Artificial Intelligence
Autumn Term
University of Rome "La Sapienza"
December 12th, 2023

Exploiting Geometric Constraints in Large-Scale Multi-Agent Pathfinding

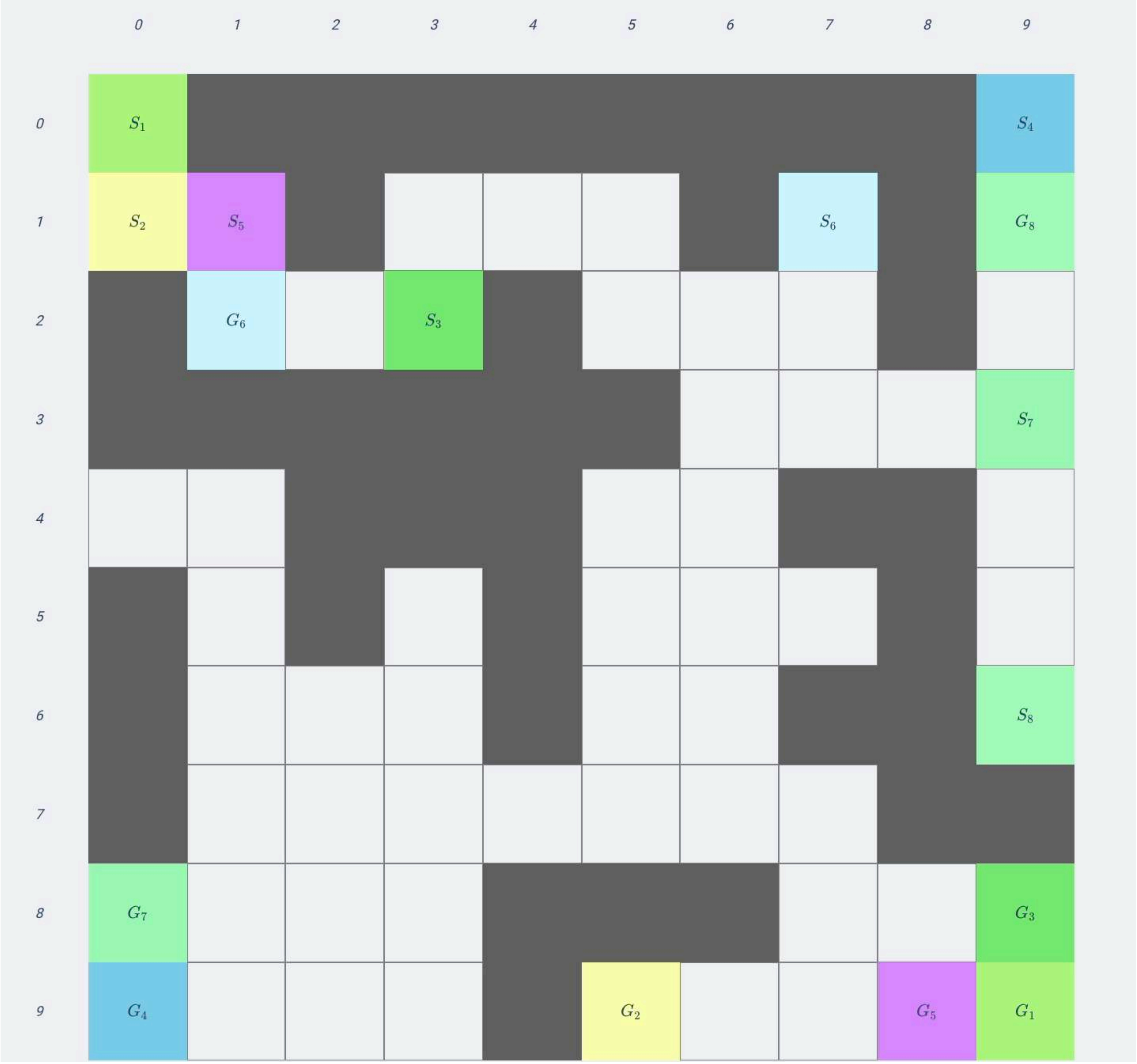
Prof Sara Bernardini
University of Rome La Sapienza
bernardini@diag.uniroma1.it
www.sara-bernardini.com

SAPIENZA
UNIVERSITÀ DI ROMA

Outline

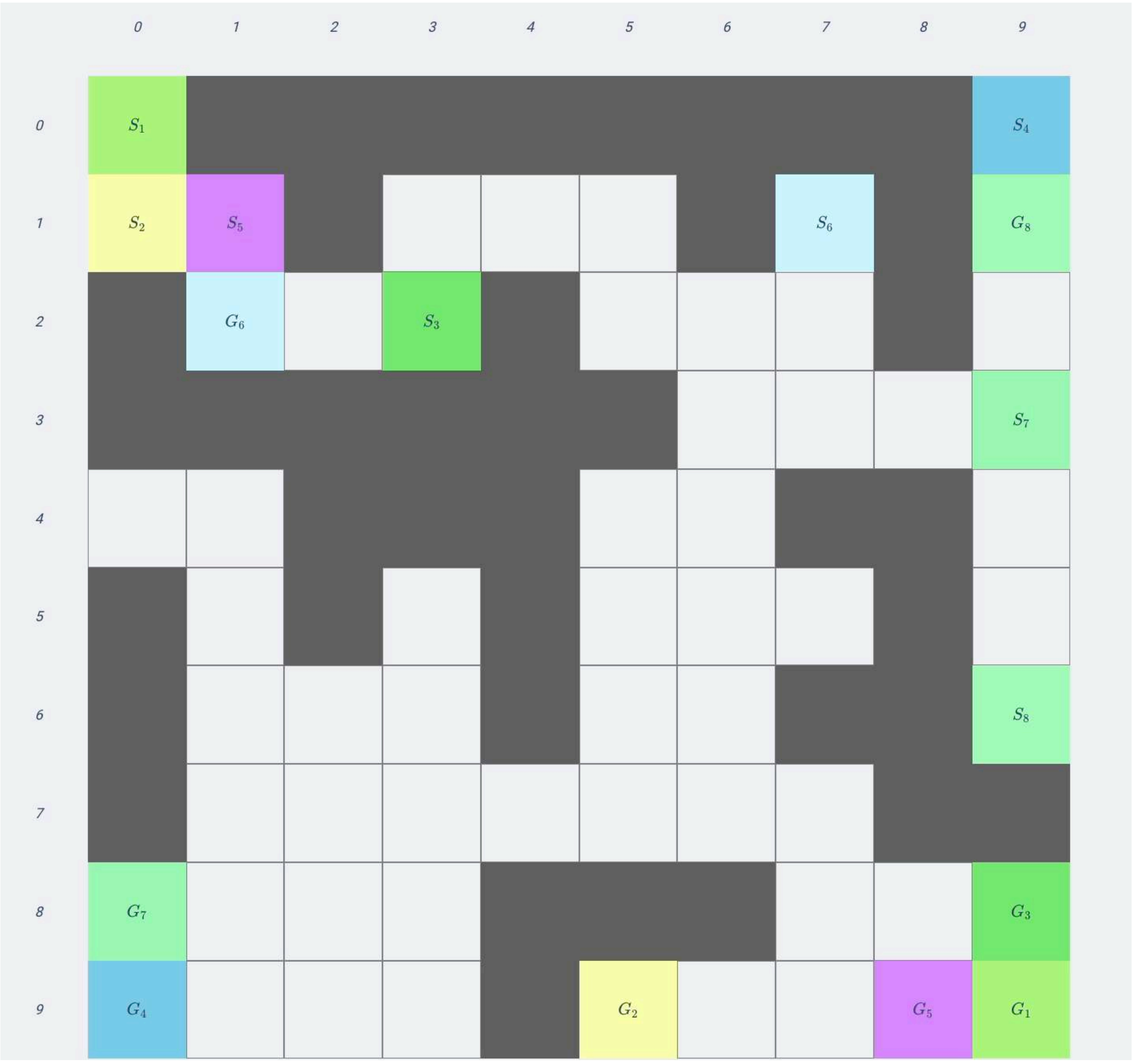
- **Multi-agent Path Finding**
  - What is MAPF?
  - Can we just use single-agent search?
  - Why is studying MAPF useful?
  - Complexity analysis
- **A robust algorithm for MAPF: Delayed Shortest Path (DSP)**
  - Safe delays
  - Geometric constraints
  - Main theorem
  - DSP Algorithm
  - Experimental Results
- **Conclusions and Future Work**

What is Multi-Agent Path Finding?

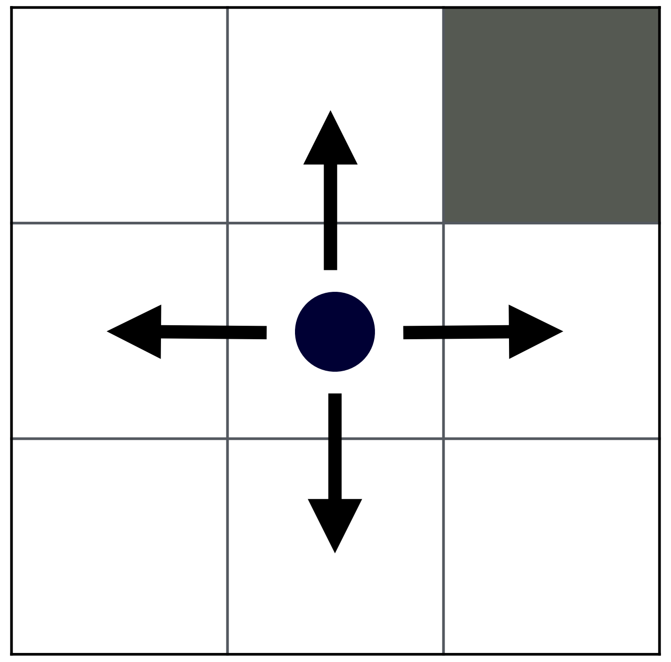


- Multi-agent path finding (MAPF): Find collision-free paths from start to goal positions for all agents
- Alternative names: cooperative path finding (CPF), multi-robot path planning, pebble motion on graphs

Formalisation of MAPF



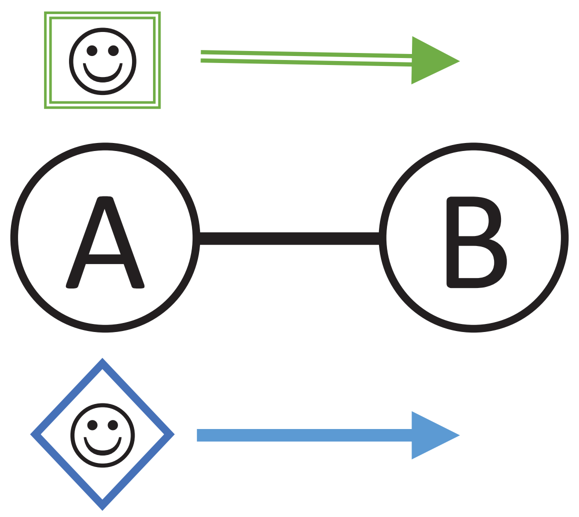
- Simplifying **Assumptions**:
  - Space is discretised: graph $\mathscr{G}$ with n vertices
  - A set $\mathscr{A}$ of k agents, each with start vertex and goal vertex
  - Point robot (has no volume and no shape)
  - No kinematics constraints
  - Time is discretised: at each time step, each agent is at one of the vertices and can perform a single action
  - Actions: wait; go to adjacent vertex (go up, down, right, left)



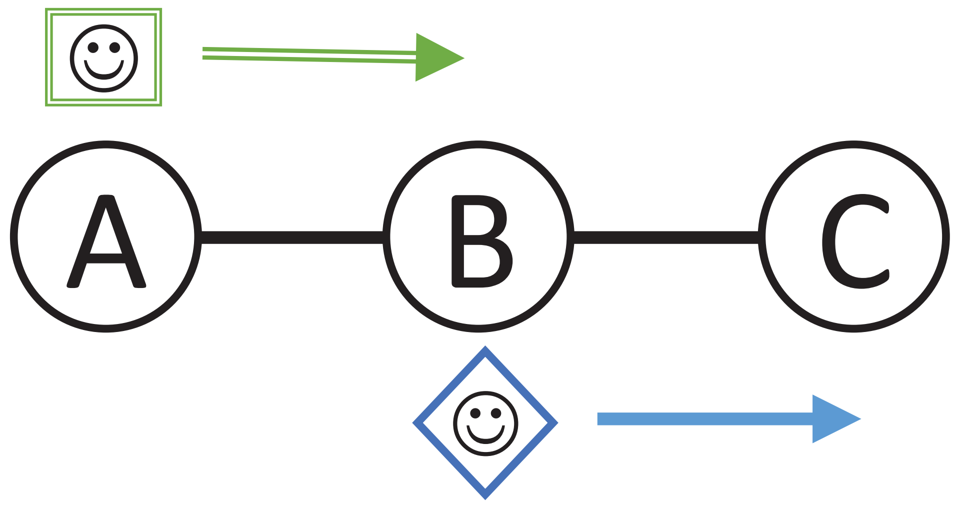
- **Goal**: find collision-free paths from start to goal for all agents

[8] Stern, R. et al. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. SoCS 2019.

