Multi-Agent Path Finding with Continuous Time Using SMT($\mathcal{LRA}$)

15th Alpine Verification Meeting (AVM'23)

Tomáš Kolárik[1,2] Stefan Ratschan[1] Pavel Surynek[2]

Czech Academy of Sciences

Czech Technical University in Prague

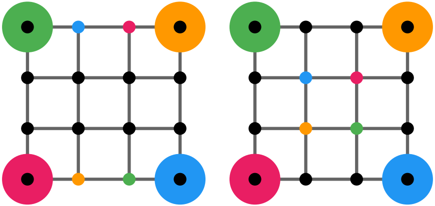
12th September, 2023

Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
    - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

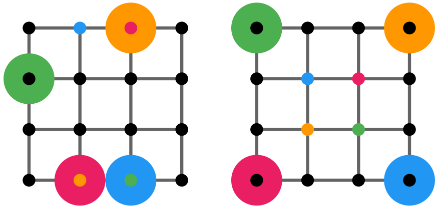


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

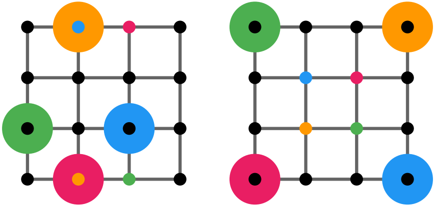


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

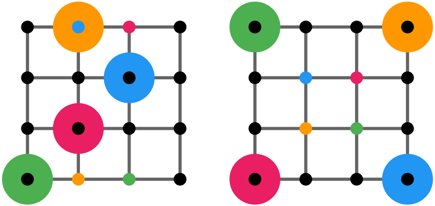


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

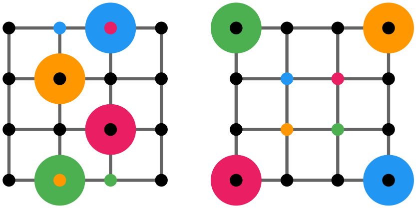


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

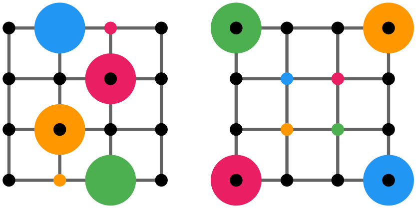


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
    - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

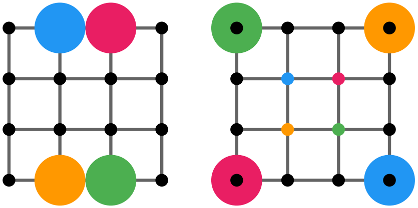


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

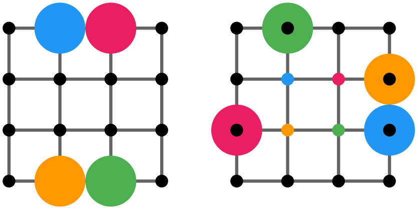


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

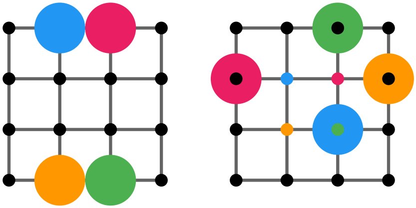


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

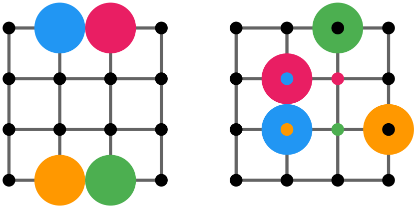


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

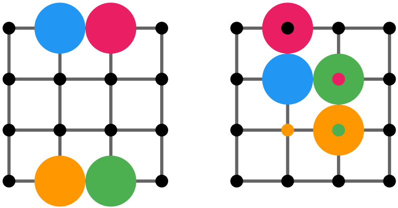


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

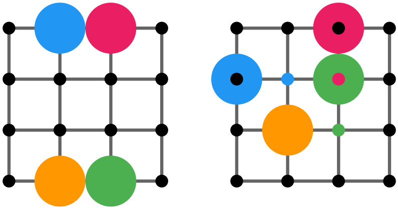


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

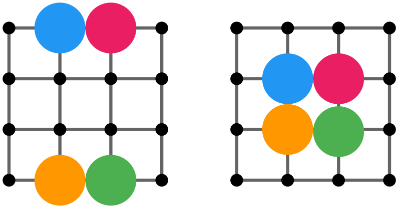


Introduction (1)

