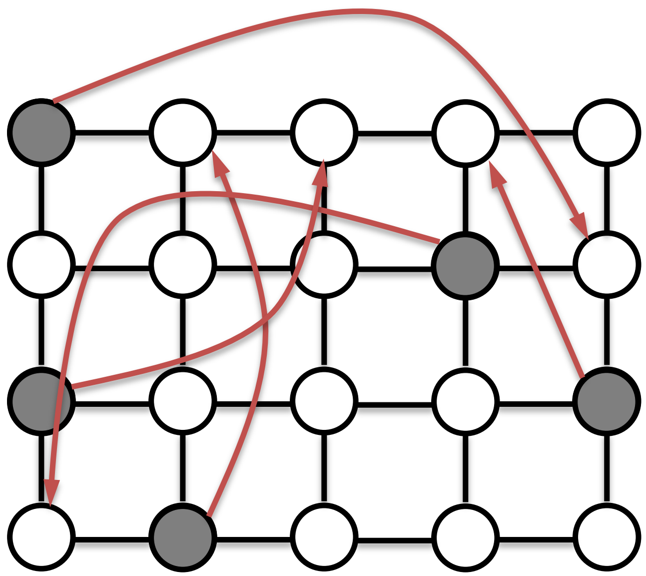
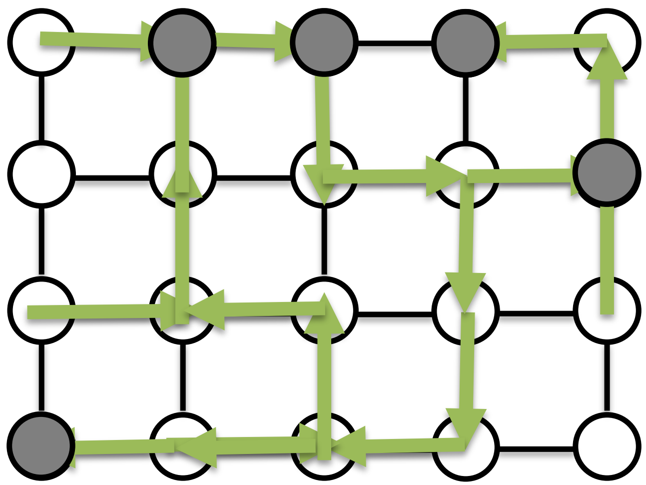
Multi-Agent Pathfinding

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

What is **multi-agent path finding** (MAPF)?





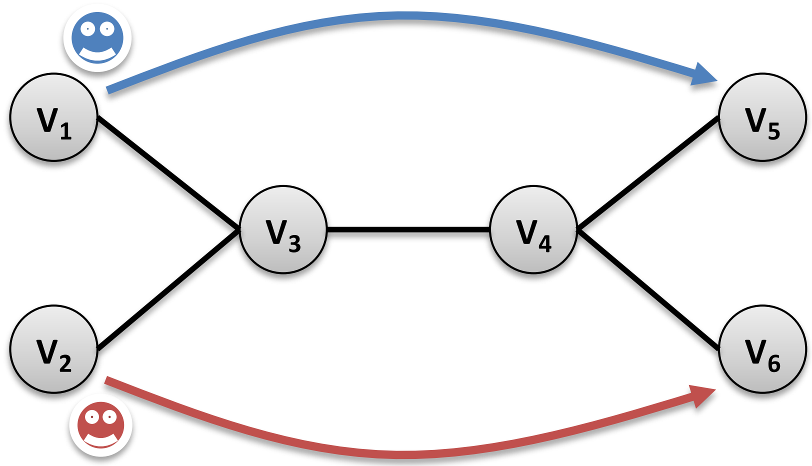
MAPF problem:

Find a **collision-free** plan (path) for each agent

Alternative names:

cooperative path finding (CPF), multi-robot path planning, pebble motion

- a **graph** (directed or undirected)
- a set of **agents**, each agent is assigned to two locations (nodes) in the graph (start, destination)



Each agent can perform either **move** (to a neighboring node) or **wait** (in the same node) actions.

