ZEROSWARM

D4.3 Self-learning modules for robotic and human behaviours.

ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems

Project funded by Horizon Europe, Grant Agreement #101057083

Topic	HORIZON-CL4-2021-TWIN-TRANSITION-01-08
Project Title	ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems
Project Number	101057083
Project Acronym	Zero-SWARM
Contractual Delivery Date	M15
Actual Delivery Date	M16
Contributing WP	WP4
Project Start Date	01/06/2022
Project Duration	30 Months
Dissemination Level	Public
Editor	RWG
Contributors	all

Author List

Leading Author (Editor)			
Surname	Initials	Beneficiary Name	Contact email
Maccioni	RM	RWG	raffaele.maccioni@rwings.tech
Co-authors (in alphabetic order)			
Surname	Initials	Beneficiary Name	Contact email
Lepore	LM	RWG	mario.lepore@mathbiology.tech
Serra	DS	RWG	domenico.serra@modelite.tech
Contributors (in alphabetic order)			
Surname	Initials	Beneficiary Name	Contact email
Chatzidiamantis	NC	CERTH	nestorash@iti.gr
Deshmukh	SD	NX-SE	shreya.deshmukh@se.com
Fritz	AF	NX-SE	artur.fritz@se.com
Lazaridis	GL	CERTH	glazaridis@iti.gr
Mpatziakas	SM	CERTH	ampatziakas@iti.gr

Reviewers List

List of reviewers (in alphabetic order)			
Surname	Initials	Beneficiary Name	Contact email
Drosou	AD	CERTH	drosou@iti.gr
Khodashenas	PK	HWE	pouria.khodashenas@huawei.com
Liakh	TL	LTU	tatiana.liakh@ltu.se
Mendes	BM	UBI	bmendes@ubiwhere.com
Santiago	RS	UBI	rsantiago@ubiwhere.com
Simeao	HS	UBI	hsimeao@ubiwhere.com

Document History

Document History			
Version	Date	Author	Remarks
00.00	15/02/2023	R Maccioni	Table of Content
00.02	29/04/2023	R Maccioni	Introduction + Chapter 2: practical scenarios and application - draft version
00.03/ 00.14	12/06/2023	R Maccioni	Intermediate revision with partners contributions
00.15	30/08/2023	R Maccioni	Release for Reviewers
00.16	18/09/2023	D Serra	Changes after review
1.0	22/09/2023	A Drosou	Final submission

DISCLAIMER OF WARRANTIES

This document has been prepared by Zero-SWARM project partners as an account of work carried out within the framework of the contract no 101057083.

Neither Project Coordinator, nor any signatory party of Zero-SWARM Project Consortium Agreement, nor any person acting on behalf of any of them:

- makes any warranty or representation whatsoever, express or implied,
    - with respect to the use of any information, apparatus, method, process, or similar item disclosed in this document, including merchantability and fitness for a particular purpose, or
    - that such use does not infringe on or interfere with privately owned rights, including any party's intellectual property, or
- that this document is suitable to any particular user's circumstance; or
- assumes responsibility for any damages or other liability whatsoever (including any consequential damages, even if Project Coordinator or any representative of a signatory party of the Zero-SWARM Project Consortium Agreement, has been advised of the possibility of such damages) resulting from your selection or use of this document or any information, apparatus, method, process, or similar item disclosed in this document.

Zero-SWARM has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101057083. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

Executive Summary

The concept of self-learning in the context of robotic and human behaviors is paramount in today's manufacturing landscape.

These self-learning modules empower autonomous robots, such as Automated Mobile Robots (AMRs), to acquire knowledge from their own experiences, enabling them to adapt and optimize their behavior in real-time. Moreover, such techniques and technologies might be used to capitalize on the expertise of human operators, augmenting their performances and interactions with robots.

Machine Learning algorithms and intelligent agents to support modern manufacturing and automated intra-logistics processes should come with a distribution in the cloud-to-edge continuum of the intelligent agents, which also include math-optimization and near-real-time simulation models.

Implementing self-learning modules and distributing intelligent agents also present certain challenges, such as managing the asynchronous learning process or making math optimization compatible with real-time scenarios.

The document analyzes such aspects, defining the novel zero-Swarm approach for learning processes and Intelligent agent deployment within the Edge-Cloud continuum to offer a strategic advantage by capitalizing on the strengths of localized Edge computing and robust Cloud resources. The primary challenge lies in effectively implementing scalability through the concept of federated learning. Additionally, there's a pressing need to reduce computational times for mathematical optimization to enable near-real-time decision-making. Starting from state-of-the-art, we introduce key novelties to address a robust and responsive application in real industrial contexts of cloud-to-edge concepts and asynchronous learning. Considering the use cases, in particular the AMR fleet management, we address the following key novel aspects:

- Define a concise guideline for managing asynchronous learning and distribution of Machine Learning algorithms in the Cloud-to-Edge;
- enable federated learning processes, optimizing communication overhead in manufacturing contexts.
- Combine the use of learning processes and math options. In particular, the use of metamodels to support the adoption of near-real-time optimization algorithms and overcome the limit of math optimization in terms of computational effort. When embedded in this cloud-to-edge architecture, these models can ensure optimal resource utilization, cost efficiency, and reduced latency, respecting the needs of near-real-time or even real-time decision-making.
- Combine the use of learning processes and simulation models to bring a simulator to the shop floor, again for near-real-time use of simulation models.

For the nature of the use-cases considered as a stimulus of this deliverable, we mainly refer to robots, but the methodological approach and the architecture also remain valid in the cases of data coming from human beings equipped, for example, with wearable devices.

Contents

List of Acronyms

Acronym	Description
5G	Fifth-Generation Wireless Communications
AE	Alarm And Events
AFoF	AALTO Factory of the Future
AGV	Autonomous Guided Vehicles
AI	Artificial Intelligence
AIC	Automotive Intelligence Centre
AIIC	AALTO Industrial Internet Campus
AR	Augmented Reality
CAT	Composite Automation Type
CPSoS	Cyber-Physical System of Systems
CUC	Centralized User Configuration
DA	Data Access
DFA	Demonstration Factory Aachen (DFA)
DLFi	Distributed Learning Framework
DSS	Dss Dynamic Spectrum Allocation
E2E	End-To-End
eMBB	Enhanced Mobile Broadband
gPTP	Generalized Precision Time Protocol
HDA	Historical Data Access
ICT	Information Communication Technologies
IDS	International Data Spaces
IICF	Industrial Internet Connectivity Framework
IIoT	Industrial Internet Of Things
IIRA	Industrial Internet Reference Architecture
IMC	Intelligent Mechatronic Components
ITU	International Telecommunication Union
LBO	And Local Breakout
MAS	Multi-Agent Systems
MES	Manufacturing Execution Systems
mIoT	Massive Internet of Things
mMTC	Massive Machine Type Communication
MR	Mixed Reality

NASA	National Aeronautics and Space Administration NASA
NASA	Telematic Data Collector
NPN	Non-Public Networks
OLE	Object Linking and Embedding
OPC	Open Platform Communication
OPC-UA	Opc Unified Architecture
PLC	Programmable Logic Controllers
PTZ	Production Technology Center (PTZ)
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
SNPNs	Stand-Alone Npns
SOAP	Simple Object Access Protocol
SoS	Systems Of Systems
TSN	Time Sensitive Networking
UA	Unified Architecture
UACP	Unified Architecture Connection Protocol
UES	User Equipment
UPF	User Plane Function
uRLLC	Ultra-Reliable Low Latency Communication
VR	Virtual Reality
AMR	Autonomous Mobile Robots
ANN	Artificial Neural Network
OR	Operation Research
AI	Artificial Intelligence
ID	Identifier
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
KPI	Key Performance Indicator
ML	Machine Learning
NaNs	Not a Numbers
H-AMR-SP-BWC	Heterogeneous AMR Scheduling Problem with Battery and Weight constraints
VRP	Vehicle Routing Problem
AMR-IRP	AMR Item Routing Problem
Ptr-Net	Pointer Network
DT	Digital Twin
RNN	Recurrent Neural Network

List of Figures

List of Tables

1 Introduction

1.1 Purpose of the document

In the era of artificial intelligence and interconnected systems, the optimization of predictive models and intelligent agents within the cloud-to-edge architecture is gaining substantial significance.

This deliverable delves into the concept of intelligent agents sustained by asynchronous learning processes, exploring its practical applications and guidelines considering the entire zero-Swarm framework.

Through the lens of general real cases and the zero-warm use cases, this document navigates the complexities of distributing machine learning algorithms and intelligent agents, facilitating efficient learning and decision-making.

Generally, the prediction concept has a wide practical meaning, depending on the context, the processes, and the type of decision.

In real and near-real-time decision-making, predicting the future might require different techniques or a combination of methods. In the decision-making cycle by intelligent agents, prediction represents one of the components, and machine learning has a significant role, often combined with statistical data representation, math optimization tools, and simulation models. For example, to predict the system's status, represented by certain variables, it could be needed to apply the same logic as in the real world will be used; an optimization model could express such logic, or a simulation model could predict the effect of a certain decision in a certain instant.

In the document, we explore such a scenario considering the cloud-to-edge continuum.

Automated Mobile Robots (AMRs), considered in the trials, offer a good conceptual and practical context for the deliverable. AMRs are versatile robotic systems that play a crucial role in various industries. These robots are capable of autonomous movement and can perform a wide range of tasks, making them suitable for applications in logistics, manufacturing, and beyond. AMRs can be designed in different forms, serving as either Automated Guided Vehicles (AGVs) or drones, depending on the specific use case and requirements.

We explore common principles and challenges regarding the Asynchronous Learning concepts to deploy intelligent agents, also considering the combination of machine learning and math optimization to enable real and near-real-time decision-making.

1.2 Structure of the document

The document is structured as follows:

- **Chapter 1** introduction and scope of the document;
- **Chapter 2** This chapter sets the stage for asynchronous learning. This chapter defines its fundamental concepts and highlights its relevance in the cloud-to-edge continuum. We examine practical scenarios where asynchronous learning and distributed machine learning algorithms/intelligent agents offer valuable solutions, enhancing efficiency and adaptability. The section Guidelines for Management provides essential guidelines for effectively managing

the asynchronous learning process and the distribution of machine learning algorithms within the cloud-to-edge architecture.

- **Chapter 3** in this chapter we dive into the data model overview, specifically focusing on the application in AMR Fleet Management. This includes data exchange mechanisms and sequence diagrams. The Efficient Predictive Models section, explores the creation of intelligent agents capable of efficient predictive modeling. The chapter covers the process, from data collection to model construction, using the AMR context. The Federation Learning section Uncovering the opportunities, methods, and implications of federation learning within the cloud-to-edge continuum. On the other side, the Optimizing Communication Overhead section, provides insights into optimizing communication overhead, ensuring seamless data transfer and decision-making efficiency.
- **Chapter 4** the chapter is a comprehensive overview of the scheduling algorithm within the cloud-to-edge architecture, facilitating optimal resource allocation and task management. The Chapter includes a discussion on the routing algorithm's role in guiding intelligent agents and predictive models for efficient decision-making.

  It includes an exploration of intelligent agents powered by math optimization and machine-learning-based metamodels. The Section Digital Twin and Simulation Highlights the synergies between asynchronous learning and digital twin/simulation technologies, offering new dimensions of optimization.
- **Chapter 5:** conclusions
- **Chapter 6:** references

2 Asynchronous learning of predictive models & intelligent agents distributions

In the rapidly evolving landscape of artificial intelligence, the synchronous training of models (e.g., predictive) can often be restrictive, especially in environments where continuous data influx and real-time adaptations are imperative. This chapter delves into the innovative realm of asynchronous learning, providing insights into its applications. This chapter's contents also constitute a backbone for applying the metamodels described in Chapter 4.

2.1 Introductory Elements

Cloud-to-Edge architecture offers several advantages in terms of reduced latency [1], bandwidth optimization, and enhanced privacy and security. However, it also presents challenges related to scalability, algorithm distribution and management, data synchronization and consistency, and network connectivity and reliability, which need to be carefully addressed to achieve optimal system performance.

Cloud-to-Edge architecture refers to a distributed computing model where data processing and computation are performed both in the cloud (centralized data centers) and at the edge (devices or nodes at the network's periphery) [2]. This approach has several advantages in certain scenarios:

Reduced Latency: Edge devices are geographically closer to the data source, which can significantly reduce the latency in processing time-sensitive data. For applications that require real-time or near-real-time responses, such as industrial automation, autonomous vehicles, or augmented reality, Cloud-to-Edge architecture can provide substantial performance benefits by minimizing data transfer delays and improving overall system responsiveness.

Bandwidth Optimization: Cloud-to-Edge architecture can help optimize network bandwidth by processing data at the edge before transmitting it to the cloud. This reduces the amount of data that needs to be transmitted to the cloud, thereby reducing the bandwidth requirements and associated costs. This can be especially beneficial in scenarios where bandwidth is limited or expensive, such as in remote or rural areas.

Enhanced Privacy and Security: Edge devices can process sensitive data locally without transmitting it to the cloud, which can help address privacy and security concerns. By keeping data locally on edge devices, Cloud-to-Edge architecture can mitigate risks associated with data breaches, unauthorized access, and compliance issues, as data is not transmitted over the network or stored in a centralized cloud.

However, there are also **crucial issues** that can impact the performance of Cloud-to-Edge architecture:

Scalability: Edge devices typically have limited computational resources compared to the cloud, which can impact the scalability of the system. Complex or resource-intensive applications may face limitations in terms of processing capabilities, memory, and storage at the edge. This can require careful distribution of tasks and workload management to ensure efficient utilization of resources.

Algorithm Distribution and Management: Deciding which algorithms should be deployed at the edge and which should be executed in the cloud can be challenging. Some algorithms may require significant computational resources and may be better suited for cloud execution, while others may need low-latency processing and are more suitable for edge execution. Efficiently managing the distribution of algorithms across the cloud and edge, and dynamically adapting the distribution based on changing conditions, can impact the overall system performance.

Data Synchronization and Consistency: In Cloud-to-Edge architecture, data may be processed and stored at different locations, leading to challenges in data synchronization and consistency. Ensuring that data remains consistent across cloud and edge, and managing data updates, caching, and versioning, can be complex and impact system performance.

Network Connectivity and Reliability: Cloud-to-Edge architecture relies heavily on network connectivity between the cloud and edge devices. Unreliable or intermittent network connections can disrupt data processing, result in delays, and impact system performance. Ensuring reliable and robust network connectivity, especially in remote or harsh environments, is crucial for the success of Cloud-to-Edge architecture.
Talking about predictive models and intelligent agents, it is necessary to clarify the application and the relation between different techniques under the "artificial intelligence" umbrella.

Let's start by asserting that many discussions are still open about where artificial intelligence starts and if techniques such as statistics and operations research are, or not, part of artificial intelligence. Surely, they are strictly connected, an intelligent agent might use one or more analytical techniques, and with increasing complexity, different techniques are combined.
OR and statistics are related to AI, even if they are distinct fields with their techniques and methods. Recent research has shown that integrating these fields with AI can significantly improve performance and efficiency.
Operations Research is a field that uses mathematical models and analytical methods to make better decisions. For example, OR techniques are used in AI to optimize machine learning algorithms, resource allocation, scheduling, and other tasks [3].
On the other hand, statistics is a branch of mathematics that deals with data collection, analysis, interpretation, presentation, and organization. Statistics is used in AI to help machines learn from data, identify patterns, and make predictions.
Recent research in AI has focused on integrating OR and statistics into machine learning algorithms to improve their performance [3]. For example, OR techniques permit optimization of the allocation of computing resources in deep learning networks, and statisticians have developed new algorithms for training models on large datasets.

Generally, we could distinguish:

- agents able to predict the future of certain variables,
- agents able to classify events and scenarios,
- agents able to establish the optimal or near-optimal value of a control or decisional variables,
- agents able to reproduce an approximated behavior (simulation or emulation) of physical or Cyber-Physical systems,
- agents able to create contents and scenarios or data set

Such agents might interact to provide a solution to a specific problem, for example the prediction might feed an optimization model to optimally plan resources for the next time windows, a stochastic simulation agent might be used as a black-box for an optimization agent, or a machine-learning agent could be trained based on data generated by a simulation agent so as to obtain a faster meta-model of the simulation.
Obviously, which techniques to use and the advantages that could be obtained are strictly connected with the problem to solve, the nature of the data, and the time response requirement.

Anyway, considering the zero-Swam goal, as a platform and as a method to manage, simulate, optimize CPSoS (Cyber Physical System of Systems) we must consider that any type of combination of the above-mentioned agents and learning process could be part of the solutions. One of the scenarios is represented by the need of Asynchronous learning process and distribution of intelligent agents.

If we consider real cases, such as those used as trials in the zero-swarm projects, we could couple the above generic type of agents with the following practical needs of modern management of CPSoS.

Agents able to predict the future of certain variables.

Such types of agents can, for example, provide the prediction of a workload that will impact resources such as workcenters, or handling robots, operators.
Different types of prediction agents can be used to predict the probability of failures within a certain time period or the expected battery discharge curve of single AGV, like in the use-case.

Agents able to classify events and scenarios.
Such agents can be used, for example, to classify images of products establishing different levels of quality, or to sort items to be treated by different type of machines and processes.
These types of agents can also be used to create dynamic quality control plans.

