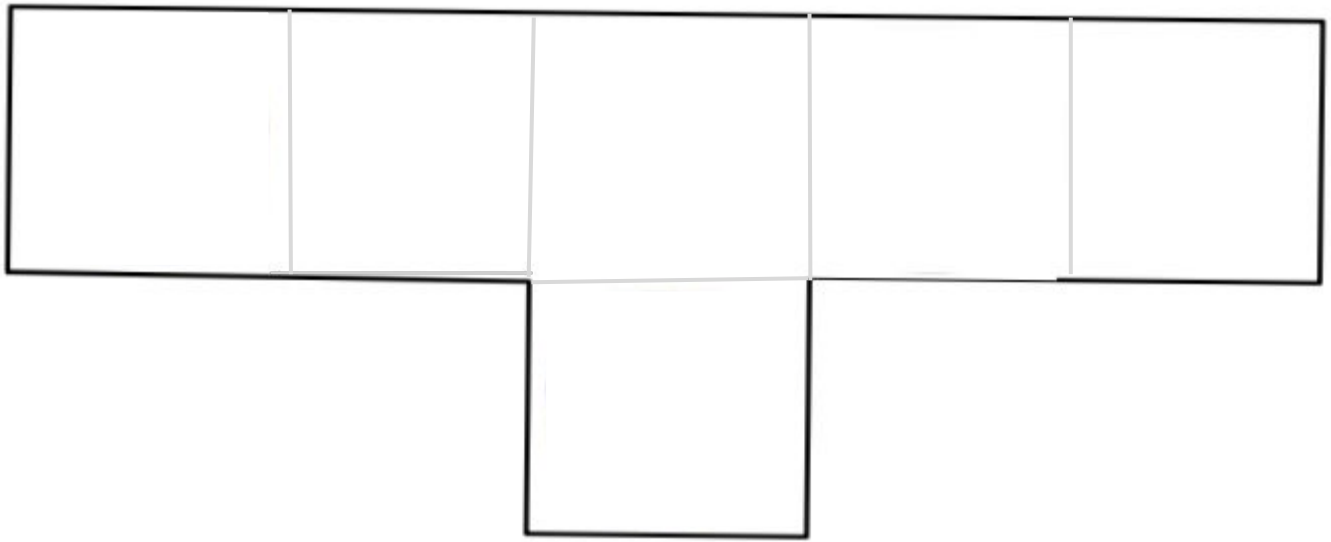
Multi-Agent Path Planning

A comparative study of Prioritized Planning and Conflict-Based Search

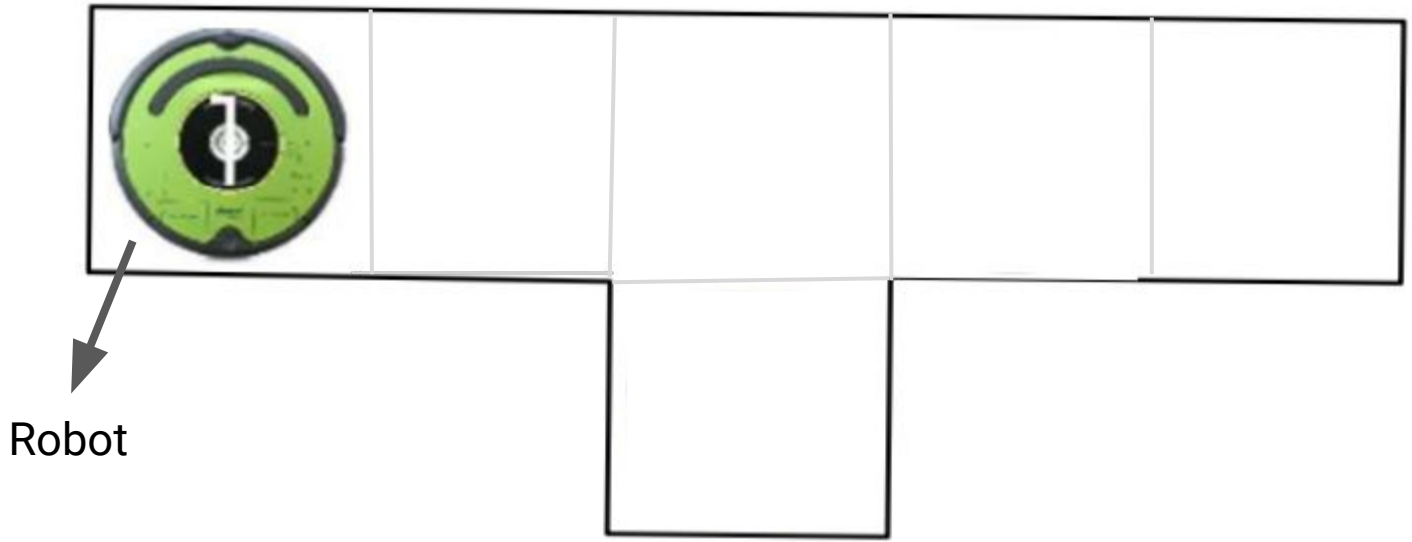
Muhammad Saud Ul Hassan

Path Planning



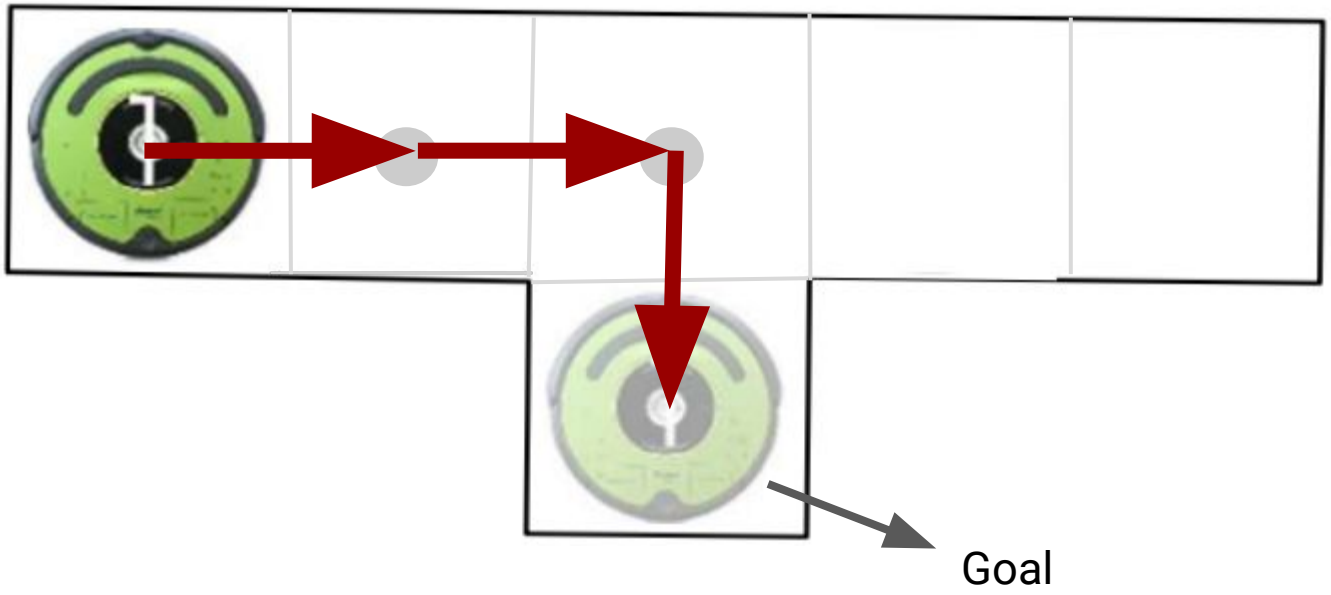
Adapted from: W. Hoenig *et al.*, “Multi-Agent Path Finding with Kinematic Constraints”

Path Planning



Adapted from: W. Hoenig *et al.*, “Multi-Agent Path Finding with Kinematic Constraints”

Path Planning



Adapted from: W. Hoenig *et al.*, “Multi-Agent Path Finding with Kinematic Constraints”

Single-Agent Path Planning

