Decentralized Multi-Agent Strategy Synthesis via Exchange of Least-Limiting Advisers

Georg Friedrich Schuppe and Jana Tumova

KTH Royal Institute of Technology

{schuppe, tumova}@kth.se

Motivation

Heterogeneous robots in shared environment might occasionally be required to collaborate, even though they were originally not deployed to operate as a team.

Example:

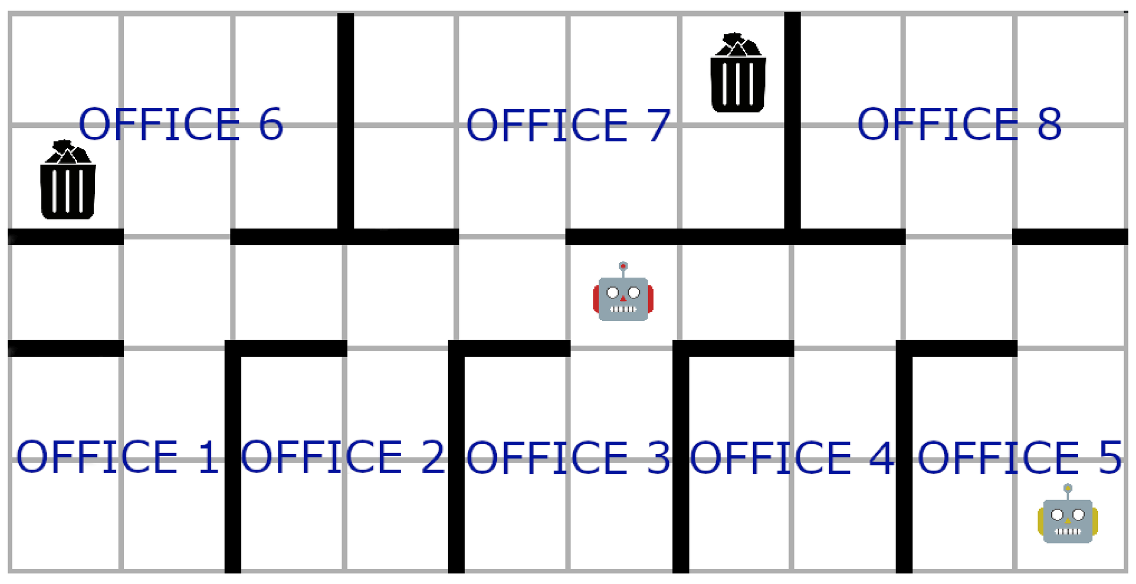


Figure 1: A partitioned office-like environment. A state of a robot is determined by its orientation and the cell it occupies. In each state, a robot can choose to stay, move forward, or turn 90°.

- Bin-emptying robots are tasked to empty k specific bins in the offices
- Cleaning robots need to clean detected spillages in certain offices and guarantee that none of the bin-emptying robots enter the affected office in order to prevent further damage

How do we express their interdependent tasks? How do we ensure that the tasks are accomplished?