Agents able to establish the optimal or near-optimal value of a control or decisional variables.
This agent can be used to schedule resources, such as work centers, or a fleet of AGV/AMR (like in the Re-Pack use-case). Can also be used to optimize processes such as cutting, or optimize operations such as picking and sorting.
Agents able to reproduce an approximated behavior (simulation or emulation) of physical or Cyber-Physical systems.
Such agents can reproduce with a certain level of accuracy the behavior of a system and process. For example, acting as a digital twin, or a simulation model to predict bottle necks. This is the case of the digital twin and the simulation model in the zero-Swarm use-case.

Agents able to create contents and scenarios or data set
Such intelligent agents might generate sets of data and information to support different needs and processes. For example, generate the list of items to inspect.

Some of the described agents require a learning process before being used and distributed.
The learning process might benefit from data produced and collected by different Systems, for example, the Predictive Maintenance machine learning related to similar work centers or robots is trained on data from all such devices. Once trained, the model, depending on the case, can be run on the cloud or the edge, depending on the specific application. For example, a model predicting the battery discharging curve trains on the cloud could be deployed directly on board to a single device on the central navigation control system.

This means that the zero-Swarm platform must enable a pretty wide selection of cases in terms of intelligent agents and learning processes, without losing generality, enabling the cloud-to-edge advantages, and limiting the related crucial issues mentioned above.

2.2 Practical cases where the asynchronous learning processes and distributing machine learning algorithms / intelligent agents in a Cloud-to-Edge architecture can be useful.

The ability to process data locally at the edge and update models asynchronously can enable intelligent decision-making, optimize production processes, improve product quality, enhance supply chain

management, and ensure worker safety, leading to increased productivity, reduced costs, and improved overall operational efficiency in manufacturing environments.

Manufacturing is one of the domains where Cloud-to-Edge architecture and the use of asynchronous learning processes and distributed machine learning algorithms can be highly beneficial.
Following some practical examples of how asynchronous learning processes and distributed machine learning algorithms in a Cloud-to-Edge architecture can be applied in manufacturing.

Predictive maintenance: Manufacturing equipment, such as production lines, robots, and conveyor belts, generate vast amounts of data that can be used for predictive maintenance.
By deploying machine learning algorithms at the edge devices or gateways in a Cloud-to-Edge architecture, data can be processed locally in near real-time to detect anomalies, predict failures, and trigger maintenance actions. Asynchronous learning processes can allow edge devices to learn from local data and update their models asynchronously, considering the dynamic nature of the data and the changing conditions of the machines.

This can help in reducing downtime, improving equipment performance, and optimizing maintenance schedules, leading to increased productivity and cost savings. In industrial settings, machines and equipment often generate large amounts of data that can be used to predict their maintenance needs.
Quality control: In manufacturing, ensuring product quality is critical. Edge devices such as cameras, sensors, and inspection systems can generate data related to product quality attributes, such as dimensions, color, and surface defects. By deploying machine learning algorithms at the edge devices in a Cloud-to-Edge architecture, data can be processed locally to perform real-time quality control checks, identify defects, and trigger rejections or rework. Asynchronous learning processes can allow edge devices to learn from local quality data and update their models asynchronously to adapt to changing quality requirements, leading to improved product quality and reduced waste.

Production optimization: Optimizing production processes is essential for maximizing efficiency and reducing costs in manufacturing. Edge devices, such as sensors, actuators, and controllers, can generate data related to production parameters, such as temperature, pressure, and speed. By deploying distributed machine learning algorithms in a Cloud-to-Edge architecture, data can be processed at the edge to optimize production processes, such as scheduling, resource allocation, and energy management. Asynchronous learning processes can allow edge devices to learn from local production data and update their models asynchronously to adapt to changing production conditions, leading to improved production efficiency and cost savings.

Supply chain management: Managing the supply chain is critical in manufacturing to ensure timely delivery of materials and components. Edge devices, such as RFID tags, barcode scanners, and tracking systems, can generate data related to supply chain events, such as shipments, deliveries, and inventory levels. By deploying machine learning algorithms at the edge devices in a Cloud-to-Edge architecture, data can be processed locally to optimize supply chain management, such as demand forecasting, inventory optimization, and route optimization. Asynchronous learning processes can allow edge devices to learn from local supply chain data and update their models asynchronously to adapt to changing supply chain dynamics, leading to improved supply chain efficiency and cost savings.

Worker safety: Ensuring worker safety is a top priority in manufacturing. Edge devices, such as wearable devices, cameras, and sensors, can generate data related to worker safety, such as location, movement, and biometrics. By deploying machine learning algorithms at the edge devices in a Cloud-to-Edge architecture, data can be processed locally to detect safety hazards, predict safety risks, and trigger timely interventions. Asynchronous learning processes can allow edge devices to learn from local safety data and update their models asynchronously to adapt to changing worker safety requirements, leading to improved worker safety and reduced accidents.

Following some **additional practical use cases** where asynchronous learning processes and distributed machine learning algorithms in a Cloud-to-Edge architecture can be useful, also considering the 5G technologies as an enabler and constitute **contiguous application areas** for the zero-Swarm platform.

Intelligent transportation systems: In transportation systems, edge devices such as vehicles, traffic lights, and road sensors generate data that can be used to optimize traffic flow, improve road safety, and enable autonomous driving. By deploying distributed machine learning algorithms in a Cloud-to-Edge architecture, data can be processed at the edge to make real-time decisions on traffic management, vehicle routing, and collision detection. Asynchronous learning processes can allow edge devices to learn from local traffic patterns and update their models asynchronously to adapt to changing traffic conditions.

Healthcare monitoring: In remote healthcare monitoring scenarios, wearable devices, sensors, and IoT devices can generate continuous streams of data related to patient health, such as heart rate, blood pressure, and temperature. By deploying machine learning algorithms at the edge devices in a Cloud-to-Edge architecture, data can be processed locally to detect anomalies, predict health risks, and trigger timely interventions. Asynchronous learning processes can allow edge devices to learn from local patient data and update their models asynchronously to personalize healthcare monitoring and improve patient outcomes.

Environmental monitoring: In environmental monitoring applications, sensors and IoT devices can generate data related to air quality, water quality, weather conditions, and other environmental parameters. By deploying distributed machine learning algorithms in a Cloud-to-Edge architecture, data can be processed at the edge to monitor environmental conditions, detect pollution events, and trigger early warnings. Asynchronous learning processes can allow edge devices to learn from local environmental data and update their models asynchronously to adapt to changing environmental conditions.

Smart cities: In smart city applications, edge devices such as street lights, waste management systems, and parking sensors generate data that can be used to optimize resource allocation, reduce energy consumption, and improve city services. By deploying machine learning algorithms at the edge devices in a Cloud-to-Edge architecture, data can be processed locally to make real-time decisions on resource allocation, traffic management, and waste management. Asynchronous learning processes can allow edge devices to learn from local data and update their models asynchronously to adapt to changing city dynamics.

2.3 Guidelines to manage asynchronous learning process and distribution of Machine Learning algorithms in the Cloud-to-Edge

The following steps represent guidelines to implement in the zero-Swarm architecture asynchronous machine learning algorithms with a distribution of the models (agents) in the Cloud-to-Edge.

Decision Making and Process needs analysis: The core and fundamental steps are an exhaustive understanding and definition of the process and decision-making needs, including the point of view of humans, systems (robots, working machines, etc,) and their interactions.
Such analysis must consider the input and output variables and the understanding of timing requirements that could influence the type of techniques to use. When models provide a prediction, another key element is the accuracy level needed for robust decision-making.

Choose appropriate asynchronous learning algorithms: Asynchronous learning algorithms are designed to handle distributed and asynchronous data sources, making them suitable for Cloud-to-Edge architectures where data may be generated at different edge devices or gateways asynchronously. Examples of such algorithms include Federated Learning, Asynchronous Stochastic Gradient Descent (ASGD), and Parameter Server-based approaches. Choose algorithms that are well-suited for the distributed and asynchronous nature of the data in your Cloud-to-Edge architecture.

Optimize communication overhead: In a Cloud-to-Edge architecture, data communication between the cloud and the edge can be a bottleneck due to limited bandwidth or high latency. It's important to optimize communication overhead to minimize the impact on the overall system performance. One approach is to use compression techniques or data aggregation methods to reduce the amount of data that needs to be transmitted between the cloud and the edge. Another approach is to leverage edge processing capabilities to preprocess data locally and reduce the amount of data that needs to be transmitted.

Utilize edge computing resources effectively: Edge devices or gateways in a Cloud-to-Edge architecture typically have limited computing resources compared to the cloud. It's important to utilize these resources effectively to minimize the impact on local operations and ensure smooth functioning of edge devices. This may involve offloading some processing tasks to the cloud, distributing the workload across multiple edge devices, or optimizing the computation and memory usage of machine learning algorithms at the edge.

Implement robust error handling and fault tolerance mechanisms: In a distributed Cloud-to-Edge architecture, failures or errors can occur at various stages, including data acquisition, communication, processing, and storage. It's crucial to implement robust error handling and fault tolerance mechanisms to ensure the reliability and availability of the system. This may involve implementing data redundancy, error detection and correction mechanisms, retry mechanisms, and graceful degradation strategies to handle failures or errors in a distributed and asynchronous environment.

Secure data transmission and storage: In a Cloud-to-Edge architecture, data transmission and storage can be vulnerable to security threats, such as data breaches or unauthorized access. It's essential to

implement appropriate security measures, such as encryption, authentication, and authorization mechanisms, to protect data during transmission and storage. This may involve using secure communication protocols, encrypting data at rest and in transit, and implementing access controls to ensure that only authorized parties can access the data.

Monitor and manage the system: Monitoring and managing the Cloud-to-Edge architecture is critical to ensure its smooth operation and performance. Implement monitoring mechanisms to track the status of edge devices, data transmission, and machine learning processes. Utilize logging, metrics, and analytics to gain insights into the system's behavior, identify performance bottlenecks, and optimize the system accordingly. Implement management mechanisms to deploy, configure, and update machine learning algorithms and models across the cloud and edge devices as needed.

Test and iterate: As with any complex system, it's important to continuously test and iterate the Cloud-to-Edge architecture to identify and address potential issues or limitations. Conduct thorough testing, validation, and performance evaluation of the system to ensure its effectiveness and efficiency. Use feedback from real-world deployments and user interactions to improve the system and optimize its performance over time.

3 Predictive Models and Intelligent Agents

In the manufacturing context, intelligent agents are crucial for optimizing the performances, increasing the flexibility and the safety of single equipment and entire systems.

Incorporating intelligent agents foster greater efficiency, quality, adaptability, and innovation, for example, intelligent agents enable:

Autonomous Manufacturing: Intelligent agents enable autonomous decision-making in manufacturing processes, reducing the need for human intervention.

They can schedule tasks, allocate resources, and adjust production parameters based on real-time data.

Intra-logistics Optimization: Intelligent agents optimize material handling and transportation within the manufacturing facility. They can plan efficient routes for automated guided vehicles (AGVs) or drones, reducing delays and enhancing throughput.

Robotic Co-robots Collaboration: Collaborative robots (co-robots) work alongside human workers, enhancing productivity and safety. Intelligent agents coordinate interactions between co-robots and human operators, ensuring smooth cooperation.

Process Monitoring and Control: Intelligent agents monitor manufacturing processes in real-time, identifying deviations from optimal conditions. They can adjust parameters to maintain consistent quality and efficiency.

Quality Control and Assurance: Intelligent agents employ computer vision and sensor data to inspect products for defects and variations. They facilitate real-time quality control, reducing waste and enhancing product quality.

Predictive Maintenance: Agents analyze data from sensors embedded in machines to predict when

maintenance is needed. This prevents unexpected breakdowns, reduces downtime, and extends equipment lifespan.

Data-Driven Decision Making: Intelligent agents process vast amounts of data to provide insights and recommendations, enabling continuous autonomous learning. Manufacturers can make informed decisions to optimize processes, improve efficiency, and reduce costs.

Inventory and Supply Chain Optimization: Agents use data to optimize inventory levels, demand forecasting, and supplier collaboration. This leads to a more efficient supply chain, reducing lead times and ensuring timely production.

Energy Efficiency: Intelligent agents monitor energy consumption and adjust equipment settings to minimize energy usage. This contributes to sustainable manufacturing practices and cost savings.

Customization and Flexibility: Intelligent agents enable agile manufacturing by adapting production processes for customized products. Manufacturers can respond quickly to changing customer demands and market trends.

Continuous Improvement: Agents gather data on process performance and identify areas for improvement. This fosters a culture of continuous improvement and innovation within the manufacturing environment.

Intelligent agents are entities equipped with the ability to perceive their environment, make decisions, and take actions in pursuit of goals. These agents leverage various technologies, including artificial intelligence and machine learning, to enhance their decision-making capabilities.

Intelligent agents enable smarter decision-making and process control. In the contest of zero-Swarm, they have a relevant role soliciting core technological and methodological aspects such as:

- the distribution of intelligence;
- computational performances, in particular, to make decision-making coherent with real-time and near-real time;
- the empowerment of the CPSoS.

Intelligent agents combine the power of artificial intelligence, in particular machine learning, math optimization (operations research), and statistics to create systems that can autonomously operate, navigate their environments and interact, learn from experience, and make optimal decisions to achieve desired outcomes.

The design of intelligent CPSoS requires finding the correct trade-off between computational performance and effectiveness, both, at the level of the single agent (logic/ algorithms) and at the level of multiple agents cooperating for a common goal. This is particularly needed in the presence of real-time or near-real-time decision-making.

There is no predefined receipt for the techniques to use. The process and system characteristics, the governance objectives, the amount of data involved, and the reaction-time are all elements to consider identifying the most appropriate technique to adopt for engineering and implementing the intelligent components.

In the 4.3 deliverable, considering the needs of the use-cases, we provide a practical example of Intelligent Agents for the fleet management of AMR, for the optimization of the jobs list and for

predictive maintenance. The cases solicit the different aspects above mentioned.

3.1 Data Model Description - the case of AMR Fleet Management

The following section describes the data model supporting the intelligent agents related to the AGV/AMR scheduler and predictive Maintenance.

The following ER diagram shows the entities and their relationships.

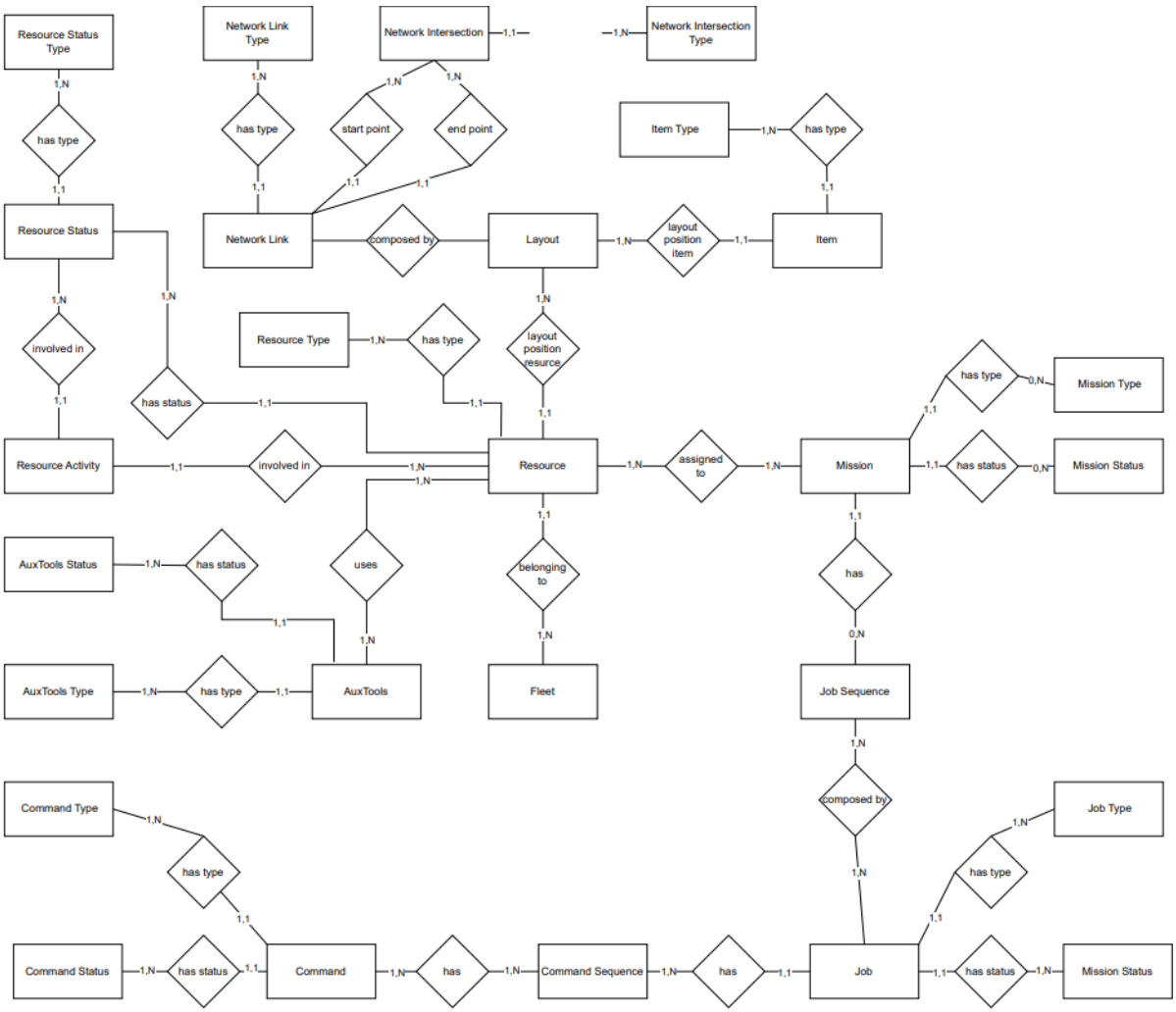

Figure 1 ER diagram

Designing the data model, considering the CPSoS paradigm, the needs of the specific intelligent agent, the data different application layers should partially share, requires to:

Limit the quantity of data a single application layer or component needs to perform its tasks and to interact with others.