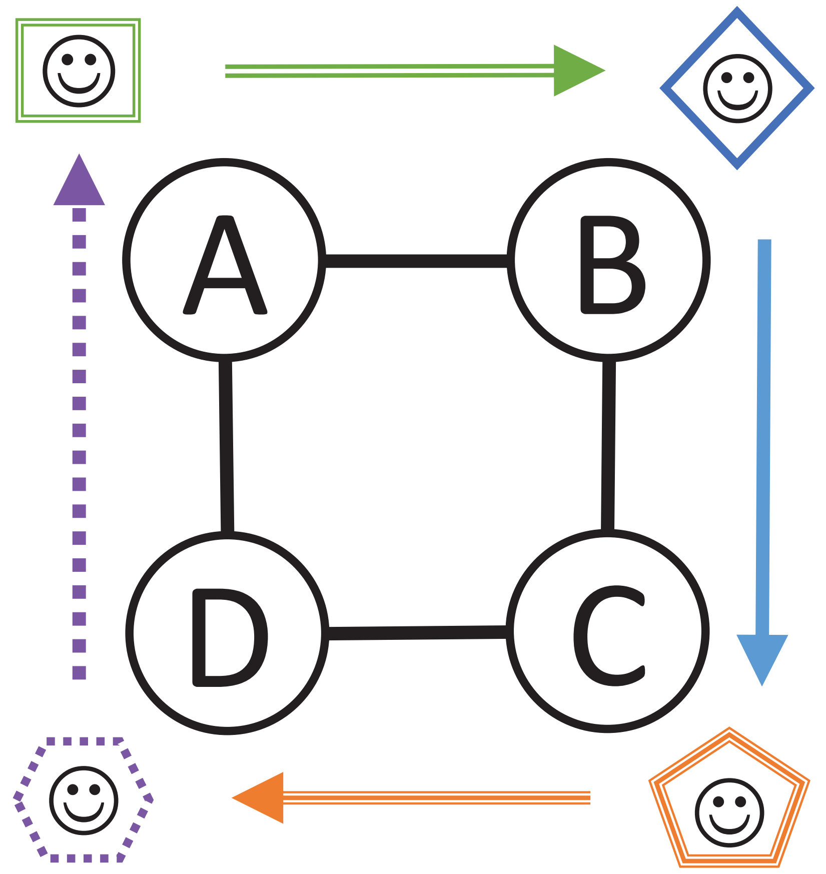
Conflicts



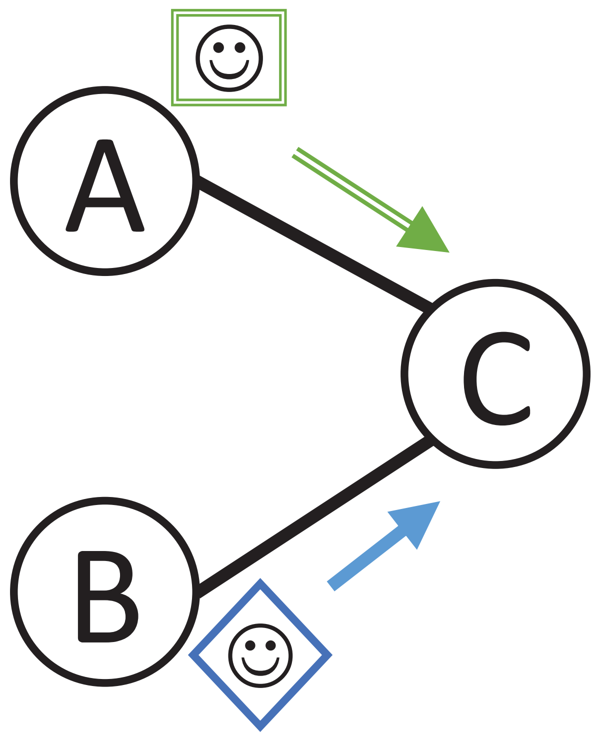
Edge conflict



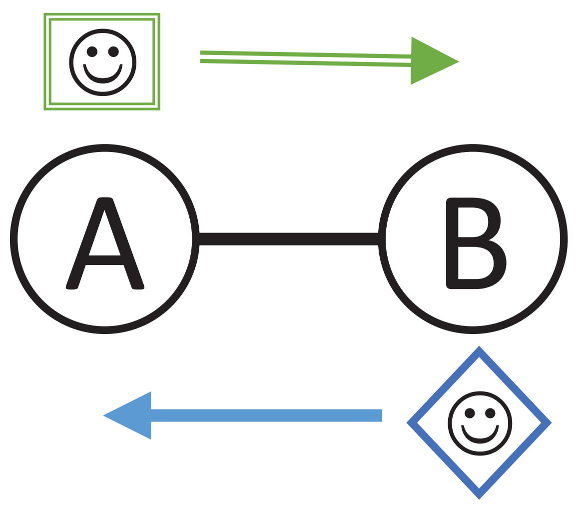
Following conflict



Cycle conflict



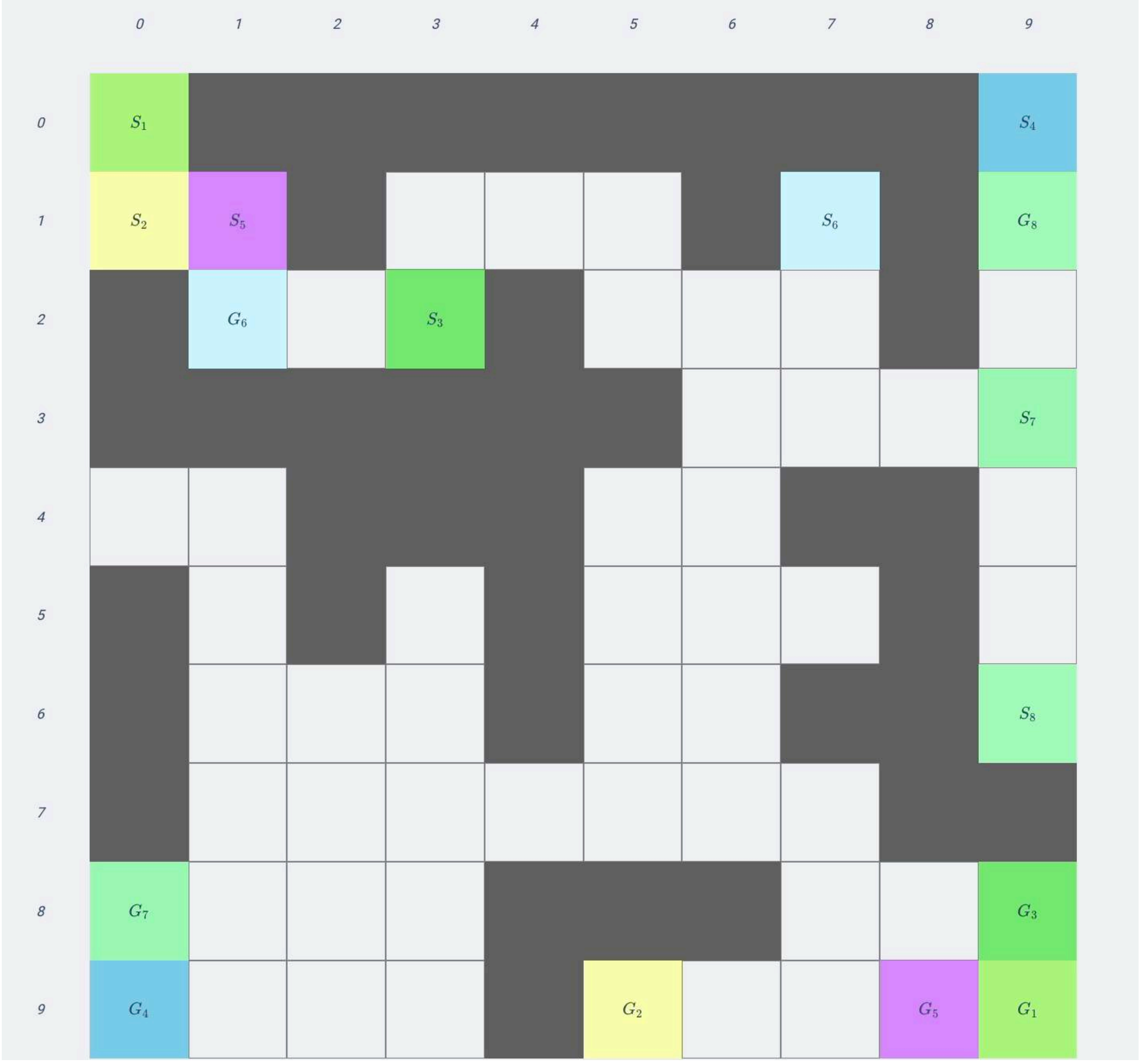
Vertex conflict



Swapping conflict

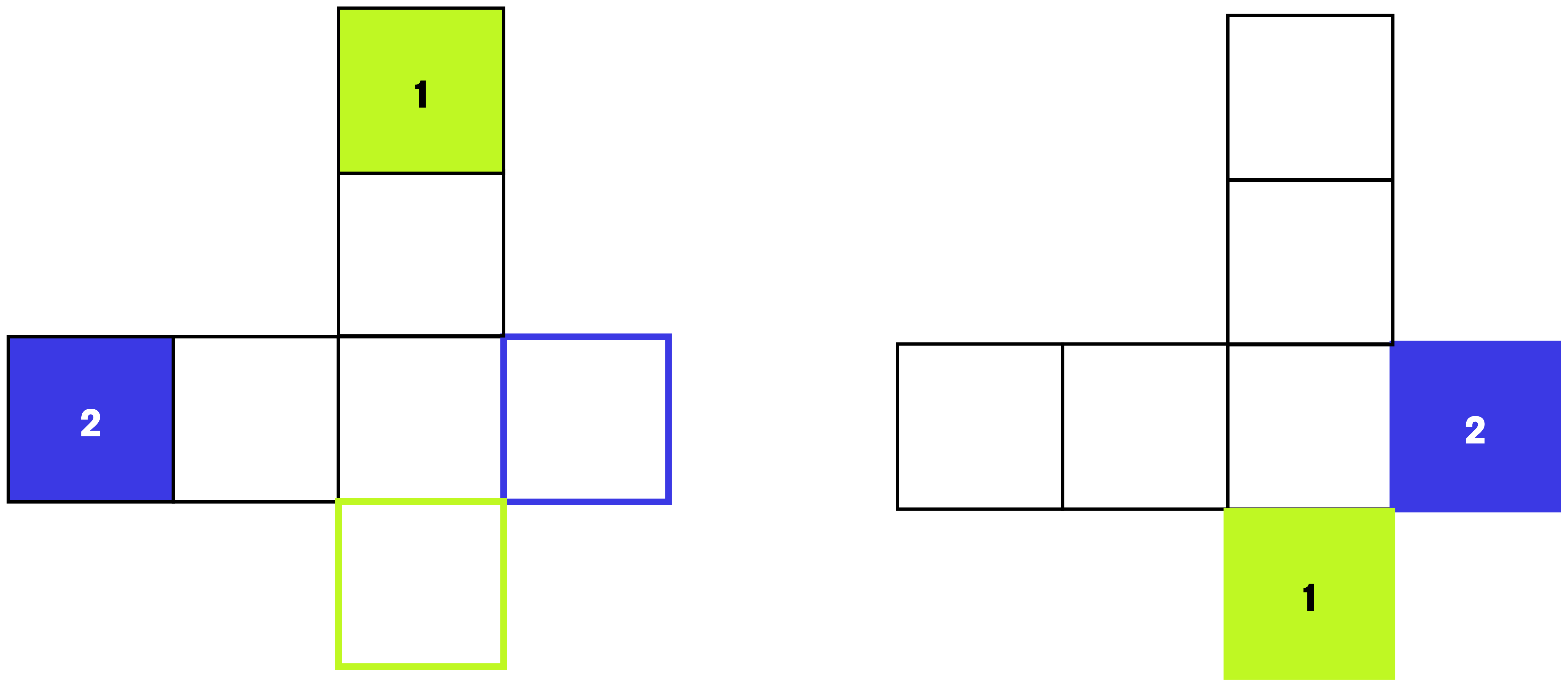
Behaviour at Target

- Agents may reach their targets at different time steps
- How does agent behave in time steps after it has reached its target and before last agent has reached its target?
  - **Stay at target**
  - **Disappear at target**



Objective Function

- How do we capture that some MAPF solutions are better than others?
  - **Makespan**: number of time steps required for all agents to reach their target
  - **Flowtime** (**sum-of-costs**): sum of time steps required by each agent to reach its target



Makespan = 4
Flowtime = 7

Classical MAPF and Beyond

- Most common setting in classical MAPF
  - Edge and vertex conflicts are forbidden
  - Stay at target
  - Flowtime

- More realistic extensions:
  - Actions have different durations
  - Online MAPF (OMAPF)
  - Life-long MAPF (LMAPF)
  - ...

Can’t we just use single-agent search?

Using Single-Agent Search

Using Single-Agent Search

Using Single-Agent Search

?

Plan for the Agents Jointly

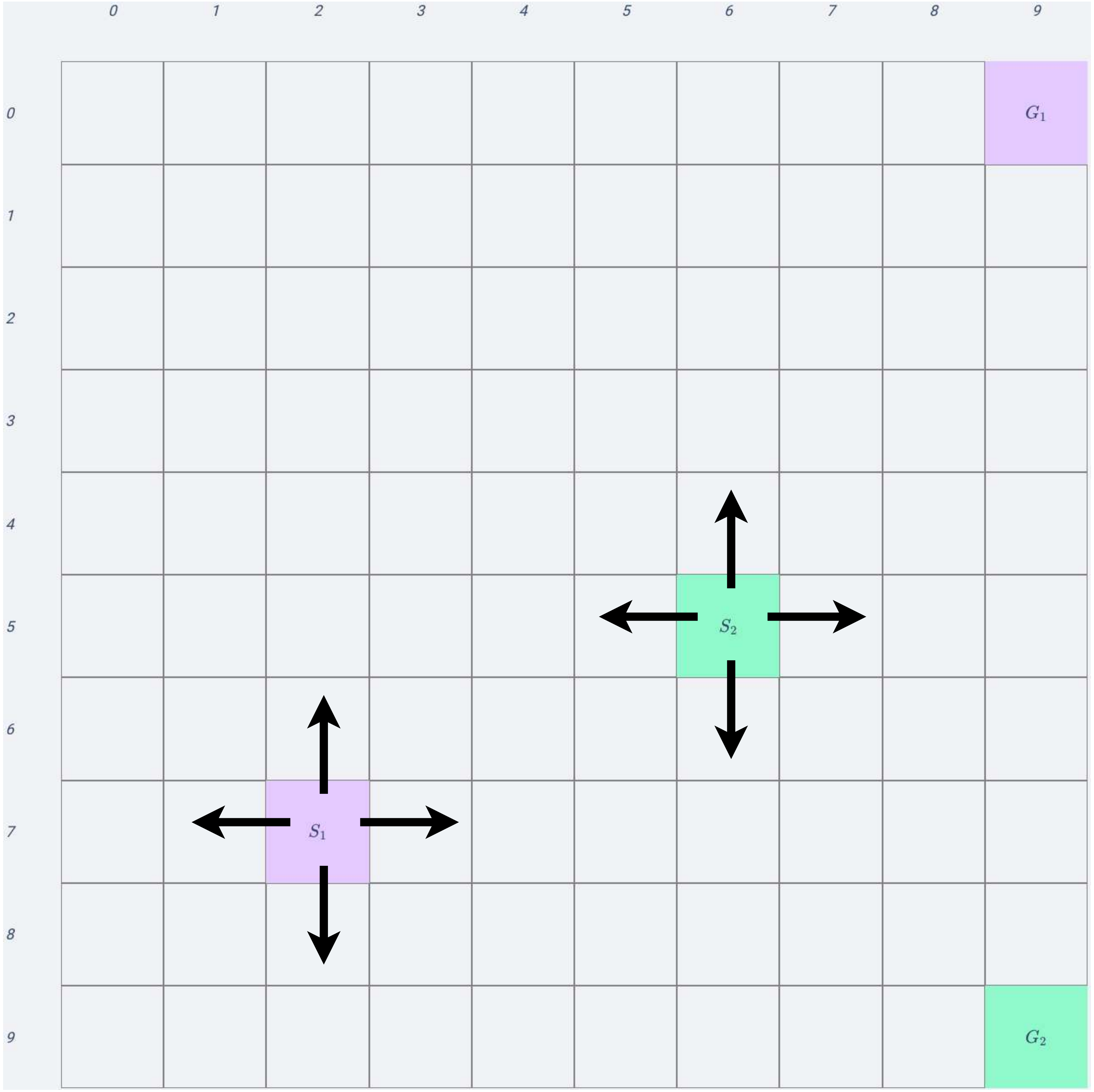
Plan for the Agents Jointly

Plan for the Agents Jointly

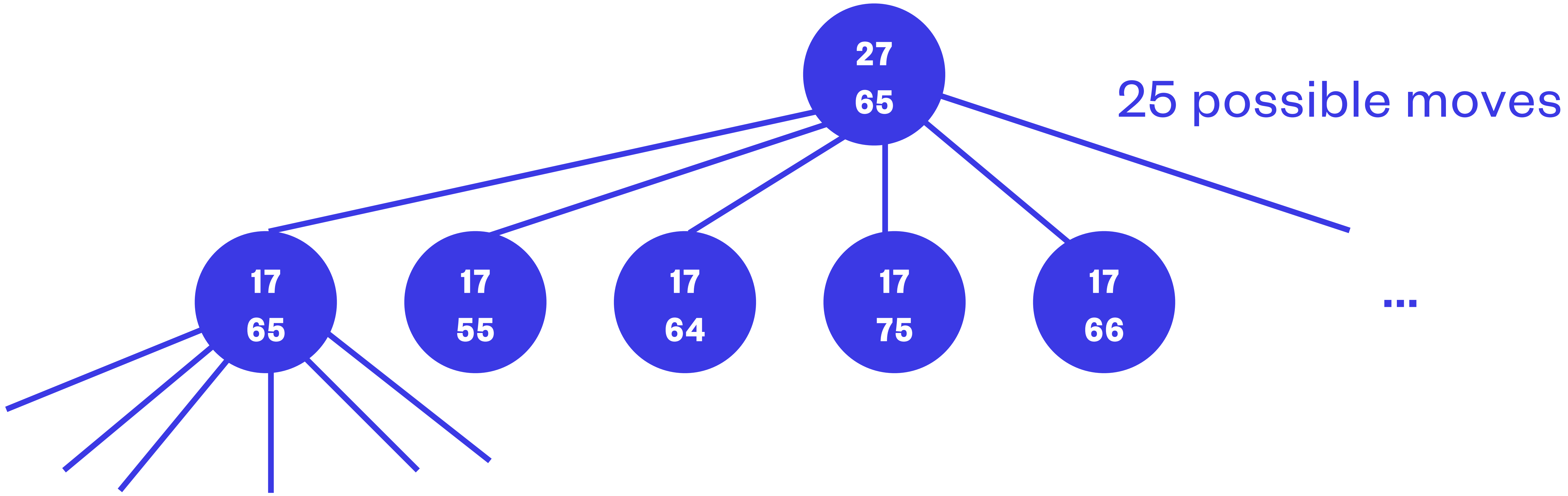
Plan for the Agents Jointly

Plan for the Agents Jointly

Combinatorial Explosion



Joint State Space



BRANCHING FACTOR = 5^2
NUMBER OF STATES = 100^2

What about if we have k agents?

[7] Standley, T. Finding Optimal Solutions to Cooperative Pathfinding Problems. AAAI 2020.