Given

- Map of environment
- Agent and its goal

What we want?

A path to the goal avoiding obstacles

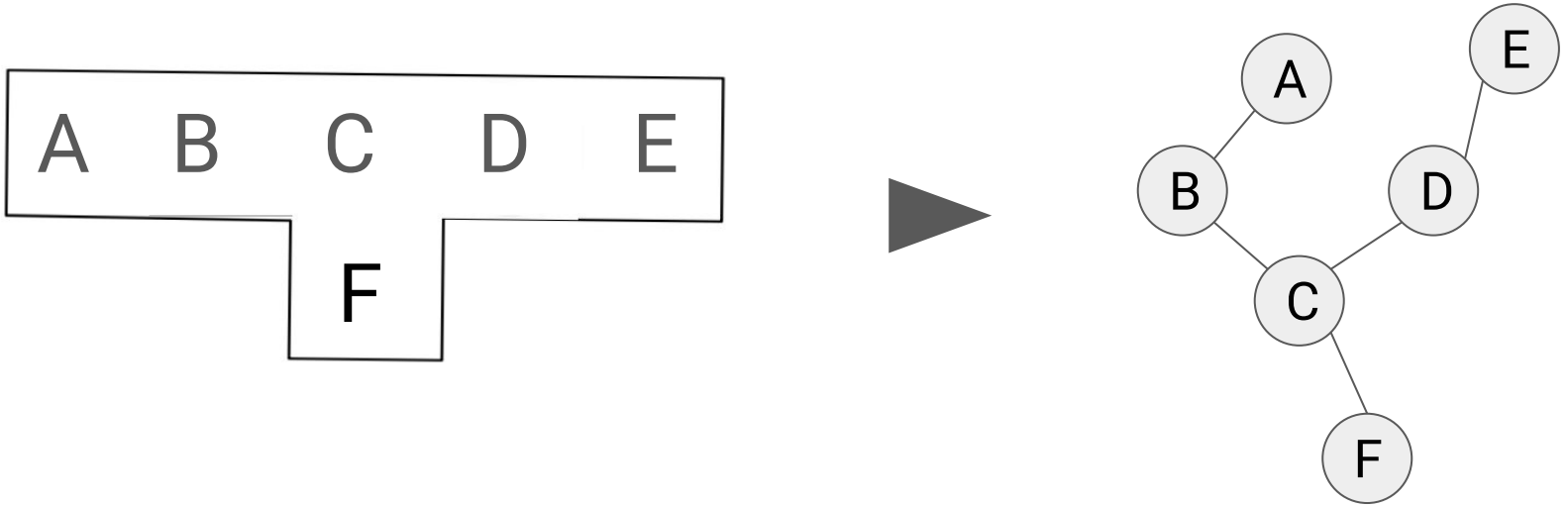
Single-Agent Path Planning

How to Solve?

Key idea:

- **Abstract** away the problem into a state-space graph
- Use a **shortest-path algorithm** (e.g. A*)

Single-Agent Path Planning



Multi-Agent Path Finding (MAPF)

Given

- Multiple agents, each with a goal
- Map of environment

What we want?

A path for each agent to its goal, avoiding:

- Collisions with obstacles
- Collisions with other agents

Multi-Agent Path Finding (MAPF)

