DALI: A Decentralized, Agent-Based Approach for Adaptive Manufacturing using Local Information

Sebastian Schmid
Friedrich-Alexander-Universität Erlangen-Nürnberg
Nuremberg, Germany
sebastian.schmid@fau.de

Andreas Harth
Friedrich-Alexander-Universität Erlangen-Nürnberg
Fraunhofer Institute for Integrated Circuits IIS
Nuremberg, Germany
andreas.harth@iis.fraunhofer.de

ABSTRACT

We study how decentralized multi-agent systems adapt on local data in dynamic environments. In this extension of our earlier work, we build on our original setting of a driverless transportation system for a manufacturing scenario with Autonomous Guided Vehicles (AGVs) to include not only disturbances but also perception and communication errors that hinder the information sharing of agents. We analyze how well the decentralized MAS can overcome and correct these errors, where corrections come from, how the errors impact the performance, and consider the communication efficiency among the MAS population.

CCS CONCEPTS

• **Computing methodologies → Self-organization**; • **Computer systems organization → Reliability**.

KEYWORDS

Dynamic Environment, Autonomous Guided Vehicles, Multi-Agent System, Self-Adaption, Intralogistics

ACM Reference Format:
Sebastian Schmid and Andreas Harth. 2024. DALI: A Decentralized, Agent-Based Approach for Adaptive Manufacturing using Local Information. In ,. ACM, New York, NY, USA, 13 pages.

1 INTRODUCTION

The crucial challenge of modern-day industry applications is to ensure that production is high-performing but still highly dynamic. Downtime shall be reduced to a minimum to maintain output, so control systems are challenged to decide between adaptation and task fulfillment. Manufacturing demands adaptability to external, pre-given production goals that comprise individual, small batch sizes and a dynamic environment that changes according to unforeseen influences, i.e. simple machine outage to pandemics [11, 40].

Especially in dynamic environments with unforeseen disturbances, centralized controlled systems (CC) optimize easily by deciding on global data, kept in one place, to adapt on the spot [14]. Still, CCs are known to have disadvantages: storage and scalability to process data with increasing load are limited, fixed links and communication channels introduce vulnerabilities, and the single, responsible node imposes the threat of a single point of failure [14, 17]. While a CC approach is efficient where data is clean and ubiquitous, manufacturing systems tend to get more complex and struggle to provide with quality and availability of such data [3, 22].

On the other hand, decentralized and distributed multi-agent systems (MAS) approaches are proposed for dynamic scenarios, as they promise robustness due to graceful degradation and scalability of problem solution by modifying the number of members [14, 16]. Existing MAS approaches already entered the dynamic domain and try to implement decentralized control, e.g. based on auctions [12], forecasting [25], negotiation [19], ant-colony intelligence [43], reinforcement learning [46] or hybrid approaches like local coordination and plan merging [1], but often centralized components remain as part of the control and extensive data for decisions is required. The question about the efficiency of agents that use exclusively local information in dynamic manufacturing environments is still open: to what extent can decentralized agents cope with the demand of dynamic environments in industry?

In our original work for the Symposium on Applied Computing 2024 [36], we studied decentralized agents that act exclusively on local information, perception, and communication (DALI) to answer the question. We motivated our cause with an example of a shop floor with a driverless transportation system (DTS), based on the Tileworld scenario [31], where an unknown list of products have to be transported to fitting workstations. In the original article, we focused on the performance difference in solving the transporation task by controlling AGVs on a dynamic shop floor either with a CC or DALIs whose interactions are assumed to be functionally correct. Here, we take a closer look at self-adaption capabilities of DALIs in dynamic environments in the presence of errors in the perception and communication of agents. Extending the existing scenario of [36], agents may now suffer from the following errors:

- **Perception error**: we introduce a probability p_P that distorts an agent's perception such that the agent perceives a wrong station
- **Communication error**: we introduce a probability p_C that during communication the sent message is distorted to contain wrong information for the receiving agent

Again, we measure the performance in comparison to an omniscient CC as well as specifically the spread of erroneous information among the population.

Consider our running example of a phone casing manufacturer (Ex. 1.1). Several AGVs, controlled by DALIs, are tasked to transport products on a shop floor to workstations that fulfill the products' needs. To self-adapt to possible disturbances, agents share information on the shop floor on a local level - but internal errors make information especially sharing hard:

,
© 2024 Copyright is held by the authors. This work is based on an earlier work: SAC'24 Proceedings of the 2024 ACM Symposium on Applied Computing, Copyright 2024 ACM 979-8-4007-0243-3. http://dx.doi.org/10.1145/3605098.3635967

Example 1.1. *A new product, a phone casing enters the shop floor at a source station at position (0,0). The product has a list of required workstation skills attached and currently seeks a workstation with "soldering" skills. AGV0 is nearby, controlled by DALI0, perceives the phone casing, and starts the transportation process as the AGV was idle before. After arriving at the source station and querying the list for the next demanded skill, DALI0 is in trouble: as the product needs a "soldering" station and DALI0 has no entry in its model, it needs to quickly find out where to locate the station to reduce downtime. DALI0 asks DALI1, in range, for information on a workstation with soldering skills. Indeed, DALI1 has an up-to-date entry and returns the position of workstation WS3 at (5,2). Unfortunately, during sending an unnoticed error occurs and the received position at DALI0 is (0,10) and marked as a correct and up-to-date entry - after all, DALI0 relies on DALI0's observation hereon. Without further information or possibilities to check, DALI0 will steer AGV0 to a position far away from the product's actual goal.*

Taking possible disturbances into account, e.g. changes on the workstation assignment, the incorrect information is not the only hardship for DALI0, as WS3 might need service, and thus the skill changes again. Observations in the agent population become outdated, too, thus, not only the information at hand is incorrect, but agents need to dissipate potentially correcting data before. A hard task for agents with limited perception and communication skills.

As with controlling a functioning population of AGVS, a CC would adapt easily to perception and communication errors (assuming that it is a random error occurring and not a permanent one, e.g. a hardware error): by retrieving the information from the shop floor directly, the CC queries all AGVs and stations periodically and retrieves current data on subsequent percepts, if one before was incorrect. By reissuing a new command, the system is again on track. Considering the needed quality of information, the CC needs various information in a sufficiently high frequency, ranging from shop floor layout, position and pairing of AGV and product, needed skill, the occurrence of the disturbance, and position of the correct station, to a reliable and homogeneous network to interact with all components.

We simulate our extended Tileworld scenario with unforeseen disturbances for a fixed, unknown list of products that have transportation tasks attached. We measure performance, communication efficiency, and error correction with local observations of the overall DALI population and study how system's performance suffers by introducing errors in DALI's perception and communication. Our contributions are as follows:

- We use model-based DALIs with reactive behavior that adapt to solve transportation tasks in a dynamic environment that extends the Tileworld to a manufacturing scenario
- We evaluate the performance of DALIs in the presence of perception and communication errors and analyze the propagation of erroneous information across the population

The remainder of the paper is structured as follows: Sec. 2 gives an overview of related work from intelligent agents and self-adaption systems in intralogistics. Sec. 3 shows briefly our approach from [36] including a formalization and presentation of the implemented algorithm for DALIs. Sec. 4 introduces the new models that are used to simulate perception and communication errors as well

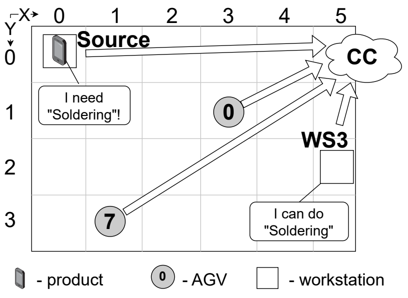

Figure 1: CC perceives all components on the shop floor, including products, workstations, and transporters (Ex. 1.1)

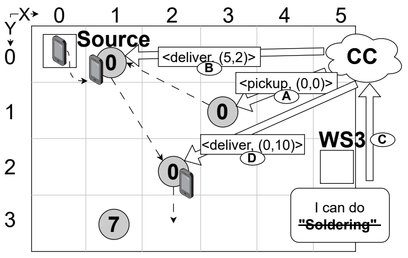

Figure 2: A: CC orders AGV0 to pick up the product at position (0,0). B: order to deliver to (5,3). C: CC perceives the disturbance. D: new order to deliver to (0,10) (Ex. 1.1).

as measurements for performance, communication efficiency, and error correction. Sec. 5 presents the experiments and results of the introduced errors and their correction, while Sec. 6 discusses the results in the context of the overall application. Sec. 7 concludes with a summary and some outlook for future work.

2 RELATED WORK AND BACKGROUND

2.1 Intelligent agents and model-based agents

Russell and Norvig [33] give structures for intelligent agents and describe basic agent programs. We follow their description of model-based agents (MBA) as agents that possess internal states, called models, and memorize parts of their environment that they cannot perceive at the moment. They may update the model as the world evolves. MBAs decide according to a set of condition-action rules with the help of the model's information and their current perception. The extension to a goal-based agent built on an MBA is straightforward when goals are defined that describe situations that are desirable for the agent. Goal-based agents can change their behavior for given goals. e.g. for pathfinding to a given destination.

