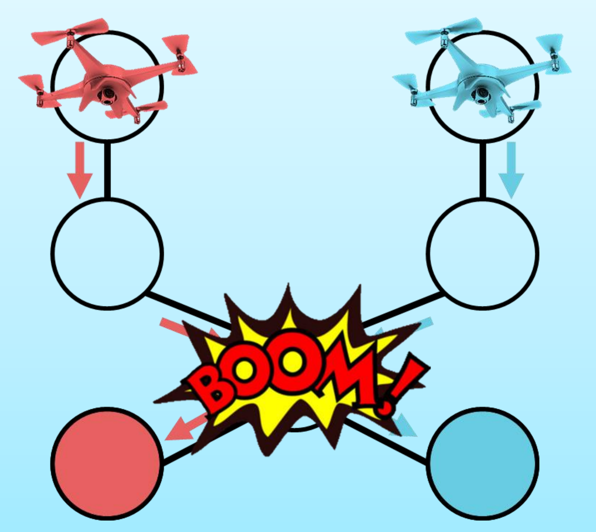
Probabilistic Robust Multi-Agent Path Finding

Dor Atzmon, Roni Stern, Ariel Felner, Nathan Sturtevant, Sven Koenig
Ben-Gurion University of the Negev, Palo Alto Research Center (PARC), University of Alberta, USC

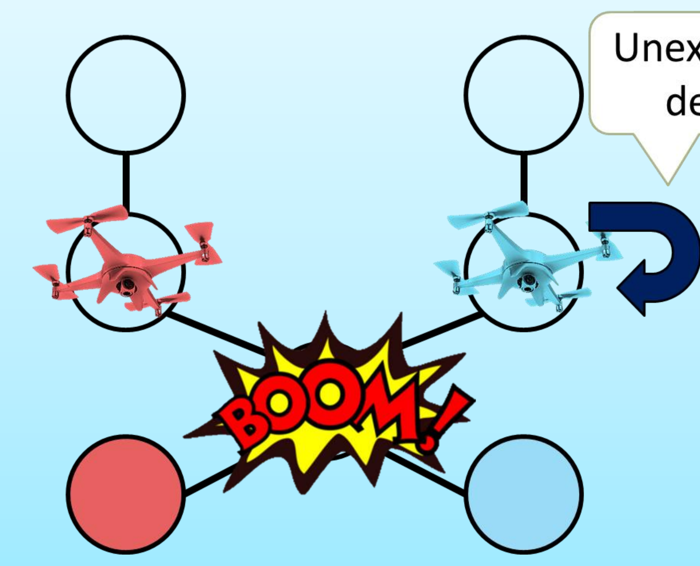
1. Multi-Agent Path Finding (MAPF)

- **Input**
  - A map with N locations
  - A set of agents, each with start and goal locations
- **Actions** - An agent can move or wait
- **Task** - Find a path for each agent
- **Constraints** - Avoid conflicts
- **Target** - Minimize the **sum of travel costs**



2. Robust solutions?

- In many real scenarios agents may get delayed unexpectedly.
  - Original plan cannot be followed
- We want a solution that is **robust** to such delays with **high probability**



3. p-Robust MAPF (pR-MAPF)

- p_d – probability for a delay
- p – probability threshold

$P_0(\pi)$ – probability that no conflict will occur in π

Plan π is p-Robust iff $P_0(\pi) \geq p$

4. Conflicts

MAPF

	0	1	2	3
red agent	s_1	A	C	g_1
blue agent	s_2	B	C	g_2

Conflict: Two agents are at the same location at the same time

pR-MAPF

		t_1		t_2	
red agent	s_1	C	...	g_1	g_1
blue agent	s_2	B	...	C	g_2

Potential Conflict: Two agents are at the same location (even in different times)

5. p-Robust Conflict Based Search (pR-CBS)

- **Conflict-Based Search (CBS)** - a state-of-the-art MAPF solver

Each agent plans independently → **Conflicts** are identified → A constraint is set on one of the agents **to avoid the conflict** → Replan the constrained agent → BFS on the constraint tree

