Some algorithmic problems related to the Autonomous Mobile Robot Orchestrator

Lou Salaün, AI Research Lab, Nokia Bell Labs, France

LIMOS Seminar, 4 April 2024

Self-introduction

- PhD thesis on resource allocation in wireless networks [Salaün2020]
- Collaboration with Pierre Bergé on the Canadian Traveller Problem [Bergé2023]
- My recent work at Nokia:
  - Radio resource allocation in future wireless networks:
    - Graph neural network [Salaün2022], deep reinforcement learning, online graph matching
    - Continuous and discrete optimization algorithms
  - Industrial robotics:
    - Collision detection and avoidance for black-box robots [Ayoubi2024]
    - Trajectory prediction: recurrent neural network, Markov model
    - Multi-agent path finding

Self-introduction

- PhD thesis on resource allocation in wireless networks [Salaün2020]
- Collaboration with Pierre Bergé on the Canadian Traveller Problem [Bergé2023]
- My recent work at Nokia:
  - Radio resource allocation in future wireless networks:
    - Graph neural network [Salaün2022], deep reinforcement learning, online graph matching
    - Continuous and discrete optimization algorithms
  - Industrial robotics:
    - Collision detection and avoidance for black-box robots [Ayoubi2024]
    - Trajectory prediction: recurrent neural network, Markov model
    - Multi-agent path finding

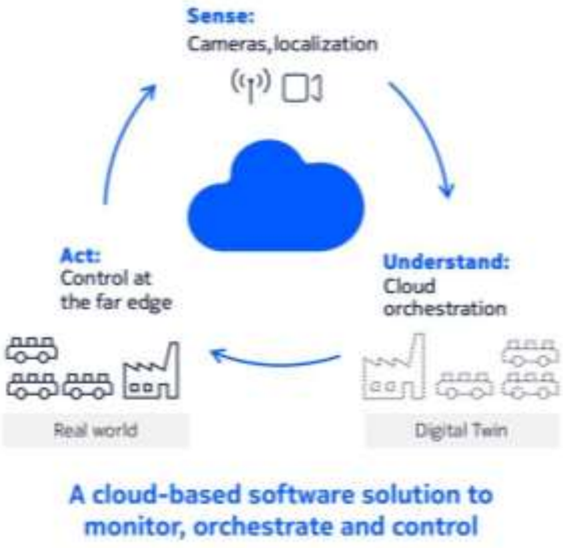
Scope of this presentation

Autonomous Mobile Robot Orchestrator (AMRO)

Objective: sense, orchestrate and control

Environment: robotics factory with multi-vendor mobile cognitive robots

- Agents controlled by AMRO:
  - AMR: Autonomous Mobile Robots (free-space mobile robot)
- Agents non-controlled by AMRO:
  - AGV: Automated Guided Vehicle (line-following robot)
  - AMR
  - Humans, forklifts, etc..

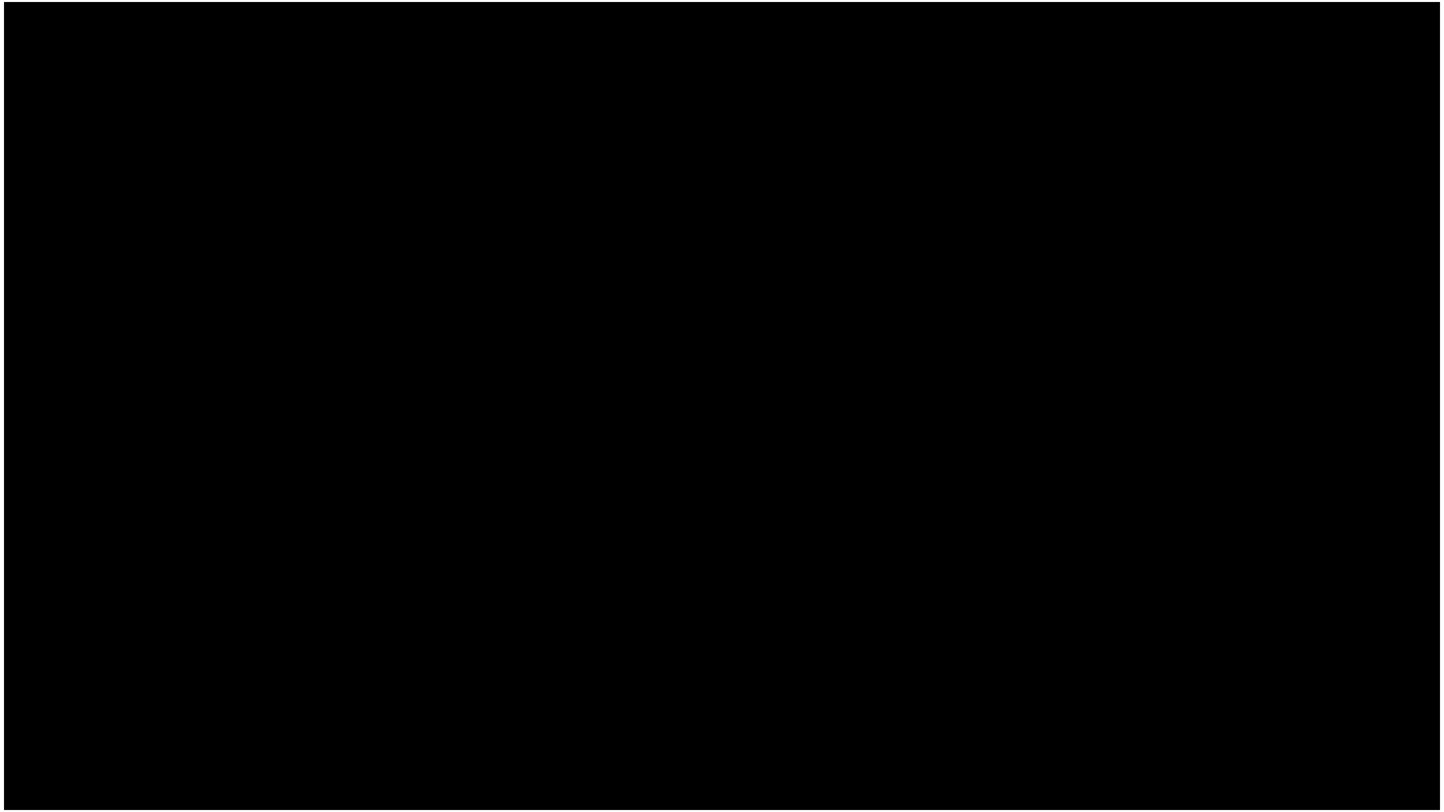


Source:

[Cloud-enhanced cognitive robotics, Nokia Bell Labs blog post, 21 November 2023]

[Commanding robots from the edge, Nokia Bell Labs blog post, 13 October 2022]

Autonomous Mobile Robot Orchestrator (AMRO)



Outline

- Collision Detection and Avoidance for Black Box Multi-Robot Navigation (CODAK) → Local planning
- Multi-Agent Path Finding (MAPF) → Global planning
- AGV Trajectory Prediction → Dynamic obstacle avoidance in global planning

CODAK

Introduction

CODAK: Collision Detection and Avoidance for Black Box Multi-Robot Navigation

Assumptions:

- Fleet of commercial industrial robots from different vendors
- Heterogeneous
- Black-box
- Shared communication channel
- Simple interface with the following actions:

CODAK

Introduction

CODAK: Collision Detection and Avoidance for Black Box Multi-Robot Navigation

Assumptions:

- Fleet of commercial industrial robots from different vendors
- Heterogeneous
- Black-box
- Shared communication channel
- Simple interface with the following actions:

1. **Set a goal**

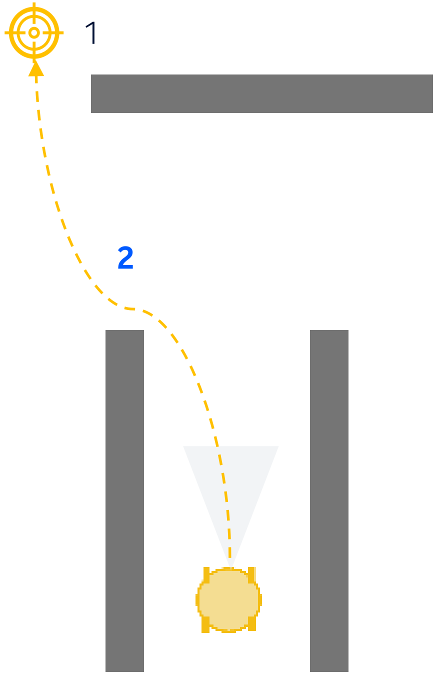
CODAK

Introduction

CODAK: Collision Detection and Avoidance for Black Box Multi-Robot Navigation

Assumptions:

- Fleet of commercial industrial robots from different vendors
- Heterogeneous
- Black-box
- Shared communication channel
- Simple interface with the following actions:
  1. Set a goal
  2. **Monitor the robot movement and plan**