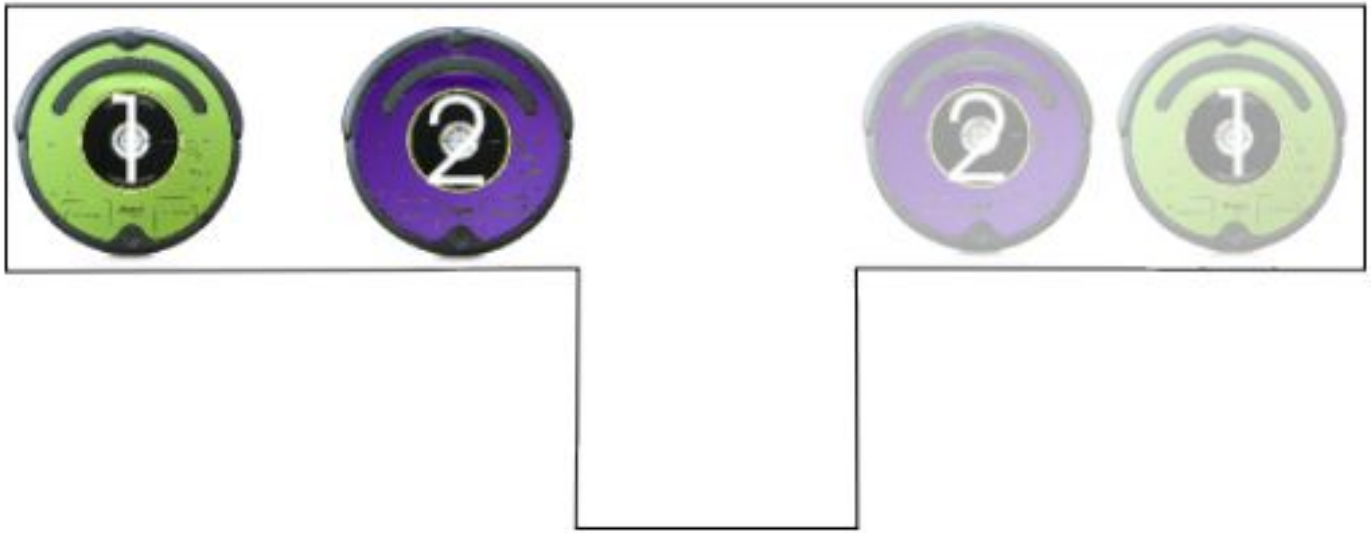


Image Source: W. Hoenig *et al.*, “Multi-Agent Path Finding with Kinematic Constraints”

Multi-Agent Path Finding (MAPF)

How to Solve?

Same as Single-Agent Planning:

- **Abstract** away the problem into a state-space graph
- Use a **shortest-path algorithm**

Multi-Agent Path Finding (MAPF)

However,

In single-agent planning, we had to look out for collisions in space only

Here,

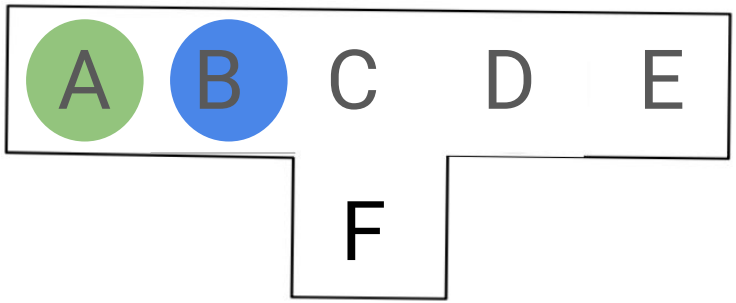
We need to look out for collisions in both space and time

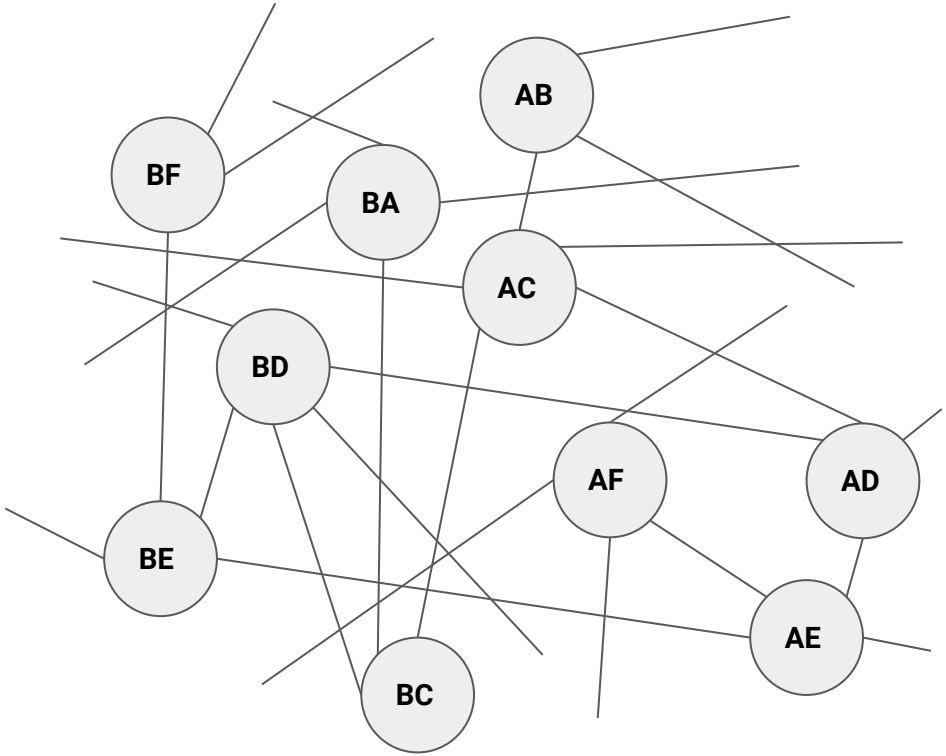
Multi-Agent Path Finding (MAPF)

How to encode the temporal nature of the problem in the state-state graph?

One Idea… don’t! (**Spoiler Alert:** Bad Idea!)