- **p-Robust CBS (pR-CBS)** - a MAPF solver that returns p-robust solutions

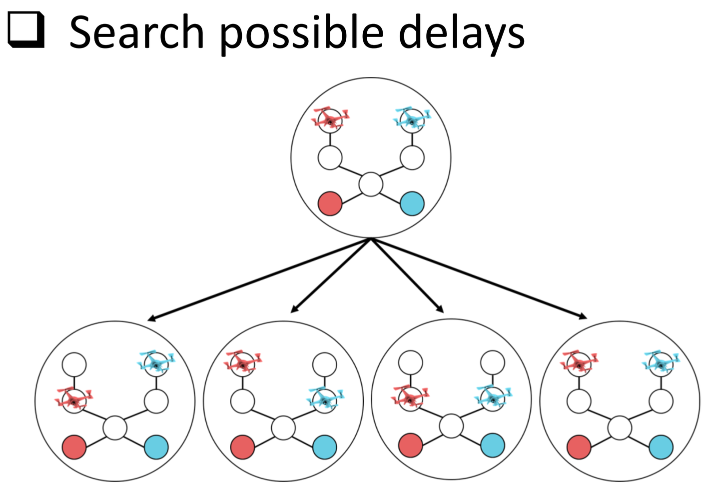
Each agent plans independently → **Potential conflicts** are identified → **Verify if $P_0(\pi) \geq p$** → A constraint is set on the agents **to avoid the conflict and to force the conflict** → Replan the constrained agent → BFS on the constraint tree

How to verify if $P_0(\pi) \geq p$?