2.2 Self-adapting agents in intralogistics

Agent-based systems in transportation attracted much interest in the past [45]. Proposals build e.g. on direct communication for local cooperation [6] or team building [41], action interpretation for cooperative cluster building [21], and range to applications in cooperative sorting in dynamic environments [37], control methods for transport flow with pheromones [23] or swarm-based urban waste collection with RFID tags [2]. Schmidt et al. [39] give an overview of approaches for decentralized intralogistics and state that multi-agent-based approaches form the majority of research approaches to implement decentralized control. Mechanisms include auctions (e.g. for in-house [12] or B2B logistics [32]), agent negotiations (e.g. [19]), or hybrid approaches like local coordination and plan merging [1]. Berndt [6] proposes a self-organizing supply system that, comparable to our approach, builds on decentralization, but uses direct communication and observations between agents to build expectations to adapt to unforeseen events. In this approach, the agents are bigger entities like consumers or manufacturers, linked by their business relationships. Klein [25] discusses decentralized strategies for dispatching, with a focus on disturbances in terms of system size change and path errors to analyze robustness. Related to our approach, but not the domain, are Lesser and Corkill [28] who present the Distributed Vehicle Monitoring Testbed, where distributed nodes share knowledge and goals, and cooperate to solve a single problem, the identification of vehicle trajectories, but stay stationary and do not further influence their environment. In contrast to other approaches, we focus on the union of the exclusive use of local communication, a dynamic environment that changes unpredictably outside agents' perception and disturbs agent's knowledge, minimal assumptions on the population and environment, and the autonomy of agents for choosing between exploration, knowledge exchange, and adaptation.

3 APPROACH AND REALIZATION

3.1 Assumptions for our approach

We approach our proposal of DALI with a simulation experiment of a dynamic shop floor that is concerning [33]:

- partially observable (information might be missed),
- stochastic (environment's state is not completely determined by the current state and agents' executed actions),
- dynamic (environment changes while agents deliberate), and
- discrete (in terms of time and states).

The task is to deliver products with an attached, ordered list of demanded skills to workstations on the shop floor that match these skills. We introduce periodic disturbances that swap the skills of stations every t_d cycles in the simulation. The DTS control system, either CC or DALI, has to steer AGVs for pickup and delivery to fulfill the task. We compare the performances for the time needed to finish a pre-specified list of products that is secret to the control system across different shop floor setups and disturbance times. We make the following assumptions about onboard AGV equipment to be used by DALI:

- Sensors to perceive nearby surroundings, including AGVs, products, and workstations including their skill
- Communication devices to contact nearby AGVs
- Automatic collision avoidance
- Lamport clocks to denote the occurrence of events [26]

AGV equipment is only relevant for DALI as CC is assumed to be omniscient and has no need to use it.

3.2 Formalization

We base our work on the MOSAIK model [10, 38], which in contrast focuses on self-organizing agents using their environment for communication, and derive our formal definitions from [9, 29].

Definition 3.1 (**DALI**). *Decentralized agents with local information are proactive components (as opposed to artifacts) that decide on their own. DALI implements a perception-thought-action cycle to influence the environment, based on defined condition-action rules. We define DALIs as the tuple* $A = \langle \mathcal{M}, perceive, apply, act\rangle$ *with*

$$\begin{gathered}
\mathcal{M},\ \textit{the agent's model and set of possible knowledge}\\
perceive : E' \times \mathcal{M} \longrightarrow \mathcal{M}, \textit{perception based on the environment},\\
apply : \mathcal{M} \longrightarrow \mathcal{M}, \textit{function for derivation rules},\\
act : \mathcal{M} \longrightarrow O, \textit{function for condition-action rules},\\
\textit{with } O, \textit{ as set of operations with } \{\langle pickup, pos\rangle,\\
\langle deliver, pos\rangle, \langle explore, pos\rangle, idle, O_{com}\} \textit{ where}\\
O_{com} : \mathcal{M} \times \mathcal{N} \longrightarrow \mathcal{M}, \textit{communication with neighbors } \mathcal{N}
\end{gathered}$$

where E' *is the set of all percepts and* E *is the set of all environmental states with* $E' \subset E$ *(cf. Def. 3.2).*

The model $\mathcal{M}$ maps entries to observed *skills* with

- a position on the shop floor F,
- a Lamport time stamp when entries are created or updated,
- a Lamport time stamp when entries are invalidated,
- a curiosity value between $[0, 1]$ that steers exploration.

Based on limited perception, *perceive* recognizes nearby surroundings, including neighboring AGVs (here used as $\mathcal{N}$), products, and workstations. *apply* revises the agent's model based on derivation rules, e.g. to determine outdated entries. *act* influences the agent's environment via operations O, where the respective operations order the AGV to go to a position to pick up or deliver a product, move to a specific part of the shop floor for exploration, or to be idle. O_{com} exchanges model entries with neighboring AGVs $\mathcal{N}$. Each DALI may submit operations for exactly one defined AGV.

Definition 3.2 (**Artifact**). *Artifacts are reactive components (as opposed to agents) that form the agents' environment [9]. Artifacts provide different ways for interaction, e.g. to read a product's demanded skill or to order an AGV to move. An internal logic defines the deterministic reaction to interactions, but artifacts do not act autonomously. We define the environment as*
$Env = \langle E, e_0, O, transfer, update, evolve\rangle$ *with*

$$\begin{gathered}
E,\ \textit{the set of all possible environment states}\\
e_0,\ \textit{the environment's initial state}\\
transfer : E \longrightarrow E', \textit{for agents to retrieve a percept } E' \subset E\\
update : O \times E \longrightarrow E, \textit{effectory function based on operations}\\
evolve : E \longrightarrow E, \textit{the environment's development.}
\end{gathered}$$

In our setting AGVs, workstations, and products are artifacts. Artifact behavior is a black box for agents, but the behavior's result is observable via environment state changes and the *transfer* function. Environment state changes, e.g. movement on the shop floor or performed work on products, may be triggered by agents as well as the physical environment via disturbances, represented by *evolve*. We treat the shop floor as a tiles lattice as in [31].

3.3 Behavior of DALI

We simulated the experiment setup with GAMA [1], an agent-based software platform. You may find our code in our online appendix[2].

We present the algorithms for the perception, communication, and cooperation of DALIs. We use examples from the perspective of AGV0 from Ex. 1.1, now controlled by DALI0.

Perception. Alg. 1 aligns the model $\mathcal{M}$ with the perceived environment D' (cf. Def. 3.1, *perceive*). DALI uses the controlled AGV t's sensors to get a percept including the AGV's state $t.state$, position pos, and nearby stations. When DALI observes a station s with a skill $skill$, DALI checks whether the model already contains data. If not, a new entry is created, with the current timestamp τ_{now}, 0 for invalidation, and curiosity c with 0.0 (l. 4). If the station is already known, DALI updates its model (l. 6) and invalidates older entries pointing to the same location (l. 12), as the observation reflects the current state of the perceivable environment. If the AGV is not busy with pickup or delivery, DALI starts to explore the shop floor to bring $\mathcal{M}$ up to date and detect further disturbances (l. 9). Finally, $\mathcal{M}$ is updated (l. 15).

Algorithm 1: Perception - *perceive*

```
   Data: inital $\mathcal{M}$, DALI's controlled AGV $t$
   Result: updated $\mathcal{M}$, $t.state$
1  $\mathcal{M} \leftarrow$ new percept from AGV $t$;
2  if AGV $t$ is next to station $s$ then
3      if $skill$ NOT in $\mathcal{M}$ then
4          $\mathcal{M} \leftarrow$ new $skill$ entry with $s.pos$ and $\tau_{now}$;
5      else
6          $inv \leftarrow$ all entries in $\mathcal{M}$ pointing to $s.pos$;
7          if $|inv| > 0$ then
8              if $t.state$ NOT "$deliver$" $\vee$ "$pickup$" then
9                  $t.state \leftarrow$ "$explore$";
10             end
11             forall $entries$ in $inv$ do
12                 invalidate and note timestamp in $\mathcal{M}$;
13             end
14         end
15         $\mathcal{M} \leftarrow$ update $skill$ entry with $s.pos$ and $\tau_{now}$;
16     end
17 end
```

Example 3.1. Fig. 3 - AGV0 moves near workstation WS3 at (5,2) with the skill "bolting". DALI D0 requests AGV0's percept of the environment. D0 checks the model $\mathcal{M}$ at "bolting" and notices that it last perceived the "bolting" skill at (0,10). As skills are unique, D0 invalidates the entry. Also, the last time D0 perceived "soldering" was at (5,2). D0 updates the entry such that $\mathcal{M}(\text{"}bolting\text{"})$ now points to (5,2). The position of the "soldering" skill is nevertheless unknown to D0, as it cannot be certain where the disturbance happened.