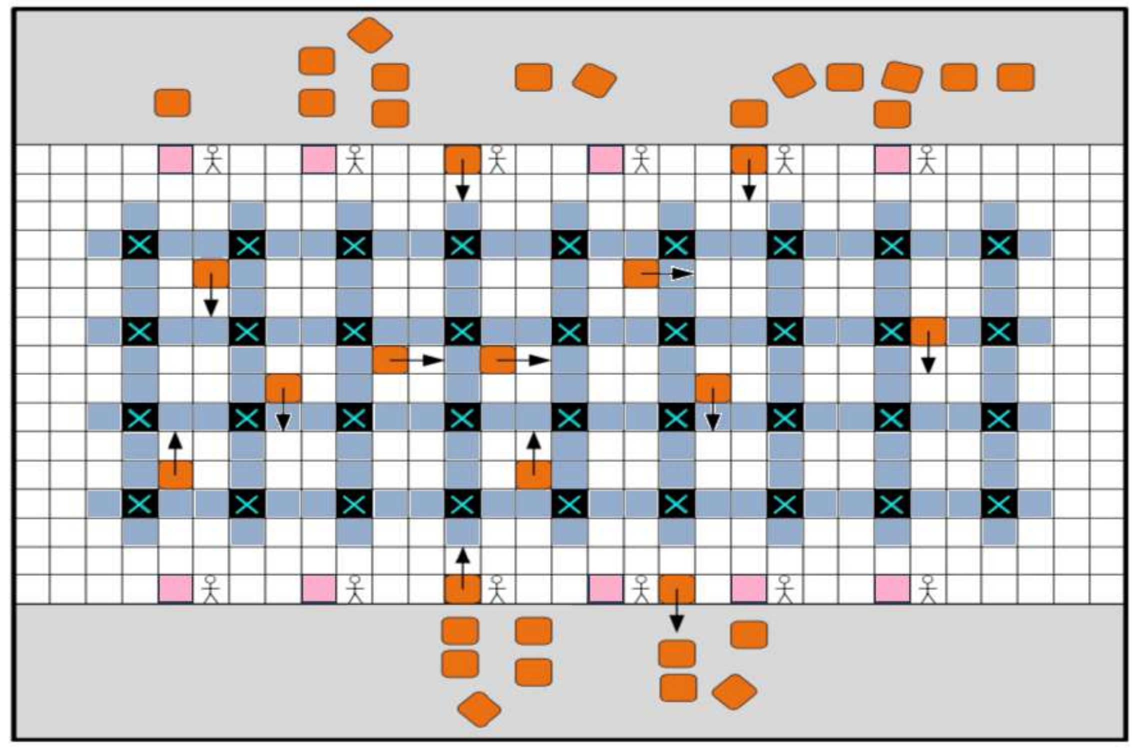
Why is MAPF Important?

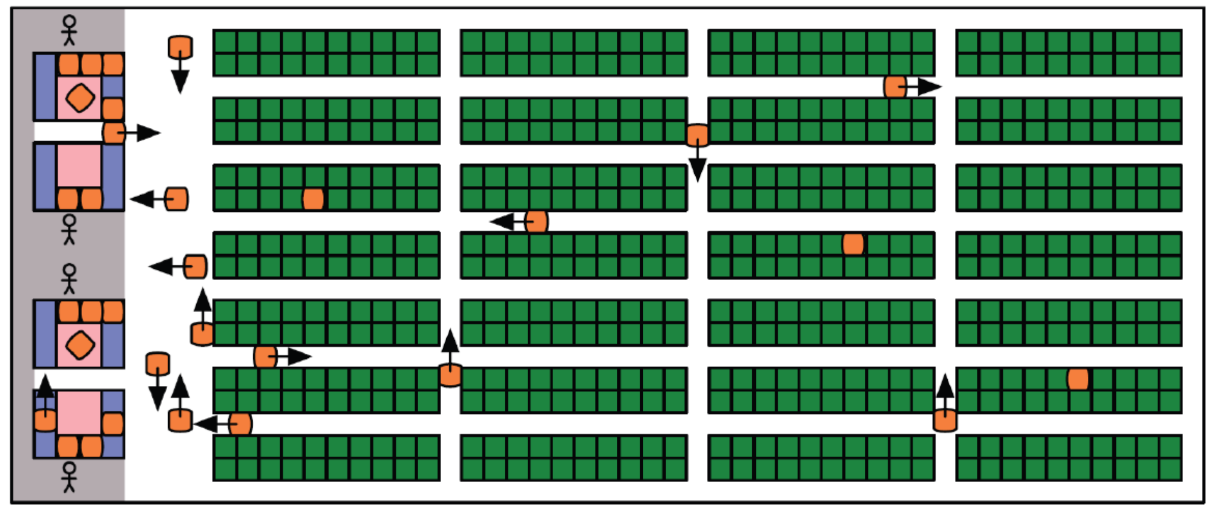
Automated Warehouse Systems

Amazon fulfilment centres

Warehousing section

https://www.machinedesign.com/mechanical-motion-systems/article/21835788/changing-the-future-of-warehouses-with-amazon-robots



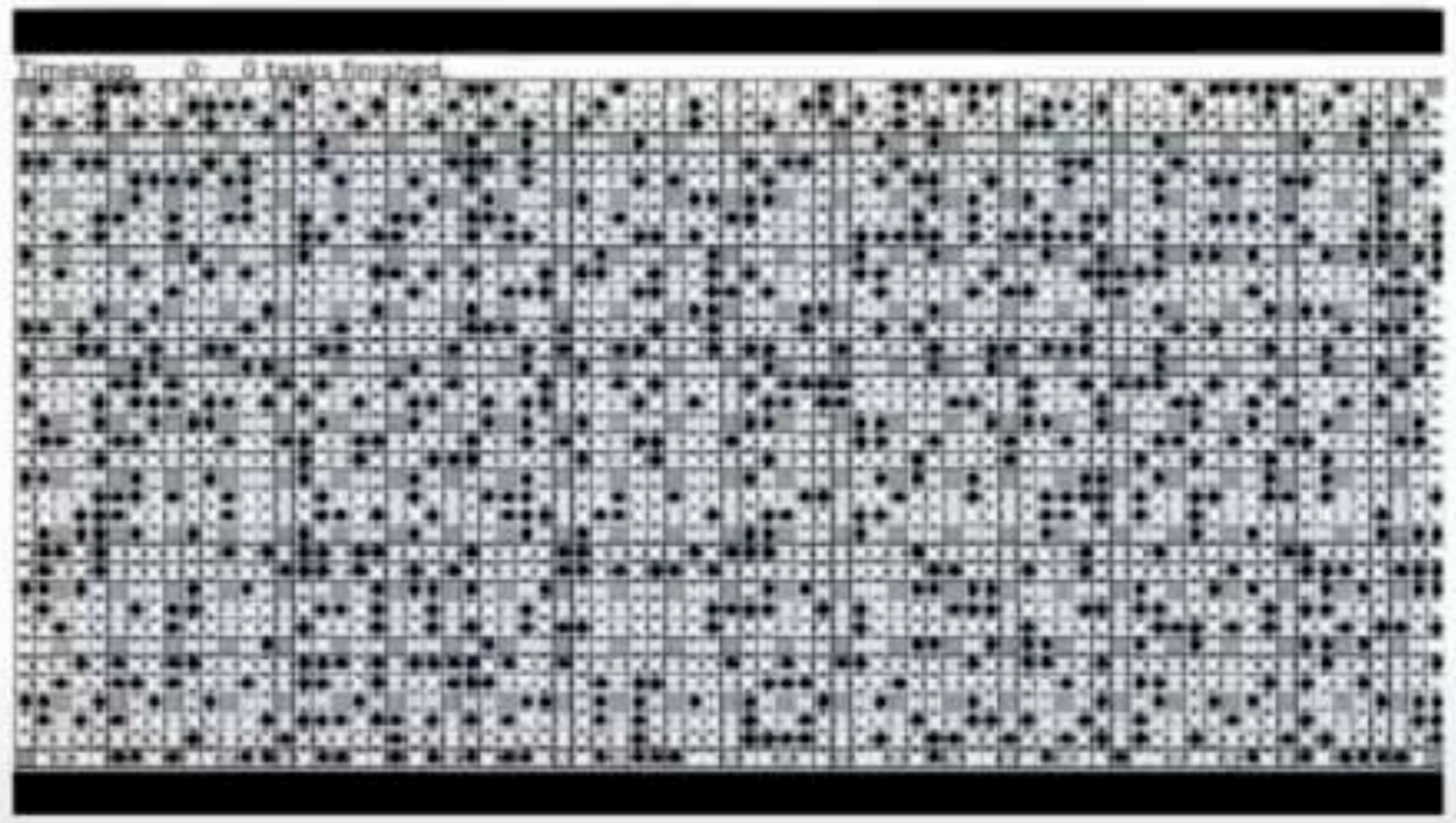


[1] J. Li et al., "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses", AAAI, 2021.

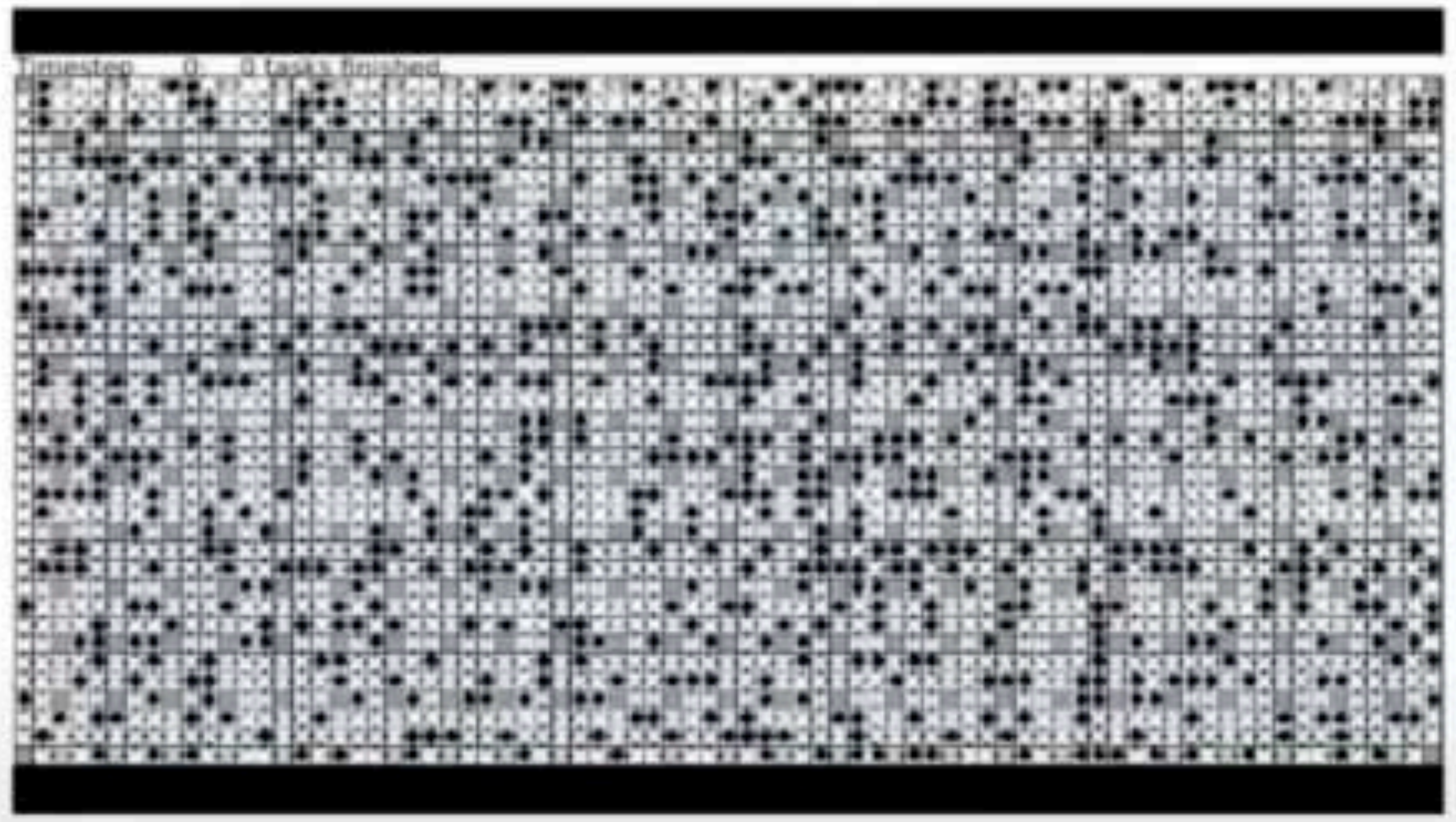
[2] J. Li et al. AAMAS-22 Tutorial on Recent Advances in Multi-Agent Path Finding. AAMAS 2022.

Scalability Problem

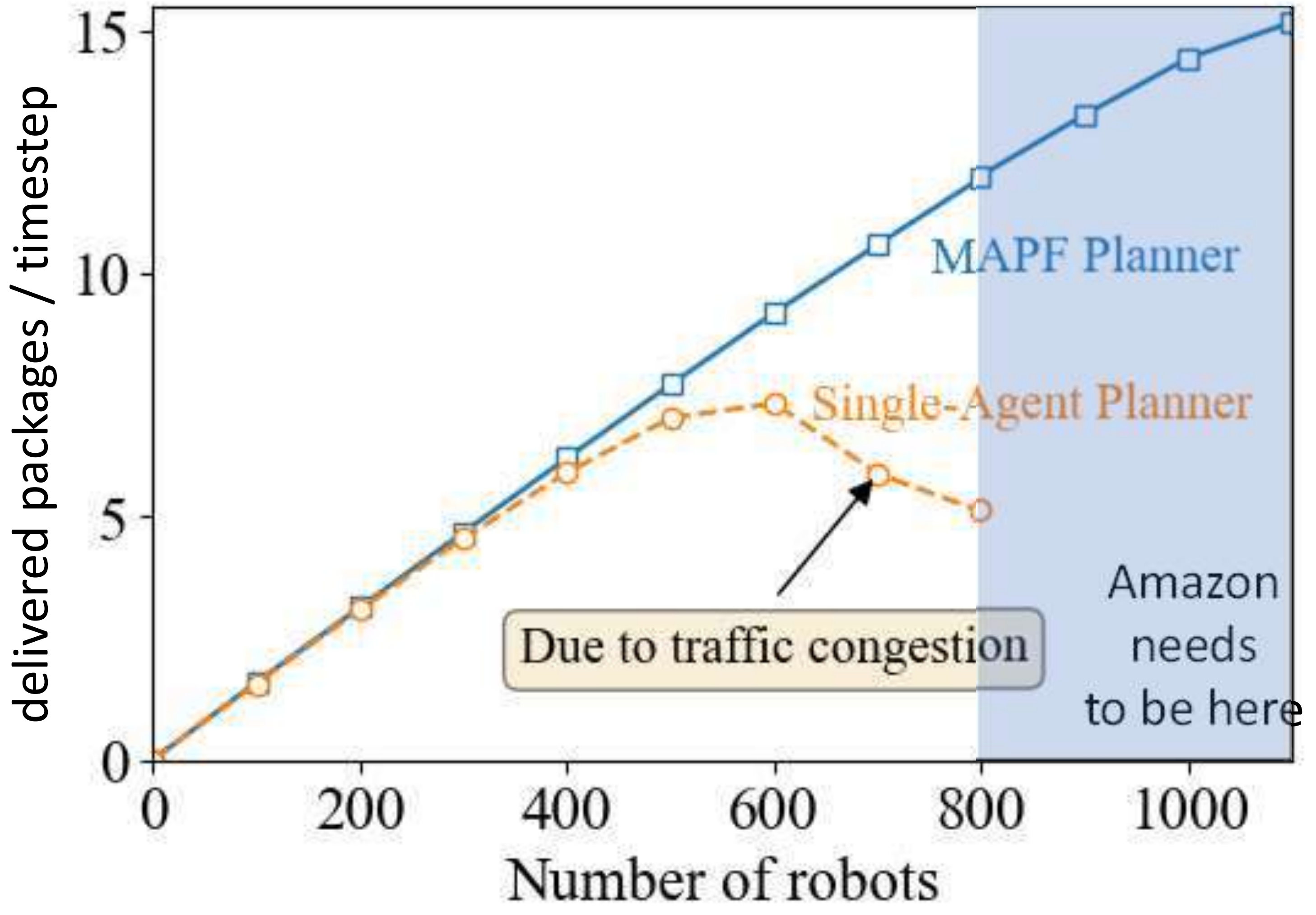
Single-agent planner with traffic rules



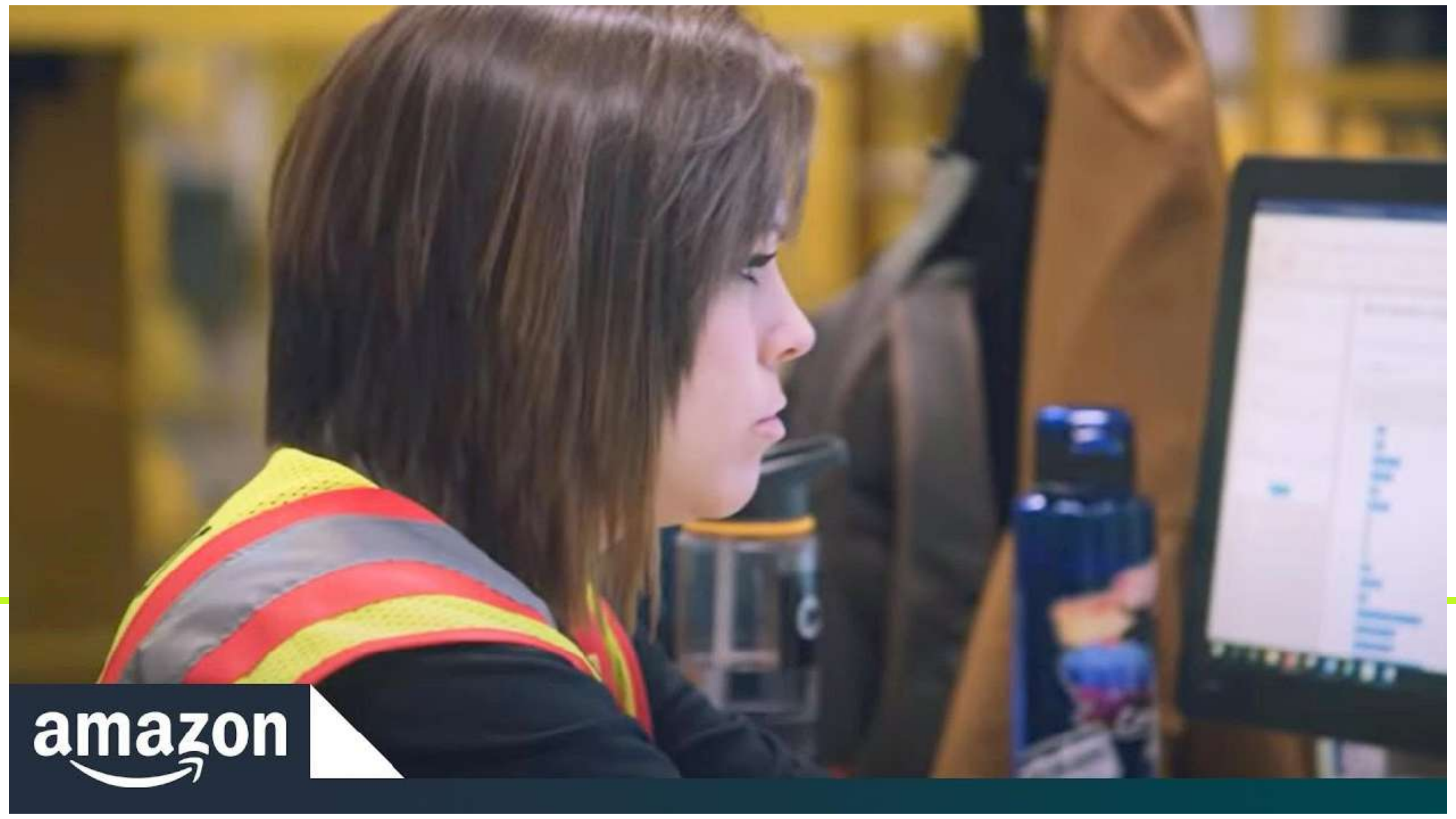
MAPF planner



800 robots (= 32% empty cells) on a 37x77 sorting center map with 50 workstations and 275 chutes