Design data objects in the most natural way to be used by the intelligent component, but at the same time, consider the most natural way for interacting with other services/applications.

Design the data model coherently to the domain ontology.

Consider data-handling performances compared to the need of the specific context.

Following a synthetic description of the main data objects involved for the AMR fleet management, underlying both the Jobs Scheduler and the predictive maintenance.

Mission Lists & Jobs List

The mission list is a list of activities to be commissioned by a fleet of AMR. The concept of mission is oriented to make the most natural as possible for the upper application level, such as an MES, to express requests to the fleet of AMR without the need to take into account other details. For this reason, missions are set of predefined instructions such as "bring the product x to the production line 2". The missions can be explored in a sequence of more jobs.

Such an explosion is performed by a component that adds all the information needed to complete the mission, for example, considering the layout and the position of the products. Examples of jobs are:

- "move from the current position to position A having coordinates x;y;z";

- "Wait for the loading product P by the operator."

Fleet (Resource Set):

An AMR (resource) is assigned to Fleet (resource set). Generally, we could have more Fleets working contemporary, similarly to more operator teams. For example, a Fleet can be assigned to certain type of activities, or operate in a certain area. AMR and Fleets can also be characterized by specific attributes permitting the governance and the jobs assignment, and the respect of specific constraints. For example, only certain types of AMRs could access a protected-clean-area or handle specific objects.

AMR Data and status.

An AMR, is considered a generic resource and is associated with the typology (resource type) characterizing it. In particular, the resource type defines all the characteristics of the AMR, for example, the dimensions and the weight, useful to respect eventual constraints, the nominal and max velocity and acceleration, and the capability to handle objects of a certain weight and type.

Layout: The Layout is an essential element in AMR's Fleet Management with multiple implications and embedding more concepts. For example:

- The layout is involved in managing the single AMR routing;
- Objects and resources must be linked and represented on the Layout (statically or dynamically);
- The Layout must permit the express constraints to be considered when resources are moving, or any entity is located/stocked in a specific position.

In the zero-warm approach, the layout is a CPSoS. It is composed of a set of networks, a group of links/paths, intersections, and/or areas with boundaries.

The data model must permit the description of all the concepts mentioned as examples and needed for a) the intelligent agent representing the layout as a System of Systems and b) the different intelligent agents interacting with the layout.

3.2 Data Exchange and sequence Diagram

The first sequence diagram in this section describes an operational scenario aimed at scheduling missions for Automated Mobile Robots (AMR) within a production environment. The scenario highlights the signals exchanged among various entities, including commands and data, to ensure the smooth execution of tasks.

The main actors involved in this scenario include the Product Manager, the Operator working on the production line, and several software modules and systems. The software modules and systems can be described as follows:

- Reenext Platform: This is a management platform used within Reepack to control and monitor operations. It serves as the main point of contact for the Product Manager, receiving and transmitting commands to other modules.
- Scheduling Module: This module is responsible for creating the mission list for the AMRs. It considers the status and availability of the AMRs.
- Artificial Intelligence (AI) Module: This module employs machine learning techniques to predict the battery discharge of the AMRs, thereby helping to optimize their usage and mission schedules.
- Navigation System: This system guides the AMRs to perform their tasks. It receives instructions from the Reenext platform and communicates with the AMRs to ensure they execute their assigned tasks correctly.
- AMRs equipped with IceBlock: These are the robots that execute the missions. IceBlock is a software or hardware component installed on the AMRs.

The process begins with the Product Manager sending a "Mission List Creation" command to the Reenext Platform. The platform then invokes the Scheduling Module with a "Mission List" command. To create this mission list, the Scheduling Module needs to know the status and availability of the AMRs. It requests this information from the Navigation System with an "AMRs Status Request" signal.

The Navigation System responds with the "AMRs Status" data. Using this data, the Scheduling Module prepares the missions and sends them back to the Reenext Platform as "AMR Jobs Assignment." Each AMR then receives a list of jobs to complete.

A cycle begins where the Reenext Platform instructs the Navigation System by sending the job ID, the start and end positions, and the ID of the AMR assigned to the job. The Navigation System then instructs the designated AMR to execute the job, providing it with the job ID.

During the job execution, the AMR communicates its status to both the Navigation System and the AI Module. This data can be used by the AI Module for various purposes including battery life prediction.

Once the job is completed, the AMR informs the Reenext Platform, which in turn is confirmed by the Operator on the field. The AMR then sends a "Job Completed" signal to the Navigation System, the AI Module, and the Scheduling Module.

This cycle is repeated until all jobs in the mission list are completed. The process ensures effective scheduling and completion of tasks, while maintaining constant communication and status updates among all involved entities.

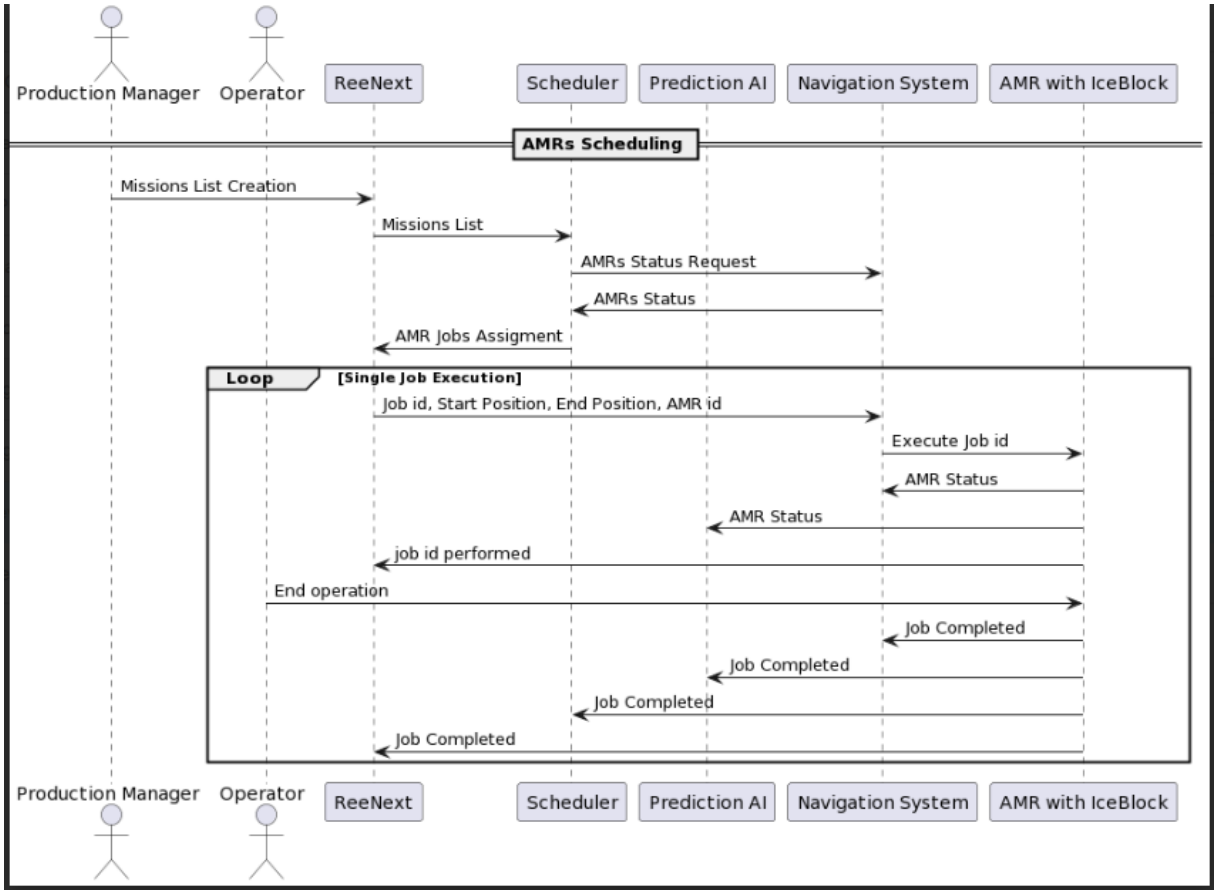

Figure 2 Sequence diagram for AMRs Scheduling scenario

Whereas, in the next scenario, we will describe the process of discovering new devices to be included in the AMR fleet, their registration within the appropriate fleet, and any subsequent updates.

The main actor involved in this scenario is the Operator who works on the shop floor and interacts with the systems to manage the AMR fleet. The software modules and systems can be described as follows:

- Reenext Platform: A management platform used for controlling and monitoring operations, serving as the main point of contact for the operator to transmit and receive commands and data.
- AMR Fleet Management Discovery Module: this module is responsible for managing the AMR fleet, including the discovery, registration, and updating of devices.
- Database Master Record: the database that contains information about the registered AMRs.
- AMRs equipped with IceBlock: these are the automated mobile robots comprising the AMR fleet.

The process of registering a new AMR within the fleet can occur through two different methods: manual and automated.

In Manual Registration the Operator manually enters the details of the new AMR into the Database Master Record using the Reenext Platform. The operator provides all the necessary information about

the AMR, including its unique ID, status, and other relevant data, through the platform.

In the Automated Registration, when a new AMR is introduced to the system and it sends its unique ID to the AMR Fleet Management Discovery Module, the module checks whether the ID is already present in the Database Master Record.

1. If the ID is found in the database, the AMR Fleet Management Discovery Module requests the Operator to provide the status of the AMR. Once the status is obtained, the module updates the relevant information in the Database Master Record.
2. If the ID is not found in the database, the AMR Fleet Management Discovery Module requests the Reenext Platform to register the new AMR. However, the registration request is forwarded to the Operator for confirmation. The Operator verifies and adds all the required data for the new AMR, and upon confirmation, the registration process is completed.

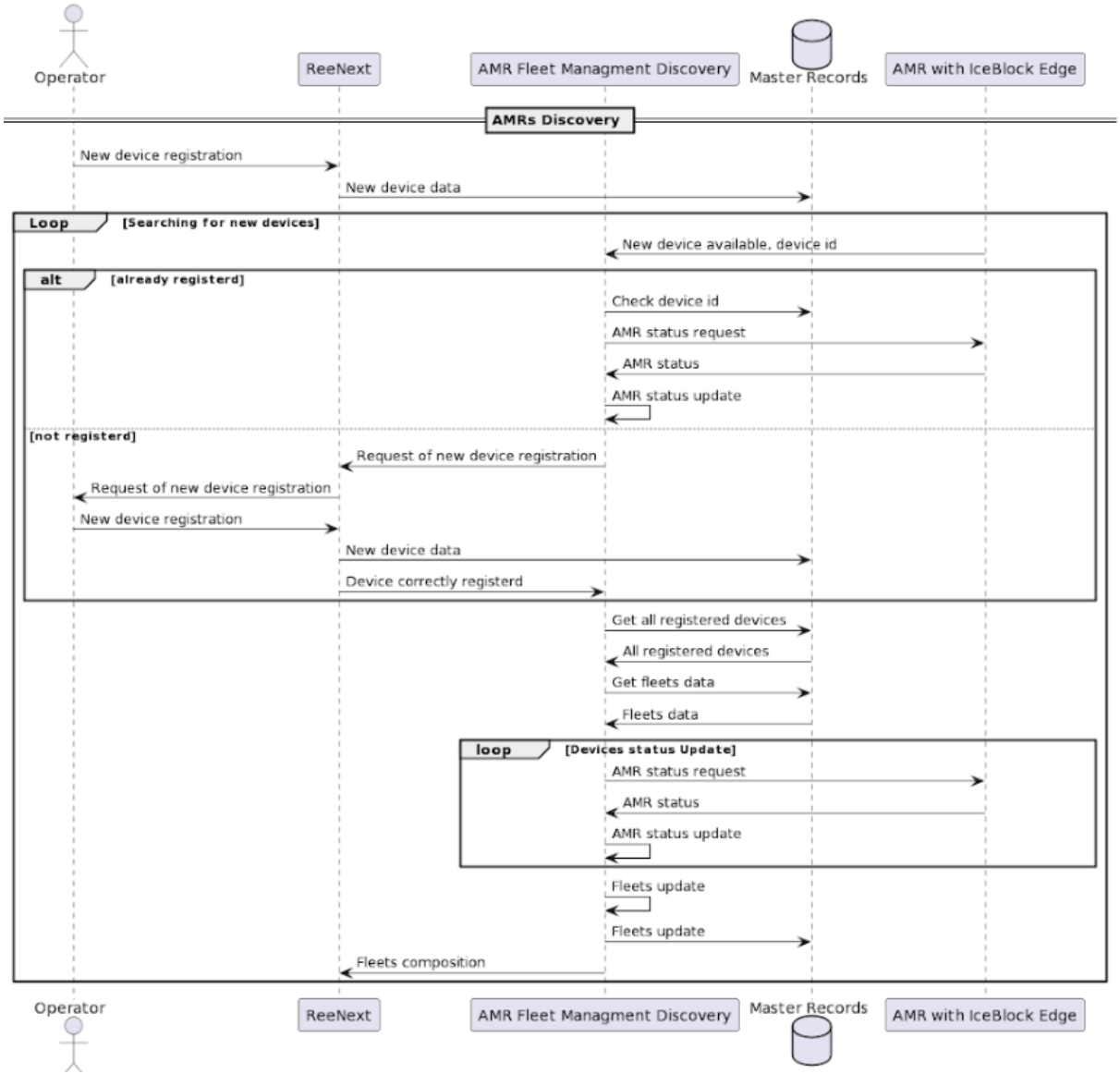

Figure 3 Sequence diagram for AMRs Discovering scenario

The complete details of the new AMR are then saved in the Database Master Record, and the AMR

Fleet Management Discovery Module updates the information related to the AMR within the fleet management module. Subsequently, during the AMR assignment phase, the AMR Fleet Management Discovery Module retrieves all registered devices from the Database Master Record, along with all available data and their respective statuses. This information is crucial for determining the optimal composition of the AMR fleet. Once the fleet composition is decided, the AMR Fleet Management Discovery Module communicates the list of fleet devices to the Reenext Platform, after ensuring that the status of each AMR in the Database Master Record is up to date.

This scenario illustrates the process of discovering, registering, and managing new AMR devices within the fleet. Proper communication and interaction between the various actors are vital to ensuring accurate information updates and efficient AMR fleet management within the automated production environment.

3.3 Intelligent Agent: Efficient Predictive Models

3.3.1 Introduction

An indispensable aspect of managing AMR fleets is optimizing battery usage to ensure smooth mission execution and maximize the longevity of costly battery systems. Considering this, we propose an innovative machine learning model for predictive maintenance, which accurately anticipates battery discharge levels and enables intelligent recharging planning.

To develop a robust predictive maintenance model, we rely on an extensive dataset sourced directly from the production lines where AMRs operate daily. This historical data encompasses detailed battery usage patterns, discharge rates, and essential mission-related characteristics, such as mission duration, distance covered, velocity, acceleration, deceleration events, weight of transported objects, and terrain complexity encountered. Our team employs rigorous data preprocessing techniques to address missing values, noise, and outliers, ensuring the quality and reliability of the training data.

A carefully curated set of features is pivotal to the success of our machine learning model. These features capture crucial aspects of AMR operations that significantly influence battery usage. Key elements include mission duration, distance covered during tasks, average velocity, frequency of acceleration and deceleration events, weight of objects handled, and the complexity of terrains encountered during missions. Moreover, the model benefits from workload predictions for future missions, providing vital inputs for precise forecasting of battery discharge trends.

To achieve high accuracy and reliability in our predictive maintenance model, we explore various machine learning algorithms. Decision Trees and Random Forests are selected for their interpretability and adeptness in handling non-linear relationships, thereby offering valuable insights into battery behavior. Gradient Boosting Machines further enhance predictive accuracy through iterative refinement of predictions. Additionally, Long Short-Term Memory (LSTM) Networks are employed to capture temporal dependencies in time-series data, such as battery discharge trends.

Recognizing the need for real-time predictions and minimal latency, our model is also deployable on the edge, directly embedded within the AMRs. This on-edge deployment empowers AMRs to perform dynamic calculations based on their current battery levels and upcoming missions, facilitating timely and precise recharging decisions. Additionally, we ensure the model's adaptability to changing

operational conditions through incremental updates, allowing it to evolve continuously in response to new data.