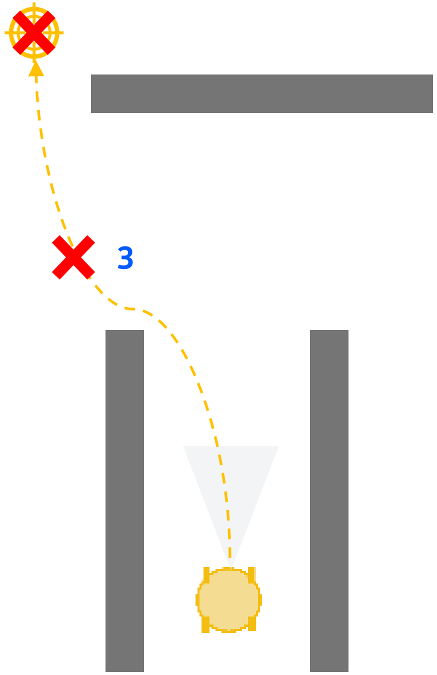
CODAK

Introduction

CODAK: Collision Detection and Avoidance for Black Box Multi-Robot Navigation

Assumptions:

- Fleet of commercial industrial robots from different vendors
- Heterogeneous
- Black-box
- Shared communication channel
- Simple interface with the following actions:
  1. Set a goal
  2. Monitor the robot movement and plan
  3. **Cancel a goal**



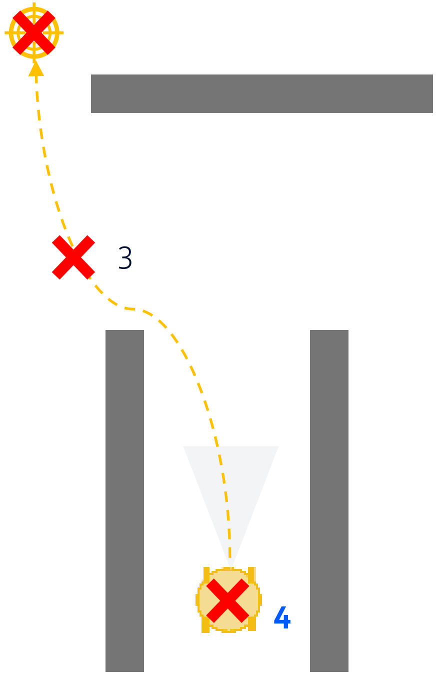
CODAK

Introduction

CODAK: Collision Detection and Avoidance for Black Box Multi-Robot Navigation

Assumptions:

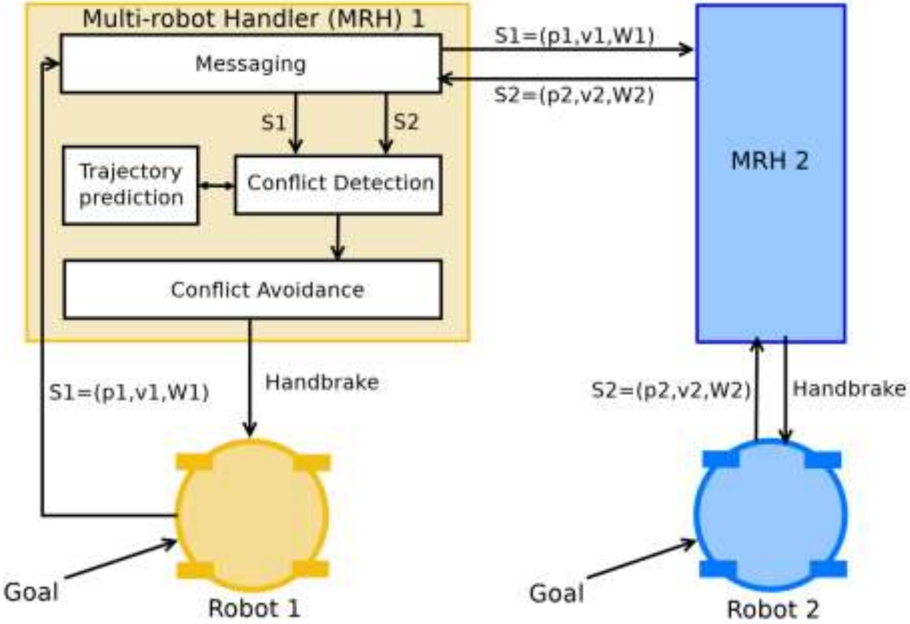
- Fleet of commercial industrial robots from different vendors
- Heterogeneous
- Black-box
- Shared communication channel
- Simple interface with the following actions:
  1. Set a goal
  2. Monitor the robot movement and plan
  3. Cancel a goal
  4. **Pull the handbrake**



CODAK

Software system overview

- Navigation stack is hidden
- Shared information:
  - **p**: position
  - **v**: velocity
  - **W**: plan (sequence of waypoints)
- Pre-assigned priority order



CODAK

Method overview

Each robot:

- Moves autonomously
- Communicates its intent (plan)
- Listen to others' plans
- Predict trajectories to estimate collision probability
- If collision is detected with a higher priority robot, run:

```
Algorithm 1: collision avoidance
Input: other_plans /* plans from higher priority robots
stop at a safe distance;
cancel goal and plan;
conflict ← True;
while conflict do
    wait;
    plan ← compute new plan (avoiding collision zone if interface allows);
    conflict ← predict collision between plan and other_plans;
end
Resume goal;
```

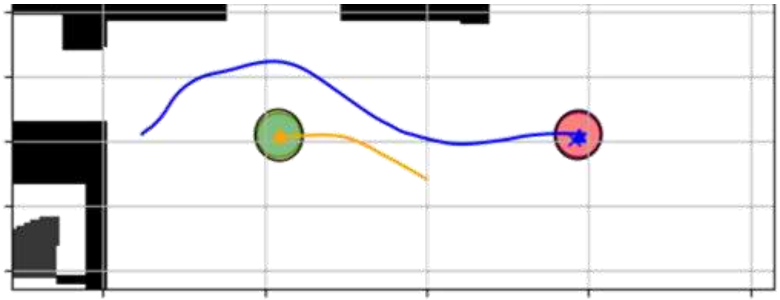


Fig. Robot's plan at $t = 0\ s$

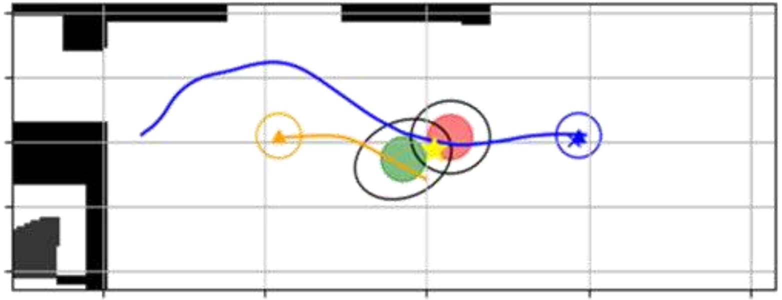


Fig. Collision predicted at $t = 4\ s$

NOKIA

CODAK

Recurrent neural network trajectory prediction

RNN

Prediction:

- Position and velocity distributions over time

Input:

- Current state: position, orientation and velocity
- Robot intent: plan
- Local/global view (optional)

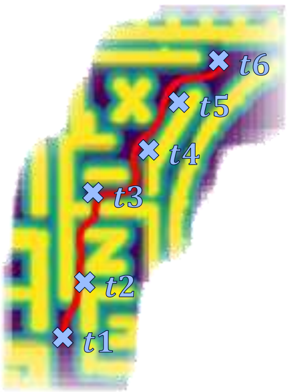


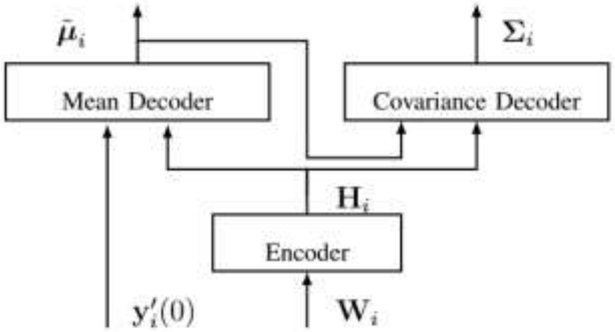
Fig. Trajectory prediction using RNN

CODAK

Recurrent neural network trajectory prediction

Fig. RNN structure for robot i

Output: Sequence of predicted states $y_i'(1) \cdots y_i'(N)$.
Each state $y_i'(k) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_i(k), \boldsymbol{\Sigma}_i(k))$.