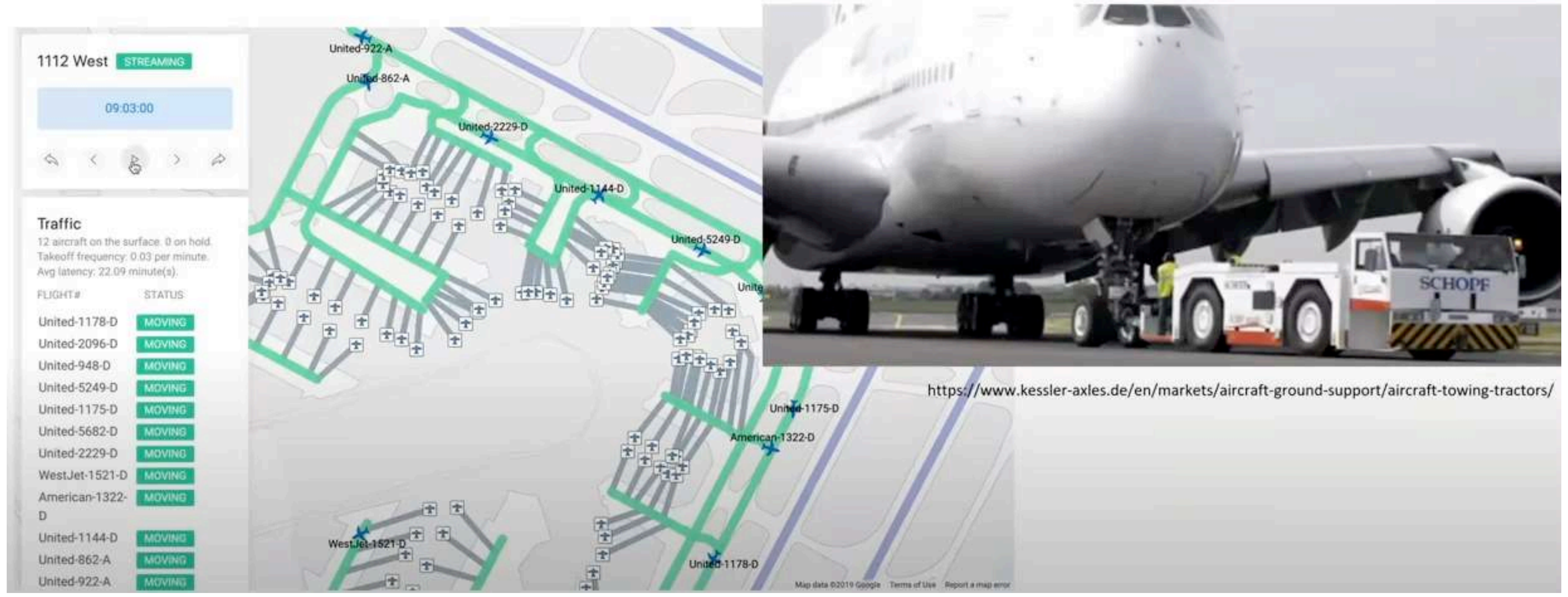


Sortation section



[1] J. Li et al., "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses", AAAI, 2021.

Not only Amazon...



Airport Taxiway Path Planning

[4] J. Li, et al. Scheduling and Airport Taxiway Path Planning under Uncertainty. AIAA 2019.

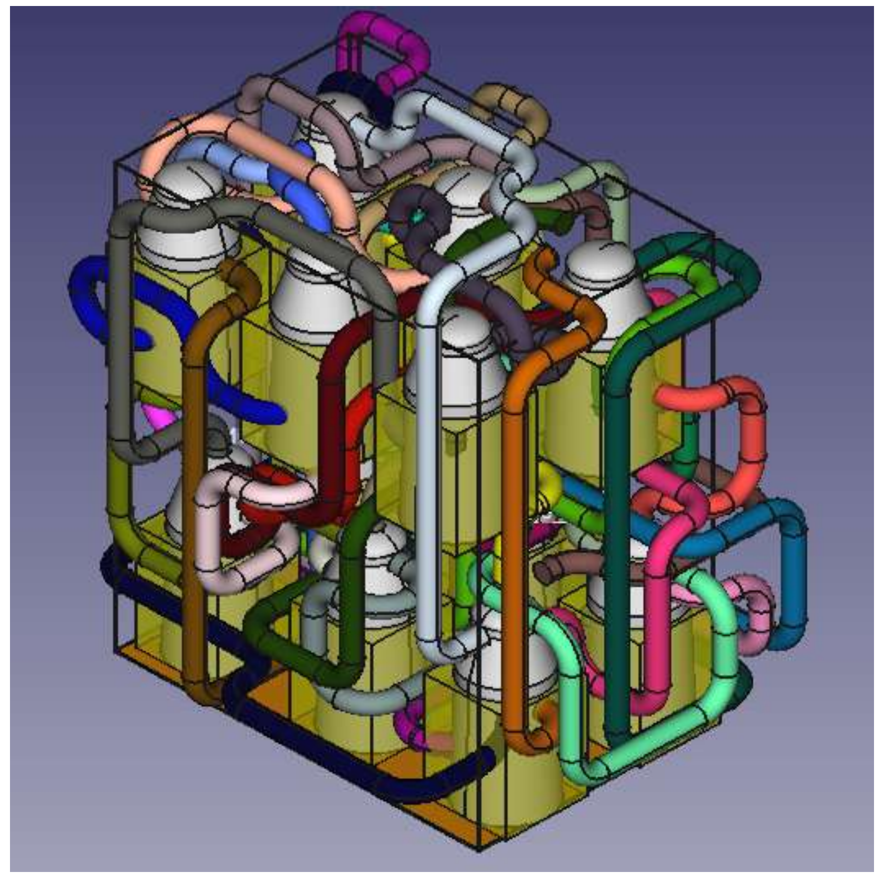
Fully Automated Warehouses

Train Scheduling and Rescheduling

[3] J. Li, et al. Scalable Rail Planning and Replanning: Winning the 2020 Flatland Challenge. ICAPS 2021.

Video Games

[5] J. Li, at al. Moving Agents in Formation in Congested Environments. In AAMAS 2020.



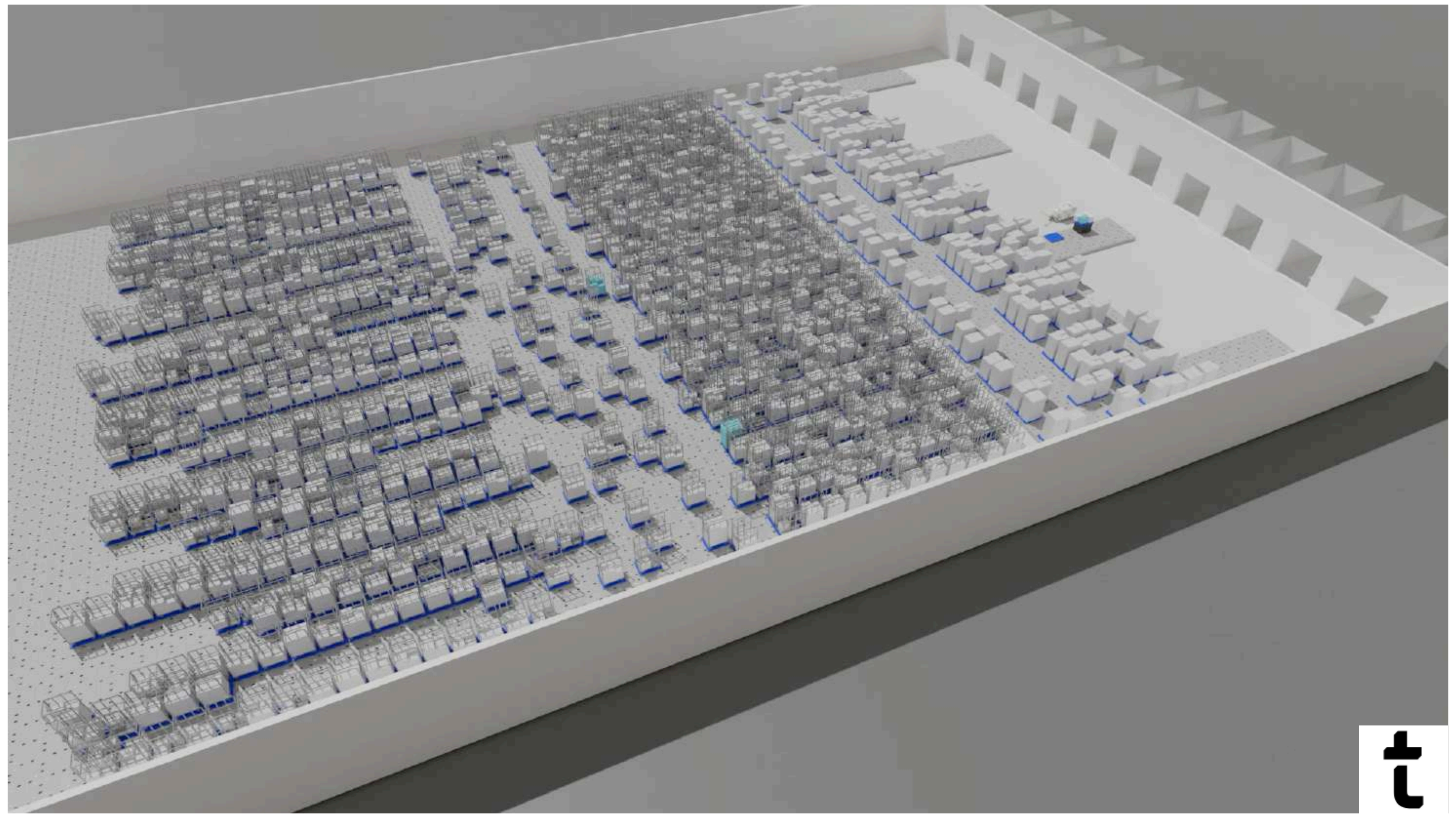
Pipe Routing

[6] G. Belov, et al. From Multi-Agent Pathfinding to 3D Pipe Routing. SoCS 2020.

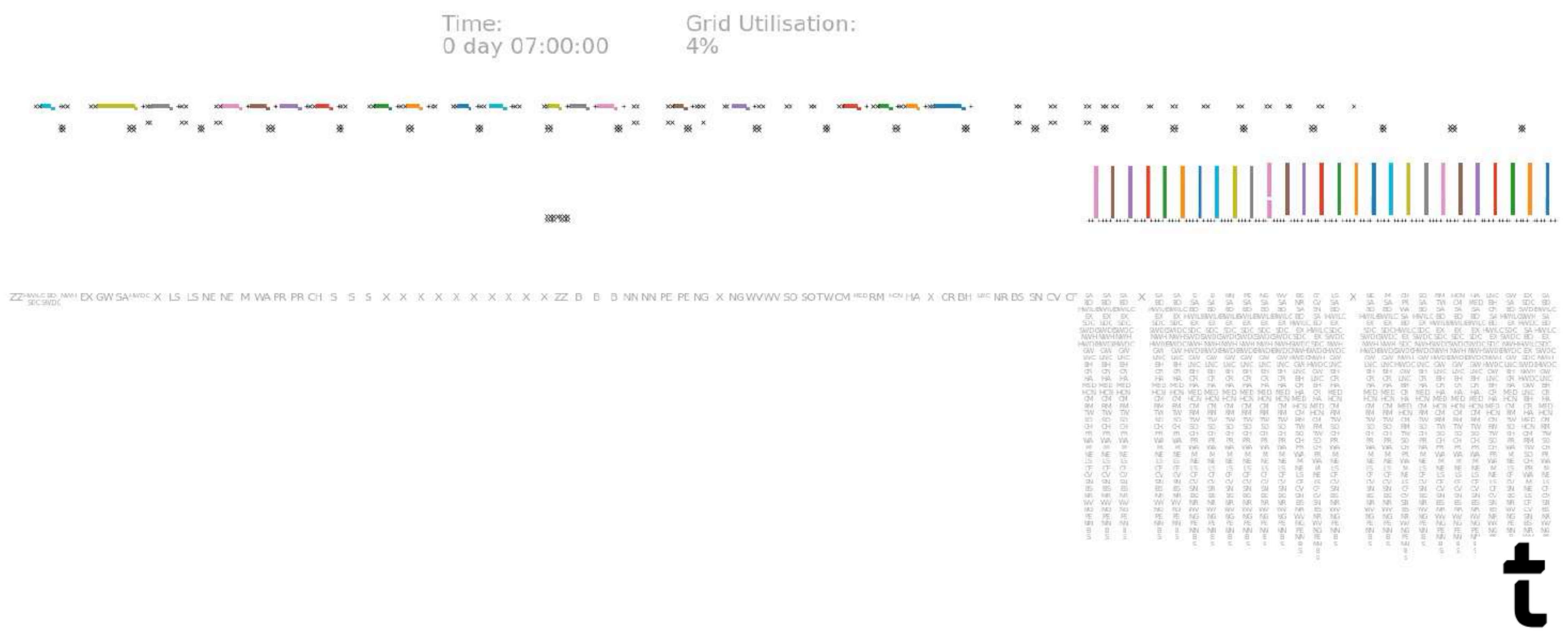
VersaTile Project

Project funded by Innovate UK and Tharsus

Pick-face Operations



Cross-dock Operations



Can we solve MAPF efficiently?

MAPF Complexity

- **Sub-optimal MAPF** can be solved in polynomial time; plans have polynomial length [Kornhauser, 1984]
  - Pebble motion on graphs
- It took a while until this result was recognised by the community! [Roger and Helmert, 2012]
- Anonymous MAPF (it does not matter which agent reaches which goal) is also tractable
- **Optimal MAPF** is NP-hard (both makespan and flowtime optimal MAPF) [Yu and LaValle, 2013]
  - Even on planar graphs [Yu, 2016] and grid-like graphs [Banfi et al., 2017]
- There are limits to approximating optimal solution for makespan optimisations [Ma et al., 2016]
  - NP-hard to find makespan-bounded suboptimal solutions with sub optimality factor of less than 4/3

Theoretical Investigation on MAPF

- B. Nebel. On the computational complexity of multi-agent pathfinding on directed graphs. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS-20)*, 2020.
- Intractability of Optimal Multi-Agent Pathfinding on Directed Graphs. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI-23)*, 2023.
- B. Nebel. The Small Solution Hypothesis for MAPF on Strongly Connected Directed Graphs Is True. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS)*, 2023.

