Multi-Agent Path Finding Problem

CogRobo Cognitive Robotics Laboratory

Sabancı Üniversitesi

Multi-Agent Path Finding Problems

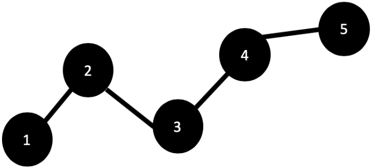
- Multi-Agent Path Finding (**MAPF**) Problem
  - Finding a plan for each agent in an environment
  - Without collisions
  - Constraints on plan length
- Optimization Variants
  - Maximum makespan
  - Total plan length
- Robotics [Lee and Yu, 2009],
  video games [Standley and Korf, 2011],
  autonomous aircraft towing vehicles [Morris *et al.*, 2016],
  traffic control [Dresner and Stone, 2008],
  autonomous warehouse systems [Wurman *et al.*, 2008].

[1] https://spectrum.ieee.org/robotics/robotics-software/three-engineers-hundreds-of-robots-one-warehouse
https://www.youtube.com/watch?v=6KRjuuEVEZs

Paths and Traversals

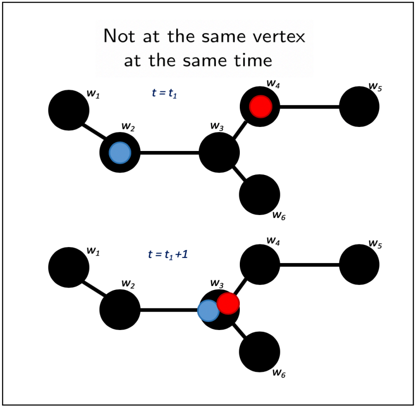
- A *path* P is a sequence of vertices $\langle w_1, w_2, \ldots, w_n \rangle$, such that every $w_i \in V$ is a vertex in a graph G and for every w_i, there is an edge to w_{i+1}.
- A *traversal f of a path* $P = \langle w_1, w_2, \ldots, w_n \rangle$ in a graph G within some time t ($t \in \mathbb{Z}^+$) is an onto function that maps every nonnegative integer less than or equal to t to a vertex in P, such that, for every w_i and w_j in P and for every $x < t$, if $f(x) = w_i$ and $f(x+1) = w_j$, then $w_j = w_i$ or $w_j = w_{i+1}$.

$$P = \langle 1, 2, 3, 4, 5 \rangle$$
$$f(P) = \langle 1, 1, 2, 3, 4, 4, 4, 5, 5 \rangle$$

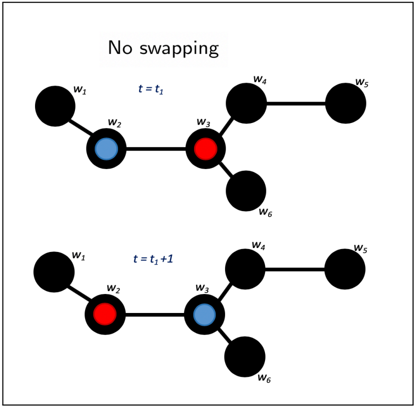


No Collisions of Traversals

- Let f_i and f_j be traversals of two different paths P_i and P_j, respectively, in a graph G within some time t.
- We say that the traversals f_i and f_j *do not collide with each other* within time t if,
  - for every $x, x' \leq t$, if $f_i(x) = f_j(x')$ then $x \neq x'$
  - for every time $x < t$, if $f_i(x) = f_j(x+1)$ then $f_i(x+1) \neq f_j(x)$



Not at the same vertex at the same time



No swapping

MAPF Problem

Input

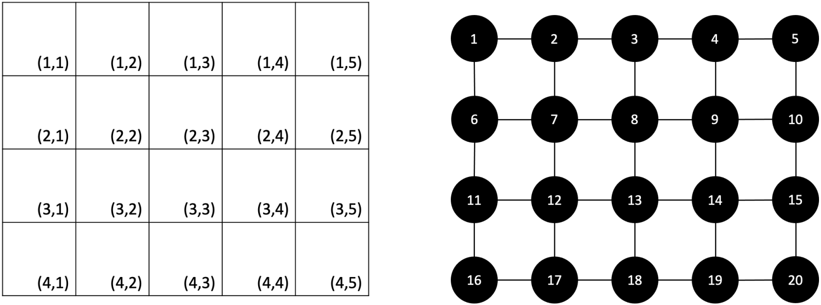
- A nonempty set $A = \{a_1, \ldots, a_n\}$ of agents $(n > 0)$.
- A graph $G = (V, E)$ (environment abstraction)
- A function $\mathit{init} : A \mapsto V$ (initial locations of agents)
- A function $\mathit{goal} : A \mapsto V$ (goal locations of agents)
- A set $O \subseteq V$ (obstacles)
- A positive integer τ (maximum makespan—plan length)