In the context of AGVs, AMRs, or drones, predictive maintenance and battery recharge prediction can be achieved through asynchronous learning and distribution of intelligent agents. Historical data on battery usage and discharging patterns can be leveraged to develop predictive models that estimate when the battery levels of AMRs will drop below a certain threshold, indicating the need for recharging. Factors such as the load carried, and operating terrain are taken into account in these models.

Using the predictive models, facility managers or dedicated intelligent maintenance planning agents can plan the recharging process in advance. This ensures that AGVs are recharged during periods of low activity, minimizing disruptions to ongoing operations. Moreover, implementing predictive models for battery recharging helps extend battery life by avoiding overcharging or deep discharging, which can damage the batteries and reduce their lifespan.

In real-life practical scenarios, fleets of AMRs may vary in size and composition. Different types of AMRs may be employed to handle tasks of varying natures, involving objects of different weights and requiring varying amounts of energy. The math-optimizing agent, responsible for ensuring high productivity, must correctly plan the assignment of time slots for individual AMR battery recharge. Predicting battery level trends is critical information for the scheduling agent to plan recharging appropriately.

The prediction model is trained based on log data generated by the fleet, which traces all operations performed by individual vehicles and their characteristics, such as duration, distance, average velocity, acceleration, deceleration, weight of moved objects, and other consumption-influencing conditions. This prediction model receives data representing the expected missions required for every vehicle in the following period and provides the discharging trend, which can be utilized by the scheduling logic or the fleet supervisor.

In many facilities, the workload for the fleet may exhibit seasonality, such as intra-day, weekly, or monthly variations. To address this, the prediction model could include different versions capable of considering such seasonality. Regular re-training and redistribution of the model to the edge permit dynamic real-time trend calculations based on the current battery level and the next predicted mission.

Considering the philosophy of the zero-swarm projects and the concept of Cyber Physical System of Systems (CPSoS), it becomes valuable to calculate the expected remaining fleet capacity in terms of energy for the next time frame. This predictive KPI can be useful for the plant supervisor, especially in cases of critical expected picks, to ensure optimal planning and resource allocation.

The integration of predictive maintenance and battery recharge optimization for AMR fleets represents a significant advancement in the field of logistics and industrial automation. Through the development and deployment of advanced machine learning models, facility managers can intelligently plan recharging, enhance productivity, and prolong battery life, leading to smoother and more efficient operations. As technology continues to evolve, predictive maintenance models will play an increasingly vital role in shaping the future of autonomous robotic fleets.

3.3.2 Background knowledge

The goal of improving the quality of cooperative-based internal logistics is the effective use of energy resources. Automated mobile robots (AMRs) are an integral part of enterprises in Industry 4.0 [4] [5] [6] [7]. One important feature is that they are a huge source of information. While driving along a given route and performing the tasks set before them, a variety of streaming information is collected and recorded. This information is sorted by the types that are defined within the frame.

In flexible and efficient internal transport systems, it is necessary to use the battery of the AMR as efficiently as possible. Therefore, determining the remaining battery charge is a very important task. To date, it is forbidden to discharge a battery by less than 10%. This is due to its technological features. Thus, when 20% is reached, the AMR should go to the recharging station. There are several approaches for predicting the AMR battery discharge rate. These include an open-circuit voltage method [8]. The state of the charge of an AMR battery is often investigated through a simulation experiment using the Kalman filter method [9].

The performance of an AMR is increased by minimizing the time it takes to recharge a battery and optimizing its energy by managing battery replenishment [10]. If more than one AMR is used at an enterprise, energy consumption can be saved based on a heuristic path planning algorithm [11]. The publication [12] presents an effective algorithm for the multi-objective optimization of costs and energy efficiency at the expense of minimizing the working times of an industrial robot arm, the travel times of the AMRs, and their trajectories. Systems for the intelligent management of the lithium-ion batteries that are based on AMRs have also been developed [13].

Today, Machine Learning (ML) methods are increasingly used to forecast the AMR battery discharging rate. Most ML is based on artificial neural networks (ANN) [13] [14] [15]. This is due to their ability to learn from historical data. However, difficulties arise when using ANN [16] [17] [18].They are associated with the preliminary preparation of the training data set. Namely:

- detecting lost data;
- recovering partially lost data;
- eliminating any cases of noise;
- eliminating any emissions;
- discarding a large number of the null values of features;
- normalizing the values;
- transforming the symbolic attributes;
- selecting a subset of features.

To support the repeatability of the ML process, which is based on the collected experimental data set, an information model was developed for the new generation of production systems [19] [20]. With its help, not only can data be collected, but it also offers repeated and selective access to the collected information.

The main goal of these studies is to apply the most common ML methods for forecasting AMR battery discharging. This should be done for more efficient use of the AMR battery charge, especially the remaining battery charge. ML methods should include the following steps:

- collection, detection, and recovery of lost data;

- finding correlation dependencies between parameters describing battery cell voltage;
- normalization of data and creation of training and test samples;
- training and testing of the developed artificial neural network model.

3.3.3 AMR data collection and setup configuration

An important part of the task is to configure the data that is collected from the production systems in a form that is suitable for further processing by the analytic ML-based system. The OPC UA protocol can be used to retrieve the measured data from the Automated Mobile Robots (AMRs). The informative components of the data-vector can be extracted from the received measurements and can be converted into a suitable format for further processing. The standard OPC UA client returns only the data changes for the configurated properties. The “OPC Historical Access” interface of the UAExpert OPC UA client from the Unified Automation GmbH [21] can be used to obtain the data from the AMRs according to the filters that had been configurated for the required properties.

Data from the AMRs can be recorded in a database on a server so that data can be placed in the cloud. Then, they can record in a file with the csv extension using for example UA Expert [21].

The received results for the requested frame can be stored locally for further pre-processing and can be used to prepare the ANN. However, to create the ANN-training and the ANN-testing data sets, the historical data for all of the informative parameters with the same and uniform sample rate are required.

Data can be collected for a given route by Reepack. An AMR route map can be prepared by the operator on which all the possible obstacles on the AMR’s route are taken into account. All the obstacles on the map should be indicated. As an obstacle, for example, could be a building construction element, furniture, work areas, etc.

The data can be aggregated at the stop points. To enhance the AMR-motion map's clarity, it is recommended to overlay a uniform grid and divide it into cell squares (or sectors). Along the AMR route, small black circles can be strategically placed, further dividing the map into segments. These points should be strategically positioned at locations where measurements can be taken or at the destinations where the AMR needs to execute specific tasks within the scope of its mission. The lengths of the segments can be not equal and moreover, the end distribution of the segments across the grid cells can also be not uniform.

The rate at which the battery discharges can be measured for each segment. As a result, the remaining battery charges (along with other diagnostic parameter values) can be assessed at each segment's endpoint as the AMR passes through on its route between the stop points.

During the experiment, the main purpose can be to obtain the historical data for all possible AMR movement cases on its way. The measurements can be taken when an AMR is carrying a specific load and when an AMR follows its route with no load. The AMR should traverse the selected route going forward and backward. The AMR motion can be paused sporadically to simulate an unexpected incidence of a worker crossing the path of an AMR. At the start of the experiment, the battery should be charged 100% and it should be stopped once the remaining battery charge was about 10%. According to the algorithm to be deployed, an AMR can be directed to a charging station when the battery had discharged to 20%. Even when a battery has discharged to 20%, an AMR can still perform

a certain type of work. Additionally, the residual charge should be enough for it to reach the charging and station before it had discharged below 10%. To make this improvement, it is necessary to determine how the battery discharges after an AMR completes this extra work. Since there must be enough battery power to reach the nearest charging station on ocean AMR completed this extra work, a highly accurate short-term forecasting is essential. In this way, the AMR tasks can be planned more accurately, and a better performance can be obtained when using an AMR.

3.3.4 Data read from an AMR

The data that should be collected for the route to be implemented are associated with specific data-frames and their corresponding properties are listed as follows:

[TS] TIME STAMP

[GS] GENERAL SIGNALS

[SS] SAFETY SIGNALS

[LED] LED STATUS

[ES] EXCLUSIONS STATUSES

[OS] OTHER STATUSES

[WS] WEIGHT STATUSES

[G1LDS] GROUP 1 – LEFT DRIVE SIGNALS

[G2RDS] GROUP 2 – RIGHT DRIVE SIGNALS

[G1BS] GROUP 2 – BRAKES SIGNALS

[G2PAS] GROUP 3 – PIN ACTUATOR SIGNALS

[G3LPS] GROUP 3 -LIFTING PLATE SIGNALS

[AI] – ALARM INFORMATION

[WI] – WARNING INFORMATION

[MI] MESSAGE – INFORMATION

[ODS] – ODOMETRY SIGNALS

[ENC] – ENERGY SIGNALS

[IS] – INCLINATION SIGNALS

[NNS] – NATURAL NAVIGATION SIGNALS

[NNCF] – NATURAL NAVIGATION COMMAND FEEDBACK

The data that can be retrieved via UAExpert for the AMR include the following parameters:

[ENC] – Energy Signals (Momentary current consumption, Battery cell voltage, Momentary power

consumption, Momentary energy consumption, Cumulative energy consumption)

[ODS] – Odometry Signals (Cumulative distance left, Cumulative distance right, Momentary frequency of left encoder pulses, Momentary frequency of right encoder pulses)

[NNS] – Natural Navigation Signals (Current segment, Heading, X-coordinate, Y-coordinate)

[G1LDS] GROUP 1 – LEFT DRIVE SIGNALS (ActualSpeed_L)

[G2RDS] GROUP 2 – RIGHT DRIVE SIGNALS (ActualSpeed_R)

The AGV battery discharge process is characterized by parameters in the [ENC] Battery cell voltage frame. To identify the most informative parameters related to battery discharge, correlation analysis methods can be employed. For this, can be used for example the three most common correlation coefficients: Pearson's correlation coefficient, Spearman's rank correlation coefficient and Kendall correlation.

3.3.5 Data preprocessing

Through experimental analysis, it can be identified that the incoming data could contain some flaws, such as missing data and spontaneous spikes, which could adversely impact the training of the artificial neural network (ANN). To address these issues, algorithms should be developed and implemented for padding data gaps, detecting peaks, and correcting data irregularities. The algorithms are designed with specific ramp periods to ensure they do not interfere with the ANN training process. As different parameters exhibited varying value ranges, normalization was essential to bring all the data within a common range from 0 to 1, ensuring uniformity for all parameters. The approach to handling lost data involved the following data padding techniques:

- Replacing any NaNs with the previous available values.
- Suppression of accidental peaks: Values exceeding 1.5 times the variance (e.g., RMSE) were replaced with moving average values.
- Trimming start and end values within the time window range.
- Scaling data to fit within the range of 0 to 1.

3.3.6 Constructing the model

With the aid of Python, a versatile programming language well-suited for machine learning tasks, we embark on the development of our predictive maintenance model. Utilizing powerful machine learning libraries, we train the model on the extensive historical dataset obtained from real-world AMR operations. Subsequently, the trained model is deployed in the cloud, making it accessible to intelligent maintenance planning agents and enabling seamless integration into existing infrastructure. To assess the effectiveness of our model, we employ a range of performance evaluation metrics. The Mean Absolute Error (MAE) is utilized to measure the absolute deviation between predicted and actual battery levels, providing crucial insights into the model's prediction accuracy. The Root Mean Squared Error (RMSE) quantifies the overall prediction performance, while precision, recall, and F1-score assess the model's ability to accurately identify critical battery discharge levels.

The primary criteria for selecting the appropriate ANN type are the simplicity of implementing its algorithm in Python and the accuracy of training and forecasting, while the learning speed should not be a decisive factor.

A comprehensive analysis of various ANN models and types, considering their respective purposes, should be conducted. The condition under which a specific network performed optimally should be carefully evaluated, considering the algorithm's complexity and computer performance requirements. Ultimately, the multilayer sequential model ANNs can be chosen due to their simplicity of implementation in Python. The sequential model in Python allows for the creation of a straightforward stack of layers, each with one input tensor and one output tensor. This model builder facilitates the extraction of intermediate level results into a sequential model, making the process convenient and efficient. Transfer training can be applied by freezing the lower layers in the model and training only the upper layers.

For Python-based training, the selected ANN can be trained using Keras, a popular deep-learning framework running on the TensorFlow machine learning platform. Its rapid experimentation capabilities enabled quick progress from ideas to results. The model can be trained using the fit() method, which slices the data into batches and iterates over the entire dataset for a specified number of epochs. The model's performance on the test data can be evaluated using evaluate(), with metrics such as mean squared error and mean absolute percentage error.

The developed ANN model combined the simplest neural networks, allowing data transfer to occur in models with several inputs and outputs. This combination can be achieved using layer concatenation in Python. Each of the simplest neural networks has 12 inputs, 8 neurons in the hidden layer, and one output neuron. The proposed ANN model with several inputs and outputs involves parallel data processing on the combined hidden layer for each parameter. The number of outputs of the ANN corresponded to the number of parameters for which "time windows" forecasting can be applied.

An additional advantage of this model is the simultaneous data transfer to models with several inputs and outputs. The total number of inputs to the model equaled the number of inputs of each ANN multiplied by the number of parameters related to the decrease in an AGV's battery charge. The model's outputs can be determined by a combination of the outputs from each of the simplest ANNs. The computational results will be part of the WP6 use case integration activities.

3.4 Federation Learning - opportunities, methods and Implications

In Industry 4.0 processes and business models rapidly expand the use of modern technologies towards the implementation of true smart industrial systems. Big data analytics, Internet of Things, Artificial Intelligence, Machine Learning, and computing on the Edge have a key role in new applications of the domain. These technologies in synergy are used to form data processing pipelines from the phase of data collection to storage in the cloud and analysis and up to the creation of intelligent services using machine learning. It is evident that Industry 4.0 can leverage the power of these technologies to drive innovation and improve productivity. Companies and manufacturers often are unwilling to share data outside their premises, restricting the availability of their data for internal purposes only. Data collected from a single company cannot accurately represent global contexts leading to AI models with limited capabilities unable to handle unfamiliar cases that might occur.

Federated learning was introduced by Google in 2017 [22] aiming to address the challenges of training machine learning models on data distributed across many devices without needing the data to leave the devices. This setting leverages the collective intelligence and computing capabilities of various devices such as smartphones, IoT devices and edge nodes to train a model collaboratively and in a privacy preserving manner. FL being a collaborative AI approach allows it to incorporate large datasets and computation resources, which are distributed in the network, improving the quality of the resulting AI models, and alleviating the limitations that centralized training faces regarding volume of available data and limited computational capabilities.

The above characteristics alongside the ability to address privacy concerns and efficiently train models in distributed environments has provided popularity to the Federated Learning paradigm in the domains of Internet of Things and the Industry where collaborative and decentralized data processing is of the essence. Integration of federated learning with the industry and industrial IoT offers important benefits [23] such as:

- *Privacy Enhancement*: During training a model in a federated learning setting only the updates are required to be transmitted over the network to a central server, while the data remain private on the devices. In addition to the privacy by design imposed by FL, many algorithms and techniques have been developed to further enhance privacy of FL systems [23] [24].
- *Low-latency Network Communication*: Not centralizing vast volumes of data to the server, communication costs are greatly reduced, saving this way valuable resources. Again, research has been focused to further reduce the communication imprint of FL systems by applying compression schemes on the updates exchanged during training [25].
- *Improved Learning Quality*: The ability to utilize large resources, data, and computation, distributed in an industrial network significantly improves the quality of the resulting AI models. Like the previous benefits also new algorithms and novelties have been developed to accelerate convergence rate of the training process and reduce time required to most Federated Learning systems implementations regard synchronous FL where the central server updates the shared global model after all the client updates have been collected. Due to device heterogeneity, where processing capabilities and other aspects differ from device to device, makes device availability and computation time to vary. This makes synchronization difficult in real world federated learning scenarios. Research in distributed stochastic gradient descent focused on asynchronous training [26] to address stragglers and latency and lead to the introduction of asynchronous federated learning. In asynchronous federated learning the server updates the shared global model as soon as a new update is collected [27] [28].

Various studies have been focused on refining and optimizing the asynchronous federated learning approach to make it more effective and practical for real-world applications. In [29] an asynchronous FL system is proposed based on two key ideas:

- solve regularized local problems to guarantee convergence.
- and use a weighted average to update the global model, where the mixing weight is set adaptively as a function of the staleness.

The proposed system facilitates non-blocking communication between the server and the clients. The server consists of two main components the scheduler and the updater which run asynchronously and in parallel. The scheduler's primary role is to trigger training tasks at specific intervals while also controlling the staleness of the information used by the updater. Staleness is represented by the difference between the current time (t) and the timestamp (τ) of the data received in the updater thread. The updater receives models from the workers and is responsible for aggregating and updating the global model. Staleness is used to determine the mixing value that is used to weight the local update during aggregation. Large staleness incurs greater error during aggregation thus the mixing value is decreased to reduce it.