Problem Formulation

- Each agent modelled an MDP $\mathcal{M}_i$

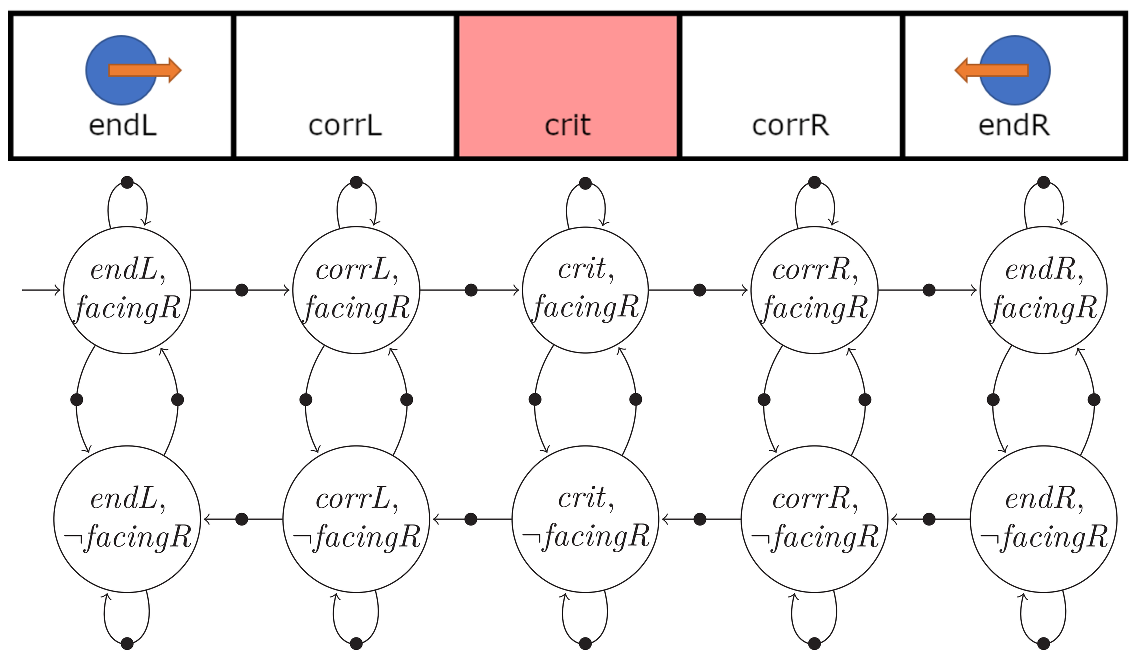


Figure 2: A small example of an MPD modeling the left robot in the corridor illustrated above. A state of a robot is determined by its orientation and the cell it occupies, the actions are to move, turn around, or stay.

- Each agent given an LTL$_f$ specification ϕ_i

$\longrightarrow$ $$\phi_i = \bigwedge_{\forall k \in \{1,\dots,\ell\}} F\, bin_{i,k},$$

$\longrightarrow$ $$\phi_j = F\, off_{j,o_j} \wedge G(\bigwedge_{\forall i \in \{1,\dots,n\}} \neg off_{i,o_j})$$

- Develop an efficient procedure to synthesize reactive strategies for all $\mathcal{M}_i$ such that all ϕ_i are satisfied, i.e. avoiding to construct a centralized Product MDP

Contributions and Approach

- A *reactive synthesis*-based approach for multi-agent high-level task planning
- A novel, decentralized approach via exchange of least-limiting advisers
- Demonstrating the scalability of the approach on selected use-cases