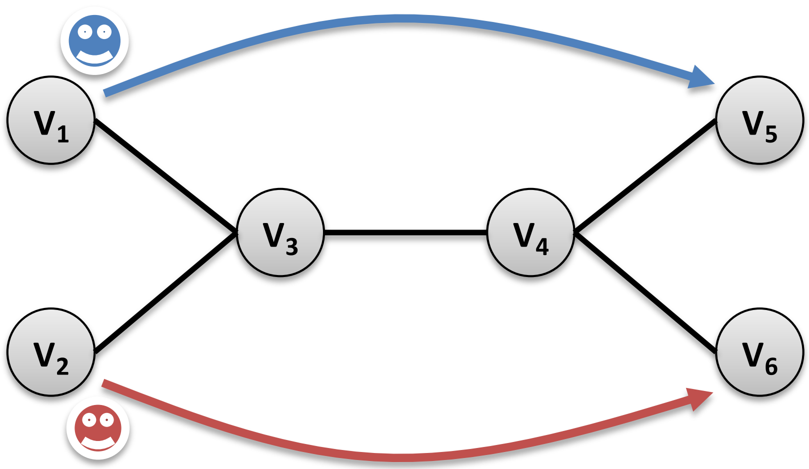
Typical assumption:

all move and wait actions have identical durations (plans for agents are synchronized)

Plan is a sequence of actions for the agent leading from its start location to its destination.

The **length of a plan** (for an agent) is defined by the time when the agent reaches its destination and does not leave it anymore.

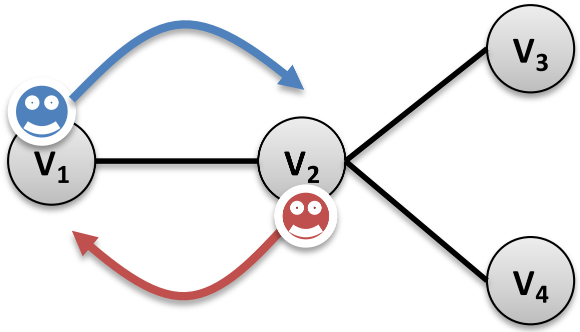
Find **plans** for all agents such that the plans **do not collide in time and space** (no two agents are at the same location at the same time).



time	agent 1	agent 2
0	$\mathbf{v_1}$	$\mathbf{v_2}$
1	wait $\mathbf{v_1}$	move $\mathbf{v_3}$
2	move $\mathbf{v_3}$	move $\mathbf{v_4}$
3	move $\mathbf{v_4}$	move $\mathbf{v_6}$
4	move $\mathbf{v_5}$	wait $\mathbf{v_6}$

Some necessary **conditions for plan existence**:

- no two agents are at the same start node
- no two agents share the same destination node (unless an agent disappears when reaching its destination)
- the number of agents is strictly smaller than the number of nodes



Agent at $\mathbf{v_i}$ cannot perform **move** $\mathbf{v_j}$ at the same time when agent at $\mathbf{v_j}$ performs **move** $\mathbf{v_i}$

Agents may swap position

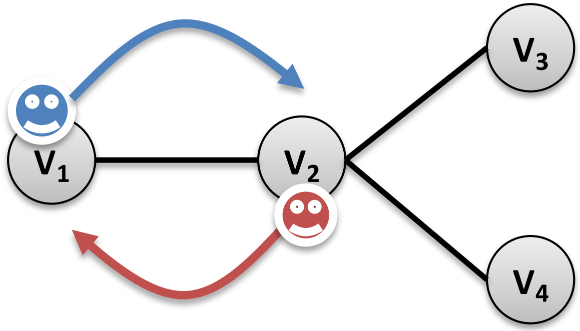
time	agent 1	agent 2
0	$\mathbf{v_1}$	$\mathbf{v_2}$
1	move $\mathbf{v_2}$	move $\mathbf{v_1}$

Agents use the same edge at the same time!