A solution may contain potential conflicts

Deterministic Verifier

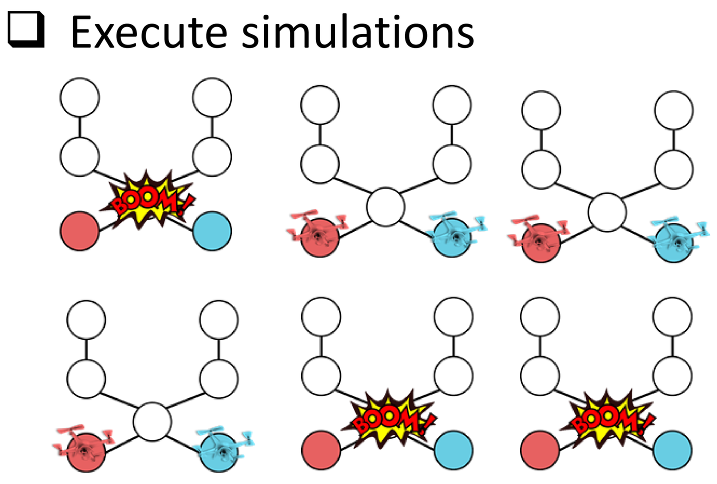
- Search possible delays



- Calculate $P_0(\pi)$
- Verify $P_0(\pi) \geq p$

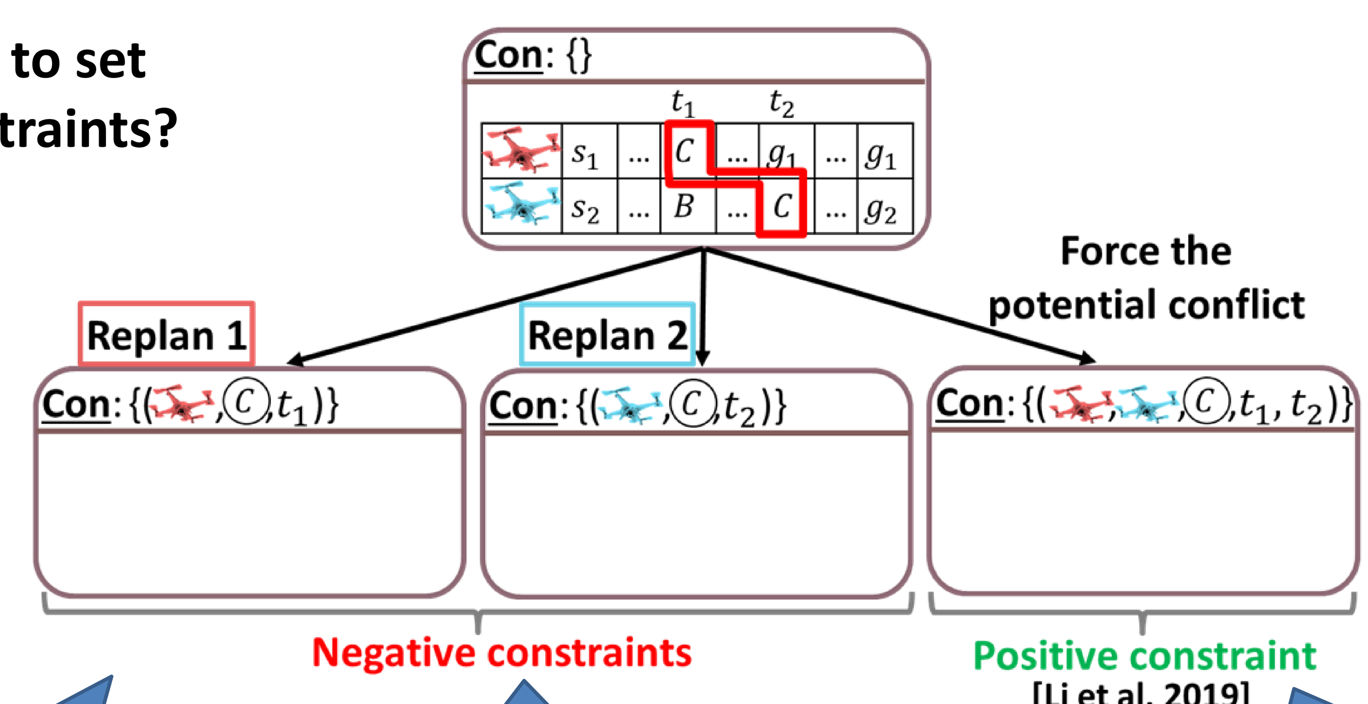
Monte-Carlo Verifier

- Execute simulations



- Verify statistically $P_0(\pi) \geq p$

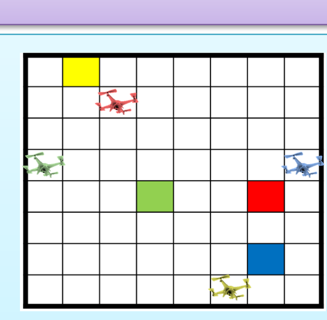
How to set constraints?



- **cannot occupy** C at timestep t_1
- **cannot occupy** C at timestep t_2
- and **must occupy** C at timesteps t_1 and t_2

6. Experimental Results

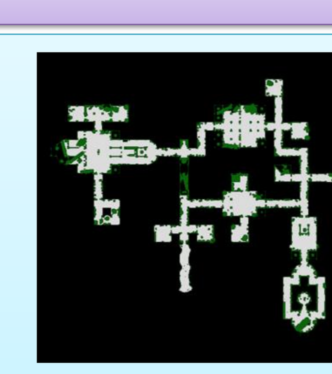
- 8x8 empty grid
- 8 agents
- $p_d = 0.1$
- pR-GCBS – Greedy version
- MC – Monte-Carlo Verifier
- DT – Deterministic Verifier



	Cost			Runtime (ms)		
	$p=0.6$	$p=0.7$	$p=0.8$	$p=0.6$	$p=0.7$	$p=0.8$
CBS	35.5	35.5	35.5	7	7	7
***p*R-GCBS**	35.7	36.0	39.7	154	218	374
***p*R-CBS (MC)**	35.5	35.6	35.9	2,811	3,505	4,823
***p*R-CBS (DT)**	35.5	35.6	35.9	9,445	27,545	62,325

- MC is faster than DT
- pR-GCBS is faster than both but not optimal
- CBS is the fastest but does not consider p

- Brc202d map
- $p_d = 0.2$
- pR-GCBS – Greedy version
- kR-GCBS – k-Robust CBS
- R – percentage of 50 simulations with no conflicts



	R			Runtime (ms)		
#Agents	**10**	**20**	**30**	**10**	**20**	**30**
CBS	0.77	0.54	0.37	268	888	2,377
***k*R-CBS (k=5)**	0.93	0.76	0.57	7,219	27,086	61,832
***k*R-CBS (k=7)**	0.96	0.85	0.73	12,014	41,913	104,449
***p*R-GCBS (p=0.6)**	0.96	0.90	0.86	8,833	36,051	70,815
***p*R-GCBS (p=0.8)**	0.99	0.94	0.92	10,141	53,091	87,206

- CBS is the fastest but causes many conflicts
- pR-GCBS causes the fewest conflicts

7. Future Work

- Adapting other solvers to find p-Robust solutions
- Integrating p-Robust solutions with execution policies