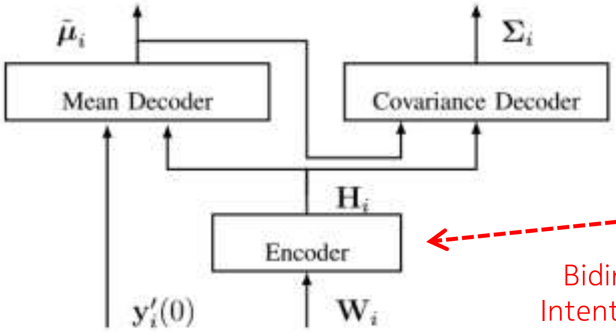
Input: Initial state (x, y, θ, v, w, t) | Sequence of waypoints (plan) $w_i(0) \cdots w_i(N)$

CODAK

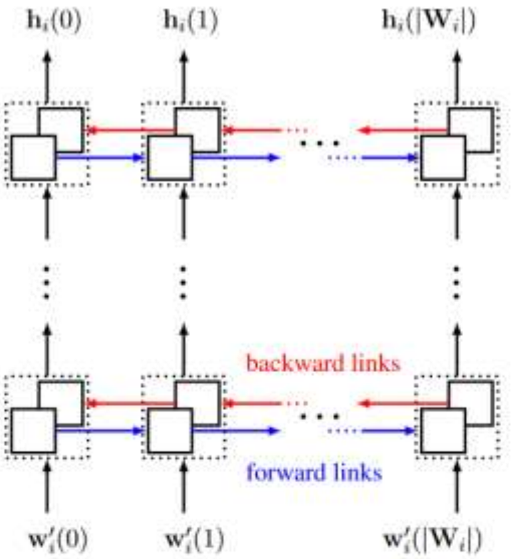
Recurrent neural network trajectory prediction

Fig. RNN structure for robot i

Output: Sequence of predicted states $y_i'(1) \cdots y_i'(N)$.
Each state $y_i'(k) \sim \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_i(k), \boldsymbol{\Sigma}_i(k)\right)$.



Bidirectional RNN Encoder
Intent $\boldsymbol{W}_i \longrightarrow$ internal repr. $\boldsymbol{H}_i$



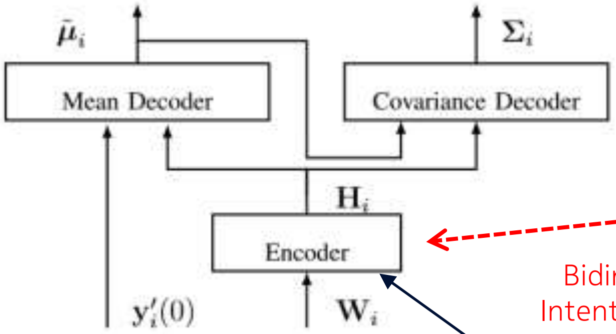
Input: Initial state (x, y, θ, v, w, t) | Sequence of waypoints (plan) $w_i(0) \cdots w_i(N)$

CODAK

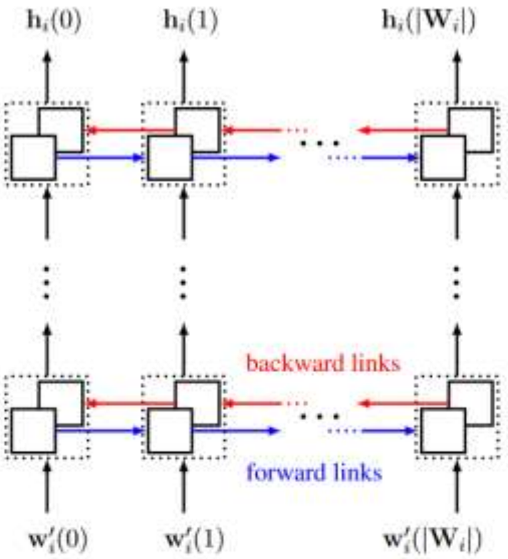
Recurrent neural network trajectory prediction

Fig. RNN structure for robot i

Output: Sequence of predicted states $y_i'(1) \cdots y_i'(N)$.
Each state $y_i'(k) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_i(k), \boldsymbol{\Sigma}_i(k))$.



Bidirectional RNN Encoder
Intent $\boldsymbol{W}_i \longrightarrow$ internal repr. $\boldsymbol{H}_i$



Input: Initial state (x, y, θ, v, w, t)

Sequence of waypoints (plan) $w_i(0) \cdots w_i(N)$

Optional "contextual" input: static map, costmap

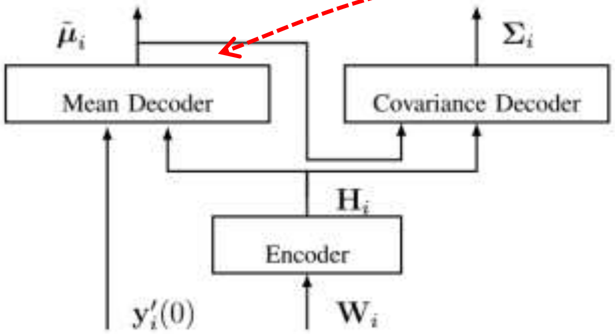
CODAK

Recurrent neural network trajectory prediction

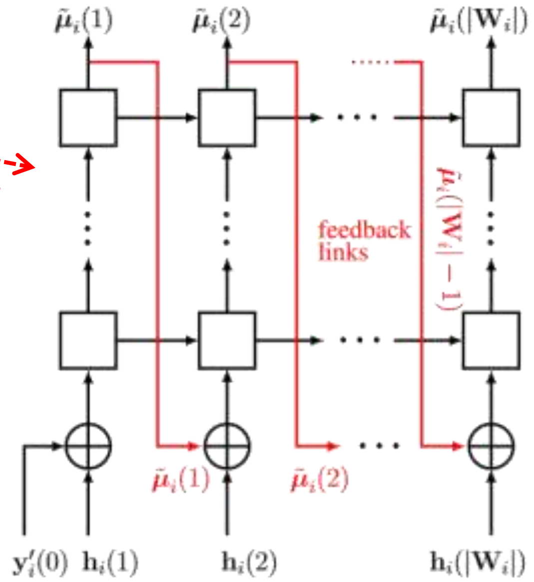
Fig. RNN structure for robot i

Previous state
$y_i'(t-1), h_i(t)$
$\Downarrow$
Next state $y_i'(t)$

Output: Sequence of predicted states $y_i'(1) \cdots y_i'(N)$.
Each state $y_i'(k) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_i(k), \boldsymbol{\Sigma}_i(k))$.



Mean Decoder with feedbacks
"predict average positions"



Input: Initial state (x, y, θ, v, w, t)

Sequence of waypoints (plan) $w_i(0) \cdots w_i(N)$

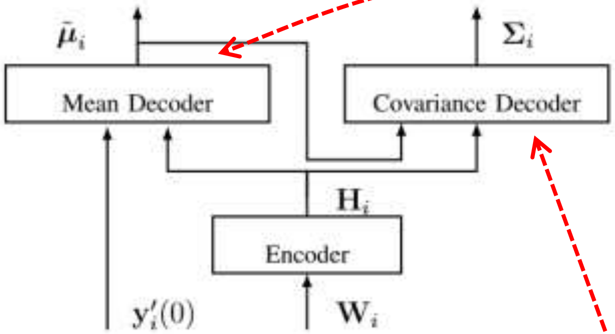
CODAK

Recurrent neural network trajectory prediction

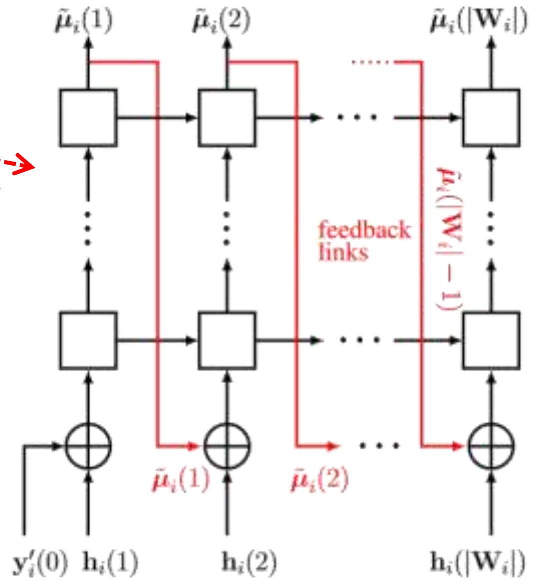
Fig. RNN structure for robot i

Previous state $y_i'(t-1), h_i(t)$ ⇓ Next state $y_i'(t)$

Output: Sequence of predicted states $y_i'(1) \cdots y_i'(N)$.
Each state $y_i'(k) \sim \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_i(k), \boldsymbol{\Sigma}_i(k)\right)$.