Swap is not allowed.

time	agent 1	agent 2
0	$\mathbf{v_1}$	$\mathbf{v_2}$
1	move $\mathbf{v_2}$	move $\mathbf{v_3}$
2	move $\mathbf{v_4}$	move $\mathbf{v_2}$
3	move $\mathbf{v_2}$	move $\mathbf{v_1}$

No-train constraint



Agent at $\mathbf{v_i}$ cannot perform **move** $\mathbf{v_j}$ if there is another agent at $\mathbf{v_j}$

Agent can approach a node that is currently occupied but will be free before arrival.

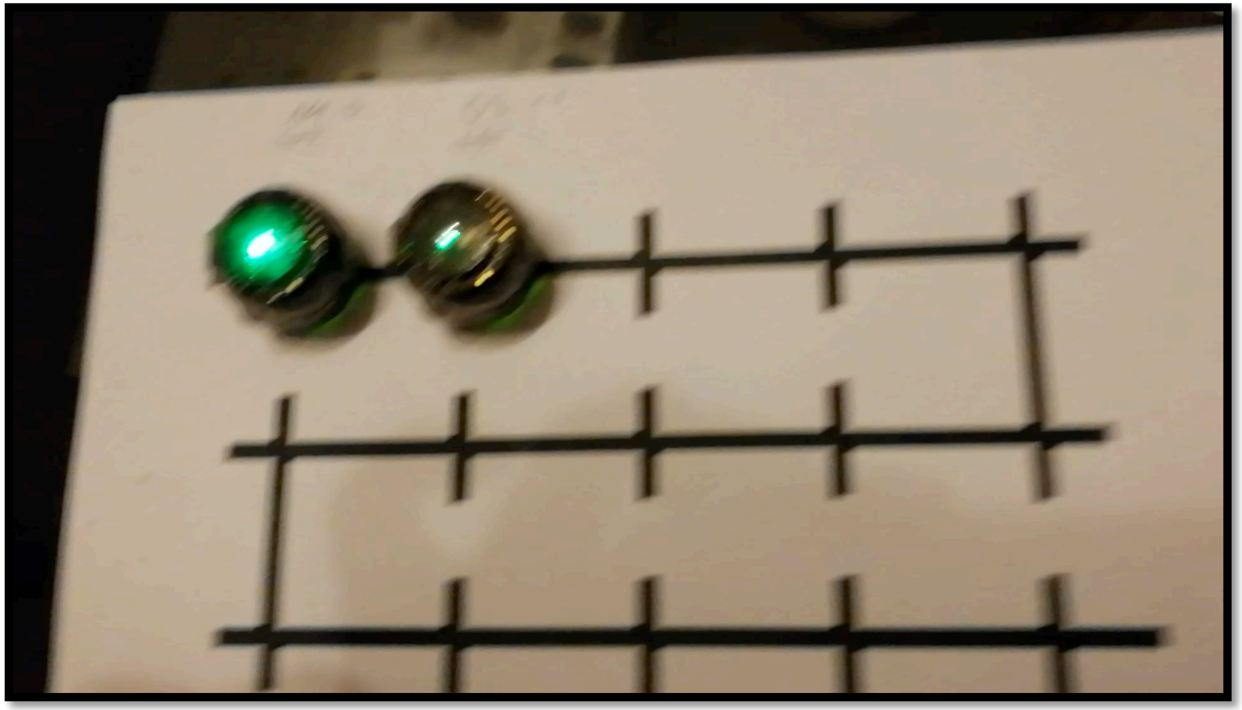
time	agent 1	agent 2
0	$\mathbf{v_1}$	$\mathbf{v_2}$
1	move $\mathbf{v_2}$	move $\mathbf{v_3}$
2	move $\mathbf{v_4}$	move $\mathbf{v_2}$
3	move $\mathbf{v_2}$	move $\mathbf{v_1}$

Agents form a **train**.