Multi-Agent Path Finding:

- given an undirected graph (without weights of edges),
- given a set of agents, each with a starting and goal vertex,
- find a **collision-free plan** that is **optimal** wrt. given cost function.
  - E.g. **makespan**—number of hops of the longest path of all agents.
- Agents are modeled as discs with the same radius.

Examples:

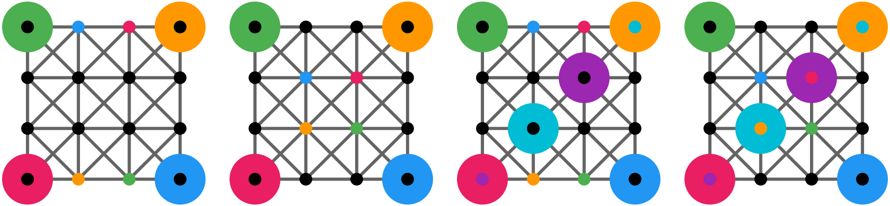


Introduction (2)

Multi-Agent Path Finding with Continuous Time:

- given an undirected graph with rational weights of edges,
- given a set of agents, each with a starting and goal vertex,
- find a collision-free plan that is optimal wrt. given cost function of weights of the used edges (not just hops).
- Agents are modeled as discs with possibly different radia, and they move with possibly different constant velocity.

Examples:

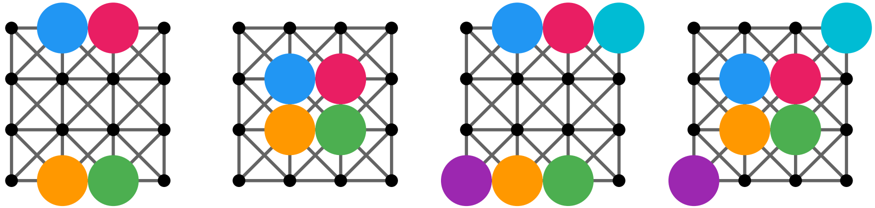


Introduction (2)

Multi-Agent Path Finding with Continuous Time:

- given an undirected graph with rational weights of edges,
- given a set of agents, each with a starting and goal vertex,
- find a collision-free plan that is optimal wrt. given cost function of weights of the used edges (not just hops).
- Agents are modeled as discs with possibly different radia, and they move with possibly different constant velocity.

Examples: demo ...



Discussion

- There may be a lot of collisions of the agents.
- State-of-the-art algorithms are usually based on searching the shortest paths of particular agents, and on resolving the conflicts separately.

→ Optimization from below.

Discussion

- There may be a lot of collisions of the agents.
- State-of-the-art algorithms are usually based on searching the shortest paths of particular agents, and on resolving the conflicts separately.

$\rightarrow$ Optimization from below.

- Finding the exact optimum can be needlessly difficult.
  - Especially in the case of continuous time.
  - The models are still far away from being precisely realistic (!)

Discussion

- There may be a lot of collisions of the agents.
- State-of-the-art algorithms are usually based on searching the shortest paths of particular agents, and on resolving the conflicts separately.

$\rightarrow$ Optimization from below.

- Finding the exact optimum can be needlessly difficult.
  - Especially in the case of continuous time.
  - The models are still far away from being precisely realistic (!)

Our approach

- We use **SAT** for efficient handling of **mutual exlusions** of agents.
- We use **SMT($\mathcal{LRA}$)** for efficient handling of **timing** constraints.
- We use **simulations** for evaluation of **conflict intervals** of agents.
- We relax the problem to finding **bounded suboptimal** plans.