[1]https://gama-platform.org/

[2]https://github.com/wintechis/dali-adaption

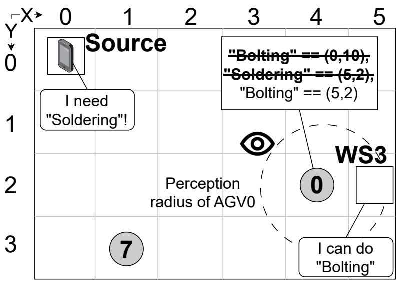

Figure 3: DALI D0 perceives WS3 and identifies the exposed skill "Bolting" to correct its internal model (see Ex. 3.1)

Communication. We model communication in Alg. 2 as an anti-entropy approach of simple epidemics as 1-to-1 pull communication [15]. Epidemic algorithms (EA) can eventually distribute knowledge and help AGVs save power, as EA reduces the number of messages between nodes, e.g. compared to broadcasts [35]. The initiating DALI asks a random neighboring DALI for information on model entries with skill and position (l. 1) and adds any new entries (l. 6). Existing observations are updated depending on the timestamp if the shared observation is more recent (l. 6) and otherwise ignored. The process is monotonic as information is only added to the model.

Algorithm 2: Communication - *act*

```
   Data: $\mathcal{M}$, neighboring DALI $D_N$
   Result: updated $\mathcal{M}$
1  $observations \leftarrow$ ask neighbor $D_N$;
2  forall $entry$ on $skill$ in $observations$ do
3      if $skill$ NOT in model $\mathcal{M}$ then
4          $\mathcal{M} \leftarrow$ new entry on $skill$ as in $D_N$;
5      else
6          if $skill$ in $\mathcal{M}$ but $D_N$'s entry is more recent then
7              $\mathcal{M} \leftarrow$ updated entry on $skill$ as in $D_N$;
8          end
9      end
10 end
```

Example 3.2. Fig. 4 - AGV0 carries a phone casing that needs a "soldering" skill. D0 has invalidated "soldering" at (5,2) before and now searches for new information to fulfill the request. AGV0 passes AGV7 and AGV8 at the same time. D0 evaluates the percept, chooses randomly AGV8, and asks its DALI D8. D8 shares a current observation for "soldering" at (0,10). D0 compares its model with D8's answer and replaces the invalidated entry. D0 concludes that phones casing's request for "soldering" can be fulfilled at (0,10). D0 orders AGV0 to move to (0,10).

Curiosity and exploration. DALIs have to fulfill the transportation task as fast as possible so DALIs have to detect disturbances quickly. Instead of controlled forgetting [7] or commitment [24], DALIs may decide to re-evaluate their model and deliberately move AGVs to stations to check if disturbances happened, even if the AGV

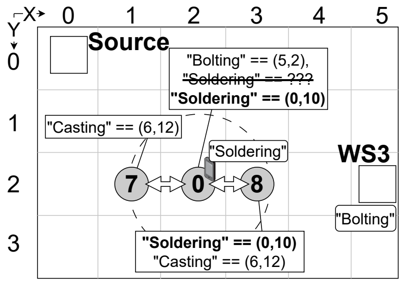

Figure 4: Searching "soldering", DALI D0 contacts neighboring AGVs, and receives an observation for "soldering" at (0,10). D0 updates its model and sends AGV0 to (0,10) (see Ex. 3.2).

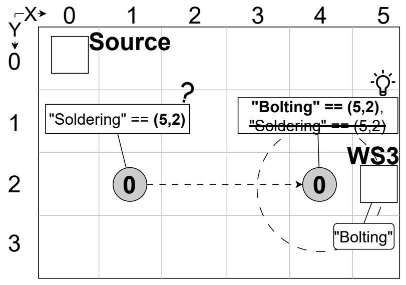

Figure 5: DALI D0 gets curious about the WS at (5,2) and moves AGV0 there. Here, D0 notices "bolting" at (5,2) and updates the model. D0's curiosity is satisfied (see Ex. 3.3).

carries no product that justifies moving to the station - which seems counter-intuitive as DALIs produce empty runs.

DALI's so-called curiosity, Alg. 3, keeps a balance between committing to visit a station and being free enough for exploration. During action selection (cf. Def. 3.1, *act*) the curiosity value acts as the probability of entering a "explore" state to move to a station. If an AGV is idle, DALI takes a random entry (l. 1) and increases the curiosity for a visit to the known location (l. 2). When the station is reached, the curiosity is satisfied and reset (l. 4). Without a target, the AGV stays, until DALI's curiosity motivates it to move the AGV or a disturbance causes exploration (Alg. 4), coming from perception (Alg. 1) or communication (Alg. 2). The curiosity mechanism also counters deadlocks as blocked stations with products may get serendipitous visits from AGVs.

Algorithm 3: Curiosity - apply

```
Data: $\mathcal{M}$, constant for curiostiy increase $cur$
Result: curiosity of $\mathcal{M}$
1 $skill \leftarrow$ choose random entry from $\mathcal{M}$;
2 $\mathcal{M}(skill).c \leftarrow$ increase $\mathcal{M}(skill).c$ by constant $cur$ ;
3 if $AGV\ t$ is next to station $s$ with $skill$ then
4     $\mathcal{M}(skill).c \leftarrow 0.0$ ;
5 end
```

Example 3.3. *Fig. 5 - As AGV0 is idle, DALI D0 increases its curiosity to visit (5,2) where it last perceived "soldering" and orders AGV0 to move to (5,2). D0 notices that "bolting" is now at (5,2) and updates the model (see. Ex. 3.1). The curiosity is satisfied and D0 decides to stay at (5,2). Over time, D0's curiosity rises to visit (0,0). Unknown to D0, a new phone casing waits at (0,0), outside any AGV's perception. As D0 ordered AGV0 to (0,0), D0 may discover the phone casing.*

When DALI concludes that a disturbance happened, it suspends the transportation task and explores the shop floor to correct the model. When the model seems to be correct again (via exploration or communication), the transportation task is resumed.

We base exploration (Alg. 4) on frontier-based exploration [44], where DALIs save checked tiles and unexplored tiles. Initially, no tiles are checked, and the current neighboring positions are the unchecked floor tile frontiers. Checked tiles are saved, and new perceived, unchecked tiles are added to the remaining ones. Two exploring DALIs also exchange information about their explored and unexplored areas. DALI chooses as best option a frontier from its neighbors which borders most checked tiles (l. 2), or if no such option exists, the closest frontier tile (l. 4) or a random neighbor (l. 7). The next target to check is then chosen from these best options (l.9). We stress that we leave the option for the shop floor to change completely, so DALIs make no fixed assumptions on the layout.

Algorithm 4: Exploration - apply

```
Data: $checkedTiles, remainingTiles$
Result: $target$ for AGV
1 $options \leftarrow$ closest neighboring frontier tiles;
2 $bestOp \leftarrow$ tiles that border most checked tiles;
3 if $bestOp = \emptyset$ then
4     $bestOp \leftarrow$ closest frontier tiles in general ;
5 end
6 if $bestOp = \emptyset$ then
7     $bestOp \leftarrow$ any neighboring tiles;
8 end
9 $target \leftarrow$ choose randomly from $bestOp$;
```

Example 3.4. *Fig. 6 - AGV0 at (5,3), DALI D0 explores the shop floor to find a station with "soldering" skill, to fulfill phone casing's request. D0 already checked parts of the shop floor and now moves AGV0 to (6,8), the nearest frontier. On its way, AGV0 meets AGV1, another AGV exploring "soldering", controlled by DALI D1. D0 and D1 exchange information about their explored areas. D1 shares that it already explored (6,8), but does not know where "soldering" is, so D0 adds (6,8) to its explored area. D0 concludes that (6,8) does not need to be checked anymore and instead moves AGV0 to frontier (7,7). On the way, AGV0 meets AGV8, controlled by D8. D0 and D8 exchange information (cf. Ex. 3.2), where D0 learns about "soldering" at (0,10). D0 ends the exploration and moves AGV0 directly to (0,10).*

4 ERROR AND MEASUREMENTS

We extended our study by introducing two new variations of error types and study how agents can correct and adapt to these additional disturbances. Besides the dynamics of the environment,

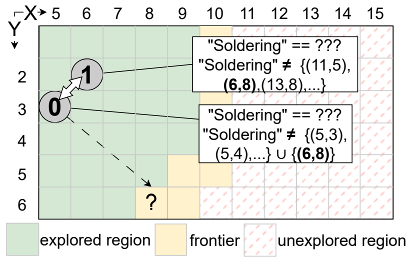

Figure 6: DALI D0 looks for "soldering" and has already explored parts of the shop floor (green). On meeting D1, D0 learns that D1 already checked (6,8), but knows nothing about "soldering". D0 concludes that visiting (6,8) is useless and selects another frontier target. (see Ex. 3.4)

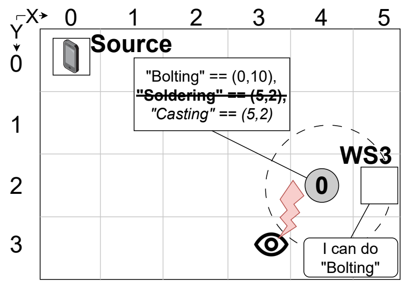

Figure 7: DALI D0 perceives WS3, but instead of identifying the correct "Bolting" skill, D0 perceives "Casting" such that D0's model contains now two wrong entries: both positions for "Bolting" and "Casting" are wrong.

agents also have to take their own errors into account that extend to both senses agents use to receive knowledge about their environment: their perception and communication capabilities.

Note that we use global information for the manipulation: we choose data from the setup of the environment to insert in the agent without its knowledge, thus the agent is unaware of the manipulation and the global information used - the agent has still only access to its model for its decision making. The only way for the agent to correct its wrong model is then either by exploring and finding out the truth fast enough, or receiving a correct update via communication. For both, however, the probability exists again to be distorted via our manipulations.