In ASO-Fed [30] an ASynchronous Online Federated Learning framework is presented which focuses on an asynchronous system with convergence guarantees which maintains optimal performance. The work in ASO-Fed is closely related to IoT and Industrial settings since it considers not only heterogeneous devices and data but also considers the existence of continuous data streams and addresses it by employing an online learning procedure. Gradients resulting from training are being balanced with the help of an iterative procedure. To ensure an optimal global model the deviation of the local gradients is constrained with the use of a threshold imposed locally on the devices. The server maintains a global model and each device keeps a copy of the global model in their memory. The server aggregates each client update the moment it arrives. The aggregated model undergoes feature learning to extract a cross-client feature representation which is used to dynamically modify learning rates for the clients. The inspiration for feature learning comes from attention mechanisms, which have proven to be highly effective in identifying crucial features and representing them in a meaningful way. When new data are collected on the device the global model is requested and is trained on them, the balance between the previous model and the new one is achieved using a decay coefficient. The dynamic learning step size which is calculated based on feature representation is used to address stragglers and improve performance. Stragglers are considered to have a smaller activation rate when using the global model and thus their learning step size is increased.

The work in [31] presents a system that tries to combine the best of both worlds. Incorporates increased scalability through asynchronous aggregation and alignment with privacy preserving algorithms designed for synchronous FL such as differential privacy and secure aggregation. It is evident that previous work mainly addressed the scalability issues in the presence of stragglers but neglected privacy guarantees which is the most valuable characteristic of an FL system. The main difference to the previous implementations is that it features a semi asynchronous algorithm which instead of aggregating the global model each time a client sends an update, stores a number of client updates to a buffer before aggregating them which allows the use of privacy enhancing algorithms. Note, that the number of client updates stored in the buffer each time is not equal to the number of clients participating.

In [32] on top of the straggler issues another key challenge of federated learning systems is addressed, that of communication cost. Repeatedly sending the large global model to the clients may overload clients with limited communication bandwidth. To this end, much research has been focused on the compression of the model to reduce the size of the data that is exchanged [33] [34]. This work, Quantized Asynchronous Federated Learning (QuAFL), is an extension of the classical FedAvg algorithm

in which asynchronous aggregation and compression mechanisms are also included.

In [35] adopts the FedBuff asynchronous federated learning system and extends it to support four aspects of vanilla synchronous federated learning. Specifically, they propose mechanisms for client selection, secure aggregation, client replacement and fast aggregation all in the context of asynchronous federated learning.

It is evident that the benefits of FL become even more apparent in the domains of IoT and the Industry. Federated Learning ensures privacy by design by letting data remain on the devices during training which is also beneficial in terms of communication costs. Lastly, learning quality is enhanced by leveraging the vast resources distributed in industrial networks, resulting in better AI models.

The extension of vanilla synchronous federated learning to asynchronous has addressed the issue of device heterogeneity and the existence of stragglers allowing non-blocking communication and the ability to adapt to varying device availability and computation times. Recent studies [36] in AFL have offered improvements in terms of scalability, privacy and communication cost though also revealed critical challenges like device heterogeneity, data heterogeneity, privacy and security on diverse devices.

In the context of Zero-Swarm an Asynchronous and Online Federated Learning system will be designed based on the work of [28]. The proposed algorithm is more closely related to the setup of Zero-Swarm where data streams exist on the edge devices. In addition, towards a truly optimized FL system compression mechanisms provided by the literature will be applied to reduce the communication cost.

3.5 Optimize communication overhead - insights

Another key aspect of the proposed algorithm is the optimized communication overhead. A federated learning system aims to train a model collaboratively across decentralized devices; thus, the success of the system heavily depends on the efficiency of the communication between the cloud, which will serve as the central server, and edge devices. Facilitating a large number of edge nodes and complex AI models can make the communication overhead a critical bottleneck. Zero-Swarm will utilize compression mechanisms to optimize the communication overhead incurred by the exchange of data to provide an efficient and scalable AFL algorithm.

To address these challenges research [37] [38] has been focused on applying ML compression techniques. The aim of the compression mechanisms employed are to reduce the size of data exchanged between the server and the edge without compromising the learning quality. For example, quantization and sparsification are two kinds of compression methods that have been widely used to minimize the amount of information exchanged.

In [39] the objectives of the compression schemes applied in the context of federated learning fall into three categories as follows:

- Methods that reduce client-to-server communication. This is achieved by compressing the gradients obtained after local training on the device and which are used to update the

global model through aggregation.

- Methods that reduce server-to-client communication. This regards the compression of the global model which is shared to the edge nodes in the network. The global model is used as a starting point for local training on the edge.
- General methods that aim to improve the overall efficiency of the global model training procedure.

The integration of gradient compression with Asynchronous Federated Learning (AFL) encounters novel challenges, particularly in the resource-constrained edge/IoT computing environment where aggregation operations are more frequent [36]. Frequent aggregation and compression operations lead to increased server-side computational demands compared to traditional FL. Also, diversity of the edge nodes presented in AFL systems in the IoT and Industry domain pose increased disparities in computation power between edge nodes.

To address challenges emerged by AFL settings studies have been focused on developing compression algorithms tailored to the needs of AFL. Similarly, to traditional FL these algorithms aim to optimize the compression process and ensure communication efficiency in resource limited settings without compromising learning quality. Addressing the computational constraints and enhancing the compression techniques, scalability and efficiency can be assured.

4 Optimization Algorithms

In this section, we introduce the fundamental concepts required to define our dispatching algorithm, which efficiently assigns tasks to different Autonomous Mobile Robots (AMRs). To achieve this goal, we establish two mathematical models for addressing specific problems: the scheduling problem and the vehicle routing problem. We also define a meta model, which makes use of Machine Learning techniques to solve the problem in question in a short time.

The dispatching problem, in its simplest form, can be treated as a scheduling problem, where we schedule tasks for AMRs within a given timeframe. However, as we account for various complexities and features of the problem, it becomes necessary to formulate it as a vehicle routing problem, aiming to optimize the routes taken by AMRs for task completion.

Before delving into the specific chapters for scheduling and vehicle routing, we provide some essential concepts related to combinatorial optimization, which serves as the foundation for designing efficient algorithms for our dispatching system.

Combinatorial optimization is a branch of applied mathematics that deals with finding the most efficient or optimal solution from a finite set of possible solutions. In this field, problems involve discrete variables, and the main objective is to explore various combinations or permutations of these variables to determine the best configuration that satisfies specific constraints while optimizing a given objective function. These problems can be quite challenging because the search space often grows exponentially with the number of variables, making the process of finding the optimal solution computationally demanding. Mathematical programming, also known as mathematical optimization, is a broader term that encompasses combinatorial optimization. It refers to the process of formulating a problem mathematically, defining an objective function to be maximized or minimized, and

determining constraints that the solution must adhere to. The goal of mathematical programming is to find the values of decision variables that optimize the objective function while satisfying the given constraints.

Two common types of mathematical programming are linear programming (LP) and integer programming (IP). Linear programming deals with problems where the objective function and constraints are all linear functions of the decision variables. It is particularly useful for continuous optimization problems, such as resource allocation and production planning, where the variables can take any real value within certain bounds.

On the other hand, integer programming extends linear programming by incorporating the additional requirement that some or all the decision variables must take on integer values (whole numbers). This constraint introduces discrete decisions into the optimization problem. As a result, integer programming is particularly well-suited for problems where the variables represent items that must be selected or not, such as in project selection, facility location, or binary decision-making scenarios.

4.1 Scheduling Algorithm

The following section defines the functional requirements for the near-real-time scheduling of the jobs for the AGV/AMR fleet(s). The description refers to the data model described in the previous chapter 2.

Introduction

Let's analyze the context in which a manufacturing company uses an AMR system to deliver the materials needed to a series of workstations for production activities. Precisely in the context of the Zero-SWARM project we want to manage a fleet of AMRs serving a food packaging production line. The objectives can be multiple, such as minimizing the time required to complete the various jobs or minimizing the energy consumption needed to power the AMRs. The company has a series of packaging lines simply called “workstations” and a set of warehouses, each of which contains a specific type of product (called item).

An item in our algorithm is anything that can be carried by an AMR and can be of different types, such as:

- packaging;
- food;
- plastic film;
- ...

The items are stored in a different warehouse based on their type and require transportation to the workstations by a fleet of heterogeneous Autonomous Mobile Robots (AMRs) located at a specific point. The AMRs can operate at varying speeds. As the number of AMRs may be considerably fewer than the number of workstations, each AMR has the potential to serve multiple workstations. However, due to the AMR's capacity to carry only one item at a time, an AMR serving multiple workstations must complete multiple round trips. A round trip involves traveling from one warehouse to a specific workstation while carrying a loaded item and returning empty to the warehouse to the starting point (which can be the warehouse associated with the type of product transportable by the

AMR). In scheduling terms, these round trips are referred to as “transfer jobs”.

Each transfer job is characterized by two parameters. Firstly, the duration of a transfer job is determined by the total time required for the round-trip travel, as well as the loading and unloading operations for the package. Secondly, the weight of a transfer job is determined by the combined weight of the materials being transported. The energy cost of a transfer job is calculated according to the time required to transport the item and the weight of the item.

Each AMR possesses a battery that has a limited capacity, causing its charge to diminish because of the duration and weight of the transfer job. Consequently, an AMR might necessitate a visit to a charging station to fulfill all the assigned transfer jobs. The charging stations are situated in the warehouse that corresponds to the type of product transportable by the AMR, where the AMR must undergo a charging operation before its battery is completely exhausted. The charging process ensures a complete replenishment of the AMR's battery and adheres to a fixed timeframe, unaffected by the remaining energy. Nonetheless, it is worth emphasizing that this assumption gains further support when considering the battery's life cycle and the distinctive features of the charging procedure. Significantly, minimizing the number of charging cycles is essential due to the inherent limitations associated with battery charge cycles. On this basis, it is possible to consider a charging operation as a special job with a duration equal to the charging time and null weight.

Intelligent Real-time Agent

Within the Zero-SWARM system, our algorithm is invoked to handle the scheduling and rescheduling of jobs. This process is triggered automatically at regular intervals, such as every five seconds, using a time-based polling mechanism. Each time the algorithm is invoked, it receives the updated list of jobs and the status of each AMR as input.

The algorithm's primary task is to determine the assignment of each job to a specific AMR. This assignment is based on various factors, including the objective function (optimization goal), constraints that need to be satisfied, and the priority of each job.

By the end of each run, a single AMR will have a sequential list of jobs assigned to it. When a new run of the algorithm is initiated, the ongoing (currently processing) jobs are kept frozen in the list. Typically, a certain number of next jobs, denoted as "x," are also maintained frozen. The value of "x" is a system parameter, and it is common to set it as 1.

In summary, our algorithm is integrated into the ZeroSwarm system, where it is invoked periodically to schedule and reschedule jobs. It considers various factors to assign jobs to AMRs, and the ongoing jobs as well as a predetermined number of upcoming jobs are frozen during each run to maintain continuity in the job sequence.

The Object Function

In the context of Zero-SWARM, there are various objective functions that can be selected for job scheduling, and these functions can be chosen via a configuration parameter.

One common objective function is Energy Saving, which aims to minimize the energy consumption of the AMRs while delivering the assigned jobs. By optimizing the assignment of jobs to AMRs, the algorithm seeks to reduce the overall energy usage and promote efficient resource utilization.

Another objective is Fulfillment Time, which focuses on minimizing the completion time of the entire job list. The algorithm aims to schedule the jobs in a way that ensures timely completion, reducing delays and maximizing productivity.

Additionally, the concept of Uniform Waiting Time plays a crucial role in enhancing the service level within the system. This objective aims to maintain a uniform and minimized waiting time for job requesters. By optimizing the job scheduling, the algorithm strives to distribute the workload evenly among the AMRs, reducing waiting times and enhancing overall customer satisfaction.

It's important to note that the choice of the objective function depends on the specific requirements and priorities of the system. The configuration parameter allows for flexibility in selecting the most suitable objective function based on the desired outcomes and operational goals of the Zero-SWARM system.

The Constraints

There are multiple constraints that the optimization algorithms must respect. The constraints set depend on the specific applications, so the single constraints can be enabled or disabled by an activation parameter. An example of Hard Constraints are the following:

1. Item type T1;T2; can be handled by AGV of type "AZ;AY"
2. Jobs of type J1; J2 can be handled by AGV of type "AZ;AY"
3. The "zone x" can only have a maximum copresence of #x AGV
4. The "zone x" can only accept AGV Type "AZ;AY"
5. Min working battery level for AGV Type "AZ;AY"
6. Min working battery level for AGV Type "AZ;AY"
7. Item type T1;T2; can be handled by AGV having an additional resource type "R1;R2" (for example, an additional robot arm)

Soft Constraints (possibly respected):

1. Respect for Jobs' due time: as each job might have a due time to be accomplished, the optimizer must ensure that a job is delivered by the maximum due time. This is not a hard constraint, as it could easily make the problem with no solutions in case of the pick of works for the fleet. In this situation, the system has the possibility to delay the delivery of some jobs.
2. Respect for Jobs' minimum due time: in some cases, jobs might be required by the requester with a min due-time, for example, "take the product P1" not before 3:10 PM. This permits requesters to book some activities in advance.

For our problem, we can classify the constraints into two categories: 1) Typical constraints for a scheduling problem; 2) Constraints related to the nature of the problem.

Typical constraints

Below we report a set of typical constraints for a scheduling problem:

- **Precedence Constraints**: Certain tasks or jobs must be executed before or after other tasks, based on dependencies or order requirements.
- **Sequence Constraints**: The order in which jobs are scheduled must adhere to a specified

sequence or sequence-dependent rules.

- **Time Constraints**: Specific time windows or deadlines within which jobs must be scheduled or completed. (SOFT)
- **Availability Constraints**: Constraints on the availability or non-availability of certain resources or facilities at specific times.
- **Setup and Transition Constraints**: Some tasks require setup or transition time before they can be executed, impacting the scheduling decisions.
- **Resource Constraints**: Limited availability of resources, such as machines, equipment, or personnel, that need to be considered when scheduling jobs.
- **Preemption Constraints**: Whether or not tasks can be interrupted or preempted once started, or if they must be completed without interruption.
- **Concurrency Constraints**: Limitations on the number of jobs or tasks that can be executed simultaneously.

These are just some of the common constraints encountered in scheduling problems. The specific constraints may vary depending on the nature of the problem and the specific requirements of the scheduling scenario, we want to consider only a subset of the proposed constraints. Moreover, the order in which the constraints are presented reflects their relative importance. When implementing our algorithm, we prioritize the constraints listed earlier and ensure their fulfillment, while the constraints mentioned later may not be guaranteed for implementation.

Typical AGV constraints

Presented below is a list of constraints that are relevant to our problem and arise due to the involvement of AMRs in performing scheduled jobs. The order in which these constraints are listed signifies their relative significance:

- **Battery Constraints**: AGVs have a limited battery capacity, and scheduling must consider the battery levels to ensure that AGVs have sufficient charge to complete their assigned tasks. This includes accounting for charging or battery swap requirements.
- **Weight and Load Constraints**: AGVs have a maximum weight or load capacity that must be considered when assigning jobs. Jobs exceeding the AGV's capacity need to be appropriately allocated or split among multiple AGVs.
- **AGV Accessibility Constraints**: Some areas or zones may have restrictions on AGV access due to safety, security, or operational considerations. Scheduling should adhere to these constraints to ensure AGVs do not enter restricted areas; or certain environmental conditions or constraints may impact AGV operations, such as temperature sensitivity, humidity, or cleanliness requirements. The scheduling algorithm should consider these constraints when assigning jobs to AGVs.
- **AGV Speed and Travel Time Constraints**: AGVs have different travel speeds, and the scheduling algorithm should consider these speeds when calculating travel times and optimizing the overall schedule. Travel times may vary depending on the AGV's location, terrain, or other factors.
- **Collision Avoidance Constraints**: AGVs should be scheduled in a way that minimizes or avoids collisions between AGVs and obstacles or other AGVs. This involves considering the paths and trajectories of AGVs and incorporating collision avoidance mechanisms.

- **AGV Maintenance Constraints**: AGVs may require periodic maintenance or servicing. The scheduling algorithm should account for these maintenance requirements and avoid scheduling tasks during maintenance periods.
- **AGV Availability Constraints**: The availability of AGVs may vary due to maintenance, charging, or other operational factors. The scheduling algorithm should consider AGV availability when assigning jobs to ensure that there are enough AGVs to handle the workload.

These constraints are specific to scheduling jobs for AGVs and need to be considered to optimize the scheduling process and ensure efficient and effective use of the AGV fleet. Again we address only a subset of the constraints listed above.

Problem Definition

In this section we describe the Heterogeneous AMR Scheduling Problem with Battery and Weight constraints (H-AMR-SP-BWC).

The H-AMR-SP-BWC involves determining the scheduling of transfer jobs and charging operations on an AMR fleet minimizing a certain objective function such as the AMRs energy consumption. Ensuring the timely completion of each job is as essential as minimizing energy consumption in the problem. However, this aspect is incorporated by defining the problem's constraints.

Table 1 Example Parameters of a job

Job-ID	Time	Energy	Weight
J1	1	6	3
J2	2	3	4
J3	3	1	6
J4	4	6	7

Table 2 Example Parameters of an AMR

AMR-ID	Max-Energy	Max-Weight
AMR-1	8	5
AMR-2	8	8

Therefore, the solution for H-AMR-SP-BWC encompasses two critical aspects: assignment decisions, which involve matching jobs with AMRs, and scheduling decisions, which entail sequencing the jobs on each AMR.