Trains may be forbidden.

time	agent 1	agent 2
0	$\mathbf{v_1}$	$\mathbf{v_2}$
1	*wait* v_1	move $\mathbf{v_3}$
2	move $\mathbf{v_2}$	*wait* v_3
3	move $\mathbf{v_4}$	*wait* v_3
4	*wait* v_4	move $\mathbf{v_2}$
5	*wait* v_4	move $\mathbf{v_1}$
6	move $\mathbf{v_2}$	*wait* v_1

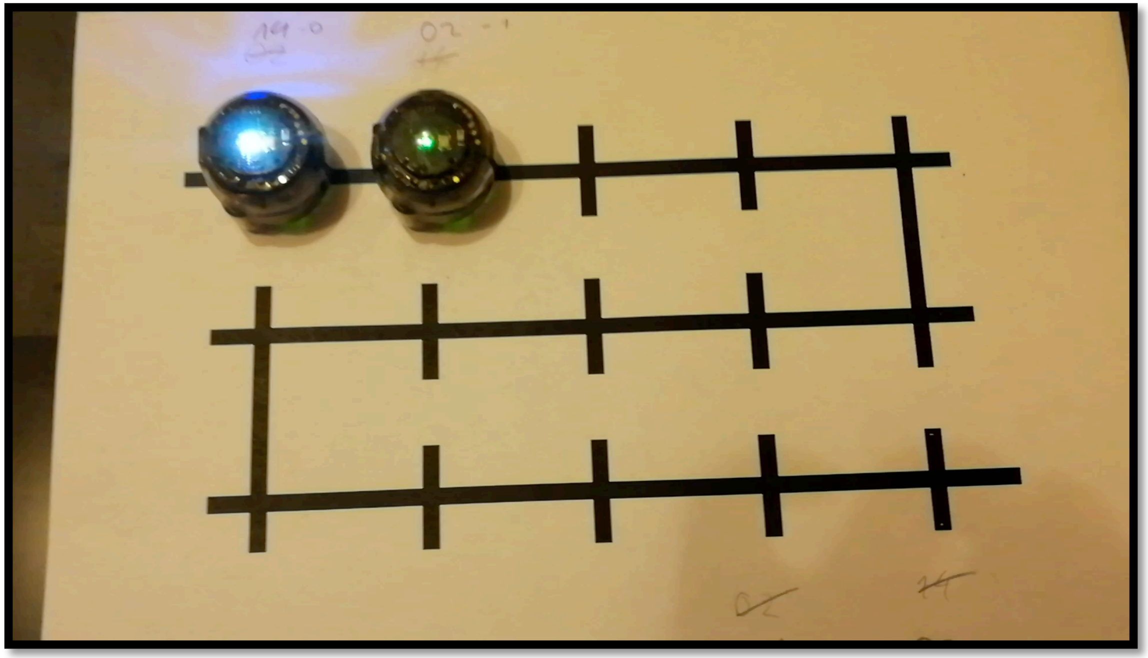
If any agent is delayed then trains may cause collisions during execution.



To prevent such collisions we may introduce more space between agents.

k-robustness

An agent can visit a node, if that node has not been occupied in recent *k* steps.



1-robustness covers both no-swap and no-train constraints

- No plan (path) has a cycle.
- No two plans (paths) visit the same same location.
- Waiting is not allowed.
- Some specific locations must be visited.
- …

Vertex conflict – two agents are at the same time at the same vertex

Edge conflict – two agents use the same edge at the same direction

Swapping conflict – two agents use the same edge at different direction

Following conflict – one agent follows another one (train)

Cycle conflict – agents are following each other forming a "rotating cycle" pattern

How to measure quality of plans?

Two typical criteria (to minimize):

- **Makespan**
  - distance between the start time of the first agent and the completion time of the last agent
  - maximum of lengths of plans (end times)
- **Sum of costs (SOC)**
  - sum of lengths of plans (end times)

time	agent 1	agent 2
0	$\mathbf{v}_1$	$\mathbf{v}_2$
1	wait $\mathbf{v}_1$	move $\mathbf{v}_3$
2	move $\mathbf{v}_3$	move $\mathbf{v}_4$
3	move $\mathbf{v}_4$	move $\mathbf{v}_6$
4	move $\mathbf{v}_5$	wait $\mathbf{v}_6$

Makespan = 4
SOC = 7

Optimal single agent path finding is tractable.

– e.g. Dijkstra's algorithm

Sub-optimal multi-agent path finding (with two free unoccupied nodes) is tractable.