Linear Real Arithmetic

Satisfiability Modulo Theories (SMT) is a general framework that is based on Boolean satisfiability (SAT).

A **theory** $\mathcal{T}$ is a parameter of the framework $\rightarrow$ SMT($\mathcal{T}$).

Linear Real Arithmetic

Satisfiability Modulo Theories (SMT) is a general framework that is based on Boolean satisfiability (SAT).

A **theory** $\mathcal{T}$ is a parameter of the framework $\rightarrow$ SMT($\mathcal{T}$).

An example of a theory:

$\mathcal{LRA}$: Linear Real Arithmetic

- Handles **linear** constraints on **rational** variables.
  - Not enough for modeling conflicts of agents (with continuous time).
- **Efficient** algorithms (e.g. solving systems of linear equations).

Linear Real Arithmetic

Satisfiability Modulo Theories (SMT) is a general framework that is based on Boolean satisfiability (SAT).

A **theory** $\mathcal{T}$ is a parameter of the framework $\rightarrow$ SMT($\mathcal{T}$).

An example of a theory:

$\mathcal{LRA}$: Linear Real Arithmetic

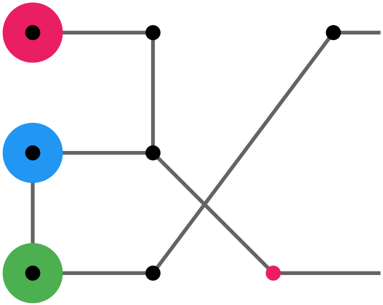
- Handles **linear** constraints on **rational** variables.
  - Not enough for modeling conflicts of agents (with continuous time).
- **Efficient** algorithms (e.g. solving systems of linear equations).

How we exploit $\mathcal{LRA}$

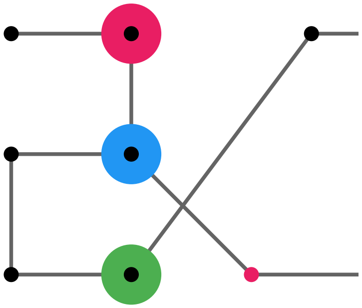
- We use $\mathcal{LRA}$ only for modeling time constraints of the agents.
- We **incrementally add conflict** interval constraints in a **lazy fashion**.

$\rightarrow$ Efficient **avoidance** of discovered conflicts.

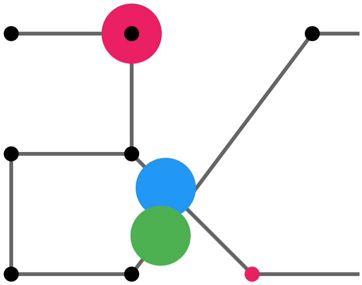
Conflict Reasoning



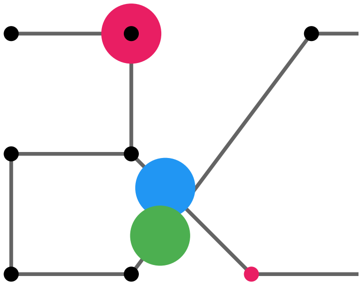
Conflict Reasoning



Conflict Reasoning



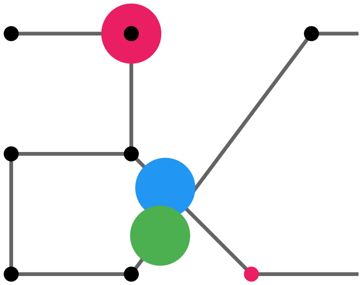
Conflict Reasoning



Conflict clause:

$$\neg\Big(\mathit{blue}.V[1] = 5 \wedge \mathit{blue}.V[2] = 8 \wedge \mathit{green}.V[1] = 7 \wedge \mathit{green}.V[2] = 2$$
$$\wedge\ \mathit{blue}.T[1] - \mathit{green}.T[1] < \mathbf{3.743} - 2$$
$$\wedge\ \mathit{green}.T[1] - \mathit{blue}.T[1] < \mathbf{3.310} - 2\Big)$$