Idea 1: Joint Location Space Planning





Idea 1: Joint Location Space Planning

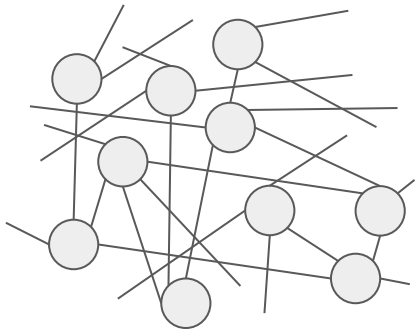
For a map of size |C| with |R| agents, the # of vertices will be:

$$|V| = \frac{|C|!}{(|C| - |R|)!}$$

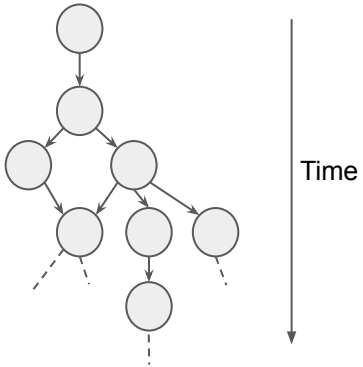
In words, the number of vertices is exponential in the number of robots

Idea 2: Space-time Tree

- Encode time as a state in the graph
- Since time progresses linearly ⇒ the graph simplifies to a tree



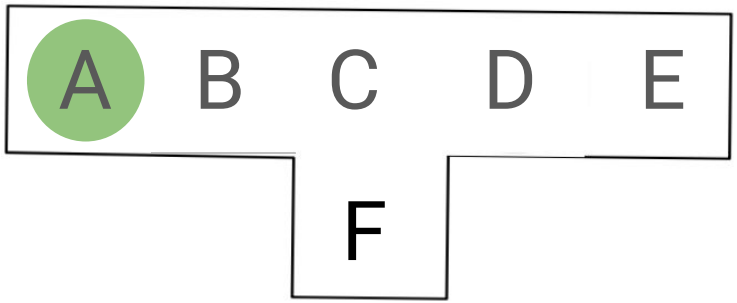
Joint Location Space Graph

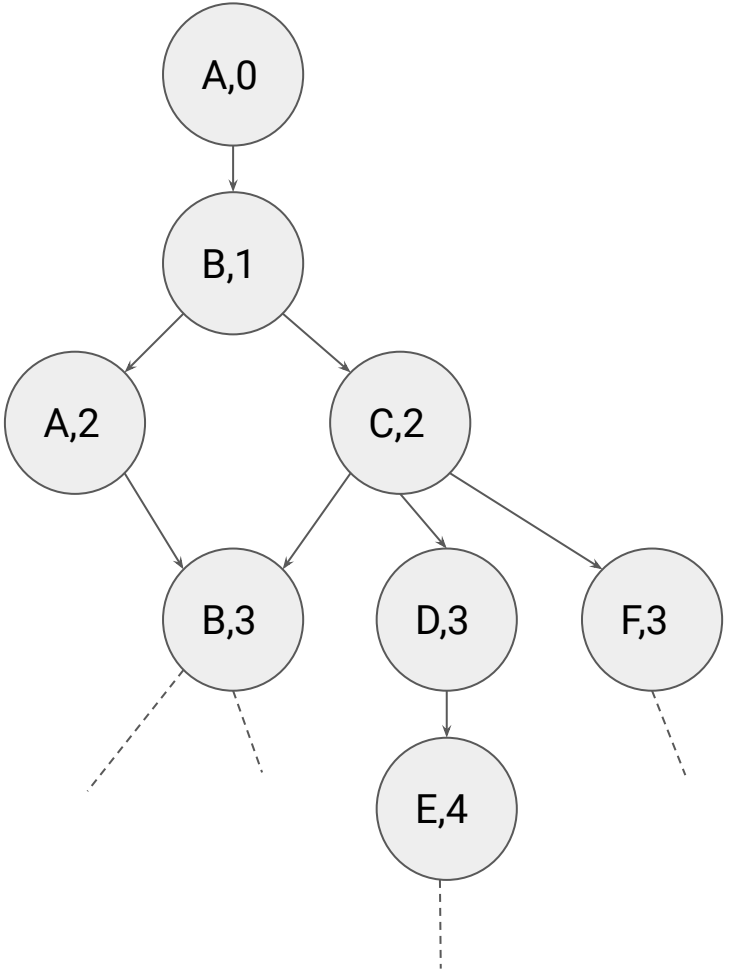


Space-time Tree

Space-time Tree

[Single-Agent Example]





Space-time Tree

How to find shortest path on a space-time tree?

- **Space-time A*:** Just A* with some modifications

Does it work on multi-agent problems?

Prioritized Planning

- Space-time A* with a twist (to account for multiple agents)

Key Idea:

- Pick a robot you want to favour.
- That robot can plan its path freely, and it’s the other robot’s headache to not collide

Prioritized Planning (Algorithm)

1. Assign priority i to each agent
2. Take agent r_i
   a. Plan the shortest-path to r_i's goal using **Space-time A***
   b. **Add constraints** that constraint the robots $r_j \, (j > i)$ to not be at the same location as r_i at time t