4.1 Perception error

DALIs have a limited perception which is a cornerstone of their purely decentralized and distributed nature. At the same time, we already saw that a limited perception entails other, more serious issues, that is, the necessity for exploration. In our initial study, we saw that exploration makes up a non-negligible part of their overhead in performance when compared to a CC.

Here, we manipulate an agent's perception before a perceived fact may be applied to the model. With relation to Alg. 1, we introduce the probability p_{err} that is evaluated after receiving the percept from the AGV (the manipulation is marked):

Algorithm 5: Manipulated perception (excerpt)

```
percept $P \leftarrow$ new percept from AGV $t$;
if $flip(p_{err})$ then
    $P.s.skill \leftarrow s'.skill$ with $s' \in S | s' \neq s$;
end
$\mathcal{M} \leftarrow P$;
if AGV t is next to station s then
    ...
end
```

With probability p_{err}, we replace the received percept with the skill of some other station. The manipulation is thus equal to a misinterpretation of the environment, e.g. represented by a false identification of a bolting station as a soldering station. Only after the manipulation is done, the agent receives the percept and applies it to the model.

The consequences are quite severe - another entry in the model for station s' will be overwritten with the position of s as entries for skills collide, even if the agent perceives s correctly in a subsequent percept. Hence, the agent's current information on the non-perceivable, far away station s' is lost, but the agent holds the belief that its knowledge is up to date, as the agent perceived the "fact" only moments ago on its own (see Fig. 7). Additionally, the agent may decide to start exploring the environment as the falsely updated model suggests a disturbance.

4.2 Communication error

With DALI using EA for communication, updates on changes may easily spread across the population, even if the range is limited. On the other hand, updates containing errors may spread easily when the only means to differentiate the information is a timestamp of observation. Here, communication is key for errors on two accounts: first, errors coming from perception errors (cf. Sec. 4.1) can be easily shared with all other stations that are pulled by a requesting agent. Second, messages can be distorted or misinterpreted to contain wrong information. Based on Alg. 2, we introduce the probability p_{com} that the agent evaluates after receiving requested observations from another agent (the manipulation is marked): As before, the

Algorithm 6: Manipulated perception (excerpt)

```
observations $\leftarrow$ ask neighbor $D_N$;
forall entry on skill in observations do
    if $flip(p_{com})$ then
        $observation.s.skill \leftarrow s'.skill$ with $s' \in S | s' \neq s$;
    end
    if skill NOT in model $\mathcal{M}$ then
        $\mathcal{M} \leftarrow$ new entry on skill as in $D_N$;
        ...
    end
end
```

manipulation cannot be detected by the agent. We replace the skill

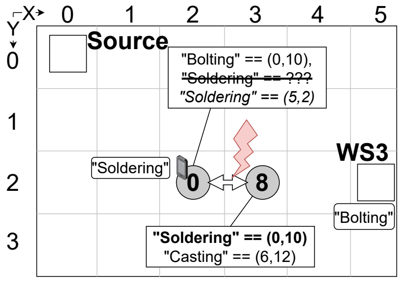

Figure 8: Searching "soldering", DALI D0 the neighboring D8. While D8 answers with the correct observation for "soldering" at (0,10), D0 receives the wrong position (5,2). With the incorrectly updated model, D0 sends AGV0 to (5,2).

of a received observation by some other station, which is applied to the model. Note that the time stamps are nevertheless checked for the original received station, so if the agent concludes to update or invalidate an entry, it does so with the correct time stamps, but for the wrong entry, possibly overwriting correct knowledge again (see Fig. 8). Concerning the sending agent, as no feedback is sent about applied or invalidated observations, the sender cannot know whether the receiver understood the message correctly.

4.3 Measured parameters

We define parameters to measure the performance of DALIs for their task as well as the efficiency of communication.

4.3.1 Task performance. We measure the system needed the time to finish the list of products (TTF) and mean time to deliver (MTTD) to judge the performance in the simulations over time [34]. To measure the *TTF*, we consider the total delivered products (TDP) over the agent population A: $\texttt{TDP}(t) = \sum_{a \in A} \texttt{deliveredProducts}(a, t)$ where $\texttt{deliveredProducts}(a, t)$ is the cumulative amount of delivered products for a single agent at time t. When the TDP is equal to the pre-given number of orders, all products were successfully delivered to shipping. The faster, the better. The resulting time when all products are delivered is the TTF. We define the *MTTD* for a product p in the set P of all successfully delivered products in one simulation run as: $\texttt{MTTD}(t) = \frac{\sum_{p \in P} \texttt{delivered}(p) - \texttt{created}(p)}{\texttt{TDP}(t)}$ where $\texttt{delivered}(p)$ is when a product p was created and $\texttt{created}(p)$ when p was successfully delivered to the specified station. The time includes waiting time at the original station plus time spent on the ride. The faster a product is delivered, the better. We evaluate both measures at the last time steps $t = TTF$ as performance indicators for the overall run. Finally, we calculate the mean over all repetitions of the scenarios.

4.3.2 Communication. We measure residue and delay based on [15] to judge communication efficiency among DALIs, but modify the formulas to fit the dynamic setting.

Residue is the percentage of agents that miss an update after a disturbance. The residue for an update u among the population A, where $S \subseteq A$ is the set of susceptible agents, at time t is: $\texttt{Residue}_u(t) = \frac{|S(t)|}{|A(t)|}$. At the last time step before the next disturbance cycle t_d, we evaluate which agents have received all updates in time and calculate the residue for the period. At $t = TTF$, the average of all residues is calculated. Residue shall be minimized as it represents the percentage of agents that missed an update.

Delay measures the average time t_{avg} it took an update to arrive at an agent after a disturbance's occurrence. We measure the time difference every t_d until an agent either notices a change on its own or gets a message that changes its model. Delay is only calculated if an agent did receive an update successfully, thus we do not consider agents that did miss updates. We calculate the average per agent over the total amount of received updates per agent, and the simulation's average delay t_{avg} over the population at $t = TTF$. Delay represents the speed of communication and shall be minimized.

4.3.3 Error correction. We measure error correction for agents that receive updates that lead to a correct model of the environment. These updates are either the observation of a fact in the environment or the receipt of a message with new information from another agent.

We measure all updates u applied at time t that changed an agent's model $\mathcal{M}$ to correctly represent the environment E at t, as well as whether the update originated from its own perception as $corr_P$ or was communicated by others as $corr_C$. Of course, we only count updates that change the state of the model from wrong to correct. Additionally, we measure how long it takes an agent to correct incorrect information in its model with the time to correct t_{corr}. Note that t_{corr} is different from delay t_{avg}: delay measures the time it takes for agents to receive an update about changes in the environment after the change has occurred. For all measures, we calculate the average per agent over the total number of updates received per agent.

5 SIMULATION SETUP AND RESULTS

5.1 Experiment setup

We extend our experiment with a shop floor of 25 × 25 floor tiles, four workstations with the skills "casting", "bolting", "molding", and "soldering", and 12 AGVs at random starting positions. The DALIs may only perceive their direct adjacency of the eight neighboring fields, as in Tileworld. Furthermore, the agents start with a correct map of the setup as with [24].

Similar to our initial study, we fix the agent's curiosity increase with $cur = 0.01$ and pre-defined lists of $P = \{50, 100, 250\}$ products with combinations of up to four distinct stations to be visited. Every t_d cycles, we disturb the experiment setup periodically as part of the environment dynamic by swapping two stations' skills and chose the periods of $\tau_d = \{100, 150, 200, 250, 300\}$ cycles and without (w.o., $\tau_d = none$). As probabilities for error, we vary the p_{com} and p_{err} each to create a respective error with a chance 1%, 5%, 10%, and 25%. We simulate two shop floor setups:

- Scenario A: source and shipping on opposite corners
- Scenario B: source and shipping close to the center

We made 50 repetitions for each scenario and stopped the simulation after all products were delivered while measuring the presented indicators from Sec. 4.3.