Two Classes of MAPF Algorithms

1. Solving MAPF **optimally**
    - Independence Detection (ID) [Standley, AAAI-10]
    - Increasing Cost Tree Search (ICTS) [Sharon et al., AIJ-13]
    - M* [Wagner and Choset, AIJ-15]
    - **Conflict-Based Search** (CBS) [Sharon et al., AIJ-15]
    - Improved CBS (ICBS) [Boyarski et al., IJCAI-15]
    - CBS+Heuristics (CBSH) [Felner et al., ICAPS-18]
    - CBSH-RM [Jiaoyang Li et al., AAAI-19]
    - Lazy CBS [Gange et al., ICAPS-19]
    - Branch-and-Cut-and-Price (BCP) [Lam et al., IJCAI-19]

Two Classes of MAPF Algorithms

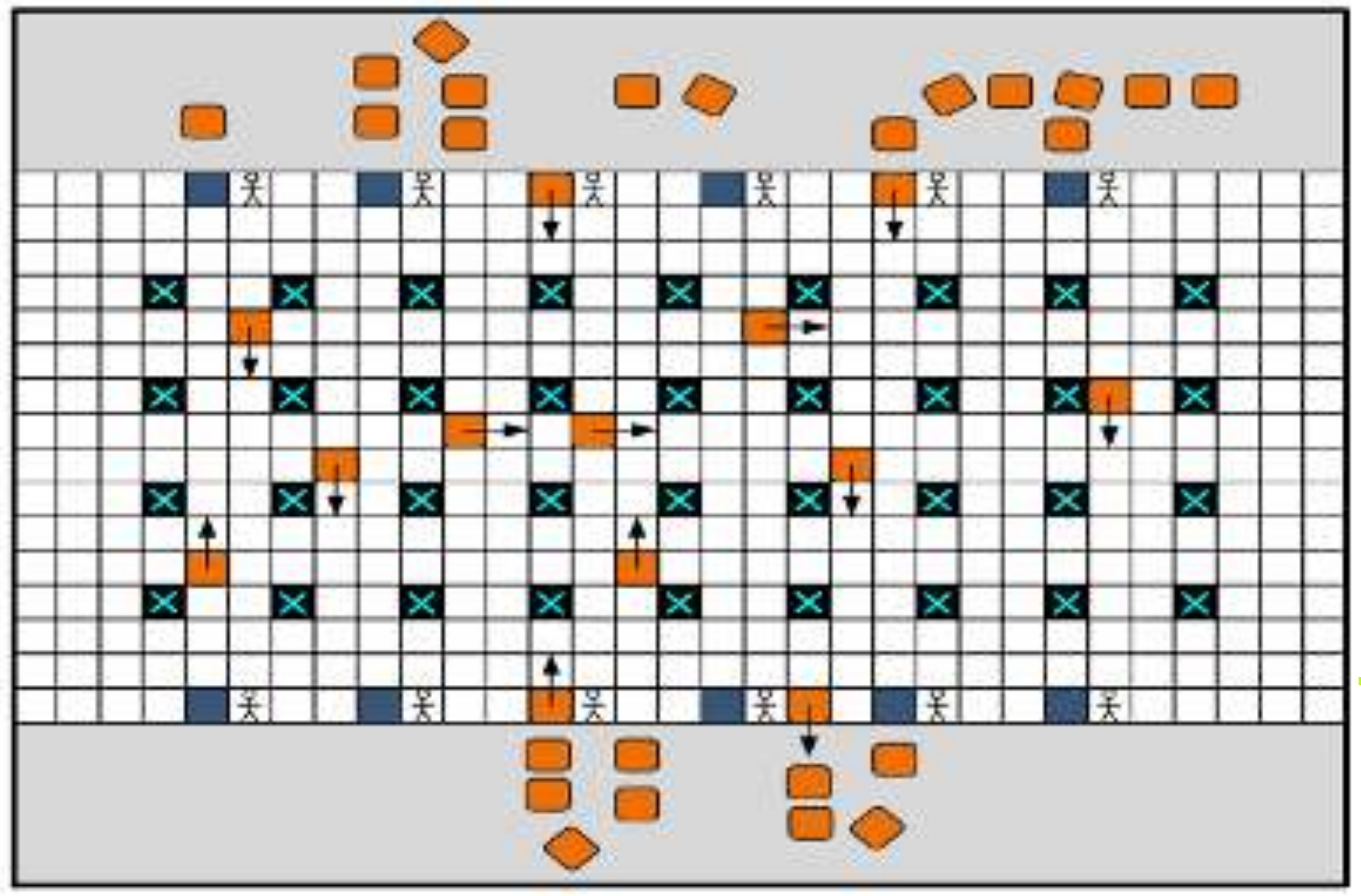
2. Solving MAPF **sub-optimally**
   - **Incomplete** methods
     - Push and Swap [Luna and Bekris, IJCAI-11]
     - **Cooperative Pathfinding** (CA*/HCA*/WHCA*) [Silver, AIIDE-05]
     - Priority-Based Search (PBS/CBSw/P) [Ma et al., AAAI-19]
     - PIBT [Okumura et al., IJCAI-19]
   - **Complete** methods
     - BIBOX [Surynek, ICRA-09]
     - **Push and Rotate** [Wilde et al., JAIR-14]
     - Enhanced CBS (ECBS) [Barer et al., SoCS-14]
     - Explicit Estimation CBS (EECBS) [Li et al., AAAI-21]

Can we design **robust** yet **fast** algorithms for large-scale MAPF?

Exploiting Geometric Constraints in Multi-Agent Pathfinding
D. Atzmon, S. Bernardini, F. Fagnani, and D. Fairbairn
Proc. of the 33rd International Conference on Automated Planning and Scheduling (ICAPS-23)

Our Setting

- All agents are present outside the environment in the beginning (time 0)
- Each agent can start moving immediately (at time 0) or can be delayed outside the environment
- Agents disappear when they arrive at their goal location
- *Assumption: shortest paths between start/goal locations of all agents are known (nothing else)*



A Simple Algorithm

- Non-conflicting paths can be generated by following three simple steps:
  1. Consider the agents sequentially according to a given **priority order**
  2. Take each agent's **shortest path** from start to goal
  3. Add **delays** at the beginning of the paths (following the priority order) to avoid conflicts

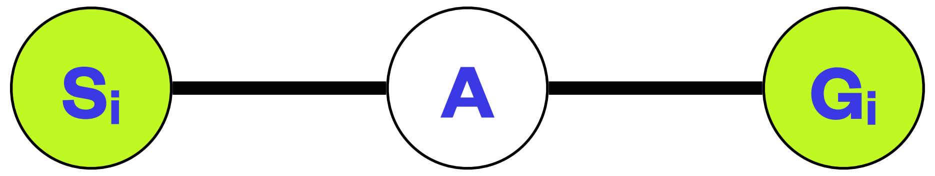
Notation

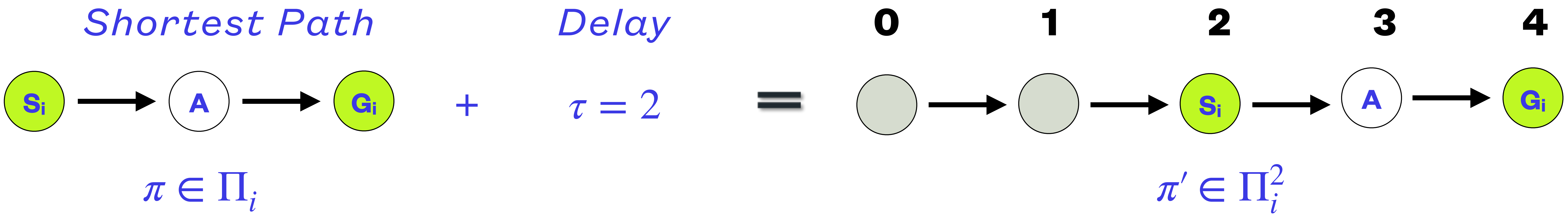
- **MAPF Instance**: $I = (\mathcal{G}, k, s, g)$
  - Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, set $\mathcal{A}$ of k agents, vectors s and g of start/end vertices
- Given $x, y \in \mathcal{V}$, $d(x, y)$ is length of **shortest path** between x and y
- A **solution** to I is a set of paths $\Pi = \{\pi_1, \ldots, \pi_k\}$ such that, for every $i, j \in \mathcal{A}$ $(i \neq j)$, π_i and π_j are non-conflicting
- Denoting **flowtime** as $C(\Pi)$ and $\mathcal{M}_I$ set of solutions of instance I, MAPF aims to solve:

$$\arg\min_{\Pi \in \mathcal{M}_I} C(\Pi)$$

Delays

- If agent has a **delay** of $\tau \in \mathbb{N}_0$, it waits $(\tau - 1)$ time steps outside the environment and then starts its motion from start to goal at time τ

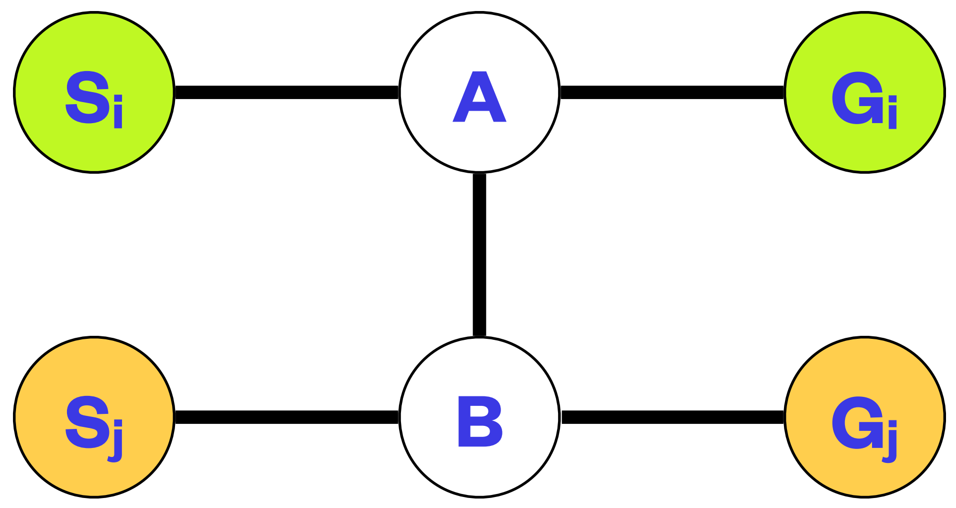




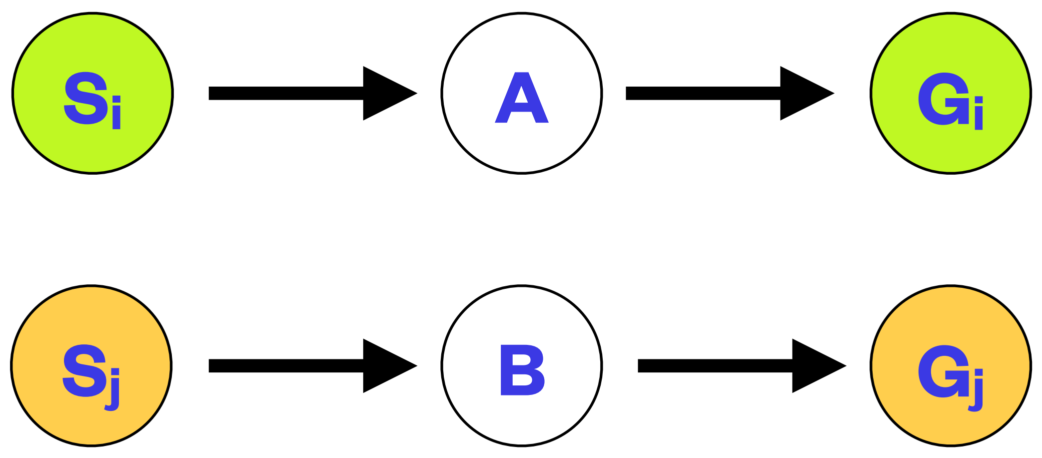
- We are interested in solutions of the form $\{\Pi_i^{\tau_i} \mid i \in \mathscr{A}\} \in \mathscr{M}_I$ for delay assignments $\{\tau_i \mid i \in \mathscr{A}\}$

Safe Delays

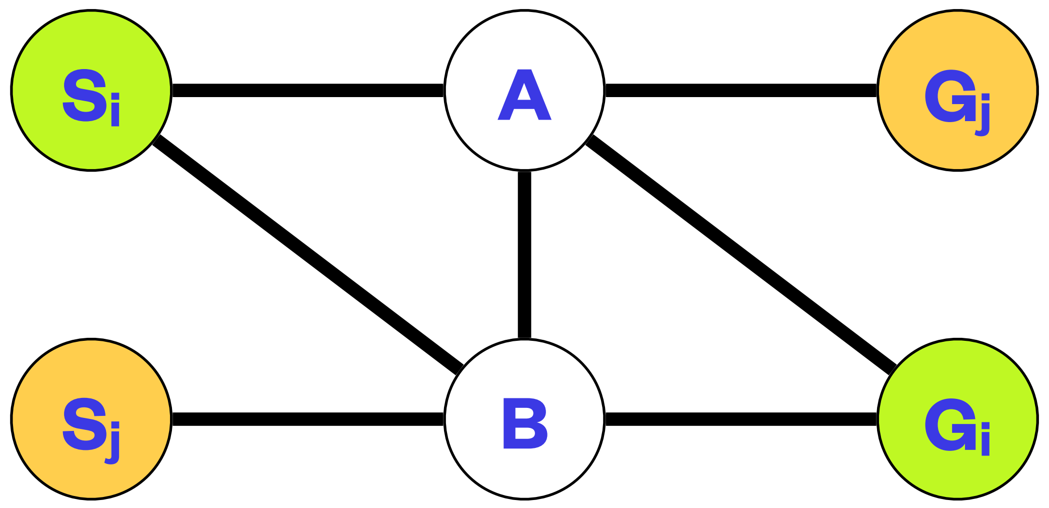
- A set of delays is **safe** if, when they are added to their corresponding shortest paths, the resulting paths are non-conflicting
- Given instance I, delay assignment $\{\tau_i \mid i \in \mathscr{A}\}$ is **safe** if, for any choice of $\pi_i \in \Pi_i^0$, the set of paths $\{\Pi_i^{\tau_i} \mid i \in \mathscr{A}\}$ is a solution to I



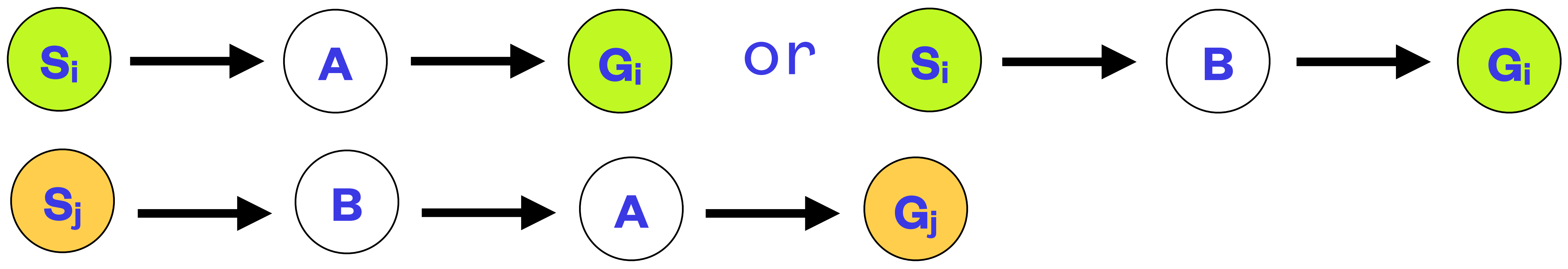
Shortest Paths



Any set of delays is **safe**



Shortest Paths



$\{\tau_i = 0,\ \tau_j = 0\}$ is **not safe!** $\{\tau_i = 0,\ \tau_j = 1\}$ is **safe**

How to ensure delays are **safe**?

Geometrical Constraints

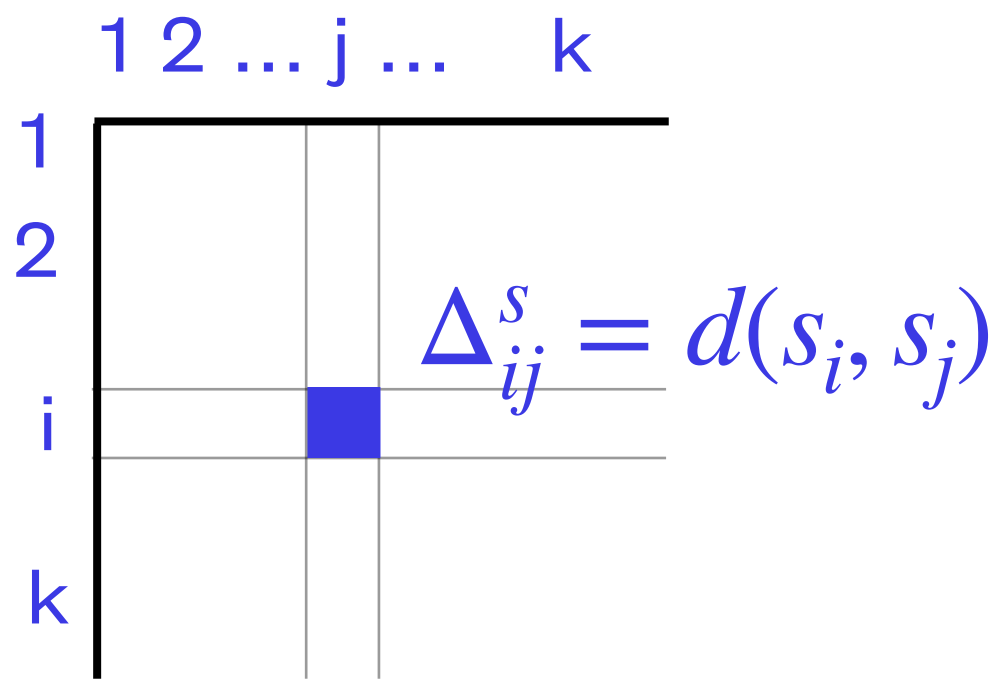
- We put forward **necessary** and **sufficient conditions** for delays to be safe (assuming only length of shortest paths between vertices is known)
- Those conditions are based on simple **geometric** relationships between the start and goal vertices of the agents
- This allows us to formulate a simple and robust, yet effective algorithm!

