigation" 2004
- **Markov Decision Process Background:** *Decision Making Under Uncertainty* by Mykel J. Kochenderfer 1980