Mean Decoder with feedbacks
"predict average positions"



Input: Initial state (x, y, θ, v, w, t)

Sequence of waypoints (plan) $w_i(0) \cdots w_i(N)$

Covariance Decoder with feedbacks
"estimate prediction uncertainty"

CODAK

Recurrent neural network trajectory prediction

- The prediction should be invariant by:
  - Translation: states and waypoints are encoded as displacements, e.g., $(dx, dy, \theta, v, w, dt)$
  - Rotation: augment training data with random rotations

- Mean-covariance training:
  - First phase:
    - Learn to predict the average positions “point-prediction”
    - Train the encoder and mean decoder with 75% of the training data using mean square error loss
  - Second phase:
    - Learn to estimate the uncertainty “covariance prediction”
    - Train the covariance decoder with 25% of the training data using Gaussian negative log likelihood loss

CODAK

RNN performance

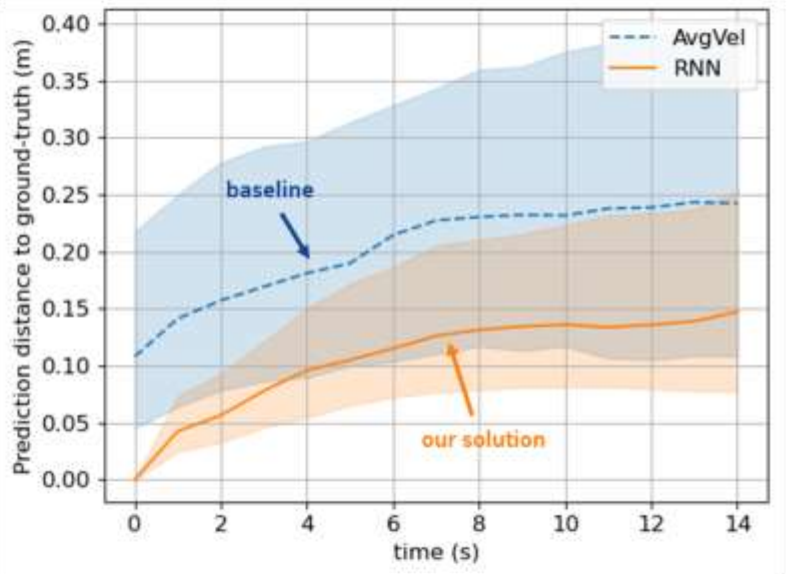


Figure. Prediction error over time of RNN and AvgVel (baseline)

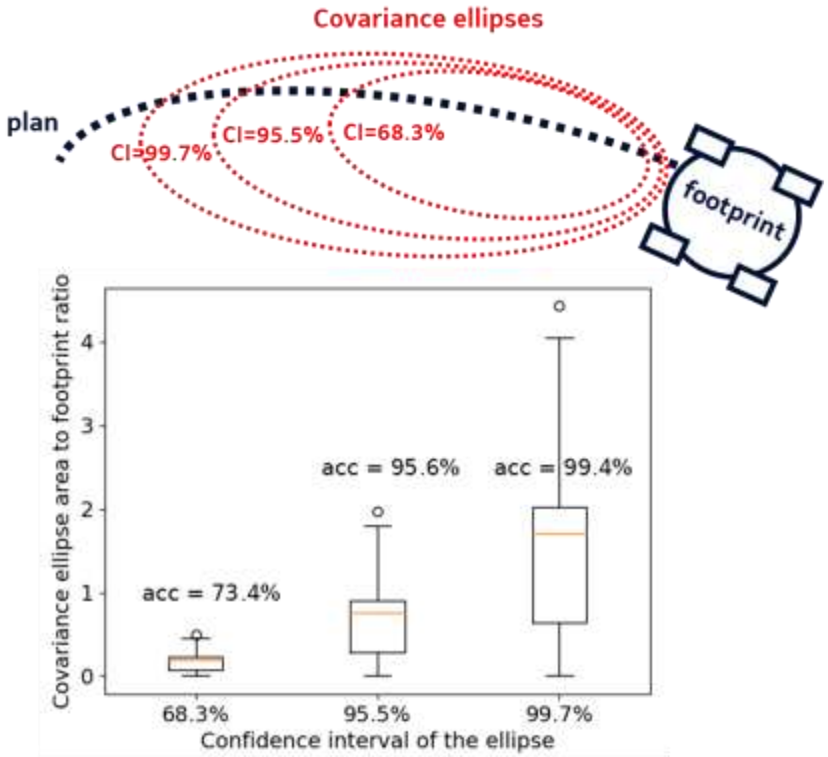
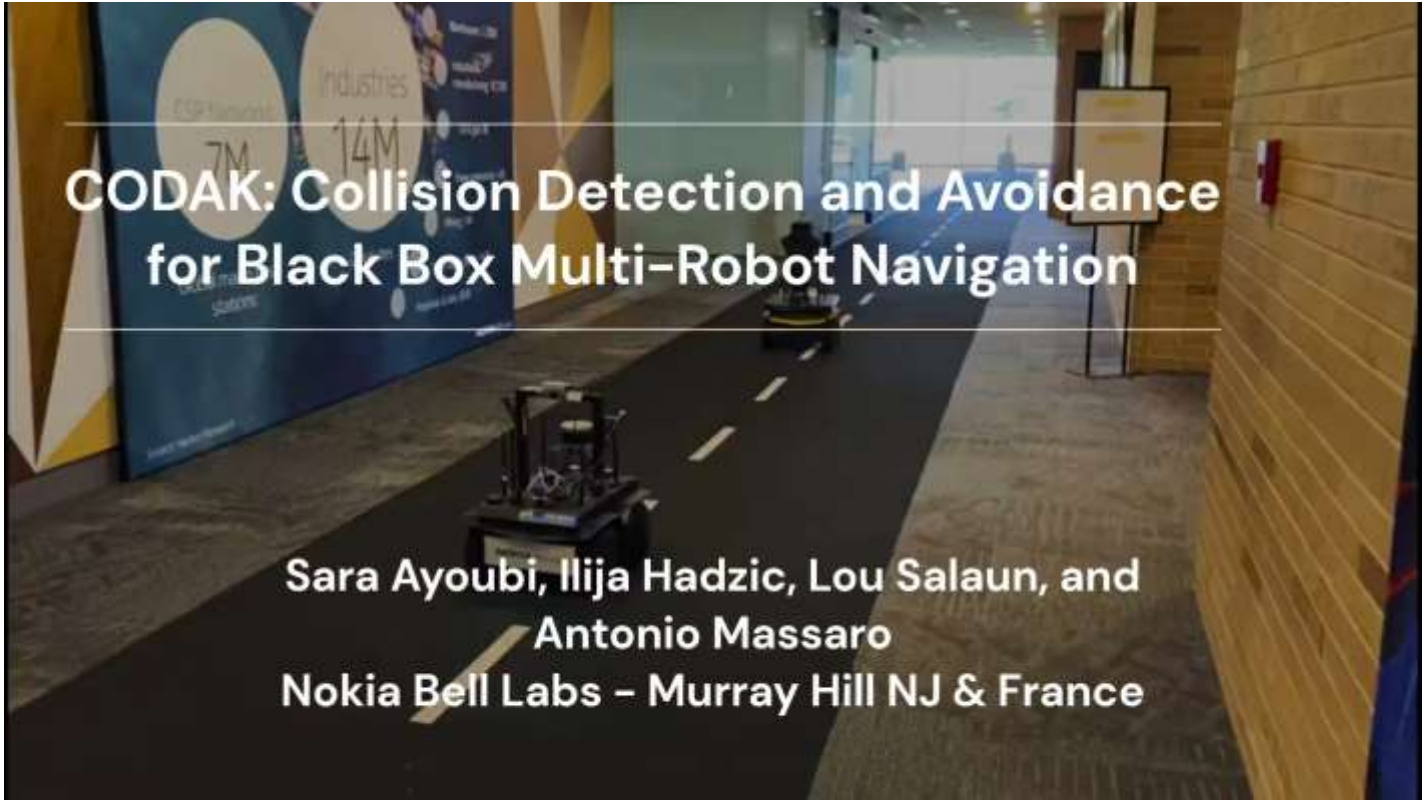


Figure. RNN covariance ellipse area vs. confidence interval. Prediction accuracy is shown on top of each boxplot

CODAK

Experiments



CODAK

Conclusion

- Avoid collision without access to the internal navigation stack
- Makespan comparable to the white-box solution NH-ORCA
- Our implementation is distributed (can also be centralized)
- Can find collision-free path but cannot avoid deadlocks

- Future work: deadlock resolution
  - Requires a free-space global planner ←
  - Robust to localization/sensor uncertainties
  - As few communication rounds as possible (latency)

NOKIA

Multi-Agent Path Finding

Problem definition

MAPF consists in finding the shortest collision-free path for each agent in a graph