Distance Profile

- Given k agents, three $k \times k$ matrices: $\Delta = (\Delta^s, \Delta^{sg}, \Delta^g)$

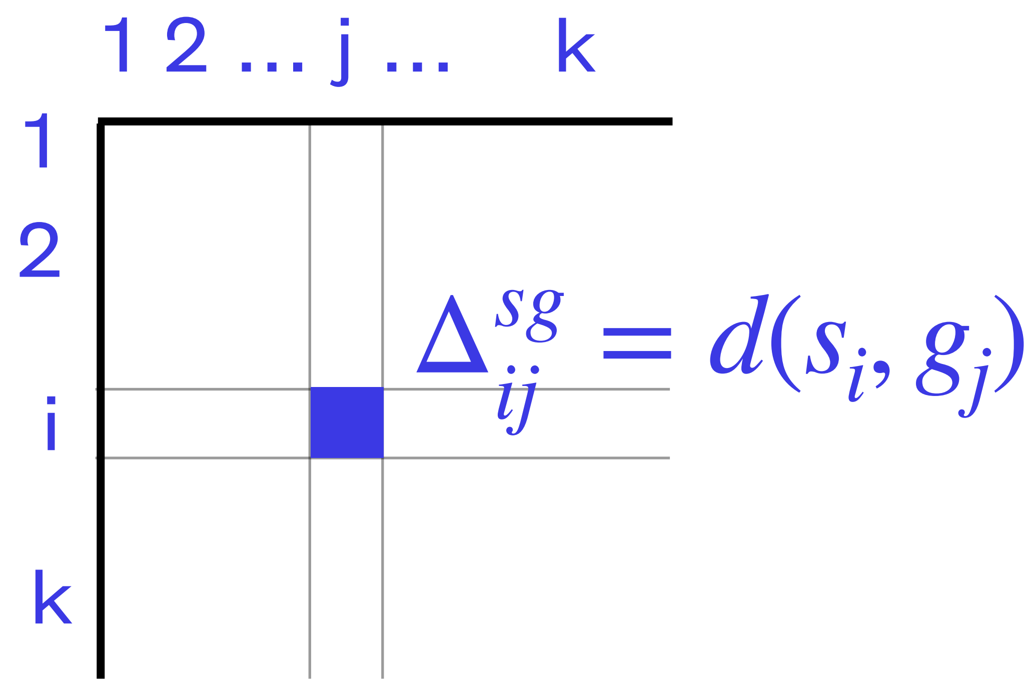
Δ^s

Distances between start positions



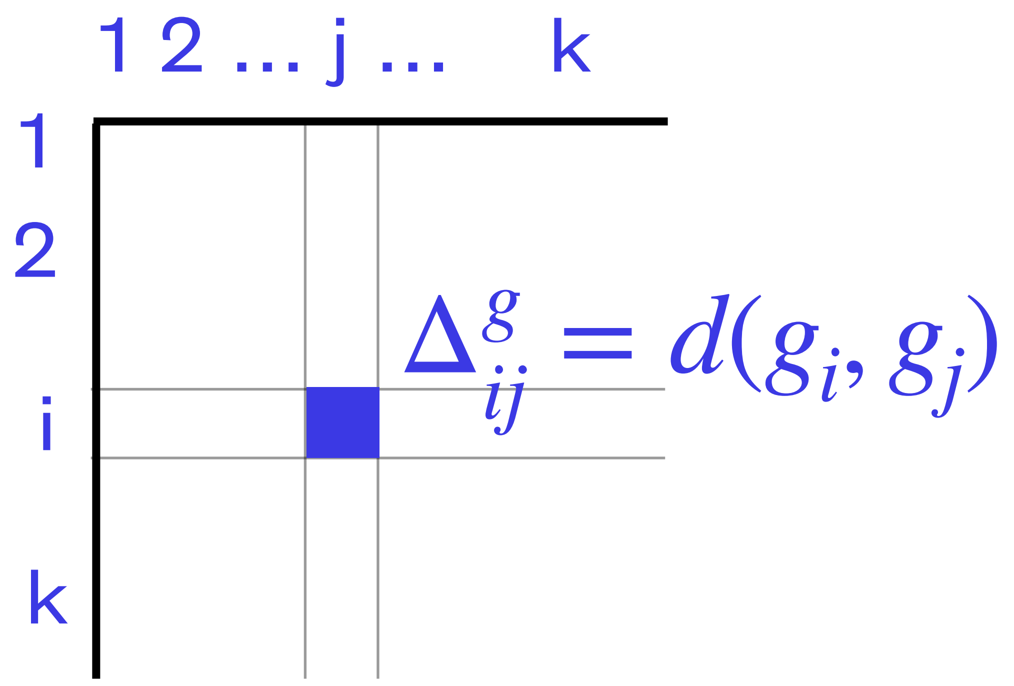
Δ^{sg}

Distances between start and goal positions

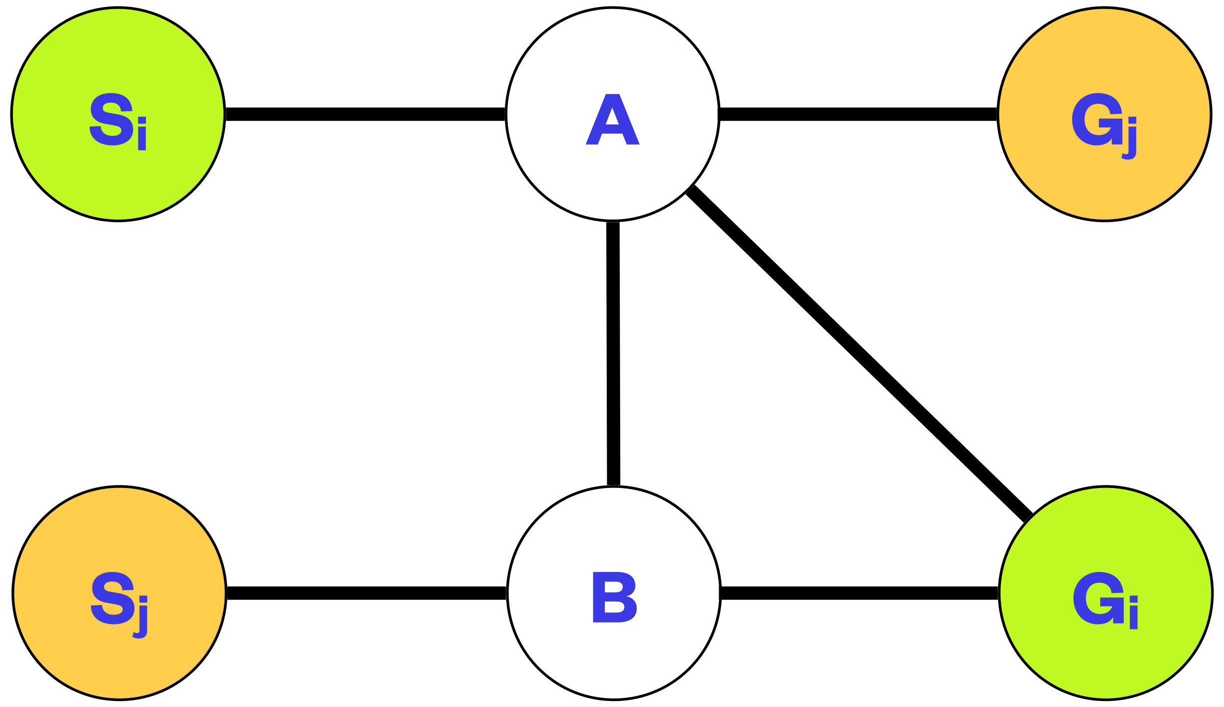


Δ^g

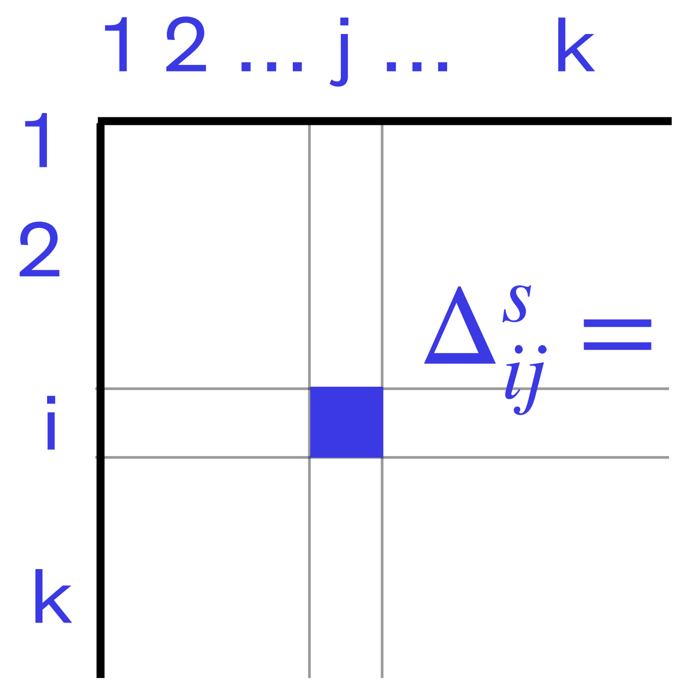
Distances between goal positions



Example

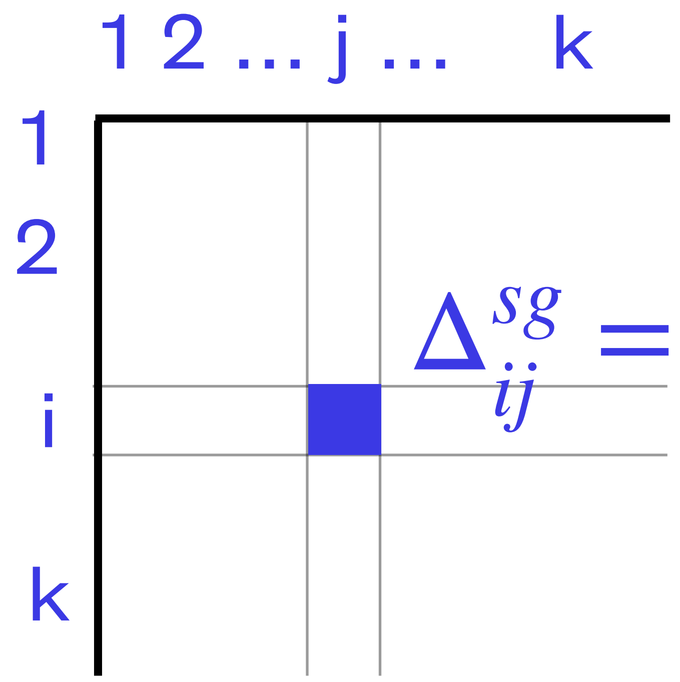


Δ^s



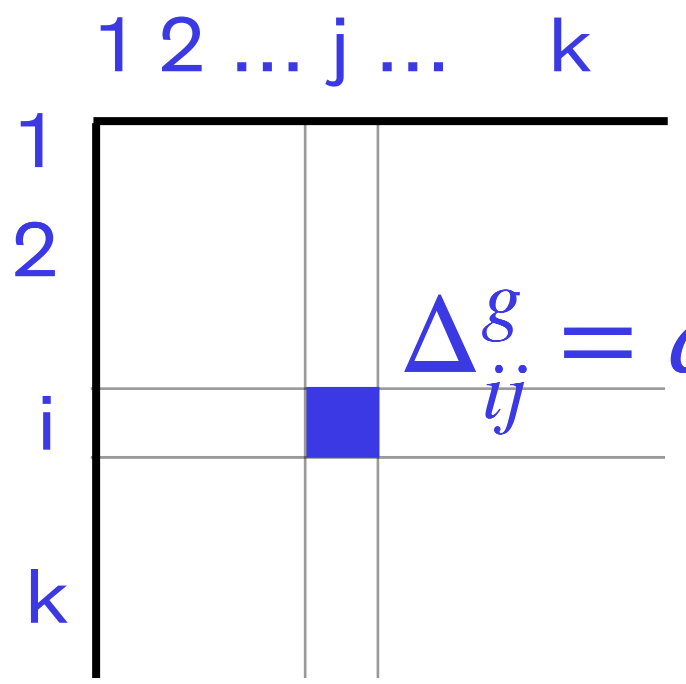
$\Delta^s_{ij} = d(s_i, s_j)$

Δ^{sg}



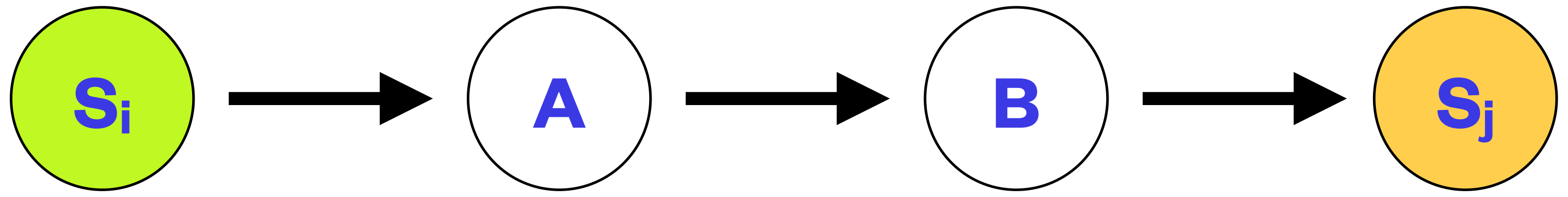
$\Delta^{sg}_{ij} = d(s_i, g_j)$

Δ^g

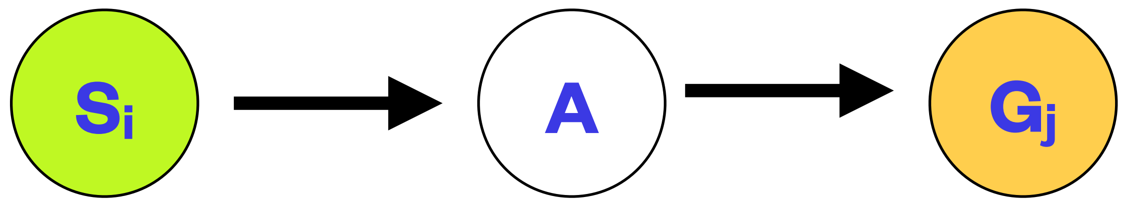


$\Delta^g_{ij} = d(g_i, g_j)$

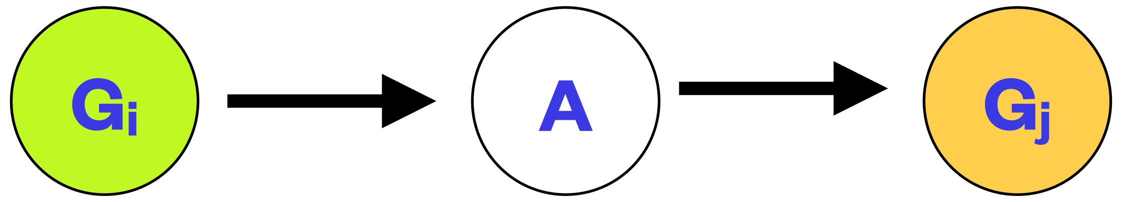
$\Delta^s_{ij} = 3$



$\Delta^{sg}_{ij} = 2$



$\Delta^g_{ij} = 2$



Minimising Safe Delays

- Given an instance I with distance profile Δ, the set of safe delay assignments τ is denoted $\mathscr{T}_\Delta$
- Our goal is to analyse the **optimisation problem**: $\arg\min_{\tau \in \mathscr{T}_\Delta} \sum_{i \in \mathscr{A}} \tau_i$

$$\arg\min_{\Pi \in \mathscr{M}_I} C(\Pi) \Longrightarrow \arg\min_{\tau \in \mathscr{T}_\Delta} \sum_{i \in \mathscr{A}} \tau_i$$

Complexity Analysis

- Decision problem corresponding to the minimisation of safe delays is **NP-complete**