To provide a better understanding of the problem, we present a solution of the H-AMR-SP-BWC on a small instance with 4 transfer jobs and 2 AMRs, as illustrated in Figure 1 using the information provided by the Tables 1-2. Table 1 provides a description of the jobs, presenting the following columns: job duration, expressed in time units, energy consumption required to complete the job, expressed in

energy units, and the maximum weight of the item to be transported in that job. If a job does not involve the transportation of an item, the weight is specified as zero.

Table 2 presents the characteristics of the AMRs, including the maximum battery energy capacity expressed in energy units and the maximum weight they can carry.

Figure 1 depicts a feasible solution of the H-AMR-SP-BWC, as we can see, AMR 1 can carry a maximum load of 5 units, while AMR 2 can carry a load of 8 units. Therefore, job 3 (J3) and job 4 (J4), which require the transportation of weights of 6 and 7 units respectively, can only be performed by AMR 2, as it has the capacity to execute both jobs without depleting its battery.

While AMR 1, with a battery capacity of 8 energy units, performs job 1 (J1) requiring 6 units, it does not have enough energy to also perform job 2 (J2) requiring 3 units of energy. Therefore, before being able to perform J2, AMR 1 needs to execute a charging job, denoted as C, to recharge its battery.

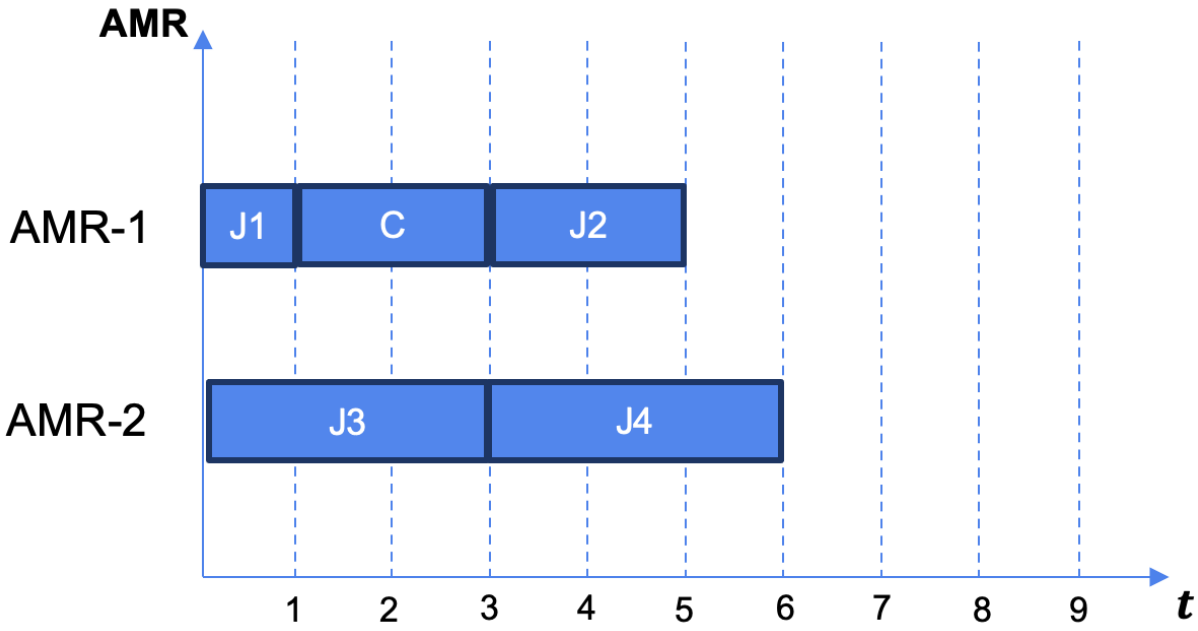

Figure 4 Possible solution for H-AMR-SP-BWC using the parameters shown in Tables 1-2.

We define timespan as the total duration or time interval required to complete a set of scheduled activities in a scheduling problem. The timespan represents the time interval between the start and end of the scheduling process, which includes the execution of all the planned activities. The timespan indicated in Figure 1 is 6.

Based on the previous discussion, we have developed a formulation for the H-AMR-SP-BWC that incorporates assignment-based considerations and implicitly considers the sequencing aspect. Based on this premise, we define $\boldsymbol{J}$ as the set of transfer jobs, where each job $\boldsymbol{j} \in \boldsymbol{J}$ has an energy consumption e_j, a processing time d_j and a weight w_j that is independent of the specific AGV assigned to perform the transfer job. Consider M as the set of AGVs, where each AGV $m \in M$ has a battery capacity of $b_m$and a maximum supported weight w^m. Additionally, let n represent the number of charging operations, and we define R as the set of charging operations, denoted by {1, ..., n}.

Considering the worst-case scenario where the job durations are near the battery capacity, it becomes necessary to charge the AMR batteries after the completion of each transfer job. To accommodate this requirement, we establish the value of n to be equal to the number of transfer jobs ($n = |J|$). As discussed earlier, the charging operations can be viewed as specialized jobs that must be assigned to the AMRs, alongside the transfer jobs. These charging operations have a fixed duration of t, representing the time it takes to recharge the AMR batteries. However, it is worth noting that the charging time of the first charging operation on each AMR can be disregarded since the AMRs start the transfer operations with fully charged batteries.

An AMR (Autonomous Mobile Robot) have a battery that degrades to a greater or lesser extent depending on its usage. The degree of battery degradation indicates how much energy the AMR disperses while performing a job. As a result, a specific job $j \in J$ may require a different execution time depending on the AMR that performs it.

In the scheduling algorithm, each job has an energy requirement value (e_j) that indicates the amount of energy needed to complete it. This value is multiplied by the lambda ($\lambda_m \in [0,1], m \in M$) associated with the AMR executing the job, which is obtained from the ML model.

The lambda coefficient (λ_m) ranges between 0 and 1 and represents the health status of the AMR's battery. A lambda value of zero indicates that the AMR can perform a job without degrading the battery, while a value close to 1 indicates that the battery is deteriorating.

By doing so, the energy required to execute a task varies depending on the AMR performing it.

Based on this notation, we can introduce the following decision variables to formulate the mathematical model:

- The variable ${x_j}^m \in \{0,1\}$ is equal to 1 if transfer job $j \in J$ is performed by AMR $m \in M$, and 0 otherwise.
- The variable ${q_r}^m \in \{0,1\}$ is equal to 1 if charging job $r \in R$ is performed by AMR $m \in M$, and 0 otherwise.
- The variable ${y_{jr}}^m \in \{0,1\}$ is equal to 1 if job $j \in J$ is performed by AMR $m \in M$ after the charging job $r \in R$.
- The variable $C_{max} \in Z^+$ represents the completion time of the transfer process.
- The variable $E_{max} \in Z^+$ represents the maximum energy consumed by the AMR fleet.

We define the mathematical model of H-AMR-SP-BWC:

$$z = min\, E_{max} \quad (0.1)$$

$$z = min\, C_{max} \quad (0.2)$$

$$C_{max} \geq \sum_{j \in J} d_j x_j^m + \sum_{r \in R \setminus \{1\}} tq_r^m \quad \forall m \in M \quad (1)$$

$$E_{max} \geq \sum_{j \in J} \lambda_m e_j x_j^m \quad \forall m \in M \quad (2)$$

$$\sum_{m \in M} x_j^m = 1 \quad \forall j \in J \quad (3)$$

$$\sum_{r \in R} y_{jr}^m = x_j^m \quad \forall j \in J, m \in M \quad (4)$$

$$2y_{jr}^m \leq x_j^m + q_r^m \quad \forall j \in J, m \in M, r \in R \quad (5)$$

$$\sum_{j \in J} e_j y_{jr}^m \leq b \quad \forall m \in M, r \in R \quad (6)$$

$$q_r^m \leq q_{r-1}^m \quad \forall m \in M, r \in R \quad (7)$$

$$q_1^m = 1 \quad \forall m \in M \quad (8)$$

$$w_j x_j^m \leq w^m \quad \forall j \in J, m \in M \quad (9)$$

$$x_j^m \in \{0, 1\} \quad \forall m \in M, j \in J \quad (10)$$

$$q_r^m \in \{0, 1\} \quad \forall m \in M, r \in R \setminus \{1\} \quad (11)$$

$$y_{jr}^m \in \{0, 1\} \quad \forall l \in J, m \in M, r \in R \quad (12)$$

$$C_{max} \in R^+ \quad (13)$$

$$E_{max} \in R^+ \quad (14)$$

To define the mathematical model, we have the option to utilize two different objective functions. We employ objective function (0.2) when our aim is to minimize the makespan, while objective function (0.1) is employed to minimize the energy required to execute the most energy consuming job. It is easy to modify this objective function to minimize the overall energy consumption of the entire fleet. Constraints (1) guarantee the coherence between the duration of the transfer process for each AGV and the makespan. Similarly, constraint (2) enforces that the E_{max} variable is equivalent to the energy consumption of the most energy-intensive task. Constraints (3) requires that each transfer job must be assigned to a single AMR. As per constraints (4), when a transfer job (j) is assigned to an AMR (m), it is required that at least one charging job (r) preceding job (j) must also be assigned to the same AMR. Constraints (5) impose upper bounds on the variables $y_{jr}{}^m$, while constraints (6) define battery capacity limitations for each charging job. Constraints (7) introduce symmetry breaking to the model, while constraints (8) establish that the first charging job on each AMR must be equal to 1, assuming that all AMRs are expected to start the process fully charged. Constraints (9) ensure that each AMR does not exceed the maximum weight it can transport. If a job requires an AMR to carry a weight greater than its capacity, that particular job cannot be assigned to that AMR. Finally, the constraints (10) to (14) define the domain of the decision variables.

4.2 Routing Algorithm

This paragraph aims to address the associated routing problem that arises from the need to move a fleet of AMRs (Autonomous Mobile Robots) that must perform multiple consecutive jobs while minimizing waiting times.

The Vehicle Routing Problem (VRP) represents a well-known challenge within the realm of combinatorial optimization. Its primary objective is to identify the most efficient delivery route for a given set of customers. As an NP-hard problem, obtaining precise solutions is computationally intensive and often unfeasible. Traditionally, numerous custom-designed heuristic algorithms have been devised to approximate near-optimal solutions within reasonable timeframes.

In the scheduling problem discussed in the previous paragraph, an Autonomous Mobile Robot (AMR) performs a series of consecutive jobs and returns to the starting point in the warehouse after completing each job. However, our objective in the following paragraph is to redefine the problem in terms of routing. This means that the AMR will no longer be required to return to the starting point before moving on to the next job. Instead, it has the flexibility to perform different tasks without necessarily going back to the starting point. For instance, an AMR can initially pick up two different types of items that are compatible with its weight and the type of product it can transport. These items can be in the same warehouse or different ones. The AMR can then drop off these items at different packaging lines or even at the same line if needed.

We refer to the AMR routing problem in the context of a food packaging production line as the AMR Item Routing Problem (AMR-IRP). In this problem, a fleet of AMRs is deployed to serve the production line, and we formulate it as a Mixed Integer Linear Programming (MILP) problem. The objective is to minimize transportation times while ensuring the efficient movement of multiple items.

In the context of the AMR-IRP, certain aspects are influenced by the electric nature of the vehicle fleet, while others are derived from the characteristics of AMR autonomy. AMRs (Autonomous Mobile Robots) offer the advantage of operating on a continuous schedule, eliminating the restrictions imposed by driver shifts.

Within the AMR-IRP, a range of additional functionalities are integrated, comprising the following characteristics:

- Effective battery management to optimize power usage.
- Intermediate stops strategically positioned for AMR recharge.
- Potential variations in AMR capacity and initial battery inventories, allowing for diverse operational requirements.
- Flexibility in initial depot assignments, enabling AMRs to be situated at different depots.
- The option for AMRs to return to optional depots, enhancing operational flexibility.
- Absence of limitations on maximum route durations, providing freedom for efficient route planning.
- Integration of the total excess ride-time of items carried by AMRs as a weighted criterion in the objective function, ensuring optimal service quality.

In the AMR-IRP, each item has a specific origin and destination within the production line. The operational constraints that need to be satisfied include time windows, precedence relationships

between items, maximum transportation times for each item, and the maximum duration of AMR routes.

The AMR-IRP is defined on a directed complete graph, $G = (V, A)$. Here, V represents the set of vertices, while A represents the set of edges. We are specifically targeting the optimization challenges in the given context, which involves a fleet of k electric AMR denoted as $K = \{1, \ldots, k\}$. These AMRs are stationed at separate origin depots, indicated by $O = \{o_1, \ldots, o_k\}$ and exhibit diverse capacities, C^k(expressed in transportable kg), and they also have a maximum battery capacity Q. It is worth noting that each AMR may possess a unique initial battery inventory, B_{o_k}.

The primary objective of the AMR-IRP is to accomplish efficient transportation of n items. Each item is associated with a specific pickup location, $P_i \in P$, $P = \{P_1, \ldots, P_n\}$, and a corresponding dropoff location, $D_i \in D$, $D = \{D_1, \ldots, D_n\}$. Additionally, time windows $[arr_i, dep_i]$ are defined for each item, dictating the desired arrival times. Moreover, each item is subject to a maximum ride-time limit, u_i, imposing restrictions on the duration they can spend on board the AMRs.

The problem scope encompasses not only the origin depots, pickup, and drop-off locations but also introduces additional key elements to the graph, including charging stations, S, and optional destination depots, F. The AMRs have the flexibility to return to one of the optional destination depots in F, allowing for versatile routing options. Notably, the number of optional destination depots, F, can exceed the number of AMRs, extending the decision-making possibilities. Within the graph, each location, $i \in V$, undergoes dynamic changes in load l_i, with positive values representing pickups ($l_i > 0$), negative values indicating dropoffs ($L_i < 0$), and zero values elsewhere ($L_i = 0$). Moreover, pickup and dropoff locations are characterized by non-zero service times, d_i, representing the time required for AMRs to handle loading and unloading operations. Considering the energy aspect, charging stations in S can only be accessed by empty AMRs and are distinguished by their recharge rates, α_s. These rates indicate the amount of energy transferred per unit time and reflect the charging speed, ranging from fast to slow charging. Partial recharging is also possible during visits to the charging stations, optimizing the utilization of the available energy resources.

In terms of battery management, it is essential to ensure that the AMR's battery levels do not fall below a predefined minimum State of Charge (SOC) upon arrival at any optional destination depot, $f \in F$, by the end of the planning horizon, T_p. This control over battery levels is achieved through a minimum battery level ratio, r, ensuring sufficient energy reserves for subsequent operations. This allows for effective planning and scheduling of the AMR's' routes, maintaining operational continuity.

The graph's edges in A represent travel times, t_{ij}, between any two locations, $i, j \in V$; $i \neq j$. Battery consumptions, β_{ij}, associated with these travel times can be estimated using an energy consumption model that considers various factors, including distance, terrain, and speed.

To model the AMR-IRP, we introduce decision binary variables, ${x^k}_{ij}$, which indicate whether AMR k sequentially visits locations i and $j \in V$. The variable ${T^k}_i$ denotes the time at which AMR k starts service at location $i \in V$, ${L^k}_i$ represents the load of the AMR after serving location i, and ${B^k}_i$ indicates the initial battery state of the AMR at the beginning of service. Recharge time for AMR k at charging station $s \in S$ is represented by ${E^k}_s$. R_i captures the excess time associated with item $i \in P$.

In summary, the AMR-IRP seeks to optimize routing decisions to minimize costs while adhering to

various constraints, including time-window restrictions, AMR capacities, and battery limitations. The objective function comprises a weighted combination of the total travel time for all AMRs and the excess ride-time of all items. Table 3 provides a comprehensive summary of the problem sets, parameters, and decision variables specific to the AMR-IRP.

Table 3 Example Parameters of an AMR

P	$P = \{1, \ldots, n\}$ Set of pickup locations.
D	$D = \{n+1, \ldots, 2n\}$ Set of dropout locations.
N	$N = P \cup D$ Set of pickup and dropoff locations.
K	$K = \{1, \ldots, k\}$ Set of available AMRs.
O	Set of origin depots for AMRs $k \in K, o_k$.
F	Set of all available destination depots.
S	Set of all charging stations.
V	$V = N \cup O \cup F \cup S$.
t_{ij}	Travel time from location $i \in V$ and $j \in V, i \neq j$.
arr_i	Earliest time at which service can begin at $i \in V$.
dep_i	Latest time at which service can begin at $i \in V$.
d_i	Service duration at location $i \in V$.
l_i	Change in load at location $i \in N$.
u_i	Maximum ride-time for item with pickup at $i \in P$.
C^k	Capacity of AMR $k \in K$.
Q	Maximum battery capacity.
${B^k}_0$	Initial battery capacity of AMR $k \in K$.
r	Minimum battery level ratio.
β_{ij}	Battery consumption between nodes $i, j \in V, i \neq j$.
α_s	Recharge rate at charging facility $s \in S$.
T_p	Planning horizon.
${x^k}_{ij}$	Binary variables 1 if vehicle k sequentially stops at location i and $j \in V$, 0 otherwise.

$T^k{}_i$	Time at which vehicle k starts its service at location $i \in V$.
$L^k{}_i$	Load of vehicle k at location $i \in V$.
$B^k{}_i$	Battery load of vehicle k at location $i \in V$.
$E^k{}_s$	Charging time of vehicle k at charging station $s \in S$.
R_i	Excess ride-time of item $i \in P$.