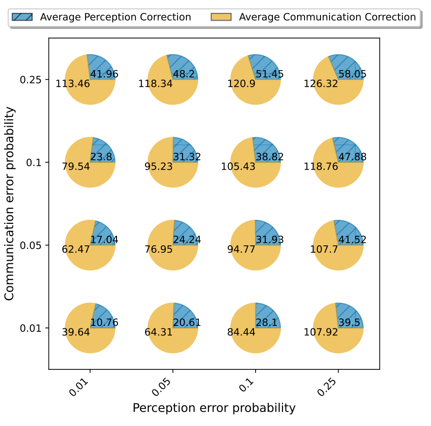

Fig. 9: 50 prod., t_d = 100 (A)

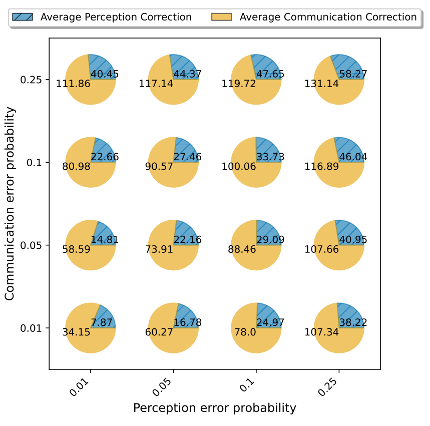

Fig. 10: 50 prod., t_d = 150 (A)

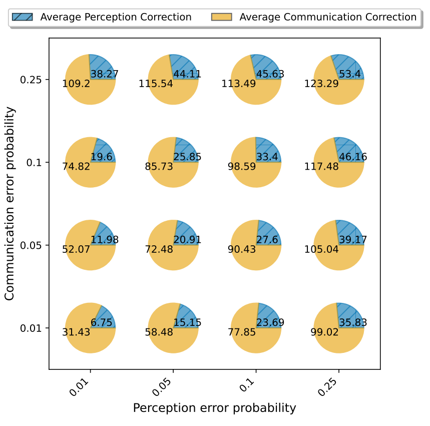

Fig. 11: 50 prod., t_d = 200 (A)

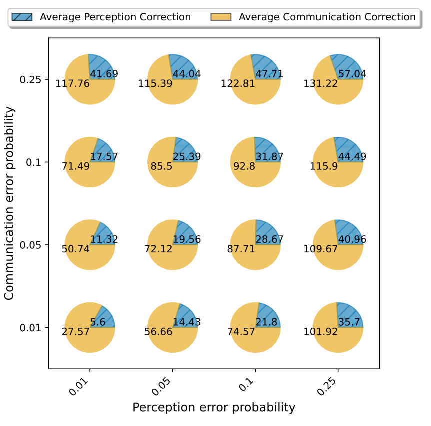

Fig. 12: 50 prod., t_d = 250 (A)

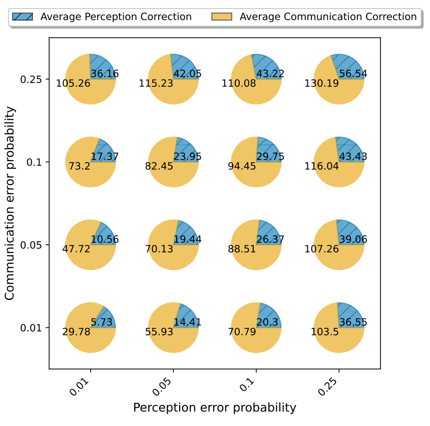

Fig. 13: 50 prod., t_d = 300 (A)

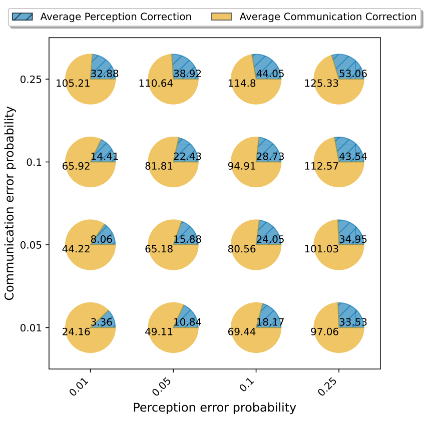

Fig. 14: 50 prod., t_d = *w.o.* (A)

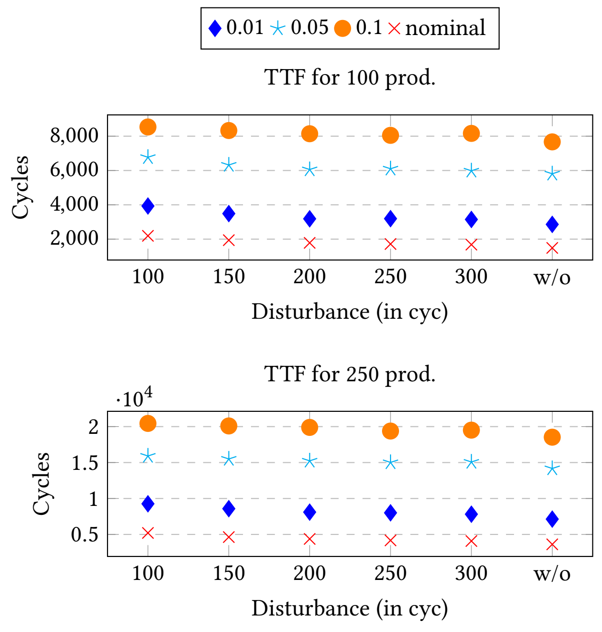

Fig. 15: TTF among disturbances of nominal DALI and with p_{err} and p_{com} as both 0.01, 0.05, 0.1

Performance with errors. Fig. 15 shows an overview of different measured TTFs across nominal DALI behavior without errors in comparison to performances with p_{err} and p_{com}. While the overall performance is qualitatively similar (performance improves with longer t_d), the effect of errors on the performance is clear: without additional errors, DALI finishes a delivery of 100 products in about 2192.9cyc for $t_d = 100$. With 1% for p_{err} and p_{com}, the delivery takes 79% longer with 3930.8cyc in the same scenario. Note that an error likelihood of 1% is still the best case. For $P = 50$ we have a factor from 1.63 to 2.05, for $P = 100$ from 1.78 to 1.92, and for $P = 250$ from 1.78 to 1.96 across all disturbances. On average, the performance is 1.86 times longer for 1%.

In the worst case, when 25% of all percepts and communication is distorted, the system can take eight times as long to finish the delivery, e.g. for 50 prod with $t_d = none$ the fault DALI finished in 6303.81cyc; a factor of 8.03 compared to 784.98cyc in the nominal case and 3.98 times worse compared to 1% with 1580.44cyc. For $P = 50$ we have a factor from 5.54 to 8.03, for $P = 100$ from 5.7 to 7.9, and for $P = 250$ from 5.66 to 7.77 across all disturbances. For the worst case, the system needs on average 6.89 times as long.

Despite the system suffering from the artificial errors we introduce, DALI still manages to deliver all products in all cases. As expected, the performance gets worse with more frequent disturbances in the environment (as with nominal) and with an increased error probability, but instead of failing for good, the DALI system degrades gracefully. Still, the effect of recovery of performance with a less frequent disturbance is getting less with a higher error probability, e.g. for 100 delivered products, the nominal performance improves from 2192.9cyc with $t_d = 100$ to 1487.9cyc with $t_d = w/o$ (improved 32%), where p_{err} and p_{com} with 1% has a similar, but worse, improvement in the same scenarios from 3930.75cyc to 2859.5cyc (improved 27%), whereas p_{err} and p_{com} with 25% improves merely from 8537.125cyc to 7668.6cyc (improved 10%).

Error correction. Fig. 9- 14 show the average proportions of corrections for scenario A with 50 products[3] for all agents among the scenarios, where we discern correction coming from local observation by agents (perception correction) and corrections coming from exchanged messaged among agents (communication correction).

We can compare both modes of correction, perception, and communication, as we assume the same range for both in DALIs - on the simulated shop floor the range includes the direct neighborhood of "floor tiles" meaning that a DALI has to be directly next to a workstation to perceive it and directly next to another DALI to exchange information. In any case, the possibilities are limited to receive updates as the range is small. If no errors were present, obviously the direct observation of an agent would be the more reliable source - the agent may directly perceive and thus update its model with a fact. Received information from another DALI can be outdated by the time it arrives at an agent, which is why DALIs are cautious to apply updates if the time stamps are not acceptable.

The proportion of sources for correcting updates shows now a different picture: communication is a far more frequent source for correcting the model as assumed - the reason is that a DALI is more likely to meet other agents on its way during delivery than meeting, increasing the probability to exchange messages. Also, the perception of a workstation is a "unique activity" (a DALI arrives at a station, scans it, then leaves and keeps its observation until contested) whereas communication can be repeated (as two DALI pass each other several rounds of exchange may happen). Still, for communication to work the time stamp has to fit - and only because it happens more frequently doesn't mean it is more reliable (with a 25% chance of two passing DALIs to introduce for 4 exchanged entries one error in each other's model on average is devastating!). Respectively, the amount of perception corrections rises with communication becoming more unreliable with rising p_{com}.

Fig. 16 shows t_{corr}, the average time needed for an agent to receive an update that corrects its model. The development shows that with the rising likelihood of errors, the time to correct the errors decreases. While the development may not seem intuitive, we see two reasonable sources:

- The more errors exist, the more often agents start to explore and thus need to invest time to update their model to reflect the (assumed) true state and share updates (as is visible with the number of corrections that rise with the probability of errors)
- With serendipity, an artificially introduced error in the model of an agent becomes true via disturbances. While the possibility of such a coincidence exists, it is very small.

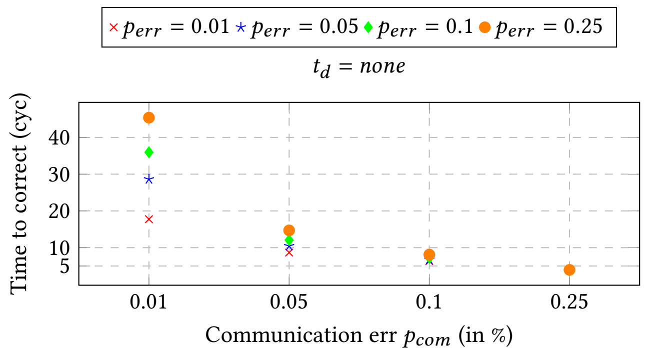

Fig. 16: Needed time to correct t_{corr} an error per agent with different p_{err} and p_{com}

6 DISCUSSION

We study an agent-based decentralized control system for intralogistics as an alternative to centralized control systems as decentralized systems promise us advantages in terms of adaptability, independence from centralized networks, and graceful degradation. While our original work focuses on designing DALIs and measuring their adaptive performance in dynamic environments, we did not consider their ability for graceful degradation. In terms of fault-stop errors where a DALI breaks down completely, it is easy to predict that a system of DALIs will perform worse over time as fewer DALIs can serve product requests until too few are in the system and deadlocks occur such that the system does not work at all.