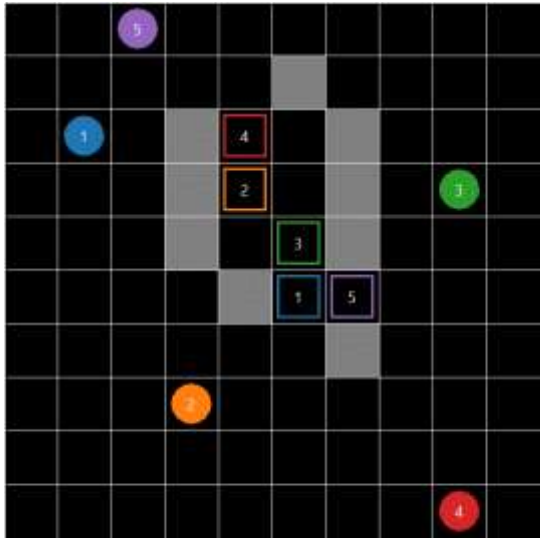


Fig. An example on a grid

Multi-Agent Path Finding

Problem definition

- $\pi_i(t)$: position (vertex) of robot i at time t
- g_i: goal position (vertex) of robot i
- Constraints:
  - Move along an edge: $(\pi_i(t), \pi_i(t+1)) \in E$
  - Vertex conflict: if $i \neq j$, then $\pi_i(t) \neq \pi_j(t)$
  - Swapping conflict: if $i \neq j$, we cannot have $\pi_i(t) = \pi_j(t+1)$ and $\pi_i(t+1) = \pi_j(t)$

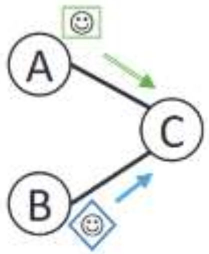


Fig. Vertex conflict

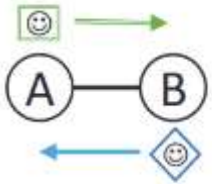


Fig. Swapping conflict
Source of the figures [Stern2019]

Multi-Agent Path Finding

Problem definition

- Objective:
  - MAPF: after some time T, for all robot i, $\pi_i(T) = g_i$
  - MAPD (multi-agent pickup and delivery): for all robot i, there is a timestep T_i, $\pi_i(T_i) = g_i$
- Metrics:
  - Makespan: T
  - Sum-of-costs: $\sum_i T_i$, where T_i is the earliest arrival time of robot i

Multi-Agent Path Finding
Algorithms

A table made a few years ago:

Algorithm	Real-time Heuristic Search	Decentralized	Complete	Optimal	Approximate
ODrM* [8]			x	x	x
PRIMAL [22]	x	x			
WHCA* [19]	x	x			
CO-WHCA* [20]	x	x			
ILP [9]			x	x	
Push and Swap [16]			x		
Push and Rotate [17]			x		
EPEA* [10]			x	x	
ICTS [11]			x	x	
Extended ICTS [23]			x	x	
MA-CBS [12]			x	x	
ICBS [13]			x	x	
ECBS [15]					x
BMAA* [21]	x	x			

Multi-Agent Path Finding

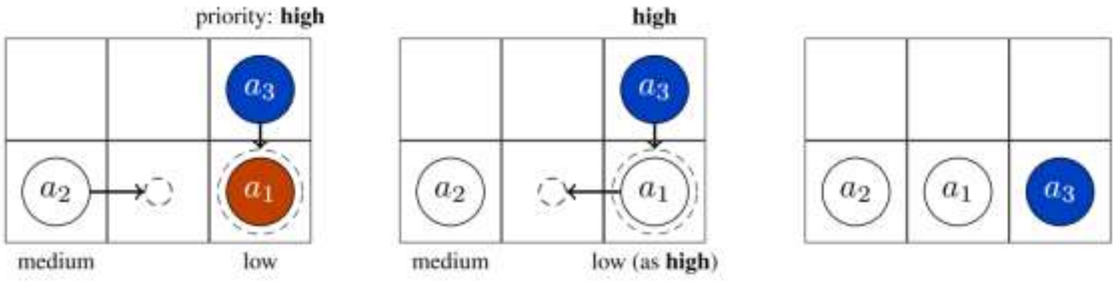
Priority Inheritance with Backtracking (PIBT)

- Introduced by [Okumura2022]
- Low complexity heuristic
- Can easily scale to hundreds of agents
- Complete for MAPD problem if graph is biconnected
- We extend PIBT to free-space scenario
- How it works?
  - Each agent follows a shortest path (e.g., Dijkstra, A*)
  - In case of conflict:
    - Priority inheritance
    - Backtracking

Multi-Agent Path Finding

Priority Inheritance

Fig. Example of priority inheritance (source [Okumura2022])

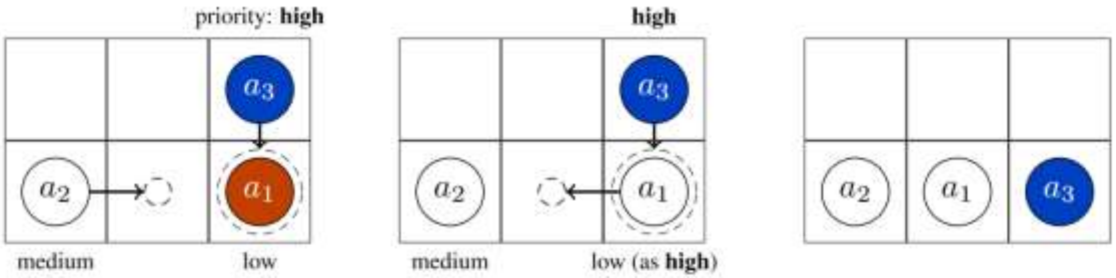


(a) Stuck agent (b) Priority inheritance (c) One timestep later

Multi-Agent Path Finding

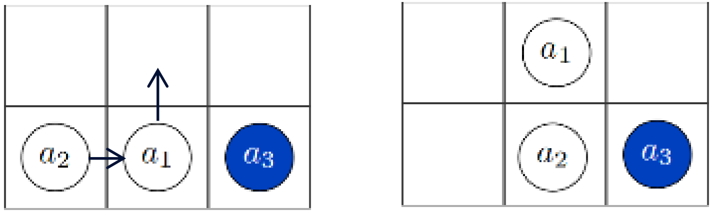
Priority Inheritance

Fig. Example of priority inheritance (source [Okumura2022])



(a) Stuck agent (b) Priority inheritance (c) One timestep later

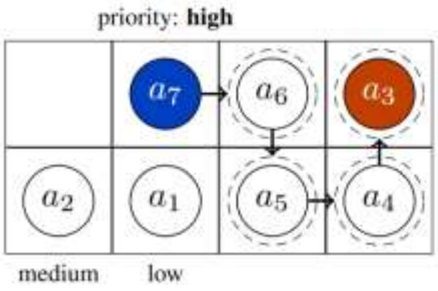
Next steps…



Multi-Agent Path Finding

Backtracking

Fig. Example of backtracking (source [Okumura2022])

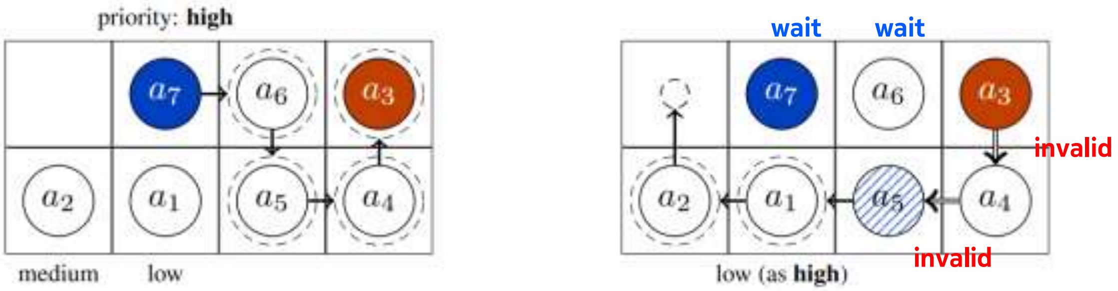


(a) Priority inheritance

Multi-Agent Path Finding

Backtracking

Fig. Example of backtracking (source [Okumura2022])