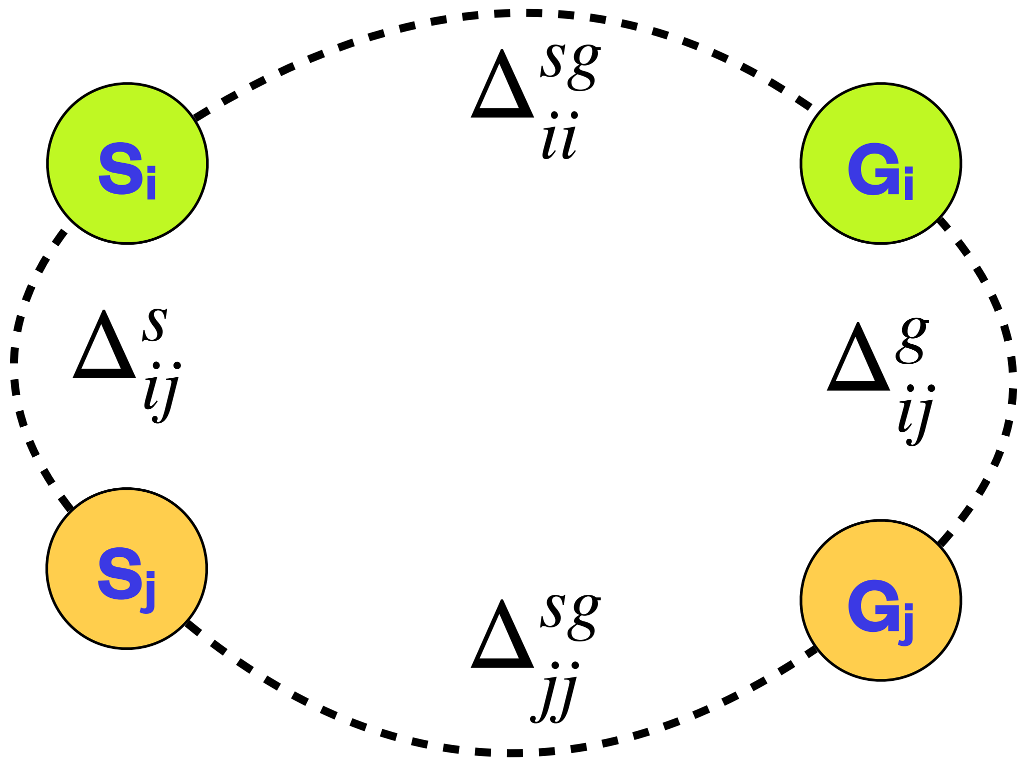
  Problem 1: Given a distance profile Δ and $m \in \mathbb{N}$, it exists $\tau \in \mathcal{T}_{\Delta}$ such that $\sum_{i \in \mathcal{A}} \tau_i \leq m$

- Problem 1 is equivalent to a well-know NP-complete **job scheduling problem** $(n \mid 1 \mid r_i \geq 0 \mid \sum_i C_i)$

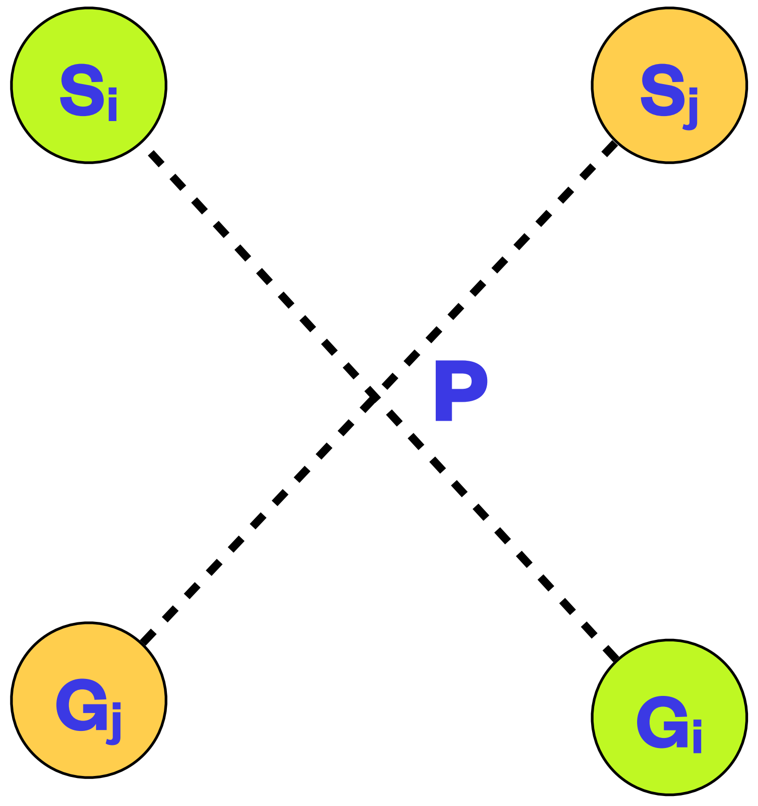
- Proof in the paper

Pairwise Geometric Constraints

- Calculate $\Psi_{ij} = \Delta^{s}_{ij} + \Delta^{g}_{ij} - \Delta^{sg}_{ii} - \Delta^{sg}_{jj}$



- If $\Psi_{ij} > 0$, agents i and j cannot possibly interfere with one another
- Hence, any pair of delays $\{\tau_i, \tau_j\}$ is **safe**!



P belongs to shortest paths from s_i to g_i and from s_j to g_j

By the **triangular inequalities**, we have:

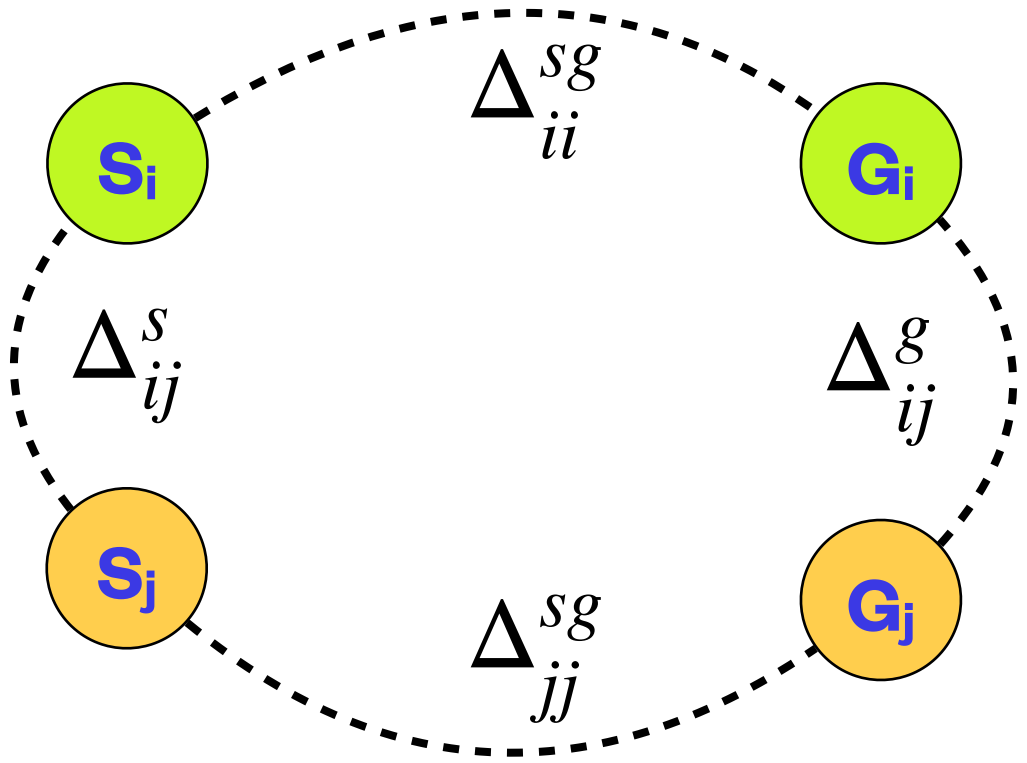
$$d(s_i, s_j) + d(g_i, g_j) \leq \underbrace{d(s_i, P) + d(P, s_j) + d(g_i, P) + d(P, g_j)}_{\substack{\parallel \\ d(s_i, g_i) + d(s_j, g_j)}}$$

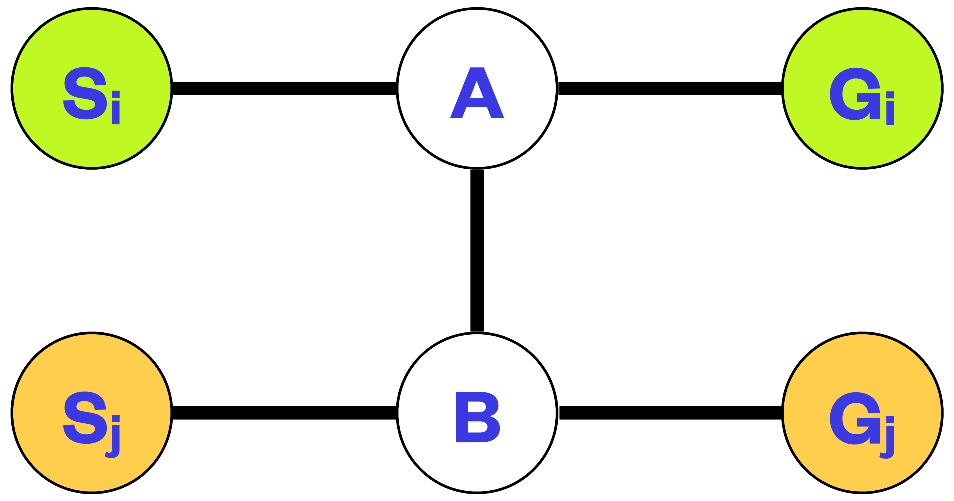
$$\Delta^{s}_{ij} + \Delta^{g}_{ij} \leq \Delta^{sg}_{ii} + \Delta^{sg}_{jj}$$

$$\Psi_{ij} = \Delta^{s}_{ij} + \Delta^{g}_{ij} - \Delta^{sg}_{ii} - \Delta^{sg}_{jj} \leq 0$$

Pairwise Geometric Constraints

- Calculate $\Psi_{ij} = \Delta^{s}_{ij} + \Delta^{g}_{ij} - \Delta^{sg}_{ii} - \Delta^{sg}_{jj}$



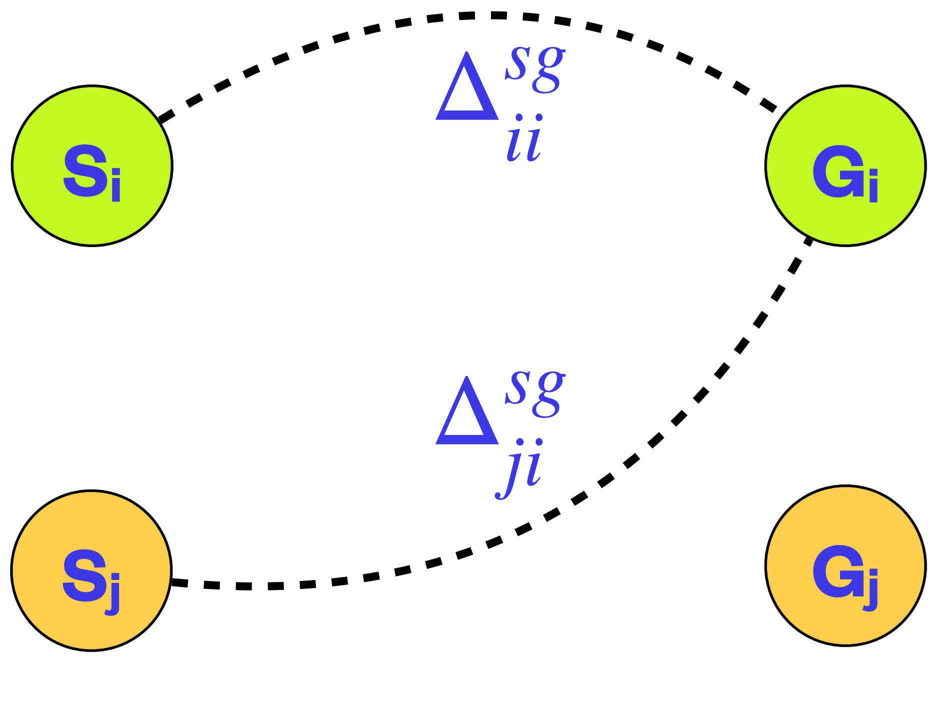


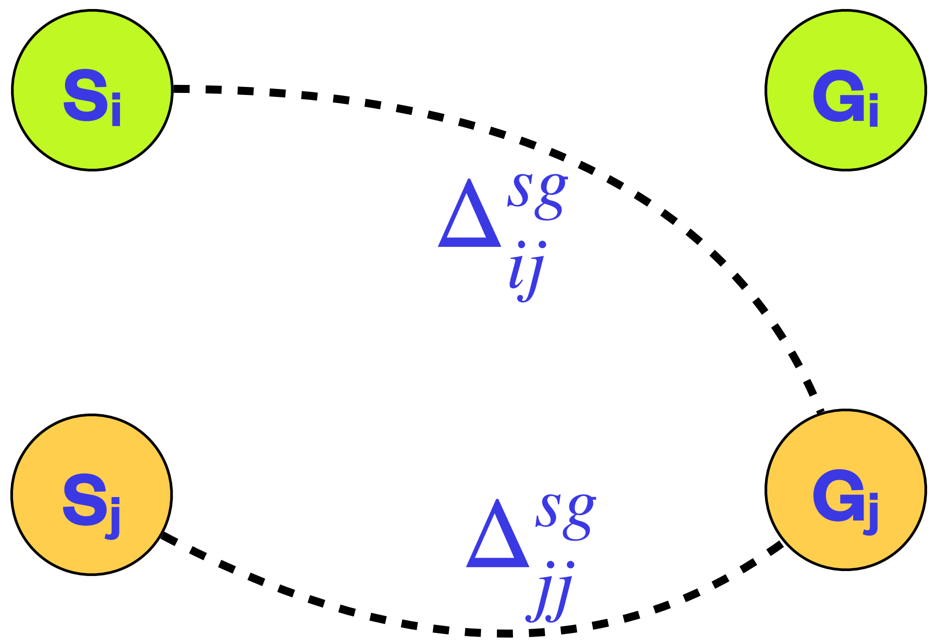
- $\Delta^{s}_{ij} = 3 \quad \Delta^{g}_{ij} = 3 \quad \Delta^{sg}_{ii} = 2 \quad \Delta^{sg}_{jj} = 2$
- $\Psi_{ij} = 3 + 3 - 2 - 2 = 2 > 0$
- Any pair $\{\tau_i, \tau_j\}$ is safe

- If $\Psi_{ij} > 0$, agents i and j cannot possibly interfere with one another
- Hence, any pair of delays $\{\tau_i, \tau_j\}$ is **safe**!

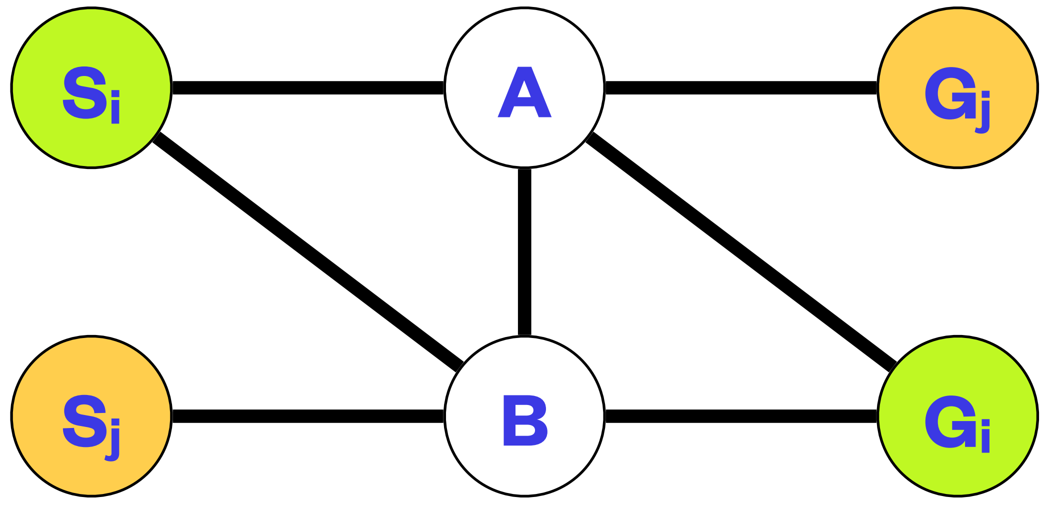
Pairwise Geometric Constraints

- If $\Psi_{ij} \leq 0$, agents i and j can potentially conflict
- To avoid that, τ_i and τ_j need to be chosen appropriately
- Calculate $\Lambda_{ij} = \Delta_{ii}^{sg} - \Delta_{ji}^{sg}$ and $\Lambda_{ji} = \Delta_{jj}^{sg} - \Delta_{ij}^{sg}$





- Any pair of delays $\{\tau_i, \tau_j\}$ such that $\tau_j - \tau_i \in (-\infty, -\Lambda_{ji}) \cup (\Lambda_{ij}, +\infty)$ is **safe**



- $\Psi_{ij} = 2 + 2 - 2 - 3 = -1 \leq 0$
- $\Delta_{ii}^{sg} = 2 \quad \Delta_{ji}^{sg} = 2 \quad \Delta_{jj}^{sg} = 3 \quad \Delta_{ij}^{sg} = 2$
- $\Lambda_{ij} = 2 - 2 = 0 \quad \Lambda_{ji} = 3 - 2 = 1$
- $\tau_j - \tau_i \in (-\infty, -1) \cup (0, +\infty)$ is safe
  - $\{\tau_i = 0,\ \tau_j = 0\}$ is **not safe!**
  - $\{\tau_i = 0,\ \tau_j = 1\}$ is **safe**

Theorem

- Theorem gives **necessary** and **sufficient** conditions for delays between agents i and j to be **safe** based on distance profile $\Delta^{(ij)}$
  - If $\Psi_{ij} > 0$, any pair of delays $\{\tau_i, \tau_j\}$ is safe
  - If $\Psi_{ij} < 0$, any pair of delays $\{\tau_i, \tau_j\}$ such that $\tau_j - \tau_i \in (-\infty, -\Lambda_{ji}) \cup (\Lambda_{ij}, +\infty)$ is safe
  - If $\Psi_{ij} = 0$, any pair of delays $\{\tau_i, \tau_j\}$ such that $\tau_j - \tau_i \in (-\infty, -\Lambda_{ji}) \cup (\Lambda_{ij}, +\infty) \cup (\{-\Lambda_{ji}, \Lambda_{ij}\} \cap (\Delta^s_{ij} + 1 + 2\mathbb{Z}))$ is safe
- No smaller delays between two agents can be calculated based on the available information (distance profile)
- Conditions do not rely on the **topology** of the graph
  - If the graph changes, but the lengths of shortest paths do not, the solution remains the same

Can we formulate an algorithm based on geometric constraints?