We report below the mathematical model:

$$z = min\, w_1 \sum_{k \in K} \sum_{i,j \in V} t_{ij} x^k{}_{ij} + w_2 \sum_{i \in P} R_i \qquad (1)$$

$$\sum_{j \in P \cup S \cup F} x^k{}_{o^k j} = 1 \qquad \forall\, k \in K \qquad (2)$$

$$\sum_{j \in F} \sum_{i \in D \cup S \cup \{o^k\}} x^k{}_{ij} = 1 \qquad \forall\, k \in K \qquad (3)$$

$$\sum_{k \in K} \sum_{i \in D \cup S \cup \{o^k\}} x^k{}_{ij} \leq 1 \qquad \forall\, j \in F \cup S \qquad (4)$$

$$\sum_{j \in V,\, i \neq j} x^k{}_{ij} - \sum_{j \in V,\, i \neq j} x^k{}_{ji} = 0 \qquad \forall\, k \in K,\ \forall\, i \in N \cup S \qquad (5)$$

$$\sum_{k \in K} \sum_{j \in N,\, i \neq j} x^k{}_{ij} = 1 \qquad \forall\, i \in P \qquad (6)$$

$$\sum_{j \in N,\, i \neq j} \sum_{j \in V,\, i \neq n+i} x^k{}_{jn+i} = 1 \qquad \forall\, k \in K, \forall\, i \in P \qquad (7)$$

$$T^k{}_i + d_i + t_{i,n+i} \leq T^k{}_{n+i} \qquad \forall\, k \in K, \forall\, i \in P \qquad (8)$$

$$arr_i \leq T^k{}_i \leq dep_i \qquad \forall\, k \in K, \forall\, i \in V \qquad (9)$$

$$T^k{}_{n+i} - T^k{}_i - d_i \leq u_i \qquad \forall\, k \in K, \forall\, i \in P \qquad (10)$$

$$T^k{}_i + t_{ij} + d_i - M_{ij}(1 - x^k{}_{ij}) \leq T^k{}_j \qquad \forall\, k \in K, \forall\, i \in V$$
$$j \in V,\ i \neq j | M_{ij} > 0 \qquad (11)$$

$$R_i \geq T^k{}_{n+i} - T^k{}_i - d_i - t_{i,n+i} \qquad \forall\, k \in K, \forall\, i \in P \qquad (12)$$

$$L^k{}_i + l_j - G^k{}_{ij}(1 - x^k{}_{ij}) \leq L^k{}_j \qquad \forall\, k \in K, \forall\, i \in V j \in V,\ i \neq j \qquad (13)$$

$$L^k_i + l_j - G^k_{ij}(1 - x^k_{ij}) \geq L^k_j \qquad \forall k \in K, \forall i \in Vj \in V,\ i \neq j \qquad (14)$$

$$L^k_i \geq max(0, l_i) \qquad \forall k \in K, \forall i \in N \qquad (15)$$

$$L^k_i \leq min(C^k, C^k + l_i) \qquad \forall k \in K, \forall i \in N \qquad (16)$$

$$L^k_i = 0 \qquad \forall k \in K,\ i \in \{o^k\} \cup F \cup S \qquad (17)$$

$$B^k_i = B^k_0 \qquad \forall k \in K,\ i \in \{o^k\} \qquad (18)$$

$$B^k_j \leq B^k_i - \beta_{ij} + Q(1 - x^k_{ij}) \qquad \forall k \in K,\ i \in V\backslash S$$
$$j \in V\backslash\{o^k\}, i \neq j \qquad (19)$$

$$B^k_j \geq B^k_i - \beta_{ij} + Q(1 - x^k_{ij}) \qquad \forall k \in K,\ i \in V\backslash S$$
$$j \in V\backslash\{o^k\}, i \neq j \qquad (20)$$

$$B^k_j \leq B^k_s + \alpha_s E^k_s - \beta_{sj} + Q(1 - x^k_{sj}) \qquad \forall k \in K, \forall s \in S,$$
$$j \in P \cup F \cup S,\ s \neq j \qquad (21)$$

$$B^k_j \geq B^k_s + \alpha_s E^k_s - \beta_{sj} + Q(1 - x^k_{sj}) \qquad \forall k \in K, \forall s \in S,$$
$$j \in P \cup F \cup S,\ s \neq j \qquad (22)$$

$$Q \geq B^k_s + \alpha_s E^k_s \qquad \forall k \in K, s \in S \qquad (23)$$

$$B^k_i \geq rQ \qquad \forall k \in K, i \in F \qquad (24)$$

$$E^k_s \leq T^k_s - t_{is} - T^k_i + M^k_{is}(1 - x^k_{is}) \qquad \forall k \in K, s \in S$$
$$i \in D \cup S \cup \{o^k\}, i \neq s \qquad (25)$$

$$E^k_s \geq T^k_s - t_{is} - T^k_i + M^k_{is}(1 - x^k_{is}) \qquad \forall k \in K, s \in S$$
$$i \in D \cup S \cup \{o^k\}, i \neq s \qquad (26)$$

$$B^k_i \geq 0 \qquad \forall k \in K,\ \forall i \in V \qquad (28)$$

$$E^k_s \geq 0 \qquad \forall k \in K,\ \forall s \in S \qquad (30)$$

The specified constraints address different aspects within the routing model. Constraints (2) guarantee that all AMRs depart from their respective origin depots follow specific actions: visiting a pickup location in P or visiting a charging station in S or proceeding to a destination depot F. Constraints (3) ensure that all vehicles return to a destination depot. It is important to note that when considering the set F, which includes the origin depots O, AMRs that are not in use will travel between their respective origin depots $o^k \in O$ and the corresponding destination depot $o^{-k} \in F$. Consequently, when $x_{o^k o^{-k}} = 1$, it implies that vehicle k is not actively utilized.

Constraints (4) ensure that each optional destination depot and charging station can be visited by the vehicles, with the condition that each of these locations is visited at most once. These nodes can be duplicated to permit numerous visits to nodes in F and S. The set of constraints (5) models the conservation of flow. Constraints (6) and (7) guarantee that each pickup node is visited precisely once and that every pickup-dropoff pair is serviced by the same AMR. Timing constraints are included to establish service start times and additional ride times. Constraints (8) ensure that pickup node i is visited prior to its corresponding dropoff node $n + i$, based on the direct travel time between the two nodes and the service duration at node i. Constraints (9) create time windows around the start of the service at each node, and constraints (10) enforce maximum ride times for the carried items.

Constraints (11) establish a lower limit on the service start time at node $j \in V$, which is visited immediately after node $i \in V$, where $M_{ij} = max\{o, dep_i + d_i + t_{ij} - arr_j\}$. Constraints (12) compute the excess ride time for the item i, where the travel time $t_{i,n+i} + d_i$ for item $i \in P$ is subtracted from ${T^k}_{n+i} - {T^k}_i$. Constraints (13) and (14) calculate the load at location $j \in V$ based on the load at the preceding location $i \in V$ and the change in load at location j (${G^k}_{ij} = min(Ck, Ck + li)$). While Constraints (14) are redundant in terms of the model formulation, they strengthen the LP relaxations. Constraints (15) and (16) establish lower and upper bounds on AMR occupancy, respectively, and Constraints (17) ensure that vehicles are empty at depots and charging stations.
Battery management constraints are added to monitor battery levels during travel between nodes and recharging periods. AMRs also have initial battery levels and must maintain minimum battery levels at the end of the planning horizon. Constraints (18) set initial battery levels for AMRs at origin depots. Constraints (19) and (20) establish the battery level state from any node $i \in V \backslash S$ to any node $j \in V \backslash \{o^k\}$. Constraints (21) and (22) establish the battery level state after visiting a charging node $s \in S$ at any location $j \in P \cup F \cup S$. Constraints (23) set upper bounds on the battery level states at charging stations, and Constraints (24) impose minimum battery levels for all vehicles returning to depots. Constraints (25) and (26) determine upper and lower bounds on the recharge time at charging station $s \in S$. Constraints (27) to (29) set integrality and non-negativity constraints.

4.3 Machine-Learning and Metamodels

4.3.1 Introduction

In the CPSoS paradigm, the machine learning algorithms have core roles related to other modeling techniques, creating the possibility of implementing metamodels.

Metamodels can be identified and trained based on optimization techniques and under specific conditions that depend on the vertical use case.

Utilizing machine learning-based meta-models provides a promising but challenging path for enhancing real-time decision-making capabilities in complex systems.

Motivations for Using Meta-Models Based on Machine Learning could be:

1. **Computational Efficiency**: ML-based meta-models can make rapid predictions without the need to solve complex optimization algorithms in real time, a vital requirement for applications with stringent time constraints.
2. **Scalability**: As the system grows, traditional models may struggle to keep up. ML-based meta-models adapt more easily to increased dimensions and variables.
3. **Adaptability**: Machine learning models can continuously update themselves based on the latest data, making them more robust to changes in system behavior.
4. **Generalization**: Properly trained meta-models can generalize well to new, unseen conditions, making them versatile tools for decision-making.
5. **Resource Allocation**: These models can be effective in dynamically allocating resources where they are most needed, especially in multi-agent systems like automated robotics in manufacturing.
6. **Interoperability**: Machine learning meta-models can be integrated with other decision-

support systems, enhancing their utility and flexibility.

On the other side, challenges in Using ML-Based Meta-Models:

1. **Data Dependency**: The quality of the meta-model is highly dependent on the data it's trained on. Inadequate or biased data can result in poor decision-making.
2. **Complexity**: The internal workings of many machine learning algorithms are not easily interpretable, leading to a "black box" problem that may be unacceptable in critical applications.
3. **Training Time**: Initially, machine learning models may require significant time and computational resources for training.
4. **Validation**: Ensuring the meta-model accurately captures the optimization logic it's approximating can be a challenging validation problem.
5. **Latency**: While faster than solving optimization problems in real time, the time required to make a prediction still needs to be accounted for in near-real-time applications.

In several cases, to support decision-making requires the use of optimization and/or simulation techniques. These might result in intensive or unsustainable computational effort, especially to support near-real time and real-time.

For example, such models can be used to coherently predict the state of the system to enable the "next decision" or adjust/correct a decision made at the previous step(s).

Let's, for example, consider a manufacturing scenario where automated robots support intensive intra-logistics operations.

Considering a certain current status of the system and a list of tasks to be performed in the next time frame, it could be useful to predict the saturation of certain buffers and automated resources, which depend on the logic used to schedule the automated resources and their tasks (for instance an optimization algorithm). In certain cases, to correctly predict the status, the "predictor" needs to "simulate" the effect of the optimization logic in near-real time. Such a task can be incompatible with the timing and the computational resources.

In this case, the concept of metamodel based on a learning process can be extremely useful. The metamodel algorithm represents, with a proper level of accuracy, the way the optimization logic acts in the real system but with a higher computational efficiency. So, the meta-model trained on the "behavior of the optimizer" constitutes a faster way to support the prediction.

In this section, we introduce, as an example fitting the SN use-case, a metamodel tailored specifically to address the complex and computationally challenging VRP, which presents greater intricacies compared to the scheduling problem discussed in Chapter 4.1. The core objective of this meta model lies in its ability to leverage advanced Machine Learning techniques to efficiently navigate and optimize the intricate routing scenarios outlined in Chapter 4.2.

The vehicle routing problem poses a significant computational burden due to its vast solution space and the need to find the most optimal routes for the Autonomous Mobile Robots (AMRs) to accomplish their assigned tasks effectively. Traditional methods may struggle to provide efficient solutions in a reasonable timeframe, making it imperative to explore novel approaches to tackle this demanding

problem.

To overcome these computational challenges, our metamodel combines the power of Machine Learning with math-optimization principles. By integrating intelligent algorithms and leveraging historical data, the meta model can rapidly evaluate numerous potential routes for the AMRs. This allows it to identify high-quality solutions swiftly, providing a practical and time-efficient approach to resolving the vehicle routing problem. The Machine Learning component empowers the metamodel with adaptability and enhanced decision-making capabilities. By learning from past routing solutions and dynamically adjusting to changing conditions, the model can continually improve its performance and adapt to diverse scenarios.

With the meta model's assistance, our dispatching system gains the capability to efficiently manage and optimize complex task assignments. By swiftly generating near-optimal routes for the AMRs, we ensure seamless coordination and timely task completion, ultimately elevating the overall effectiveness and productivity of the entire dispatching process.

Neural Networks (NN) have been applied to solve combinatorial optimization problems since the 1980s, particularly with the pioneering work of Hopfield and Tank [40], who developed the Hopfield network to tackle the Travelling Salesperson Problem (TSP). Early research predominantly focused on the TSP, exploring two main approaches: Hopfield networks and self-organizing feature maps [41]. However, recent years have witnessed a resurgence in applying NNs to combinatorial optimization, thanks to significant advancements in deep learning.

Vinyals et al. [42] introduced the Pointer Network (Ptr-Net), a sequence-to-sequence model that employs a Recurrent Neural Network (RNN) encoder and decoder with an attention mechanism to generate permutations of input sequences. The model is trained in a supervised manner and applied to various problems, such as convex hull, Delaunay triangulation, and the TSP. Test-time solutions to the TSP are obtained using a beam search procedure.

In the realm of recent advancements, notable works include Bello et al. [43], who made significant strides by enhancing the Ptr-Net using reinforcement learning in an unsupervised manner. Building on this progress, Nazari et al. [44] also delved into training a Ptr-Net, adopting a unique approach to further push the boundaries of its capabilities. Moreover, Sheng et al. [45] made a groundbreaking contribution by introducing a Ptr-Net tailored specifically for the VRP with Task Priority and Limited Resources. Their innovative approach involves leveraging a strategy gradient method to estimate gradients based on the benefits calculated from two consecutive batches of training data. These recent works have substantially expanded the possibilities of utilizing Ptr-Net in addressing complex combinatorial optimization challenges.

4.3.2 Data

As part of this deliverable, we will develop a novel approach for generating the dataset for the AMR-IRP. This innovative strategy aims to create a dataset tailored to evaluate the performance of the machine learning models and optimization techniques we are developing. The intention is to design a unique and distinctive dataset that showcases ideas and methodologies.

To construct instances of the AMR-IRP, we will utilize commercial solvers like Gurobi or CPLEX or LocalSovler, or OR-Tools. These solvers provide robust optimization capabilities for complex problems,

ensuring high-quality solutions. By the solver, we will efficiently solve the AMR-IRP instances and generate a diverse and comprehensive dataset for training and testing.

During the dataset generation process, we will introduce several random elements to simulate real-world scenarios. We will randomly select n nodes from the coordinate grid S = {xi}, where each xi ∈ [0, 1000]². Among these nodes, some will be designated as pickup locations, while others will be drop-off locations. Additionally, we will randomly choose charging stations to be included in the dataset. We also add a special token qi to each node to tell to the network if the node is a depot or not

Moreover, to reflect the variability of AMR capabilities in real-world settings, we will assign random battery capacities to each AMR. The battery capacity will play a crucial role in their ability to perform tasks and navigate through the AMR-IRP instances efficiently.

These random elements contribute to the diversity of the dataset, allowing our machine learning models to adapt to different scenarios and handle the uncertainties that might arise in actual AMR routing operations.

4.3.3 Metamodel

The proposed neural network, AMRNet, exhibits a message passing mechanism, wherein it computes messages between nodes and updates the hidden states of these nodes across T iterations. During iteration t, each node i in the graph transmits messages to its neighboring nodes, conveying its current state. The computation of messages takes place concurrently for all nodes. Subsequently, each node assimilates all incoming messages and updates its hidden state, transitioning from ${h^{t-1}}_i$ to ${h^t}_i$, based on the received information. Following this, the network calculates new messages founded on the updated hidden states and repeats the process iteratively. The ultimate output of the network is derived from the hidden states of the nodes, as proposed by Palm et al. [46].

The Recurrent Relational Network (RRN) architecture allows for the computation of either individual outputs for each node or a single output for the entire graph. In our case, we are specifically interested in generating the probabilities $p(e_{ij})$ of each edge e_{ij} being either active or inactive. To accommodate this objective, we adapt the architecture to include hidden states (${e^t}_{ij}$) for the edges as well. These edge states are updated based on the hidden states of the pair of nodes connected by the respective edge. As a result, the final output is computed using the hidden states of the edges.

Our hypothesis suggests that the relational reasoning capabilities of the RRN will provide effective representations of the solutions to the AMR-IRP. This is achieved by implicitly learning the sub-route associations of each node. Consequently, the hidden state of a node will represent the likelihood of it belonging to a specific sub-route. As the confidence of a node belonging to a particular sub-route increase, it communicates this information to its neighboring nodes, thereby influencing their likelihoods accordingly. By discerning the most probable routes for each node, the network can make more informed decisions when predicting the edge probabilities in the output, surpassing the mere utilization of relative positions of the nodes.

In our approach to tackle the AMR-IRP, we introduce AMRNet, a powerful machine learning model designed to receive input in the form of a graph that precisely describes an instance of AMR-IRP. The input graph is characterized by being fully-connected, where every AMR node is interconnected with all other nodes in the graph. Each AMR node i is equipped with an input feature vector $z_i = [x_i\ q_i]$,

which encompasses the normalized coordinates and a token signifying whether the node represents a pickup or drop-off location. Additionally, each edge e_{ij} connecting nodes i and j is associated with its normalized length d_{ij}, serving as an essential input component. To initiate the process, the hidden states of both nodes and edges are initialized to zero, ${h^0}_i = 0$, and ${e^0}_{ij} = 0$.