Here, we introduce errors that are quite malicious and in terms of graceful degradation more interesting: instead of agents receiving or perceiving obviously faulty information in their environment, e.g. nonsensical information exposed by a workstation or gibberish sent by an agent, which lets them conclude concerning their internal model that *something* went wrong, we deliberately manipulate errors such that they are believable and plausible. As a harmful consequence, such an update does not only overwrite already existing knowledge but also invalidates entries that depend on the changes. For an agent that relies purely on its model such an error has a far-reaching impact.

Still, such errors are everyday problems: a faulty sensor generates a wrong measure, a damaged label has a misleading identification, or the agents' environment was changed deliberately, e.g. because of a changed shop floor layout, but the control software was not updated - usually a temporary error but still one that entails an undesired halt of the manufacturing shop floor.

Overall, we study, how well mobile agents relying purely on local perception can adapt in dynamic environments, where corrections for possible internal errors come from most (and thus the biggest impact exists), and thus conclude where to put focus on creating an adaptive DALI system. Based on our measurements, we come to the following conclusions:

[3]The figures for 100 products (Fig. 17 - 22) and 250 products (Fig. 23 - 28) are for legibility on p. 12ff

- We see that most corrections come from communication, but as communication becomes unreliable, perception catches up and compensates for the loss of communication updates.
- An increasing error rate leads to a loss of performance, as expected, because of the need for DALIs for exploration to correct their internal map. More exploration results in more empty runs and a high MTTD, as products have a longer waiting time.
- Despite an error rate of up to 25% for communication and perceptions, all products were successfully delivered and the system did not break down completely, e.g. because of deadlocks.

As an increase in the error likelihood leads to more errors, DALIs are more likely to detect a deviation of their model to the factual environment state. As a consequence, t_{corr} decreases, as DALIs are more often in a state of exploration as their model is unexpectedly often wrong, resulting in empty runs and degrading performance. Consistent with Kinny and Georgeff [24], we observe that with reduced environmental dynamics the agent performance increases. Additionally, we observe that the qualitative behavior of the system is unaffected compared to a performance without errors.

Closely related are fundamental thoughts about distributed systems and their difficulties in reaching consensus [30]. The problem of agents that share wrong information could be regarded as a Byzantine Generals problem [27], but classical solutions are not applicable as agents do not form a stable $3m$-regular graph nor is it guaranteed that at most m agents generate wrong values as failure occur randomly. To be adaptive, DALIs use a combination of curiosity and exploration and then try to act accordingly: over time, entries in their own model are actively questioned whether they still hold, while when disturbances are noticed the ongoing task is suspended [7] until a working state is reached. Thus, DALIs swing between the states of active commitment to the tasks and suspending tasks on disturbances to gather up-to-date data by frontier-based exploration [44] and epidemic communication [15]. The task is immediately resumed when the DALI's model is plausibly up-to-date to solve the task [20]. The difference in TTF shows that despite multiple possible detours because of wrong information, the overall transportation task is nevertheless successfully solved (in the best case only 1.86 times for a 1% error chance).

In [36] we already discussed the important role of curiosity to think *ahead* of disturbances and explore the unknown. With our current results, the importance is even more emphasized as the core problem of dealing with unsure information outside the current perception range is worsened by unreliable information sources. Still, curiosity cannot guarantee a reliable adaption for all possible disturbance cases, but at least give agents an incentive to care. Several strategies exist to influence the behavior of DALIs in terms of disturbance detection (be it because of an environmental change or an internal defect): while the natural increase of curiosity *cur* can be influenced to steer a more or less curious DALI, we caution against too drastic increases or decreases of *cur* as with both the balance of a working and self-exploring DALI is disturbed. Either too much exploration happens, causing empty runs, or not enough exploration happens, causing odysseys that end up in the worst case in a random walk behavior.

The generalizability of our results is of course dependent on additional factors that influence the system performance, e.g. the shop floor area, the number and perception range of AGVs, and the station setup. We observe similar developments of our approaches in the compared scenarios for performance and communication, so we are confident our results can be reproduced in similar settings.

A future refinement of the DALI approach can - without relaxing the problem - only happen along a few parameters that change:

- how the agent models its environment,
- the communication style,
- the exploration on disturbance detection,
- or the mechanism for disturbance anticipation.

While the model is currently a mapping from workstation skill to location, time stamps for observation and invalidation, and curiosity, a more refined representation is possible, but can easily introduce additional dependencies on central components (disturbance rates for workstations, spawn rates for products, etc.). While DALIs might be able to measure these during execution, their observations will stay guesses as they are difficult to verify during execution time - in the end, the system shall solve transportation tasks and that is the metric that counts. Every other overhead that does not immediately improve the performance will be punished. The communication style is deliberately chosen as simple entropy - as we have seen in previous work [35] a 1-PULL communication style is sufficient for mobile agents and reduces the coordination overhead among agents while it is still fast enough to share information even in a distributed population. Exploration with an increased perception radius may help to gather more useful data at once (but also additional wrong data in the presence of faulty sensors), while alternatives to the frontier-based exploration [44] can increase the speed of collaborative explorations. Our work aims for a rigorous baseline performance with DALIs using only the least possible information without any centralized assumption over the environment or the agent population. When relaxing these strict requirements, e.g. allowing shared but not necessarily centrally controlled resources for agents, related variants of Yamauchi's approach may be used instead, see f.i. [4] to create shared maps, or RACER [47] which uses a pairwise interaction between agents with designated exploration areas, or [37] which uses manipulable floor tiles to act as a medium to place information in the environment as indirect communication between agents - although the argument of centralized mechanism introduced indirectly is applicable again. Disturbances anticipation ahead lies on a spectrum from rigid and simplistic to complicated: While AGVs can simply circle between stations, as discussed in [39] as a first-encountered-first-served rule, we deem such solutions that are tied to the floor layout as unsuitable for a use case with high disturbances. Burgard et al. [8] use local assignments of targets based on shared utility values based on the predicted explored area, and save other robots' targets when they are in range to coordinate the exploration efforts. In a comparison of approaches for coordinated exploration under communication constraints [13], the authors find a similar approach most efficient, where robots evaluate each other's poses for coordinated exploration and select a random frontier to explore when it checks true for other nearby robots.

All in all, we argue that a system may adapt with simple, distributed agents even in complex dynamic systems, using few assumptions about the world. In combination with our original work, we have shown that a DALI system can not only rival a centralized control approach like VDA5050 [42] in nominal situations, but also function under pressure of environmental dynamics and internal faults, albeit to the cost of a more careful and complex system design. Despite using only local communication and perception, the performance of DALI to inform all members and to deliver all products is encouraging. In our opinion, the real strength to use a DALI system plays out not necessarily as a replacement for centralized control, but as a backup system. Industry favors central control because of its simplicity, but systems with centralized control can experience failures, f.i. hardware and software defects, network failures, power outages, cybersecurity problems, or scalability issues because of high complexity, where the control system becomes nonfunctional. We believe that a fail-safe system for an AGV control system like VDA5050 is best realized by using a multi-agent system (MAS) approach like DALI, where the AGV control can be switched to decentralized when the central control fails. From an AGV's point of view, the control is the same: with a connection to CC, AGVs receive destinations and act on reflexes. Without a connection, DALIs choose the AGVs' targets, keep track of the world's state in their model, explore, and communicate with other DALIs in range. Thus, DALIs enable decentralized adaptivity as a fail-safe system in environments with difficult constraints - instead of not at all or with extensive redundant hardware. As a consequence, shop floors may use hybrid, adaptive systems that work in dynamic environments. Here lies a research direction, especially for intralogistics [18, 39] or transportation [5]. Further applications include unfeasible environments for global networks for CC (steel walls, underground mines), where expenses for redundant hardware for downtimes shall be saved, or strict safety regulations apply (oil, gas, fine dust e.g. in wheat mills). AGVs can organize with DALIs, before coming back to an area where they are controlled by a CC.

7 CONCLUSION AND OUTLOOK

We presented our decentralized agent-based approach called DALI uses local information to adapt successfully to a dynamic environment based on a driverless transportation system. We confronted DALIs with a shop floor environment that suffers from unforeseen disturbances as well as a manipulated perception and communication by distorting both to include random errors. Based on the controlled experiments, we measured the performance of given transportation tasks as well as the capabilities to self-adapt as a population by exploring the environment for disturbances and sharing updates. We conclude that DALI can adapt as a group to external and internal disturbances while degrading gracefully in their performance. We discussed several axes along which the baseline approach of DALI may be improved, including communication style, modeling, perception, and exploration, as well as the possibilities for a fail-safe system that works as a backup for centralized control. With the current system, we presented a distributed and decentralized control agent for intralogistics that works without centralized assumptions in its process.