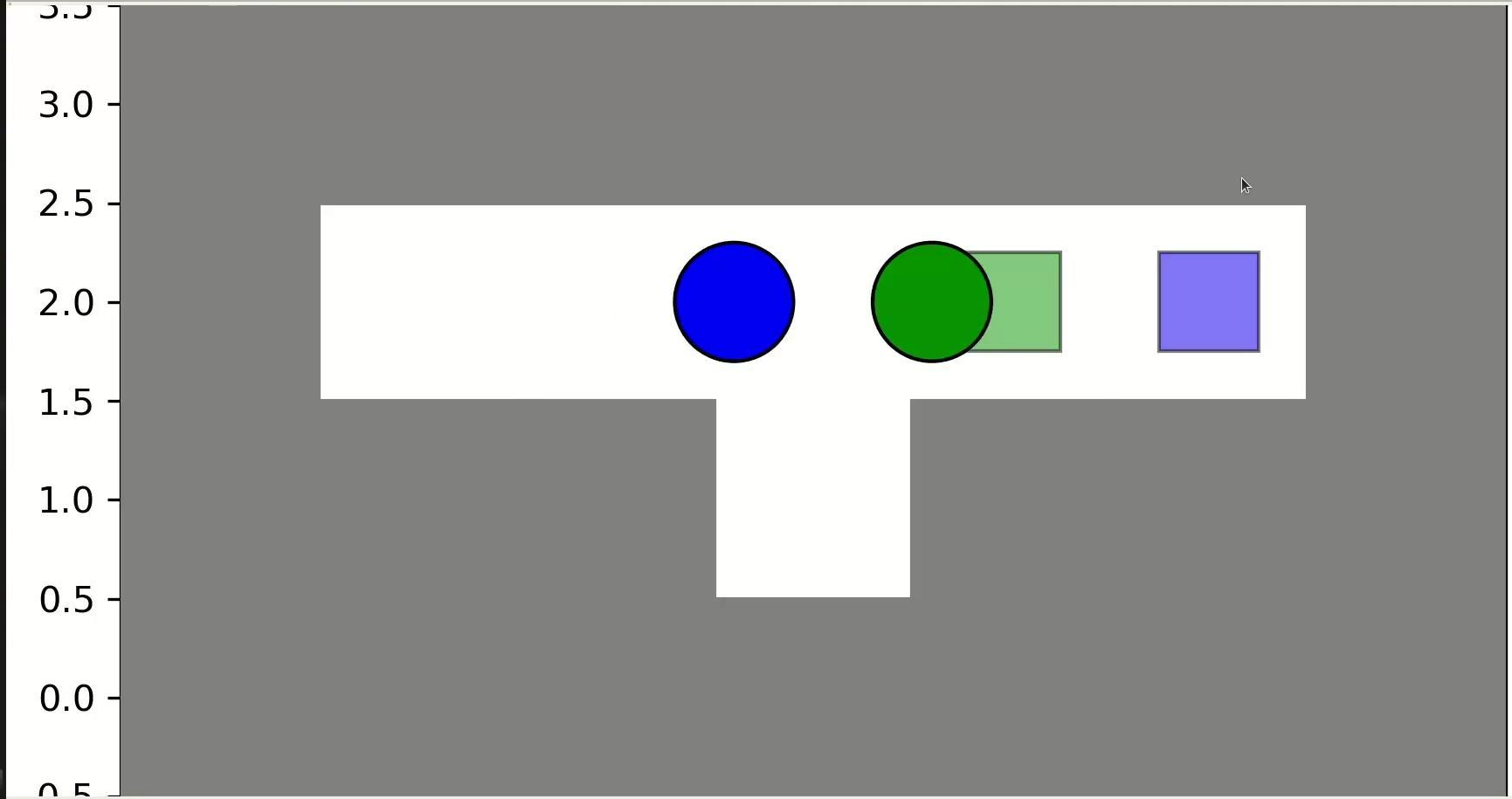
* There are also edge collisions that we need to take care of

It works... right?

It works, but in this case.

The agent priorities were just right for it to work. **What if I prioritize the other agent?**

Figure 1



Prioritized Planning

Prioritized Planning is **fast**, but

- It **lacks completeness and optimality**

This is where the **alternative, Conflict-based Search**, comes in

Conflict-based Search

Key Idea:

- Just plan a path to each agent’s goal
- We can figure out the collisions later

Conflict-based Search

Key Idea:

- Just plan a path to each agent's goal [use **space-time A***]
- We can figure out the collisions later [use **best-first search over a binary constraint graph**]

Constraints: {}

Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”

Constraints: {}
Solution: 1: [A,C,E]
2: [B,C,D]

Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”

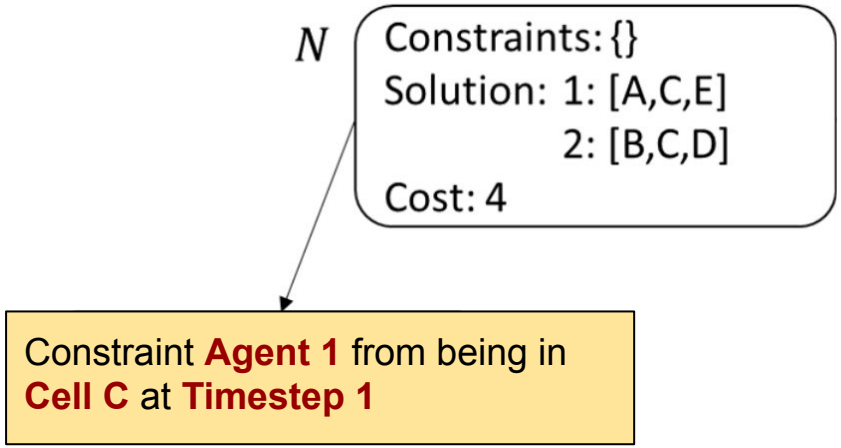
Constraints: {}
Solution: 1: [A,C,E]
2: [B,C,D]
Cost: 4

Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”

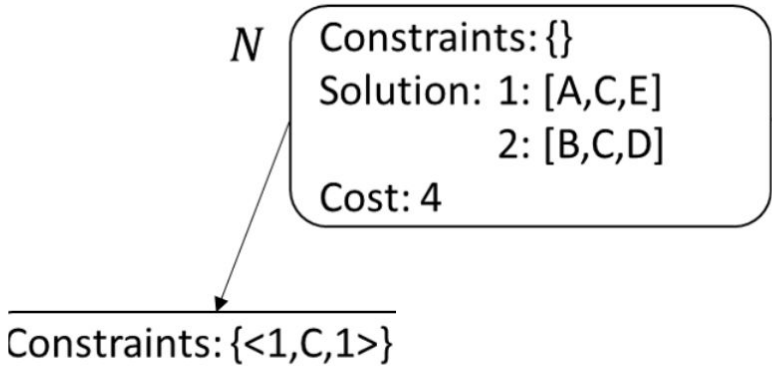
Constraints: {}
Solution: 1: [A,C,E]
2: [B,C,D]
Cost: 4

Collision at timestep 1

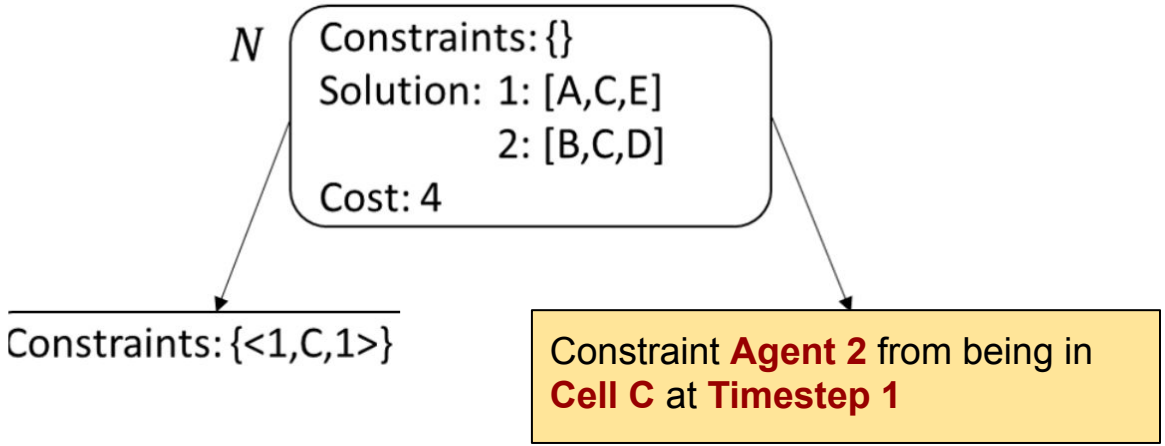
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