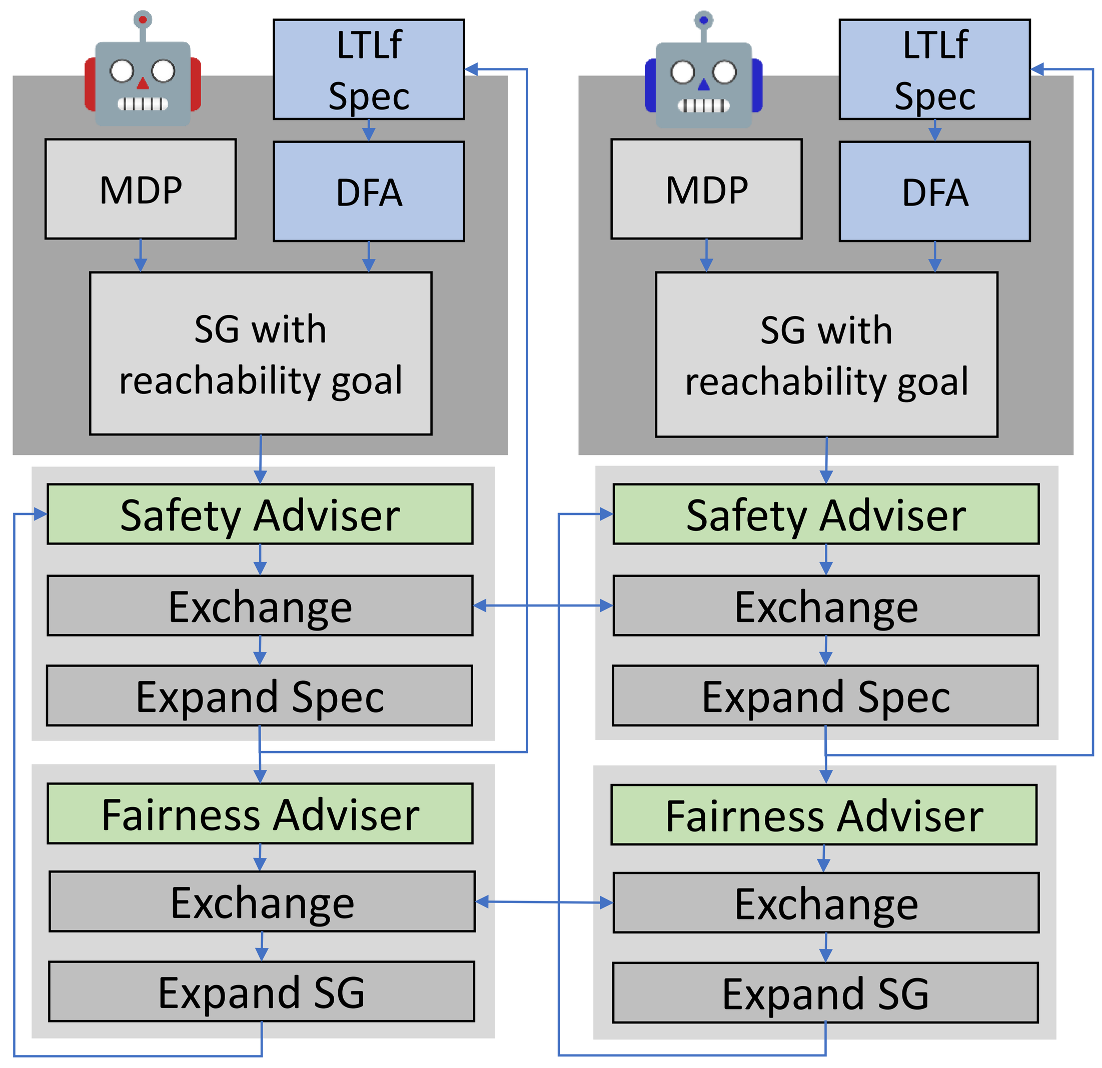


Figure 3: Schema of the approach for two agents. Each agents constructs their stochastic game locally and computes minimal, necessary assumptions on the behaviour of other agents. In an iterative process, agents incorporate advice from each other and compute additional advisers, if necessary.

Safety Advisers

Definition 1 (Minimality)**.** *A safety assumption E_s is minimal if $|E_s'| \leq |E_s|$ for all safety assumptions $E_s' \in E_2$. The unique, minimal safety assumption can be computed as*

$$E_s = \{(s, s') \in E_2 \mid s \in \langle\langle 1,2 \rangle\rangle \psi \text{ and } s' \notin \langle\langle 1,2 \rangle\rangle \psi\},$$

The assumption E_s cannot be directly communicated as an adviser to the other agents. Instead, we communicate the advice in the form of an adviser:

Definition 2 (Safety Adviser)**.** *A* safety adviser *is a set of tuples:*

$$SafeAdv = \{(pre, \sigma) \mid pre \in AP_i, \sigma \in \widehat{\Sigma}_i\}$$

Given that agent i satisfies pre, other agents should not satisfy σ in their next state.

Safety Advisers are implemented by expanding the specification formula of affected agents:

$$\phi_{(pre,\sigma),i} = G(pre \rightarrow \neg X\, proj_i(\sigma)),$$

We can incorporate all advisers from all agents into the specification of the agent i through conjunction:

$$\phi_{s,i} = \bigwedge_{\forall (pre,\sigma) \in SafeAdv_j, j \in N} \phi_{(pre,\sigma),i}$$

Fairness Advisers

Since computing a minimal fairness assumption is NP-hard, we compute *locally minimal* fairness assumptions instead. Similarly to safety assumptions, we transform fairness assumptions into Fairness Advisers:

Definition 3 (Fairness Adviser)**.** *A* fairness adviser *is a set of tuples:*

$$FairAdv = \{(pre, \sigma) \mid pre \in AP_i, \sigma \in \widehat{\Sigma}_i\}$$

Given that agent i satisfies pre, other agents should satisfy σ with non-zero probability in their next state.

Fairness Advisers are implemented through explicit modification of the stochastic games.