Conflict Reasoning



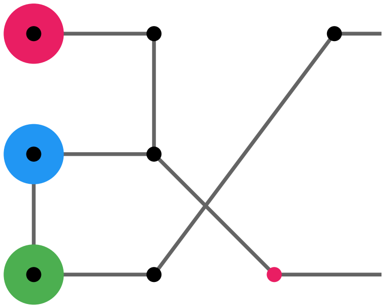
Conflict clause:

$$\mathit{blue}.V[1] \neq 5 \vee \mathit{blue}.V[2] \neq 8 \vee \mathit{green}.V[1] \neq 7 \vee \mathit{green}.V[2] \neq 2$$

$$\vee \;\; \mathit{blue}.T[1] - \mathit{green}.T[1] \geq \mathbf{1.743}$$

$$\vee \;\; \mathit{green}.T[1] - \mathit{blue}.T[1] \geq \mathbf{1.310}$$

Conflict Reasoning



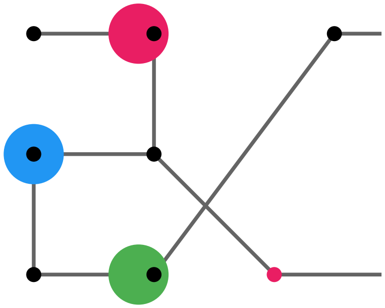
Conflict clause:

$$\mathit{blue}.V[1] \neq 5 \lor \mathit{blue}.V[2] \neq 8 \lor \mathit{green}.V[1] \neq 7 \lor \mathit{green}.V[2] \neq 2$$

$$\lor \;\; \mathit{blue}.T[1] - \mathit{green}.T[1] \geq \mathbf{1.743}$$

$$\lor \;\; \mathit{green}.T[1] - \mathit{blue}.T[1] \geq \mathbf{1.310}$$

Conflict Reasoning



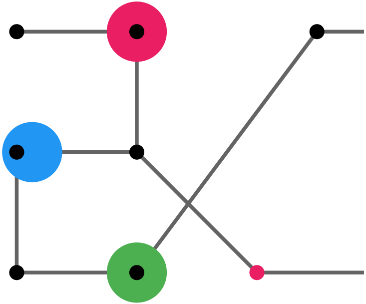
Conflict clause:

$$blue.V[1] \neq 5 \lor blue.V[2] \neq 8 \lor green.V[1] \neq 7 \lor green.V[2] \neq 2$$

$$\lor \;\; blue.T[1] - green.T[1] \geq \mathbf{1.743}$$

$$\lor \;\; green.T[1] - blue.T[1] \geq \mathbf{1.310}$$

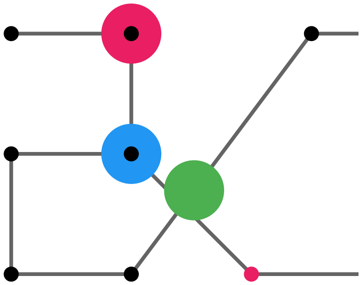
Conflict Reasoning



Conflict clause:

$$blue.V[1] \neq 5 \lor blue.V[2] \neq 8 \lor green.V[1] \neq 7 \lor green.V[2] \neq 2$$
$$\lor \; blue.T[1] - green.T[1] \geq \mathbf{1.743}$$
$$\lor \; green.T[1] - blue.T[1] \geq \mathbf{1.310}$$