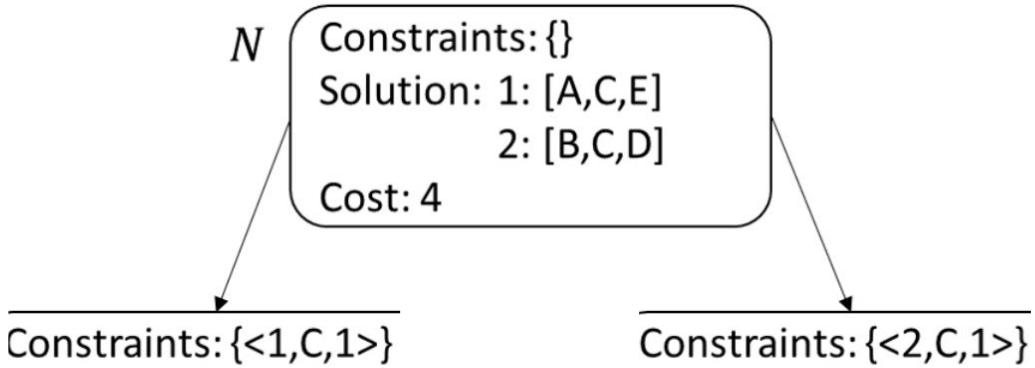
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



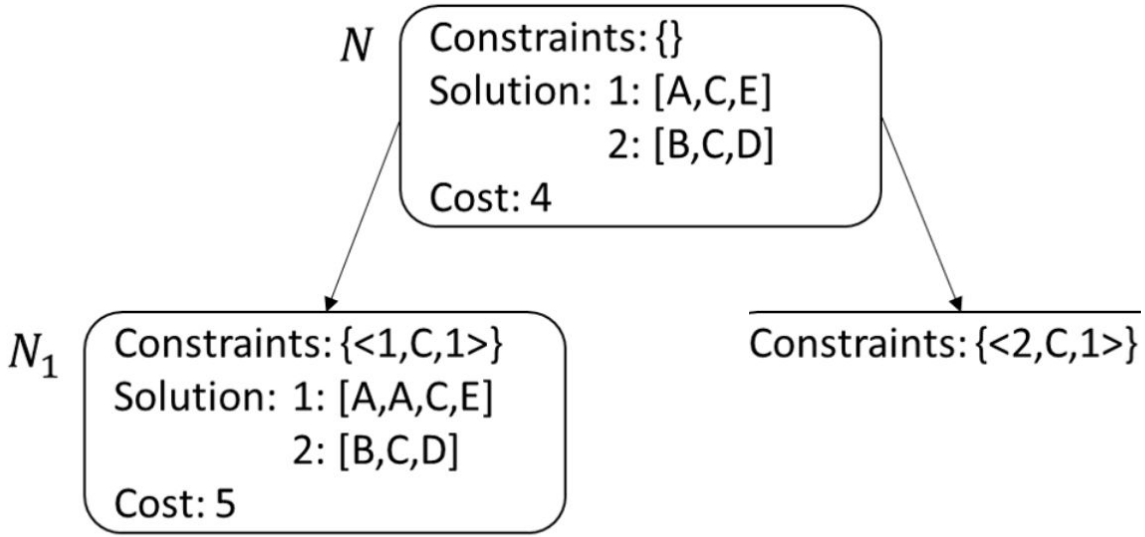
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



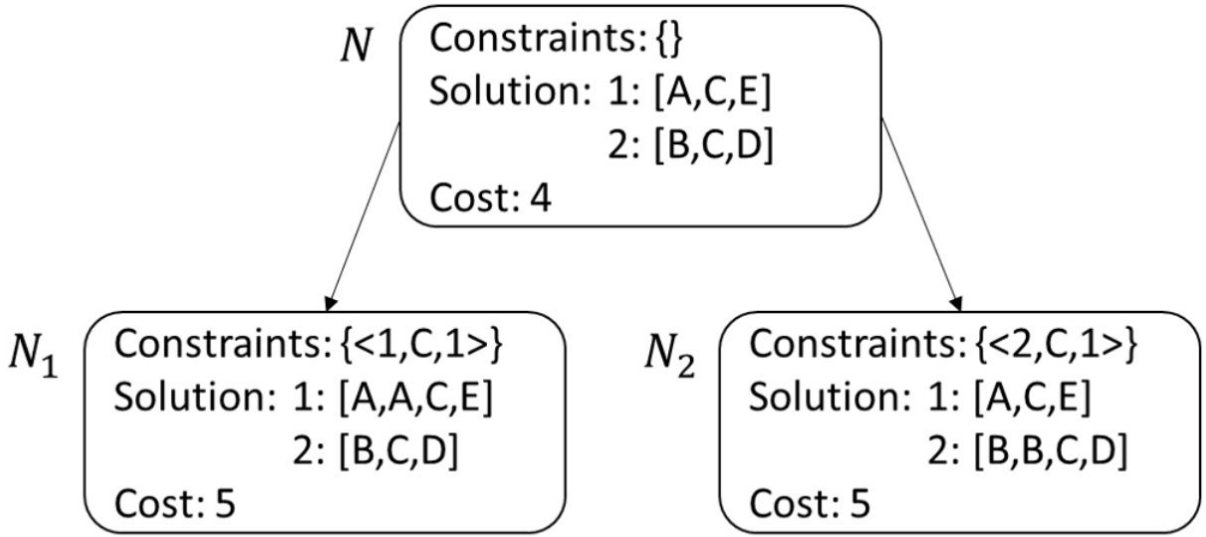
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