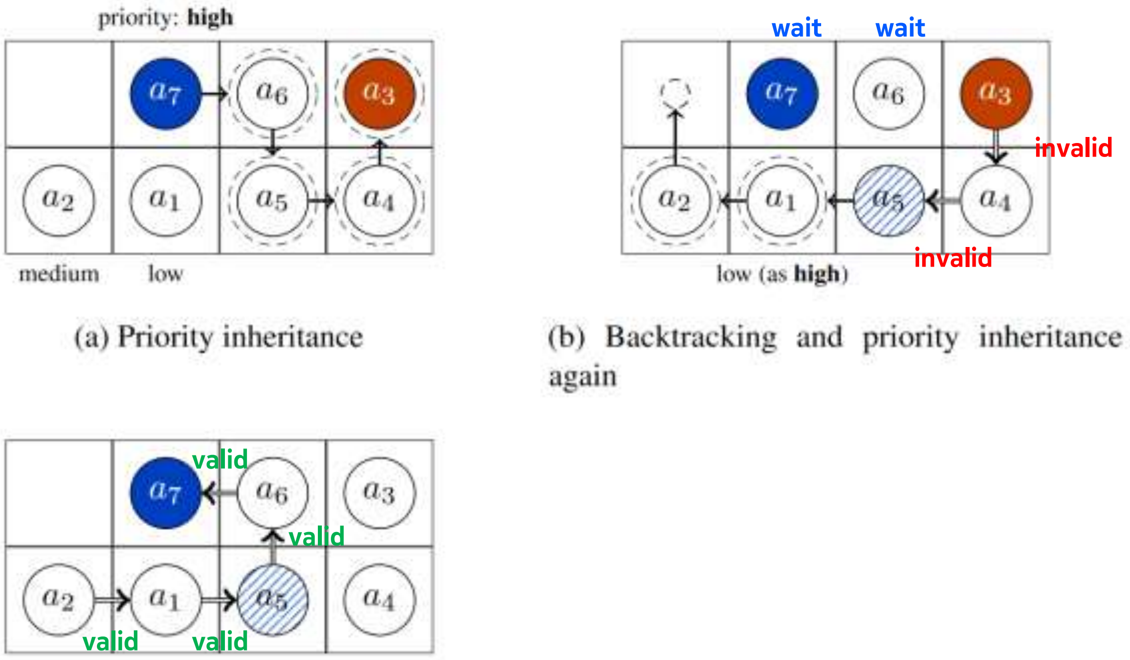
(a) Priority inheritance

(b) Backtracking and priority inheritance again

Multi-Agent Path Finding

Backtracking

Fig. Example of backtracking (source [Okumura2022])



(a) Priority inheritance

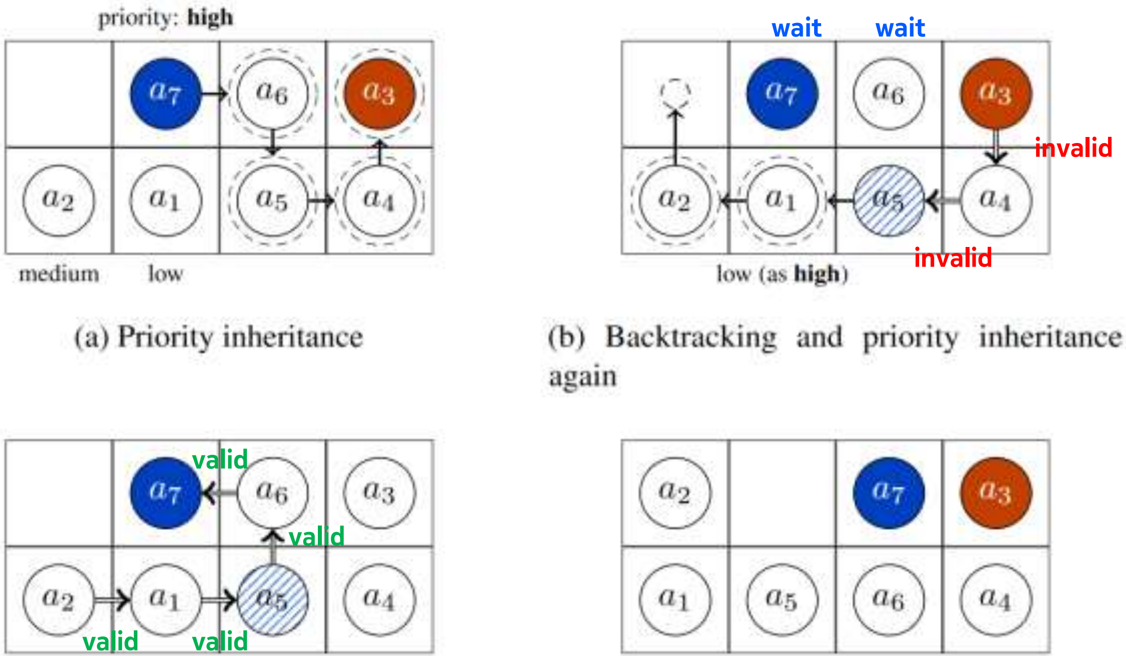
(b) Backtracking and priority inheritance again

(c) Backtracking

Multi-Agent Path Finding

Backtracking

Fig. Example of backtracking (source [Okumura2022])



(a) Priority inheritance

(b) Backtracking and priority inheritance again

(c) Backtracking

(d) one timestep later

Free-Space PIBT

Problem definition

New rules:

- Agents can be of any size, can cover more than one node
- Agents can move on different graphs
- Conflict is given by a distance function, e.g., Euclidean: $d\left(\pi_i(t), \pi_j(t)\right) < r_i + r_j$

Free-Space PIBT

Priority inheritance requires multiple steps

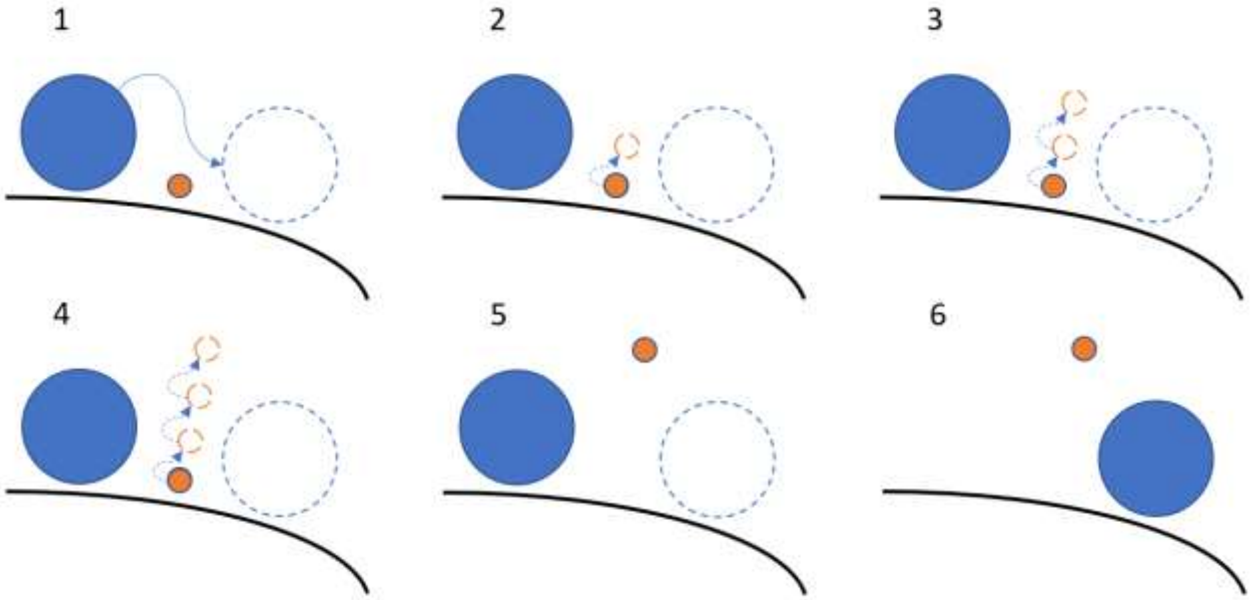
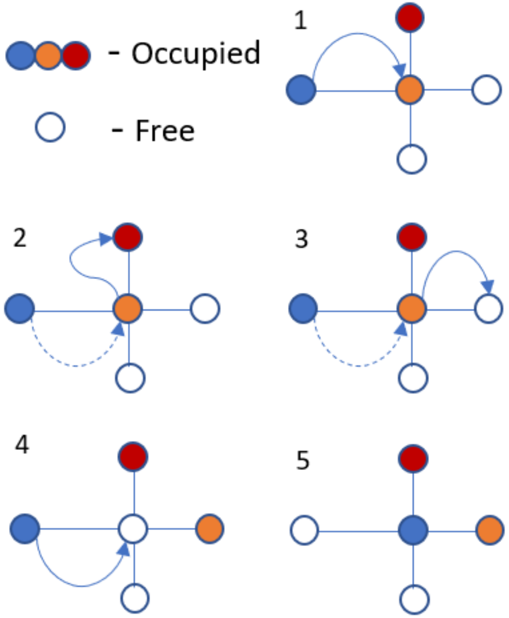