Conflict Reasoning



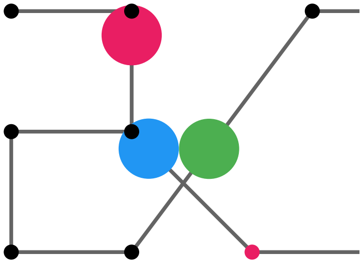
Conflict clause:

$$blue.V[1] \neq 5 \lor blue.V[2] \neq 8 \lor green.V[1] \neq 7 \lor green.V[2] \neq 2$$

$$\lor \;\; blue.T[1] - green.T[1] \geq \mathbf{1.743}$$

$$\lor \;\; green.T[1] - blue.T[1] \geq \mathbf{1.310}$$

Conflict Reasoning



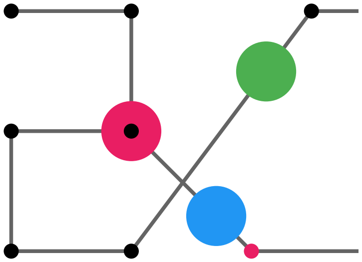
Conflict clause:

$$blue.V[1] \neq 5 \lor blue.V[2] \neq 8 \lor green.V[1] \neq 7 \lor green.V[2] \neq 2$$

$$\lor \;\; blue.T[1] - green.T[1] \geq \mathbf{1.743}$$

$$\lor \;\; green.T[1] - blue.T[1] \geq \mathbf{1.310}$$

Conflict Reasoning



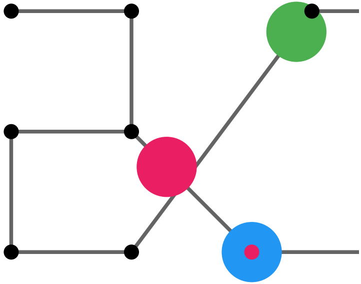
Conflict clause:

$$blue.V[1] \neq 5 \vee blue.V[2] \neq 8 \vee green.V[1] \neq 7 \vee green.V[2] \neq 2$$

$$\vee \;\; blue.T[1] - green.T[1] \geq \mathbf{1.743}$$

$$\vee \;\; green.T[1] - blue.T[1] \geq \mathbf{1.310}$$

Conflict Reasoning



Conflict clause:

$$blue.V[1] \neq 5 \vee blue.V[2] \neq 8 \vee green.V[1] \neq 7 \vee green.V[2] \neq 2$$

$$\vee \;\; blue.T[1] - green.T[1] \geq \mathbf{1.743}$$

$$\vee \;\; green.T[1] - blue.T[1] \geq \mathbf{1.310}$$

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.
- Instead: **simulation** of conflicts.
  - **Floating-point** computation.
  - It is necessary to properly **convert** to/from **rational numbers** ($\frac{1}{3}$?!).

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.
- Instead: **simulation** of conflicts.
  - **Floating-point** computation.
  - It is necessary to properly **convert** to/from **rational numbers** ($\frac{1}{3}$?!).
  - $\rightarrow$ **Overapproximation** of the time intervals.

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.
- Instead: **simulation** of conflicts.
  - **Floating-point** computation.
  - It is necessary to properly **convert** to/from **rational numbers** ($\frac{1}{3}$?!).
  - $\rightarrow$ **Overapproximation** of the time intervals.

Principle:

1. Find a plan ignoring new conflicts (obeying the already learned ones).

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.
- Instead: **simulation** of conflicts.
  - **Floating-point** computation.
  - It is necessary to properly **convert** to/from **rational numbers** ($\frac{1}{3}$?!).

$\rightarrow$ **Overapproximation** of the time intervals.

Principle:

1. Find a plan ignoring new conflicts (obeying the already learned ones).
2. Simulate motions of agents and estimate conflict intervals via binary search.

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.
- Instead: **simulation** of conflicts.
  - **Floating-point** computation.
  - It is necessary to properly **convert** to/from **rational numbers** ($\frac{1}{3}$?!).
  - $\rightarrow$ **Overapproximation** of the time intervals.

Principle:

1. Find a plan ignoring new conflicts (obeying the already learned ones).
2. Simulate motions of agents and estimate conflict intervals via binary search.
3. Overapproximate the intervals using rational numbers via simple continued fractions.