Output

For every agent $a_i \in A$, for some positive integer $u \leq \tau$,

- a path $P_i = \langle w_{i,1}, \ldots, w_{i,n_i} \rangle$ of length n_i $(n_i \leq u)$
  - that the agent a_i will follow to reach its goal location from its initial location (i.e., $w_{i,1} = \mathit{init}(a_i)$ and $w_{i,n_i} = \mathit{goal}(a_i)$),
  - without colliding with any obstacles (i.e., $w_{i,j} \in V \setminus O$), and
- a traversal f_i of the path P_i within time u, such that
  - for every other agent $a_j \in A$ with a path P_j and its traversal f_j within u, $f_i(P_i)$ and $f_j(P_j)$ do not collide with each other.

MAPF Example



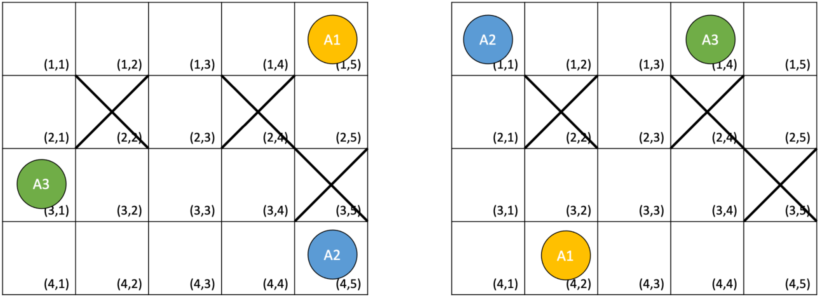
(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)

(a) A 4x5 grid

(b) Graph representation of the grid

Figure: How to represent a grid as a graph?

MAPF Example

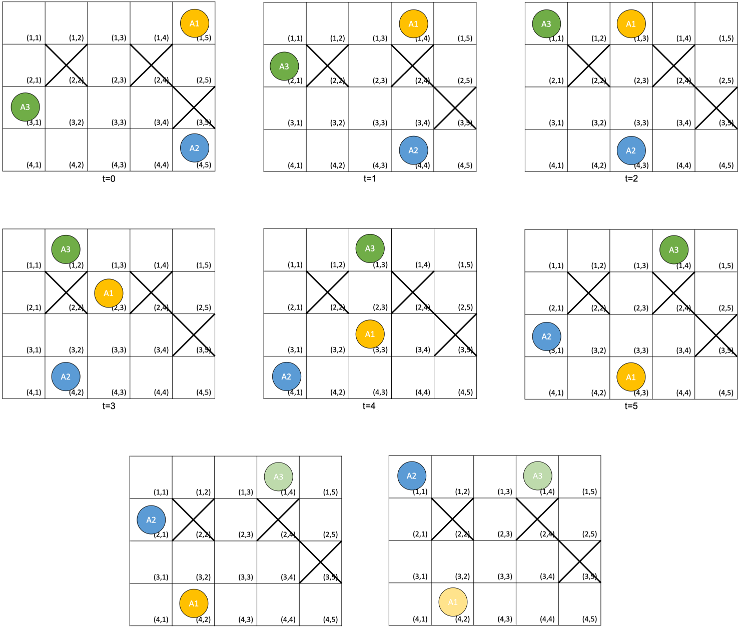


(a) Initial locations

(b) Goal locations

Figure: An example MAPF instance with 3 agents and initial & goal locations. Crossed cells denote cells with obstacles.