Delayed Shortest Path Algorithm

- DSP Algorithm:

1. Init Π with empty plan
2. For each agent i (according to a priority order *PO*)
    1. Calculate minimal safe delay τ_i based on distance profile Δ and Theorem's conditions
    2. $\Pi \leftarrow \Pi \cup \{\pi^{\tau_i}\}$
3. Return Π

- DSP performs **no search** in the graph
- DSP only uses length of shortest paths to determine safe delays
- Given a priority order, DSP provides minimal safe delays for the agents

Running DSP

	0	1	2	3	4
$\tau_1 = 0$	S_1 →	A →	B →	C →	G_1

THE FIRST AGENT HAS NO DELAY

$\tau_1 = 0$

$\Psi_{12} = 0 \quad \Lambda_{12} = 2 \quad \Lambda_{21} = 2$

$(\tau_2 - \tau_1) \in (-\infty, -2) \cup (2, \infty)$ IS SAFE

$\tau_2 = 3$

	0	1	2	3	4	5	6	7
$\tau_2 = 3$	→	→	→	S_2 →	C →	B →	A →	G_2

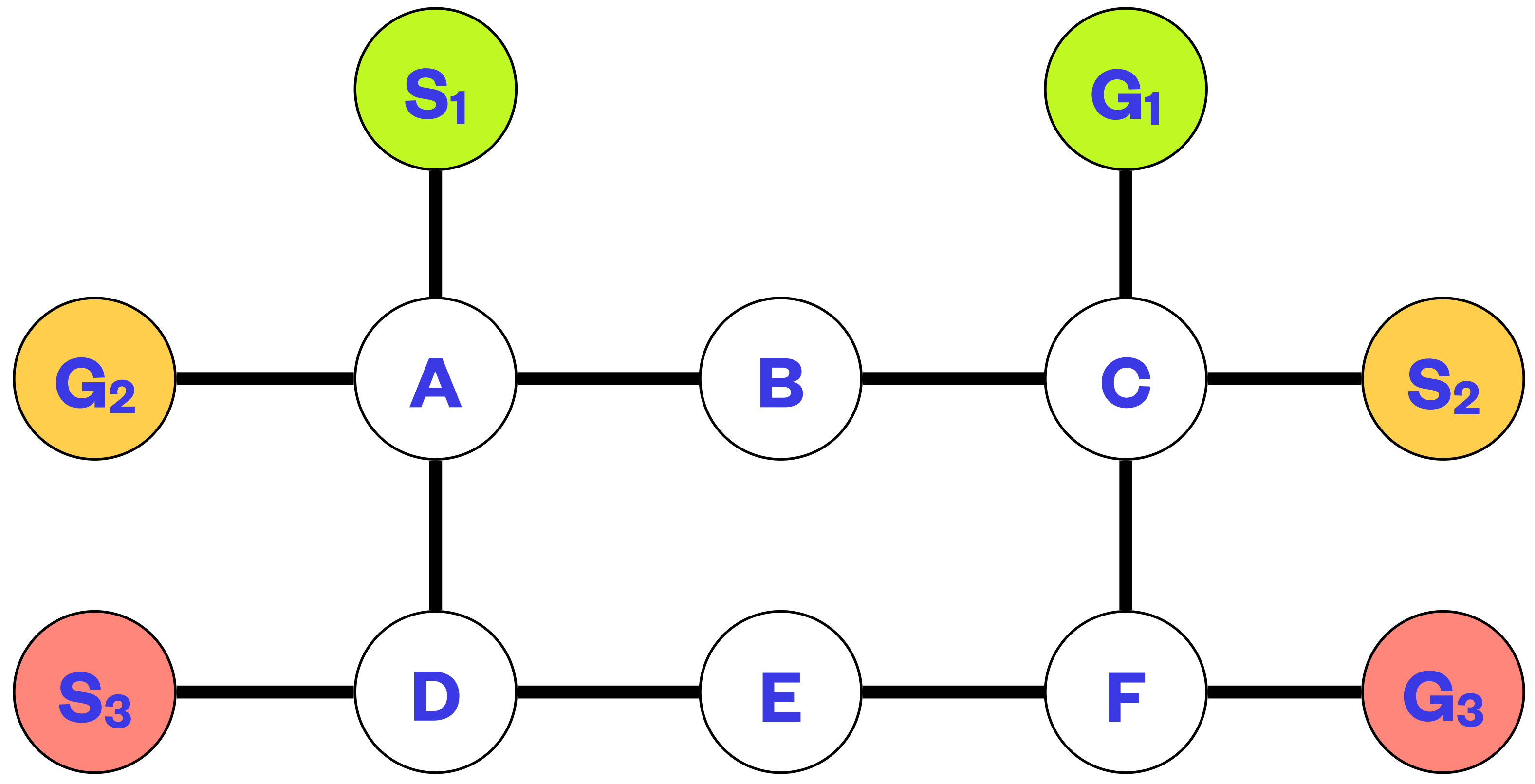
$\Psi_{23} = 2 > 0$ ANY DELAY IS SAFE

$\Psi_{13} = -2 < 0 \quad \Lambda_{13} = -1 \quad \Lambda_{31} = -1$

$(\tau_3 - \tau_1) \in (-\infty, 1) \cup (-1, +\infty)$ ANY DELAY IS SAFE

$\tau_3 = 0$

	0	1	2	3	4
$\tau_3 = 0$	S_3 →	D →	E →	F →	G_3



Priority order $PO = 1, 2, 3$

DSP’s Benefits

- **Robust**: uses very little information about the agents (shortest paths)
  - No information about the topology of the graph
- Involves **no planning**
- Minimises the **time** the agents spend on the floor, hence their **resources** (e.g. battery)
- Preserves agents’ **privacy** since it does not impose them to take any particular shortest path

How does DSP perform?

Experiments

- Algorithms:
  - SEQUENCE [Ma et al., 2021]
  - Prioritised Planning (PP) [Silver, 2005]
  - **Delayed Shortest Path (DSP)**
- Priority Orders:
  - Random (RND)
  - Shortest Path First (SH)
  - Longest Path First (LH)
  - **Lowest Delay First (LD)**

Results

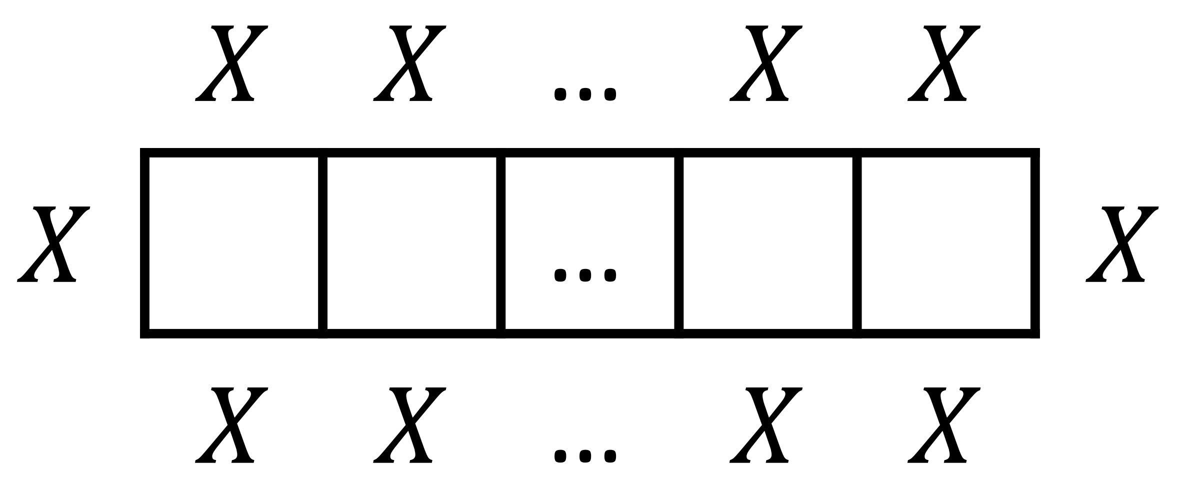
Average Plan Cost

	DSP				SEQ				PP			
#Agents	**RND**	**SH**	**LH**	**LD**	**RND**	**SH**	**LH**	**LD**	**RND**	**SH**	**LH**	**LD**
20	1.3	1.4	**1.1**	**1.1**	7.1	4.8	9.8	8.0	1.5	1.5	1.5	**1.1**
60	5.3	8.2	**3.9**	**3.9**	67.2	43.0	91.2	69.6	9.8	9.1	8.7	**3.9**
100	9.8	18.3	**6.7**	**6.7**	182.4	115.7	249.7	186.8	24.6	21.3	21.6	**6.7**

Average Running Time (sec)

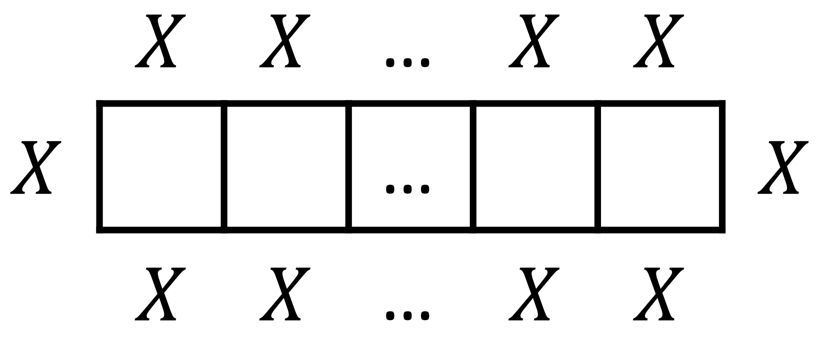
	SEQ/DSP	PP			
#Agents	**Any**	**RND**	**SH**	**LH**	**LD**
20	<0.1	0.1	0.2	0.1	0.2
60	<0.1	1.4	1.8	1.0	0.6
100	<0.1	3.7	4.6	2.5	1.0

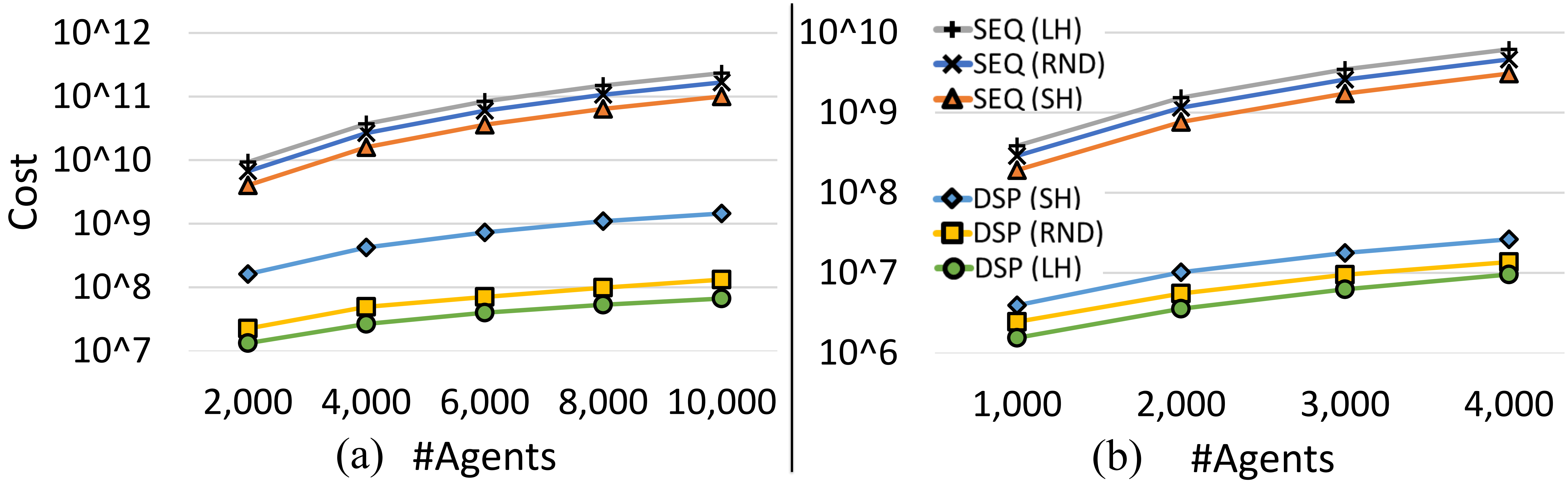
Domain: Long Corridor (1x100)

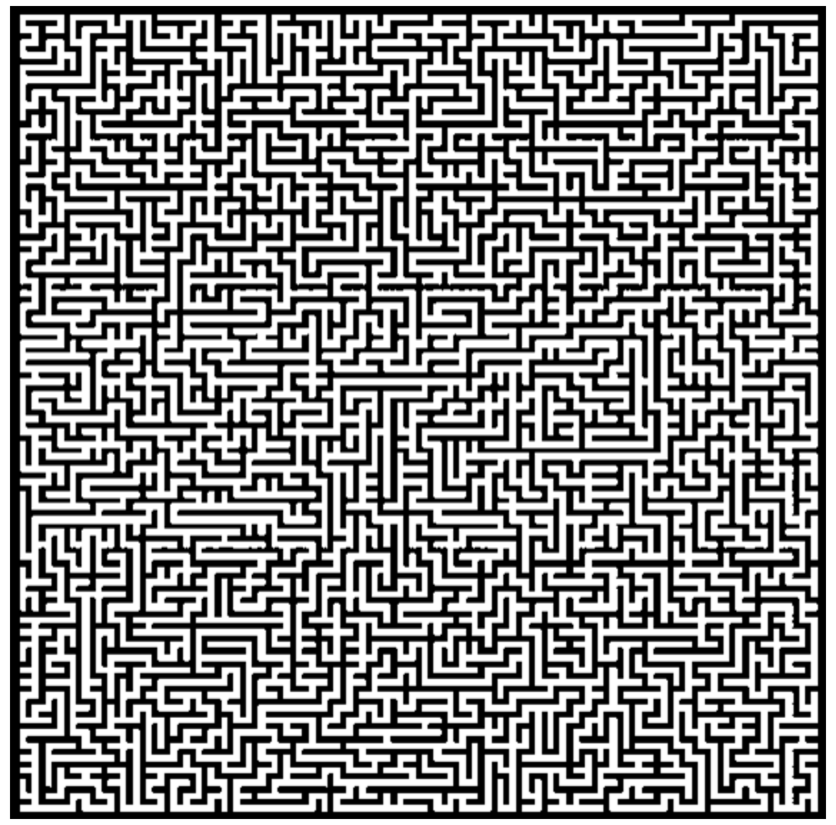


- DSP is more beneficial in **dense** domains
  - The agents use their shortest paths while, when waiting, do not block other agents with lower priority

Results







Very Long Corridor (1 x 10,000)

Maze Grid

- Our simple algorithm runs several orders of magnitudes faster than related methods while addressing problems with thousands of agents and returning low-cost solutions

Conclusions and Future Work

- Introduction to MAPF
- Delayed Shortest Path (DSP) algorithm
- Safe delays
- Geometric constraints

- Use the geometric constraints for other algorithms (e.g. Priority-Based Search)
- Extend geometric constraints for solving related problems (e.g. OMAPF and LMAPF)
- Priority orders

MAPF is a very active and open field of research with real-world applications.

Join the fun!