Simulation of Conflicts

- It is possible to model everything (including conflicts) in $\mathcal{NRA}$ ...
  - Very difficult even to formalize.
- Instead: **simulation** of conflicts.
  - **Floating-point** computation.
  - It is necessary to properly **convert** to/from **rational numbers** ($\frac{1}{3}$?!).
  - → **Overapproximation** of the time intervals.

Principle:

1. Find a plan ignoring new conflicts (obeying the already learned ones).
2. Simulate motions of agents and estimate conflict intervals via binary search.
3. Overapproximate the intervals using rational numbers via simple continued fractions.

→ Fast and **extensible** to more complicated and **realistic** motions.

Bounded Suboptimal Plans

- Given a user-specified coefficient $\omega > 1$,
  find a plan with a cost at most ω-times worse than the optimum.

Bounded Suboptimal Plans

- Given a user-specified coefficient $\omega > 1$,
  find a plan with a cost at most ω-times worse than the optimum.

$\rightarrow$ The result is not optimal, but is **guaranteed***
not to exceed given relative bound.

Bounded Suboptimal Plans

- Given a user-specified coefficient $\omega > 1$,
  find a plan with a cost at most ω-times worse than the optimum.

$\rightarrow$ The result is not optimal, but is **guaranteed***
not to exceed given relative bound.

- Safe-but-not-optimal plans are available soon ("better than nothing").

Bounded Suboptimal Plans

- Given a user-specified coefficient $\omega > 1$,
  find a plan with a cost at most ω-times worse than the optimum.

$\rightarrow$ The result is not optimal, but is **guaranteed***
not to exceed given relative bound.

- Safe-but-not-optimal plans are available soon ("better than nothing").

$\rightarrow$ Optimization from above.

Bounded Suboptimal Plans

- Given a user-specified coefficient $\omega > 1$,
  find a plan with a cost at most ω-times worse than the optimum.

$\rightarrow$ The result is not optimal, but is **guaranteed***
  not to exceed given relative bound.

- Safe-but-not-optimal plans are available soon ("better than nothing").

$\rightarrow$ Optimization from above.

- Giving up on precise optimality allows some flexibility of plans.

Bounded Suboptimal Plans

- Given a user-specified coefficient $\omega > 1$,
  find a plan with a cost at most ω-times worse than the optimum.

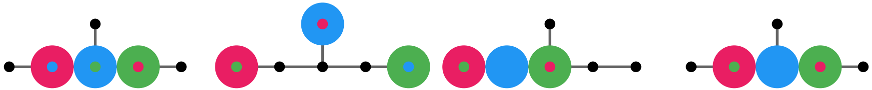
$\rightarrow$ The result is not optimal, but is **guaranteed***
  not to exceed given relative bound.

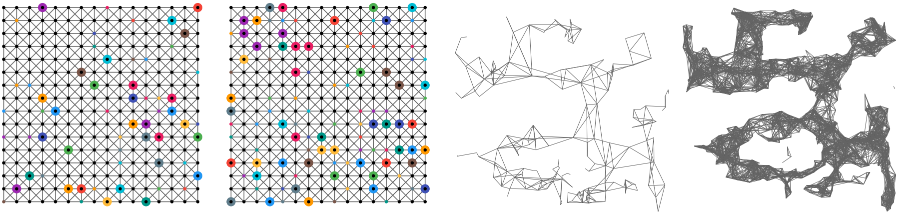
- Safe-but-not-optimal plans are available soon ("better than nothing").

$\rightarrow$ Optimization from above.

- Giving up on precise optimality allows some flexibility of plans.
  - Offline planning vs. online execution.

More Demos . . .





Future Work

- Modeling dynamic phenomena of agents.
- Allowing trajectories of agents not to be just straight lines.
  - For example roadmaps with curves.
- ...

Questions?

$\mathrm{MAPF_R}$

https://gitlab.com/Tomaqa/mapf_r

https://github.com/Tomaqa/mapf_r-visualizer

UN/SOT: UN/SAT modulo ODEs Not SOT

https://gitlab.com/Tomaqa/unsot

tomaqa@gmail.com