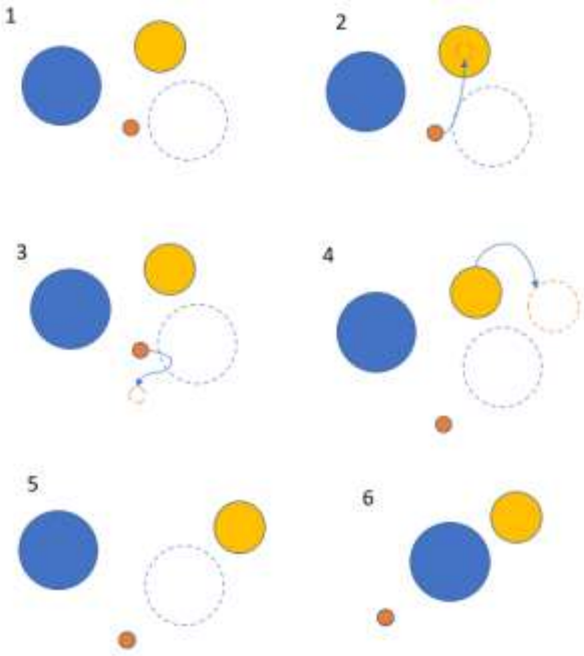
Fig. free-space priority inheritance requires multiple steps

Free-Space PIBT

Priority inherited by multiple agents



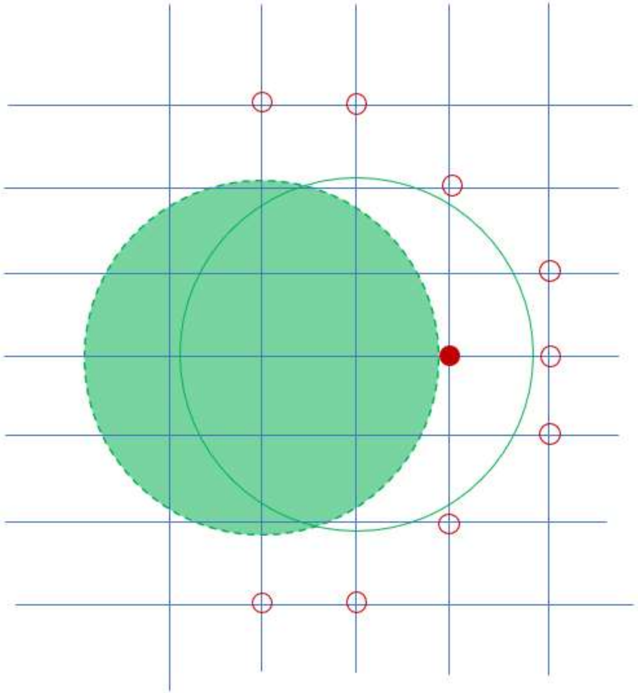
[Fig. classical priority inheritance]



[Fig. free-space priority inheritance]

Free-Space PIBT

Backtracking search space is larger



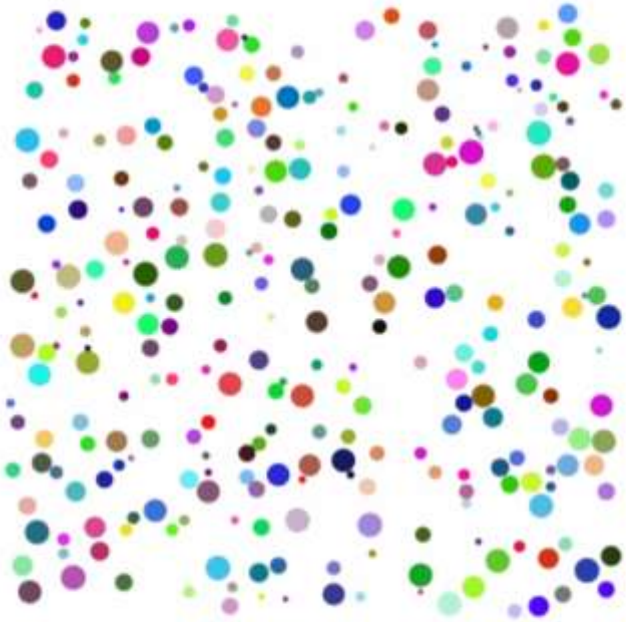
[Fig. free-space backtracking has more potential positions]

Free-Space PIBT

Preliminary solution

- 2022 Internship subject: C++ implementation
  - Round agents (Euclidean distance)
  - Square agents (Manhattan distance)

- We made arbitrary choices to handle the above 3 issues:
  - Priority inheritance requires multiple steps ⟶ not an issue
  - Pass the priority to multiple agents in arbitrary order
  - We limit the max number of steps during backtracking
  - The recursion depth can also be limited
  - Impact on completeness?



Free-Space PIBT

Open questions

- How to correctly handle these issues?
- Proof of completeness
- How to efficiently reduce the complexity in practice?
- Arbitrary shapes?

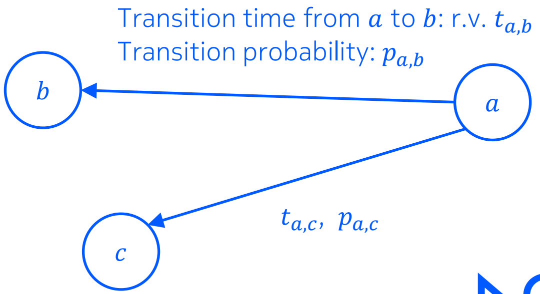
AGV Trajectory Prediction

Heterogeneous continuous-time random walks (HCTRW)

- Automated Guided Vehicle (AGV):
  - Line-following robot
  - Black-box (do not communicate with the orchestrator)
  - Noisy Localization from cameras and radio

- HCTRW [Grebenkov2018], Markov model with:
  - Transition probabilities
  - Transition time is a continuous random variable

- The prediction is used in MAPF solvers to avoid conflict AGVs:
  - Consider AGVs as dynamic obstacles (space-time reservation)
  - Compatible with most state-of-the-art algorithms
  - Improve planning quality (faster mission completion)



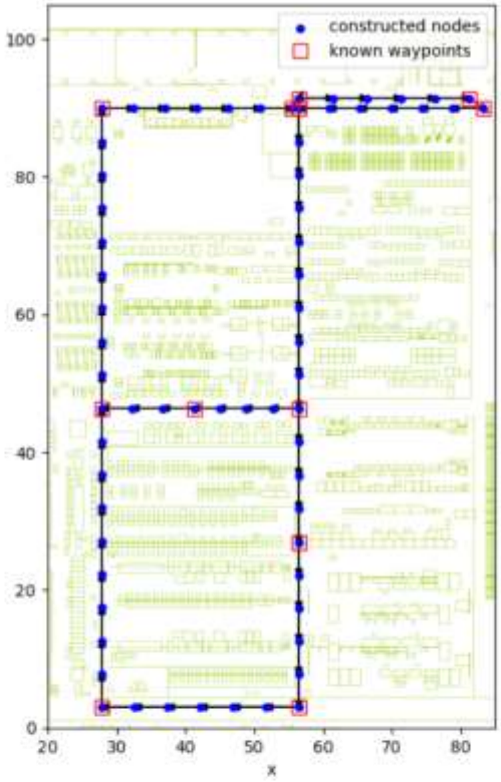
AGV Trajectory Prediction

HCTRW model learning pipeline

- Data:
  - Positions and orientations of the AGVs over time
  - Covariance (uncertainty of the localization)
  - Noisy, localization can be wrong even with low-covariance

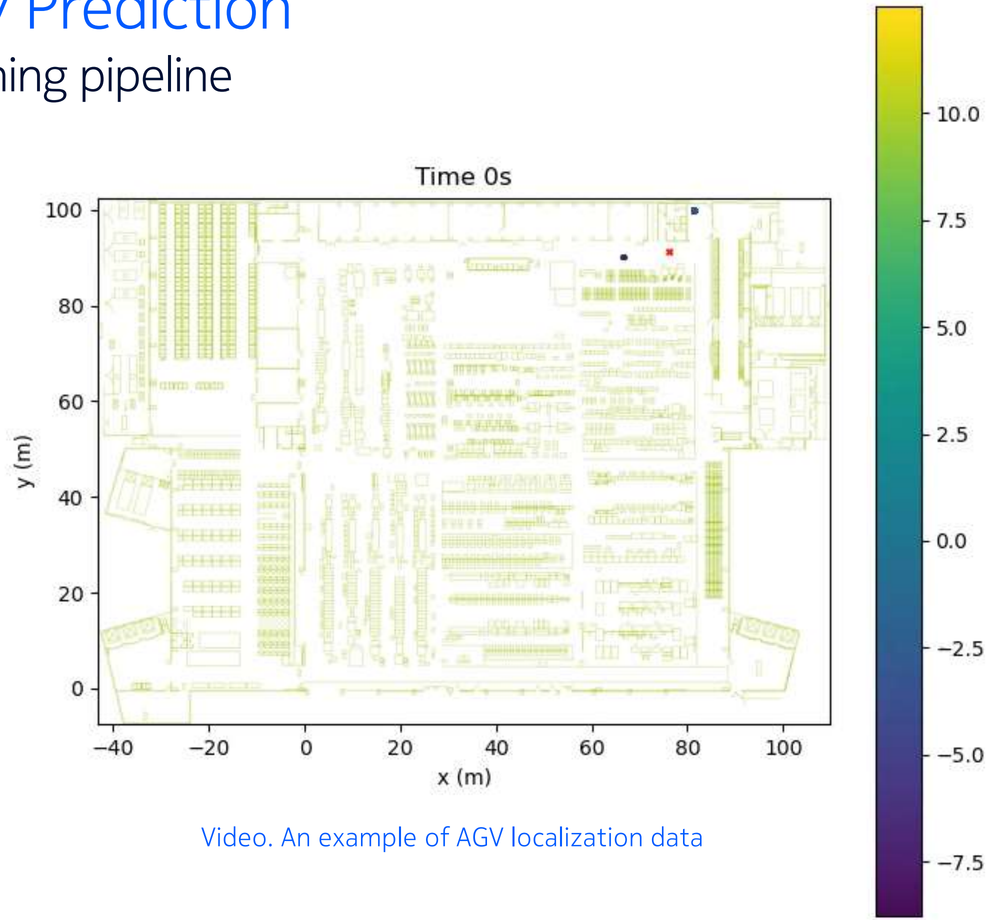
- Graph construction: based on expert knowledge
- Data preprocessing:
  - Filter out high-uncertainty data and large time gaps
  - Compute the most likely sequence of states given the noisy observations (the AGV maximum velocity is known)
- Fit:
  - Fit transition probabilities and times to common distributions (e.g., expon, powerlaw, lognorm, uniform, etc.)