4.3.3.1 Message Passing Phase

At iteration t, every node possesses a hidden state ${h^t}_i$. During this phase, nodes exchange messages with their neighboring nodes, facilitating efficient communication and information sharing. The message ${m^t}_{ij}$ from node i to node j, computed at iteration t, incorporates the hidden states of the nodes ${m^t}_{ij} = f({h^{t-1}}_i, {h^{t-1}}_j, d_{ij})$. The function f, is implemented as a Multilayer Perceptron (MLP) with ReLU activation function in its input and hidden layers, empowers the network to learn the most effective messages to transmit. Additionally, batch normalization and dropout techniques with a probability of 0.2 are applied to the first two layers, enhancing the model's robustness through regularization. Each node j takes into account every incoming message it receives by aggregating the messages from its neighborhood $N(j)$, which encompasses all other nodes in the fully-connected graph. For a node j all messages are aggregated in the following way: ${m^t}_j = \sum_{i \in N(j)} {m^t}_{ij}$.

4.3.3.2 Node Updating Phase

After gathering all the incoming messages, each node's hidden state ${h^t}_j$ is updated using the learned Recurrent Neural Network (RNN) update function g_n, implemented as a Gated Recurrent Unit (GRU) module. This update function considers the previous iteration's hidden state (${h^{t-1}}_j$), the aggregated message ${m^t}_j$, and the node's input feature vector z_j. The provision of the input feature vector at each iteration enables the update function to focus solely on processing incoming messages, rather than memorizing the original input, thereby enhancing its efficiency. So, we define ${h^t}_j = g_n({h^{t-1}}_j, {m^t}_j, z_j)$.

4.3.3.3 Edge Updating Phase

Ascertaining the states of individual nodes is not sufficient to solve the AMR-IRP effectively; we must also determine the probability of each edge e_{ij} in the graph being either active or inactive. To address this, an idea is to extend the RRN architecture of Palm et al. [46] to incorporate hidden states for each edge. Consequently, the probabilities of the edges are computed based on these hidden states.

Like the node updates, the hidden states of each edge ${e^t}_{ij}$ are updated through a learned update function g_e, which also employs the GRU module. The update function takes as input the hidden states of the nodes connected by the edge (${h^t}_i$ and ${h^t}_j$) and the edge's input feature vector (d_{ij}). The inclusion of the edge input feature vector at each iteration ensures that the update function g_e does not need to remember the original input, thus optimizing its performance. So, we set $g_e = ({e^{t-1}}_{ij}, {h^t}_i, {h^t}_j, d_{ij})$.

To determine the probabilities of edges being active or inactive, we use an output function o. This function takes the hidden state of each edge, denoted as ${e^{t-1}}_{ij}$ and produces a 2-feature vector ${p^t}_{ij} = o({e^t}_{ij})$, with two values in output representing two probabilities. One value represents the probability of the edge being inactive, and the other value represents the probability of the edge being

active. To compute these probabilities, we use a three-layer neural network known as a Multilayer Perceptron (MLP). The first two layers of this MLP use the Rectified Linear Unit (ReLU) activation function. We also apply batch normalization and dropout with a probability of 0.2 to the first two layers, which helps the model generalize better and prevents overfitting.

The raw output values $p^t{}_{ij}$ are then transformed into probabilities using the softmax nonlinearity, which ensures that the probabilities fall within the range of 0 to 1. This step is crucial to obtain meaningful and accurate probabilities for each edge.

4.3.3.4 Loss

The output target of AMRNet is represented by the solved AMR-IRP's adjacency matrix, denoted as $y = \{y_{ij}\}^n{}_{i.j=1}$, where n is the number of nodes in the respective AMR problem. Each entry y_{ij} in the adjacency matrix is binary, indicating whether the edge connecting nodes i and j is active (1) or inactive (0). In essence, this binary classification problem requires AMRNet to learn to classify each edge as either active or inactive. To achieve this, we train the network using cross-entropy loss over mini-batches of the dataset.

To balance the class distribution during the loss computation, particularly when dealing with sparse adjacency matrices where inactive edges outnumber active ones, we apply appropriate class weights, following the approach proposed by Joshi et al. [47]. During training, we minimize the loss at each iteration t of the network, as suggested by Palm et al. [46]. This approach encourages AMRNet to learn a convergent message passing algorithm, while mitigating the vanishing gradient problem. However, during validation, we only consider the output of the final iteration of the network.

4.3.3.5 Beam Search

To convert the probabilistic heat-map outputted by AMRNet into valid solutions for the AMR-IRP, we implement a beam search decoder. This search algorithm, based on limited-width breadth-first search, efficiently identifies high-probability routes by sampling a subset of possible routes on the graph. The beam search starts from the depot node and expands the b most probable edge connections in the depot's neighborhood. Subsequently, it iteratively expands the top-b most probable partial routes π' until all routes have visited every node.

To ensure the construction of valid routes without revisiting nodes, we adopt a masking strategy, similar to that proposed by Joshi et al. [47] for TSPs. However, to accommodate AMR-IRP's multiple visits to the depot, we modify the masking strategy to prevent the immediate masking out of the depot node once visited. Instead, we implement a counter to keep track of the number of visits to the depot, masking it out only when it has been visited v times. Two different strategies are used to choose the final output of the beam search decoder:

Vanilla beam search: The output of vanilla beam search is the complete route with the highest probability at the end of the search. This route might not necessarily minimize the longest sub-route, but it is useful for fast validation during training to track the improvement of heat-maps output by AMRNet.

Shortest beam search: The output of shortest beam search is the solution that minimizes the longest

sub-route. While this approach takes more time compared to vanilla beam search, as it evaluates the length of the longest sub-route for all b complete routes, it is used for the final evaluation of the trained models.

4.4 Opportunities of Asynchronous Learning couple with Simulation / Digital Twin

Detailed simulation models provide a high-fidelity system representation, often capturing complexities related to production, scheduling, and logistics [45]. They're used for optimization and for predicting KPIs such as throughput and resource utilization.

A simulator, for example, a discrete-event and stochastic simulator, of a manufacturing facility, including intra-logistics, serves multiple scopes such as dimensioning or representing the black-box of a metaheuristic or math-heuristics optimization process to evaluate in advance the performance (KPI) of the production or verify the capacity of the production to deliver the orders on time.

While highly accurate, these simulations can be computationally expensive and time-consuming, making them impractical for real-time decision-making or optimization that requires numerous simulations runs.

In other words, in some cases, the computational time/effort, which includes multiple replications, is heavy to respect the timing requirements and sustainability in terms of costs.

In such cases, the identification of metamodels, based on machine learning algorithms trained on input and output of the simulator, can represent a good approximation of the model enabling faster solution. Algorithms such as Random Forests, Neural Networks, and Support Vector Machines are often used for training the metamodel. They capture the mapping between input variables (e.g., order volume, resource allocation) and output variables (e.g., throughput, delay.

In the zero-warm approach, a metamodel of the Systems or subsystems, in the CPSoS logics, can be trained, levering the Asynchronous mechanics and, eventually, the Federation Learning concepts described in the deliverable.

A similar concept is valid in the case of the system, which is available as a Digital Twin (DT). The metamodel can be trained on the Digital Twin instead on the real-system data, with some potential advantages:

1. **Data Availability**: DTs often collect and store vast amounts of data that are more readily available for training algorithms, unlike real systems where collecting data can be costly or disruptive.
2. **Data Richness**: DTs often have comprehensive data on variables that are not easily observable in the real world, which can lead to a more robust metamodel.
3. **Experimentation**: With a DT, you can experiment freely without affecting real-world operations, allowing for more extensive data collection under varied conditions.
4. **Speed**: DTs can often be run faster than real-time, enabling quicker data collection and, thus faster training and validation of metamodels.
5. **Risk Mitigation**: Using a DT avoids the risks and costs associated with experimenting on a real-world system, such as downtime, wear-and-tear, or safety concerns.
6. **Consistency**: DTs can ensure a level of consistency in the data by controlling for external variables, making it easier to train and validate a metamodel.

5 Conclusions

Summarizing the key insights and contributions of the document, this deliverable underscores the significance of self-learning modules for robotic and human behaviors, in particular asynchronous learning combined with cloud-to-edge intelligent agents' distribution. The key driving element is enabling optimal real-time and near-real-time decision-making.

This innovative approach can potentially revolutionize the manufacturing landscape, offering more efficient and adaptive solutions.

In comparison to the existing state of the art, we have explored and introduced novel key elements for managing asynchronous and federation learning. Additionally, we explore the possibility of learning processes to develop metamodels, which make math optimization and simulation models closer to the operational computational performances needed in real industrial applications.

These elements promise to enhance the scalability and adaptability of manufacturing systems, making them better equipped to handle the complexities of modern production environments.

Within the context of our use cases, we have defined a robust mathematical model that represents the intelligent agent responsible for job scheduling. This model not only improves task allocation but also accounts for critical factors such as battery level prediction, which relies on distributed machine learning techniques. This integration seamlessly fits into the broader fleet management scenario, enhancing operational efficiency and resource utilization.

Moreover, our exploration has revealed that machine learning-based metamodels provide a promising but challenging path for enhancing real-time decision-making capabilities within complex systems. As illustrated in the document, the zero-swarm approach leverages learning-based meta-modeling, taking full advantage of asynchronous learning. This novel approach empowers manufacturing systems to adapt swiftly to changing demands and optimize their performance efficiently.

In conclusion, the zero-swarm approach represents a significant leap forward in the field of manufacturing operations. By combining asynchronous learning, advanced mathematical modeling, and machine learning-based metamodels, it paves the way for more efficient, responsive, and adaptable manufacturing systems, ultimately driving the industry toward greater innovation and competitiveness. This transformative framework capitalizes on the strengths of localized edge computing and robust cloud resources, addressing challenges related to resource management, algorithm selection, and network connectivity while ensuring manufacturing systems can thrive in the modern era.

Bibliography

[1] P. Lea, IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security, 2nd Edition, Packt Publishing, 2020.

[2] F. Al-Turjman, Edge Computing: From Hype to Reality, Springer International Publishing, 2018.

[3] L. a. D. P. a. L. I. Amorosi, Optimization in Artificial Intelligence and Data Sciences: ODS, First Hybrid Conference, Springer International Publishing, 2022.

[4] R. C. a. M. D. T. Steclik, Automatic grouping of production data in Industry 4.0: The use case of internal logistics systems based on Automated Guided Vehicles, vol. 62, p. 101693,, Journal of Computational Science, 2022.

[5] M. V. a. F. D. M. De Ryck, Automated guided vehicle systems, state-of-the-art control algorithms and techniques, Journal of Manufacturing Systems, vol. 54, pp. 152-173,. Available: 10.1016/j.jmsy.2019.12.002., 2020.

[6] M. N. a. R. Z. J. Mehami, Smart automated guided vehicles for manufacturing in the context of Industry 4.0, Procedia Manufacturing, vol. 26, pp. 1077-1086, 2018.

[7] I. R. A. E. I. A. M. U. M. B. M. a. B. A. Chaudhry, Integrated scheduling of machines and automated guided vehicles (AGVs) in flexible job shop environment using genetic algorithms. International Journal of Industrial Engineering Computations, 13(3), pp.343-362., 2022.

[8] W. L. a. A. Z. X. Liu, PNGV Equivalent Circuit Model and SOC Estimation Algorithm for Lithium Battery Pack Adopted in AGV Vehicle, IEEE Access, vol. 6, pp. 23639-23647, 2018.

[9] C.-Y. L. J. Z. L. Pian, Estimation and simulation of charged state of battery for AGV, International Journal of Mechatronics and Applied Mechanics, vol. 2, Issue 6, pp. 120 – 126, 2019.

[10] A. B. D. T. N. A. a. K. B. M. Abderrahim, Manufacturing 4.0 Operations Scheduling with AGV Battery Management Constraints, Energies, vol. 13, no. 18, p. 4948, 2020.

[11] L. Z. W. X. a. K. W. Y. Lian, An Improved Heuristic Path Planning Algorithm for Minimizing Energy Consumption in Distributed Multi-AGV Systems, International Symposium on Autonomous Systems (ISAS), 2020.

[12] F. L.-A. C. a. V. F. Rubio, Multi-objective optimization of costs and energy efficiency associated with autonomous industrial processes for sustainable growth., Technological Forecasting and Social p.121115. Change, 173, p.121115, 2021.

[13] H. L. a. L. P. H. Zhang, Study on Li-Ion Battery Intelligent Management System Based on AGV, Applied Mechanics and Materials, vol. 651-653, pp. 1101-1104, 2014.

[14] O. P. a. R. S. M. Medykovskvi, Use of Machine Learning Technologys for the Electric Consumption Forecast, IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 2018.

[15] X. J. Q. H. S. F. a. K. L. H. Hu, Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0, Computers Industrial Engineering, vol. 149, p. 106749, 2020.

[16] M. E. a. M. Tran, Development of an IoT Architecture Based on a Deep Neural Network against Cyber Attacks for Automated Guided Vehicles, Sensors, vol. 21, no. 24, p. 8467, 2021.

[17] S. S.-G. J. M. A. a. P. A. Vakaruk, Forecasting Automated Guided Vehicle Malfunctioning with Deep Learning in a 5G-Based Industry 4.0 Scenario., IEEE Communications Magazine, 59(11), pp.102-108, 2021.

[18] Yuan et al., MU R-CNN: A Two-Dimensional Code Instance Segmentation Network Based on Deep Learning, Future Internet, vol. 11, no. 9, p. 197, 2019.

[19] A. Z. D. Z. a. M. D. R. Cupek, Determination of the machine energy consumption profiles in the mass-customised manufacturing, International Journal of Computer Integrated Manufacturing, vol. 31, no. 6, pp. 537-561, 2017.

[20] Ł. G. a. A. Z. R. Cupek, An OPC UA Machine Learning Server for Automated Guided Vehicle, Computational Collective Intelligence, pp. 218-228, 2020.

[21] Unified automation. NET based OPC UA Client/Server SDK V3.2.1 — Service Release, 2022.URL: https://www.unified-automation.com/.

[22] 1. McMahan et al. 2017.

[23] N. e. a. 2. [https://arxiv.org/pdf/1710.06963.pdf.

[24] D. e. all, 2022, 5910--5924 [https://arxiv.org/pdf/2202.08312.pdf.

[25] K. e. a. 2. [https://arxiv.org/pdf/1610.05492.pdf.

[26] S. a. J. Koloskova, 17202--17215 https://openreview.net/pdf?id=4_oCZgBIVI, 2022.

[27] K. a. G. Xie, [https://arxiv.org/pdf/1903.03934.pdf, 2019.

[28] C. e. al., 15-24, https://arxiv.org/pdf/1911.02134.pdf, 2020.

[29] K. a. G. Xie, https://arxiv.org/pdf/1903.03934.pdf, 2019.

[30] C. e. al., 15-24 https://arxiv.org/pdf/1911.02134.pdf, 2020.

[31] N. e. al., https://arxiv.org/pdf/2106.06639.pdf, 2022.

[32] Z. e. al., https://arxiv.org/pdf/2206.10032v2.pdf, 2022.

[33] K. e. al., https://arxiv.org/pdf/1610.05492.pdf, 2016.

[34] K. e. al., 1–210 https://arxiv.org/pdf/1912.04977.pdf, 2021.

[35] H. e. al., 814-832 https://arxiv.org/pdf/2111.04877.pdf, 2022.

[36] X. e. al., https://arxiv.org/pdf/2109.04269.pdf, 2021.

[37] K. e. al., https://arxiv.org/pdf/1610.05492.pdf, 2016.

[38] S. e. al., https://arxiv.org/pdf/2107.10996.pdf, 2021.

[39] K. e. al., https://arxiv.org/pdf/1912.04977.pdf, 2021.

[40] J. J. H. a. DavidWTank, “neural” computation of decisions in optimization problems., Biological cybernetics, 52(3):141–152, 1985.

[41] K. A. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research., INFORMS Journal on Computing, 11(1):15–34, 1999.

[42] M. F. a. N. J. Oriol Vinyals, Pointer networks. arXiv preprint arXiv: 1506.03134, 2015..

[43] H. P. Q. V. L. M. N. a. S. B. Irwan Bello, Neural combinatorial optimization with reinforcement learning., arXiv preprint arXiv:1611.09940, 2016.

[44] A. O. L. V. S. a. M. T. Mohammadreza Nazari, Reinforcement learning for solving the vehicle routing problem., arXiv preprint arXiv:1802.04240, 2018.

[45] H. M. a. W. X. Yuxiang Sheng, A pointer neural network for the vehicle routing problem with task priority and limited resources., Information Technology and Control, 49(2):237–248, 2020.

[46] U. P. a. O. W. Rasmus Berg Palm, Recurrent relational networks. arXiv preprint arXiv:1711.08028, 2017..

[47] T. L. a. X. B. Chaitanya K Joshi, An efficient graph convolutional network technique for the traveling salesman problem. arXiv preprint arXiv:1906.01227, 2019.