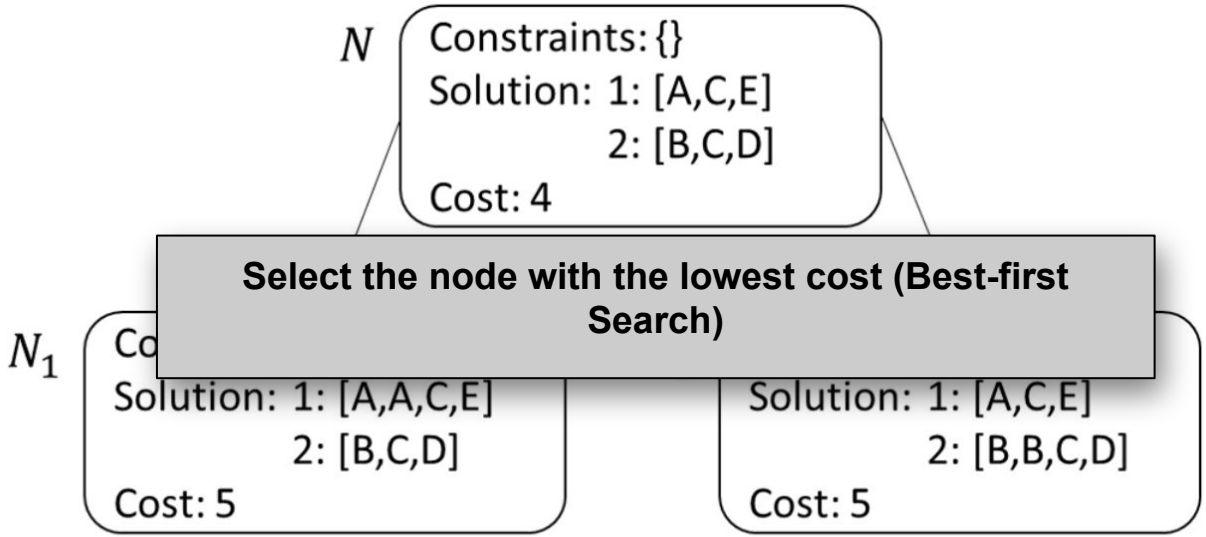
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



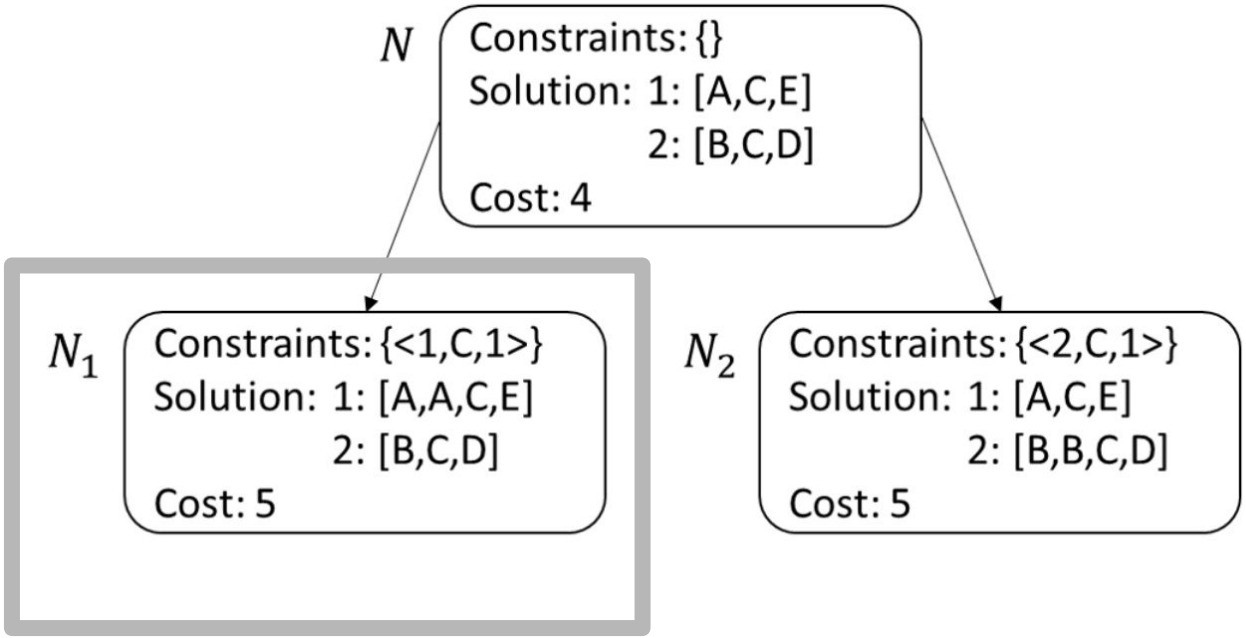
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