Fig. Constructed graph



AGV Trajectory Prediction

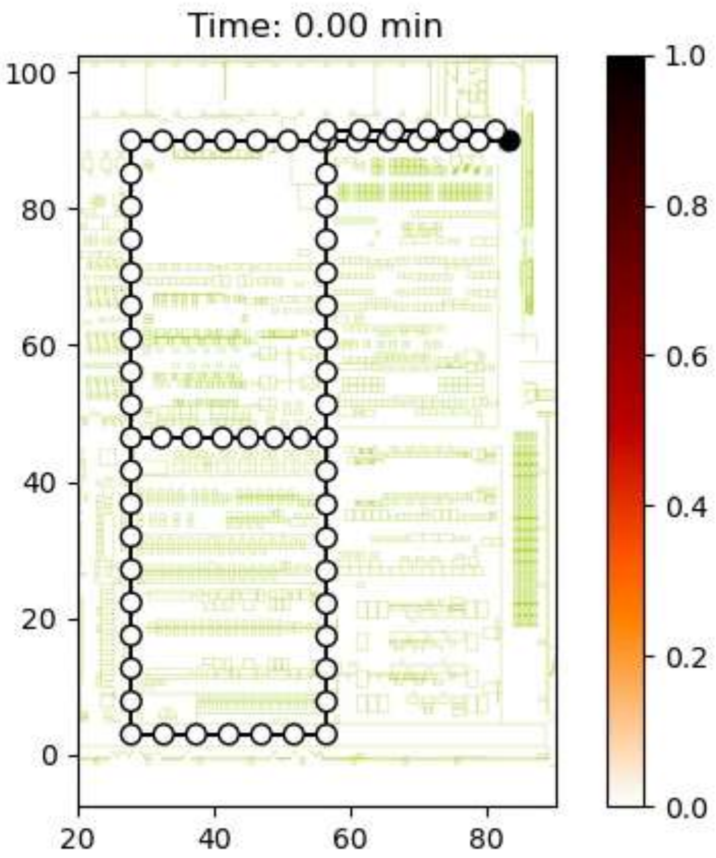
HCTRW model learning pipeline



Video. An example of AGV localization data

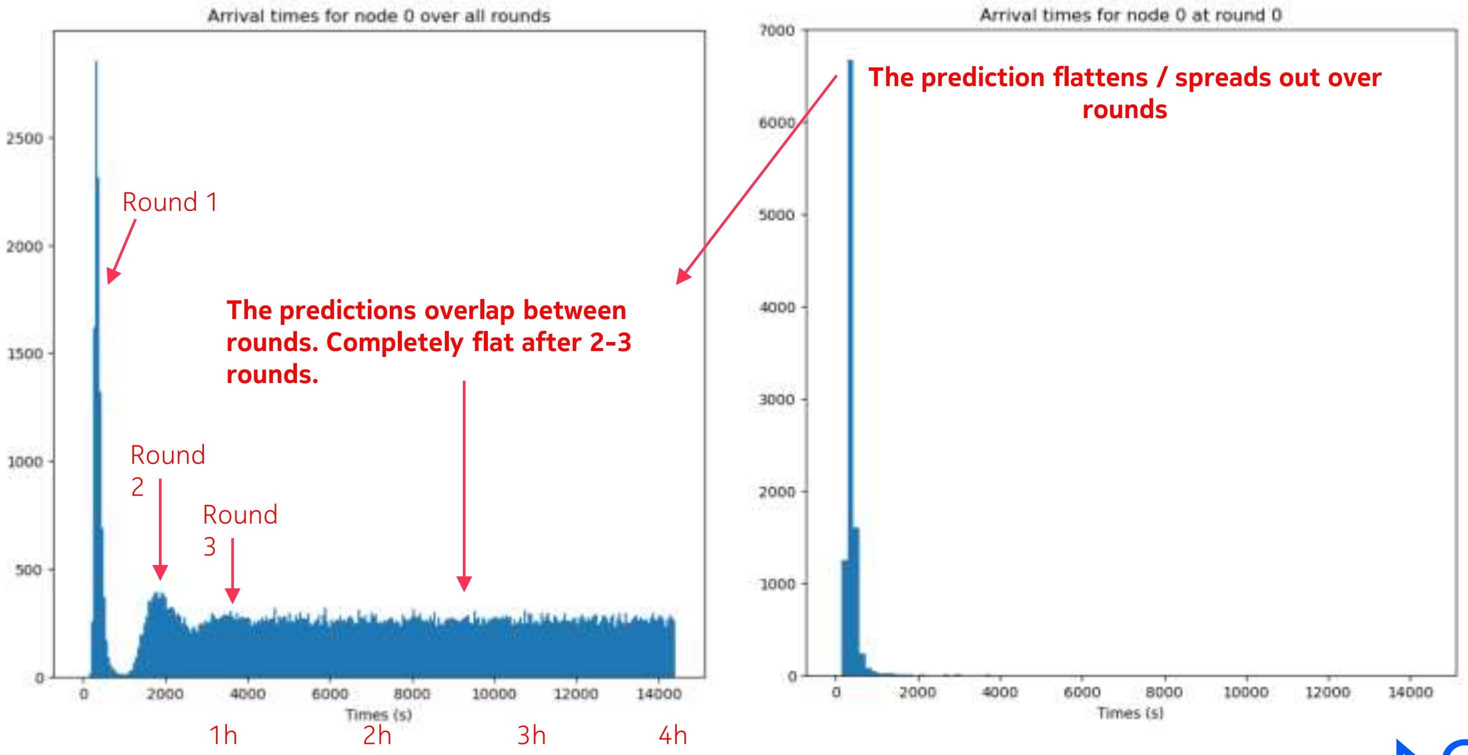
AGV Trajectory Prediction
HCTRW prediction

- Take as input the initial state of the robot
- Closed-form calculation:
  - Laplace domain
  - Only tractable for some distributions, e.g., exponential
- Monte Carlo sampling



AGV Trajectory Prediction

HCTRW prediction



Conclusion

- Feel free to ask questions!

- Credits:
  - CODAK: Sara Ayoubi, Ilija Hadzic, Antonio Massaro
  - LA-PIBT: Sara Ayoubi, Vladimir Kondratyev
  - AGV prediction: Manuel Deneu, Antonio Massaro, Liubov Tupikina

References

[Ayoubi2024] Sara Ayoubi, Ilija Hadzic, Lou Salaün and Antonio Massaro, “Collision detection and avoidance for black box multi-robot navigation”, *ICRA*, 2024.

[Bergé2023] Pierre Bergé and Lou Salaün, “The influence of maximum (s, t)-cuts on the competitiveness of deterministic strategies for the Canadian Traveller Problem”, *Theoretical Computer Science*, vol. 941, p. 221-240, 2023.

[Grebenkov2018] Denis S. Grebenkov and Liubov Tupikina, “Heterogeneous continuous-time random walks”, *Physical Review E*, vol. 97, no 1, 2018.

[Okumura2022] Keisuke Okumura, Manao Machida, Xavier Défago, et al., “Priority inheritance with backtracking for iterative multi-agent path finding”, *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[Salaün2020] Lou Salaün, “Resource allocation and optimization for the non-orthogonal multiple access”, PhD thesis, Institut polytechnique de Paris, 2020.

[Salaün2022] Lou Salaün, Hong Yang, Shashwat Mishra and Chung Shue Chen, “A GNN Approach for Cell-Free Massive MIMO”, *IEEE Globecom*, 2022.

[Stern2019] Roni Stern, et al., “Multi-agent pathfinding: Definitions, variants, and benchmarks”, in *Proceedings of the International Symposium on Combinatorial Search*, 2019.

NOKIA