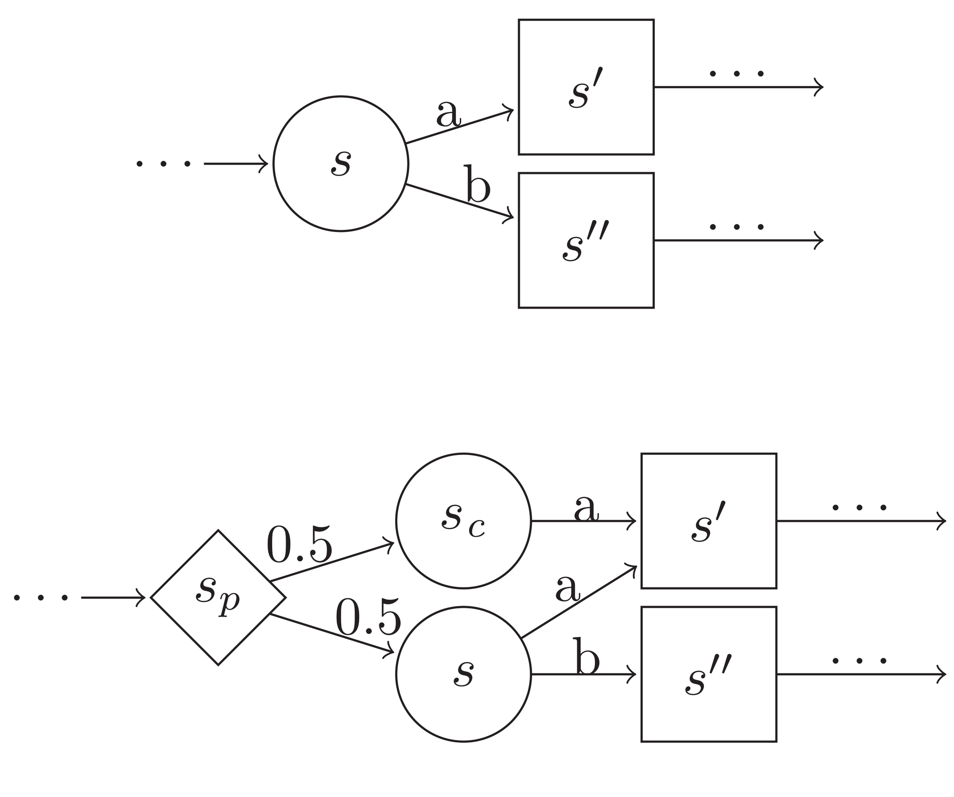
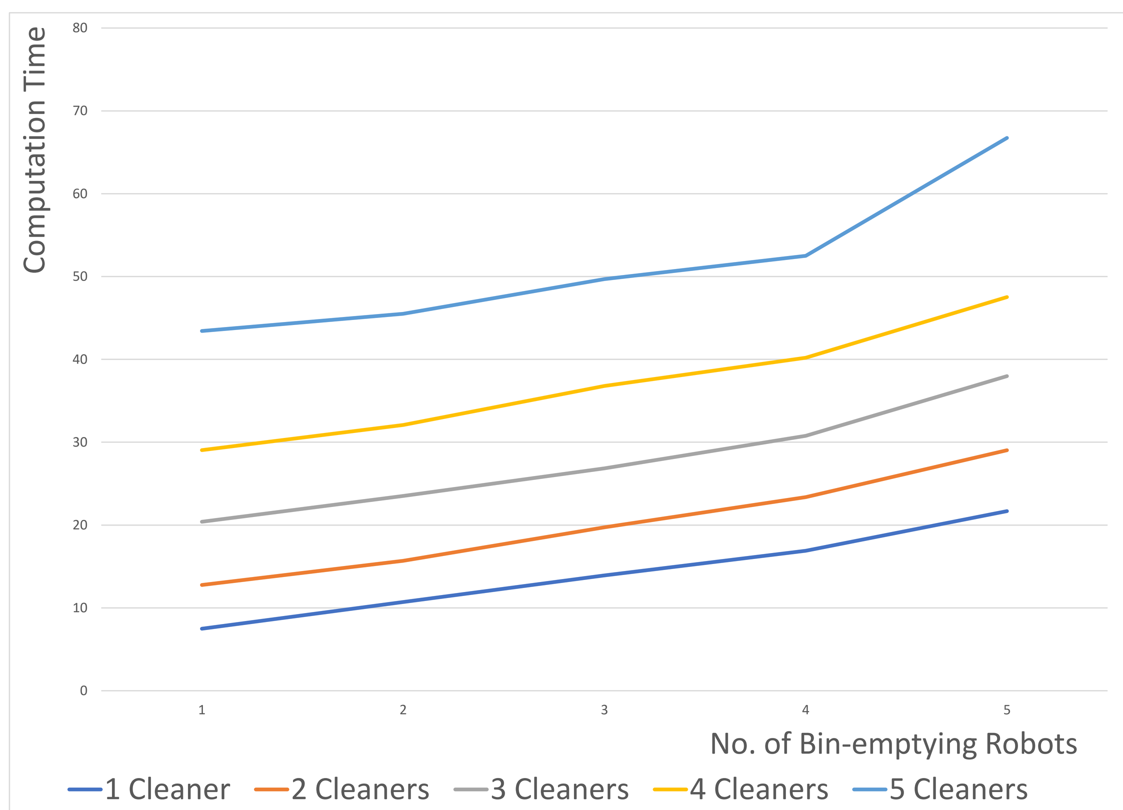


Figure 4: Enforcing fairness on (s, a, s') by prepending a probabilistic state.

Results

- The solution is sound, but conservative
- Conservativeness stems from the information gap between agents and the implementation of fairness advisers
- When dependencies between agent specifications are low, the computation time depending on the number of agents behave almost linearly
- Conveying advisers to humans poses an interesting line of future research



Acknowledgement

This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation and the Swedish Research Council (VR) (project no. 2017-05102).