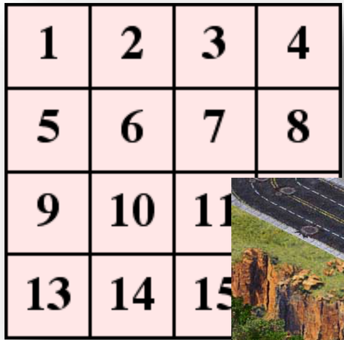
– e.g. algorithm Push and Rotate

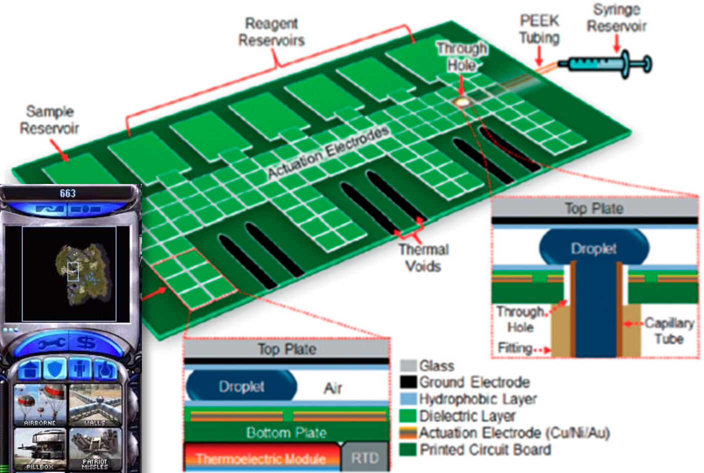
MAPF, where agents have joint goal nodes (it does not matter which agent reaches which goal) is tractable.

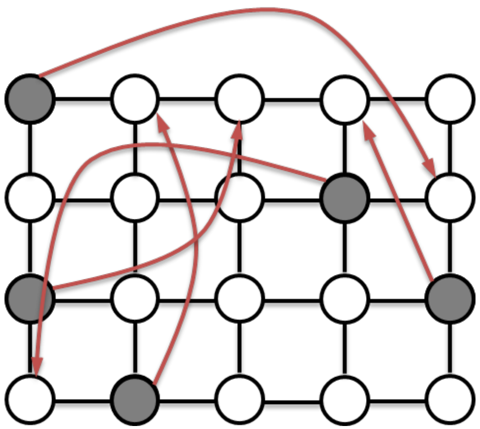
– reduction to min-cost flow problem

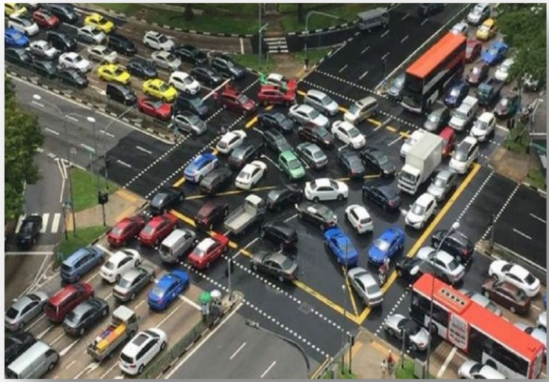
Optimal (makespan, SOC) multi-agent path finding is **NP-hard**.

Applications









Offline MAPF

Online MAPF

	Warehouse	**Intersection**
Fixed set of agents	**Fixed set of agents**	**Sequence of agents**
One task per agent	**Sequence of tasks**	**One task per agent**

Search-based techniques

- state-space search (A*)
  - state = location of agents at nodes
  - transition = performing one action for each agent
- conflict-based search

Reduction-based techniques

- translate the problem to another formalism (SAT/CSP/ASP …)

Topics

- Aplications
  - Warehouse (pickup-and-delivery)
  - Intersections
- Extensions
  - On-line MAPF
  - Robust MAPF
  - Large agents
  - Kinematics constraints
  - Continuosu time
  - Capacitated arcs
- Solvers
  - Search-based
  - Compilation-based (SAT, CSP, ASP, PDDL)
- MAPF and learning