MAPF Example



t=0 t=1 t=2

t=3 t=4 t=5

t=6 t=7

MAPF-ASP

- *Requirements:* Python 3, Clingo[1]
- *Input:* input file name, output file name (optional)
- *Output:* an output file with plans and optimization values (default name: plan.txt)

Running:

```
python3 MAPF-ASP.py input.txt
python3 MAPF-ASP.py input.txt -o out.txt
```

[1] https://potassco.org/clingo/

MAPF-ASP: Input File Format

MAPF input:

- Graph (Grid size)

Input file format:

```
s <row_size> <column_size>
```

MAPF-ASP: Input File Format

MAPF input:

- Graph (Grid size)
- Maximum makespan

Input file format:

```
s <row_size> <column_size>
m <makespan>
```

MAPF-ASP: Input File Format

MAPF input:

- Graph (Grid size)
- Maximum makespan
- Agents

Input file format:

```
s <row_size> <column_size>
m <makespan>
a <agent_count>
```

MAPF-ASP: Input File Format

MAPF input:

- Graph (Grid size)
- Maximum makespan
- Agents
- Initial & goal locations of agents

Input file format:

```
s <row_size> <column_size>
m <makespan>
a <agent_count>
```

For each agent:

```
<init_r> <init_c> <goal_r> <goal_c>
```

MAPF-ASP: Input File Format

MAPF input:

- Graph (Grid size)
- Maximum makespan
- Agents
- Initial & goal locations of agents
- Obstacles

Input file format:

`s <row_size> <column_size>`

`m <makespan>`

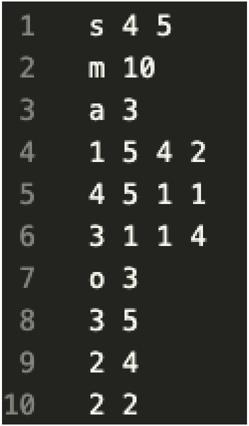
`a <agent_count>`

For each agent:
`<init_r> <init_c> <goal_r> <goal_c>`

`o <obstacle_count>`

For each obstacle:
`<obs_row> <obs_col>`

MAPF-ASP: Input File Format



```
s 4 5
m 10
a 3
1 5 4 2
4 5 1 1
3 1 1 4
o 3
3 5
2 4
2 2
```

Input file format:

`s <row_size> <column_size>`

`m <makespan>`

`a <agent_count>`

For each agent:
`<init_r> <init_c> <goal_r> <goal_c>`

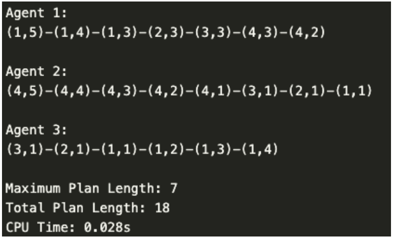
`o <obstacle_count>`

For each obstacle:
`<obs_row> <obs_col>`

MAPF-ASP: Output File Format

The output file includes:

- Plans for each agent.
  - Agent 1 is at (1,5) at time step 0.
  - Agent 2 is at (4,3) at time step 2.
- Maximum plan length.
- Total plan length.
- Computation time.



```
Agent 1:
(1,5)-(1,4)-(1,3)-(2,3)-(3,3)-(4,3)-(4,2)

Agent 2:
(4,5)-(4,4)-(4,3)-(4,2)-(4,1)-(3,1)-(2,1)-(1,1)

Agent 3:
(3,1)-(2,1)-(1,1)-(1,2)-(1,3)-(1,4)

Maximum Plan Length: 7
Total Plan Length: 18
CPU Time: 0.028s
```

Experiments

- How does the computation time change with the grid size?
- How does the computation time change with the number of agents?
- How does the makespan change with the number of agents?
- How does the total plan length change with the number of obstacles?

  …

Experiments

Input				*Output*		
Grid Size	# Agents	Maximum Makespan	# Obstacles	Maximum Plan Length	Total Plan Length	Time (s)
	⋮	⋮	⋮	⋮	⋮	⋮

References I

Kurt M. Dresner and Peter Stone.
A multiagent approach to autonomous intersection management.
J. Artif. Intell. Res. (JAIR), 31:591–695, 2008.

Jae-Yeong Lee and Wonpil Yu.
A coarse-to-fine approach for fast path finding for mobile robots.
In *Proc. of IROS*, pages 5414–5419, 2009.

Robert Morris, Corina S. Pasareanu, Kasper Søe Luckow, Waqar Malik, Hang Ma, T. K. Satish Kumar, and Sven Koenig.
Planning, scheduling and monitoring for airport surface operations.
In *Planning for Hybrid Systems, Papers from AAAI Workshop.*, 2016.

Trevor Scott Standley and Richard E. Korf.
Complete algorithms for cooperative pathfinding problems.
In *Proc. of IJCAI*, pages 668–673, 2011.

Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz.
Coordinating hundreds of cooperative, autonomous vehicles in warehouses.
AI Magazine, 29(1):9–20, 2008.