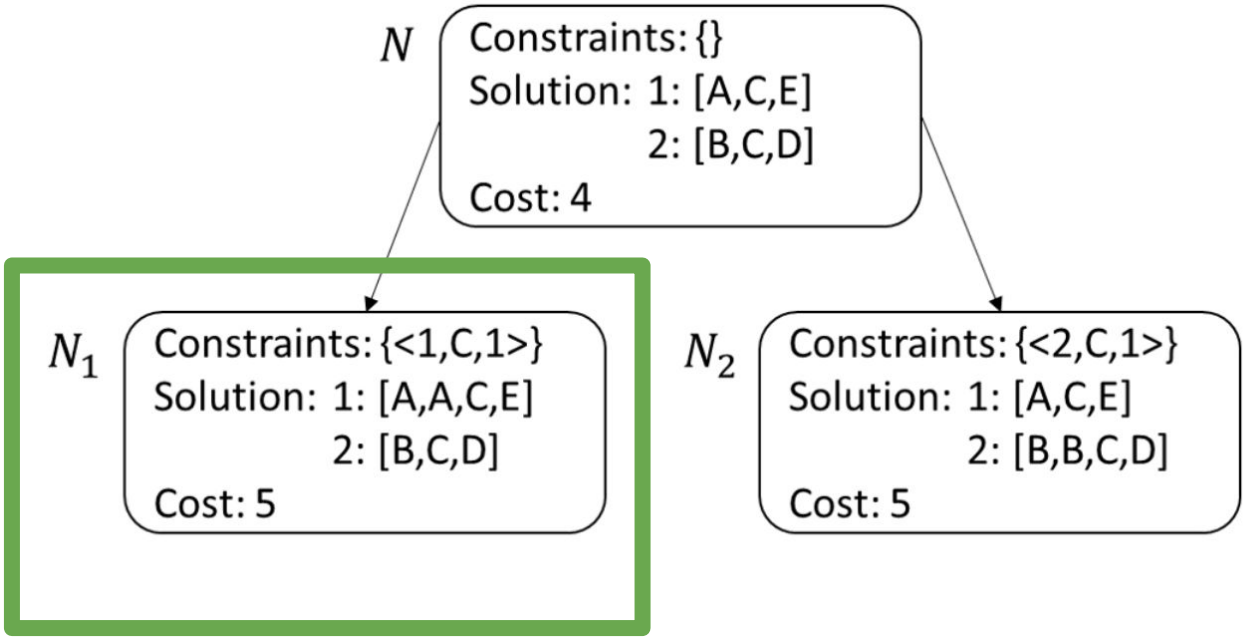
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”



GOAL!

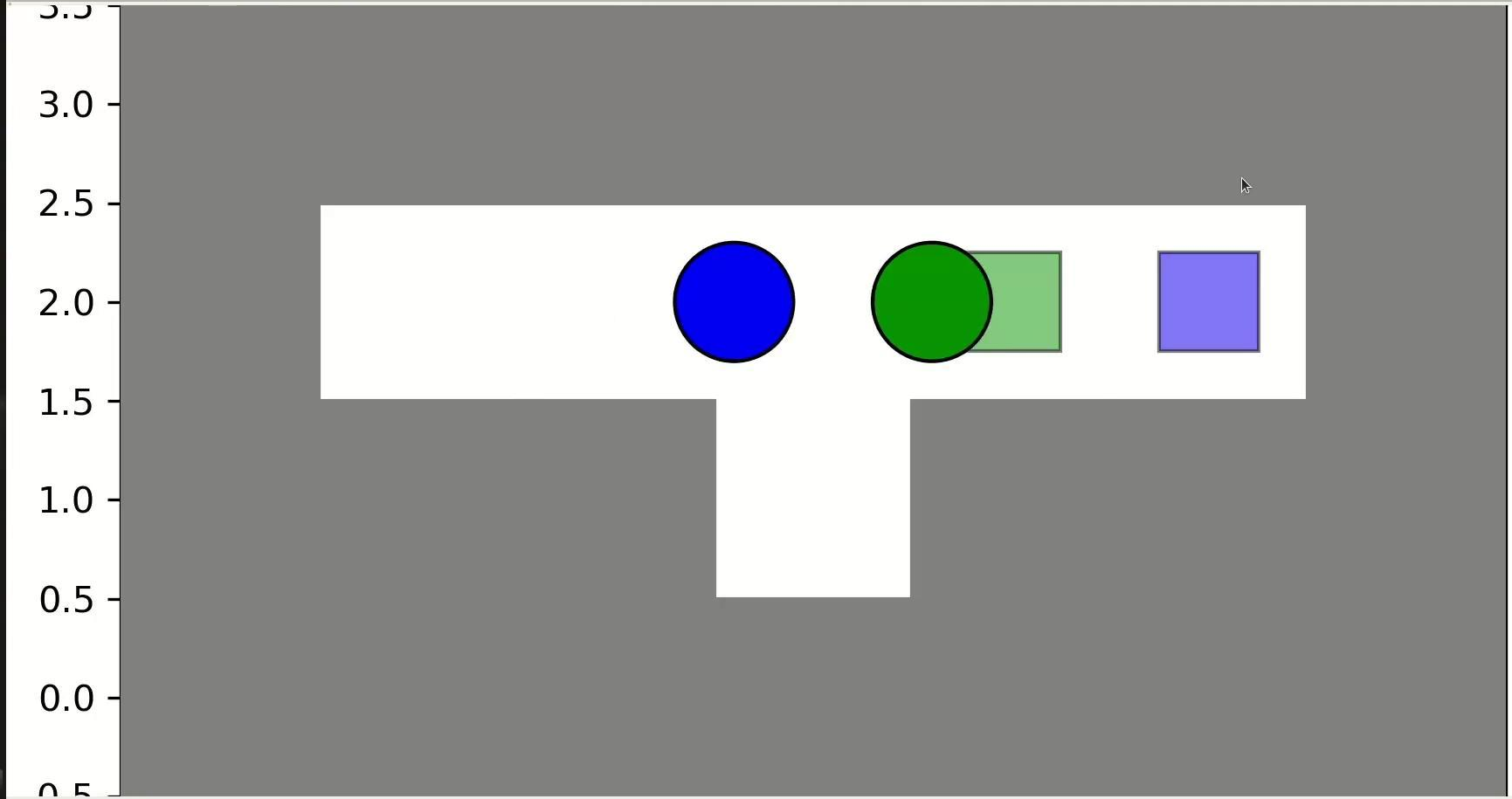
Adapted from: W. Hönig, J. Li, and S. Koenig, “Overview of Multi-Agent Path Finding (MAPF)”

Conflict-based Search

There are two levels of search, so, it is **slow**

But, it is **complete and optimal**

Figure 1



Automated Warehousing (Application Demo)

31 agents, several obstacles, narrow pathways

Warehouse Map
(Black pixels represent obstacles)

Automated Warehousing (Application Demo)

Conflict-based Search failed to return a solution in reasonable time.

- [I waited an hour and still was still running]

Prioritized Search found a solution in only 20 milliseconds!

- [The solution was not optimal, but atleast I got something]