REFERENCES

[1] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. 1998. Multi-robot cooperation in the MARTHA project. *IEEE Robotics & Automation Magazine* 5, 1 (1998), 36–47. https://doi.org/10.1109/100.667325

[2] A. Alfeo, E. Ferrer, Y. Carrillo, A. Grignard, L. Pastor, D. Sleeper, M. Cimino, B. Lepri, G. Vaglini, K. Larson, M. Dorigo, and A. Pentland. 2019. Urban Swarms: A new approach for autonomous waste management. In *2019 International Conference on Robotics and Automation (ICRA)*. 4233–4240.

[3] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. 2009. Methodologies for Data Quality Assessment and Improvement. *ACM Comput. Surv.* 41, 3, Article 16 (jul 2009), 52 pages. https://doi.org/10.1145/1541880.1541883

[4] Ana Batinović, Juraj Oršulić, Tamara Petrović, and Stjepan Bogdan. 2020. Decentralized Strategy for Cooperative Multi-Robot Exploration and Mapping. *IFAC-PapersOnLine* 53, 2 (2020), 9682–9687. https://doi.org/10.1016/j.ifacol.2020.12.2618 21st IFAC World Congress.

[5] A. Bazzan and F. Klügl. 2014. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review* 29, 3 (2014), 375–403. https://doi.org/10.1017/S0269888913000118

[6] J. O. Berndt. 2013. Self-Organizing Logistics Process Control: An Agent-Based Approach. In *Agents and Artificial Intelligence*, J. Filipe and A. Fred (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 397–412.

[7] M. Brenner and B. Nebel. 2009. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems* 19, 3 (01 Dec 2009), 297–331. https://doi.org/10.1007/s10458-009-9081-1

[8] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. 2005. Coordinated multi-robot exploration. *IEEE Transactions on Robotics* 21, 3 (2005), 376–386.

[9] Victor Charpenay, Tobias Käfer, and Andreas Harth. 2022. A Unifying Framework for Agency in Hypermedia Environments. In *Engineering Multi-Agent Systems*, Natasha Alechina, Matteo Baldoni, and Brian Logan (Eds.). Springer International Publishing, Cham, 42–61.

[10] Victor Charpenay, Daniel Schraudner, Thomas Seidelmann, Torsten Spieldenner, Jens Weise, René Schubotz, Sanaz Mostaghim, and Andreas Harth. 2021. MOSAIK: A Formal Model for Self-Organizing Manufacturing Systems. *IEEE Pervasive Computing* 20, 1 (2021), 9–18. https://doi.org/10.1109/MPRV.2020.3035837

[11] R. Chisu. 2010. *Kommunikations- und Steuerungsstrategien für das Internet der Dinge.* Dissertation. Technische Universität München. https://mediatum.ub.tum.de/doc/821003/821003.pdf

[12] H.-L. Choi, L. Brunet, and J. P. How. 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics* 25, 4 (2009), 912–926. https://doi.org/10.1109/TRO.2009.2022423

[13] Alysson Ribeiro Da Silva and Luiz Chaimowicz. 2023. Analysis of Opportunistic Interactions for Multi-robot Exploration Under Communication Constraints. In *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*. 242–247. https://doi.org/10.1109/LARS/SBR/WRE59448.2023.10332975

[14] R. de Lemos et al. 2013. *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–32.

[15] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. 1987. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing* (Vancouver, British Columbia, Canada) *(PODC '87)*. Association for Computing Machinery, New York, NY, USA, 1–12.

[16] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes. 1996. A taxonomy for multi-agent robotics. *Autonomous Robots* 3, 4 (01 Dec 1996), 375–397.

[17] E.H. Durfee and V. Lesser. 1991. Partial global planning: a coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics* 21, 5 (1991), 1167–1183. https://doi.org/10.1109/21.120067

[18] J. Fottner, D. Clauer, F. Hormes, M. Freitag, T. Beinke, L. Overmeyer, S. N. Gottwald, R. Elbert, T. Sarnow, T. Schmidt, K.-B. Reith, H. Zadek, and F. Thomas. 2021. Autonomous Systems in Intralogistics – State of the Art and Future Research Challenges. In *Logistics Research.* 14:02. https://doi.org/10.23773/2021_2

[19] S. Giordani, M. Lujak, and F. Martinelli. 2013. A Distributed Multi-Agent Production Planning and Scheduling Framework for Mobile Robots. *Computers & Industrial Engineering* 64 (01 2013), 19–30. https://doi.org/10.1016/j.cie.2012.09.004

[20] J. Harland, D. Morley, J. Thangarajah, and N. Yorke-Smith. 2017. Aborting, suspending, and resuming goals and plans in BDI agents. *Autonomous Agents and Multi-Agent Systems* 31, 2 (01 Mar 2017), 288–331.

[21] O. Holland and C. Melhuish. 1999. Stimergy, Self-Organization, and Sorting in Collective Robotics. *Artificial Life* 5 (04 1999), 173–202.

[22] T. Hubauer, S. Lamparter, M. Roshchin, N. Solomakhina, and S. Watson. 2013. Analysis of data quality issues in real-world industrial data. In *Annual Conference of the PHM Society.*

[23] Y. Imoto, Y. Tsuji, and E. Kondo. 2011. A control method with pheromone information for a transport system. *Artificial Life and Robotics* 16, 1 (01 Jun 2011), 112–115. https://doi.org/10.1007/s10015-011-0899-7

[24] D. N. Kinny and M. P. Georgeff. 1991. Commitment and Effectiveness of Situated Agents. In *Proceedings of the 12th International Joint Conference on Artificial*

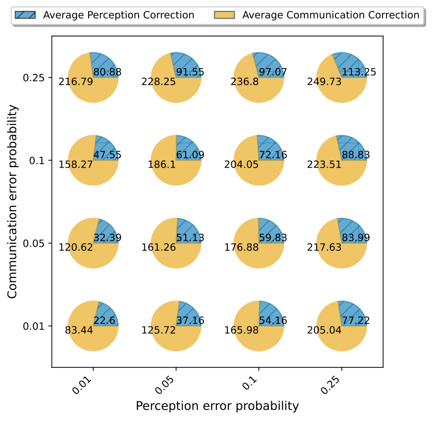

Fig. 17: 100 prod., t_d = 100 (A)

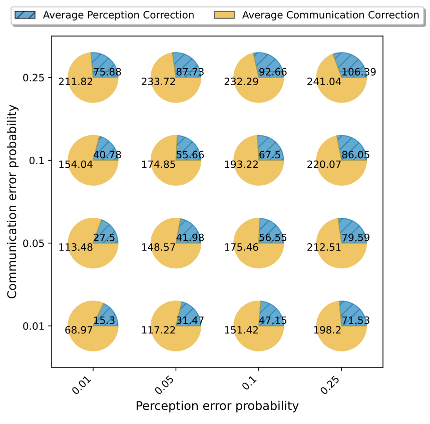

Fig. 18: 100 prod., t_d = 150 (A)

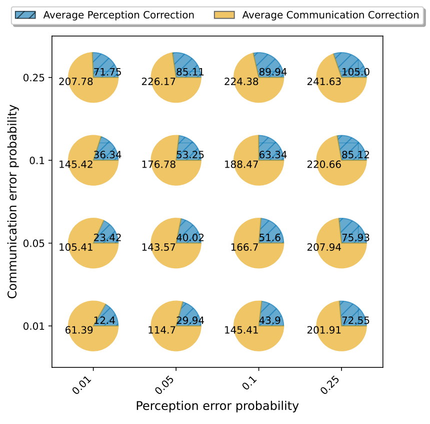

Fig. 19: 100 prod., t_d = 200 (A)

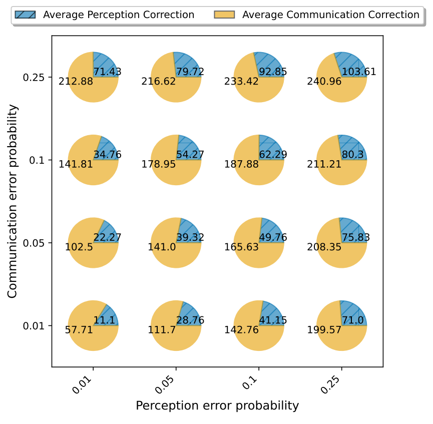

Fig. 20: 100 prod., t_d = 250 (A)

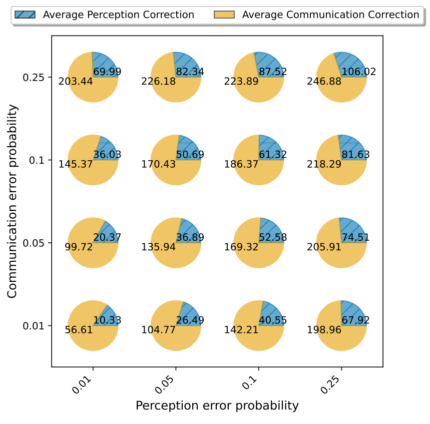

Fig. 21: 100 prod., t_d = 300 (A)

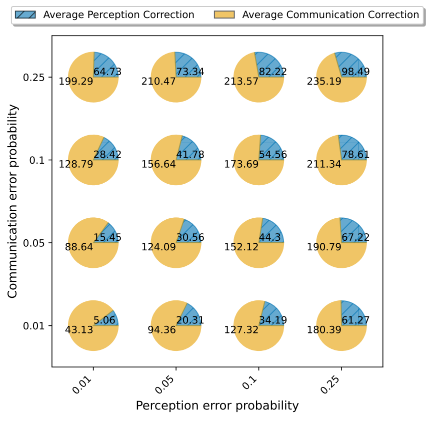

Fig. 22: 100 prod., t_d = $w.o.$ (A)

Intelligence - Volume 1 (Sydney, New South Wales, Australia) *(IJCAI'91)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 82–88.

[25] N. Klein. 2012. *The impact of decentral dispatching strategies on the performance of intralogistics transport systems.* Dissertation. Technische Universität Dresden. https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-147739

[26] L. Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM* 21, 7 (jul 1978), 558–565.

[27] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (July 1982), 382–401. https://doi.org/10.1145/357172.357176

[28] V. Lesser and D. Corkill. 1983. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine* 4, 3 (Sep. 1983), 15. https://doi.org/10.1609/aimag.v4i3.401

[29] A. Omicini, A. Ricci, and M. Viroli. 2008. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 17, 3 (01 Dec 2008), 432–456. https://doi.org/10.1007/s10458-008-9053-x

[30] M. Pease, R. Shostak, and L. Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (April 1980), 228–234. https://doi.org/10.1145/322186.322188

[31] M. E. Pollack and M. Ringuette. 1990. Introducing the Tileworld: Experimentally Evaluating Agent Architectures. In *AAAI Conference on Artificial Intelligence.*

[32] V. Robu, H. Noot, H. La Poutré, and W.-J. van Schijndel. 2008. An Interactive Platform for Auction-Based Allocation of Loads in Transportation Logistics. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track* (Estoril, Portugal) *(AAMAS '08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 3–10.

[33] S. Russell and P. Norvig. 2009. *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall Press, USA.

[34] M. Savelsbergh and M. Sol. 1995. The General Pickup and Delivery Problem. *Transportation Science* 29 (02 1995), 17–29. https://doi.org/10.1287/trsc.29.1.17

[35] S. Schmid and A. Harth. 2022. Decentralized Self-Adaption With Epidemic Algorithms for Agent-Based Transportation. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 99–104.

[36] Sebastian Schmid and Andreas Harth. 2024. Enabling Adaptation in Dynamic Manufacturing Environments with Decentralized Agent-Based Systems and Local Perception. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing* (Avila, Spain) *(SAC '24)*. Association for Computing Machinery, New York, NY, USA, 235–242. https://doi.org/10.1145/3605098.3635967

[37] S. Schmid, D. Schraudner, and A. Harth. 2021. Performance Comparison of Simple Reflex Agents Using Stigmergy with Model-Based Agents in Self-Organizing Transportation. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. 93–98.

[38] S. Schmid, D. Schraudner, and A. Harth. 2023. MOSAIK: An Agent-Based Decentralized Control System with Stigmergy for a Transportation Scenario. In *The Semantic Web*, C. Pesquita, E. Jimenez-Ruiz, J. McCusker, D. Faria, M. Dragoni, A. Dimou, R. Troncy, and S. Hertling (Eds.). Springer Nature Switzerland, Cham, 697–714.

[39] T. Schmidt, K.-B. Reith, N. Klein, and M. Däumler. 2020. Research on Decentralized Control Strategies for Automated Vehicle-based In-house Transport Systems – a Survey. In *Logistics Research*. 13:10. https://doi.org/10.23773/2020_10

[40] J. Schuhmacher and V. Hummel. 2018. Development of a descriptive model for intralogistics as a foundation for an autonomous control method for intralogistics systems. *Procedia Manufacturing* 23 (2018), 225–230. "Advanced Engineering Education & Training for Manufacturing Innovation"8th CIRP Sponsored Conference on Learning Factories (CLF 2018).

[41] A. Schuldt. 2012. Multiagent Coordination Enabling Autonomous Logistics. *KI - Künstliche Intelligenz* 26, 1 (01 Feb 2012), 91–94.

[42] VDA5050 2020. *VDA 5050 - Interface for the communication between automated guided vehicles (AGV) and a master control.* Standard. Verband der Automobilindustrie e.V. (VDA), Berlin.

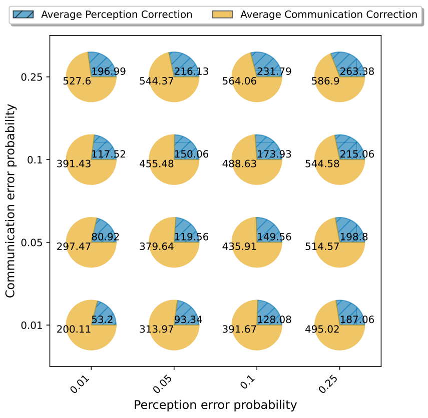

Fig. 23: 250 prod., $t_d = 100$ (A)

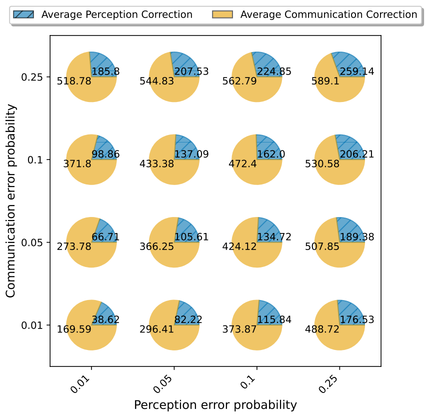

Fig. 24: 250 prod., $t_d = 150$ (A)

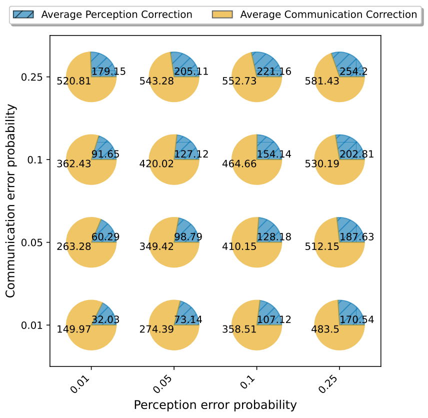

Fig. 25: 250 prod., $t_d = 200$ (A)

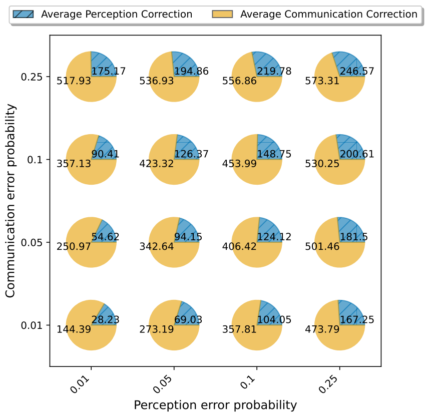

Fig. 26: 250 prod., $t_d = 250$ (A)

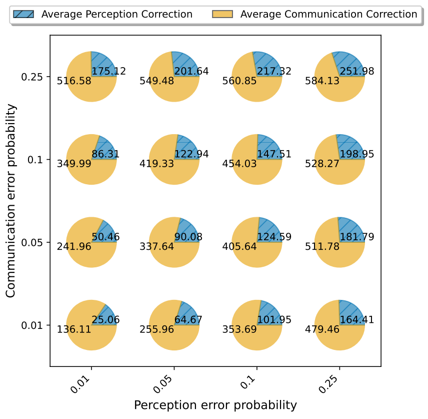

Fig. 27: 250 prod., $t_d = 300$ (A)

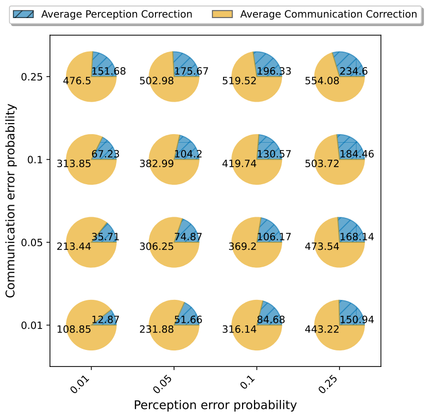

Fig. 28: 250 prod., $t_d = w.o.$ (A)

[43] W. Xiang and H.P. Lee. 2008. Ant colony intelligence in multi-agent dynamic manufacturing scheduling. *Engineering Applications of Artificial Intelligence* 21, 1 (2008), 73–85. https://doi.org/10.1016/j.engappai.2007.03.008

[44] B. Yamauchi. 1997. A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'* (1997), 146–151.

[45] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino. 2017. Multi-Agent Foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling* 5, 1 (02 Feb 2017), 3. https://doi.org/10.1186/s40294-016-0041-8

[46] K. Zhang, Z. Yang, and T. Başar. 2021. *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms.* Springer International Publishing, Cham, 321–384. https://doi.org/10.1007/978-3-030-60990-0_12

[47] Boyu Zhou, Hao Xu, and Shaojie Shen. 2022. RACER: Rapid Collaborative Exploration with a Decentralized Multi-UAV System. arXiv:2209.08533 [cs.RO] https://arxiv.org/abs/2209.08533