Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Federated Discrete Reinforcement Learning for Automatic Guided Vehicle Control

J. Enrique Sierra-Garcia [a,1,*], Matilde Santos [b,1]

[a] Electromechanical Engineering Department, University of Burgos, 09006 - Burgos, Spain
[b] Institute of Knowledge Technology, Complutense University of Madrid, 28040 - Madrid, Spain

ARTICLE INFO

*Article history:*
Received 11 May 2023
Received in revised form 25 July 2023
Accepted 20 August 2023
Available online 21 August 2023

*Keywords:*
Automated guided vehicle (AGV)
Federated learning
Industry 4.0
Intelligent control
Path following
Reinforcement learning

ABSTRACT

Under the federated learning paradigm, the agents learn in parallel and combine their knowledge to build a global knowledge model. This new machine learning strategy increases privacy and reduces communication costs, some benefits that can be very useful for industry applications deployed in the edge. Automatic Guided Vehicles (AGVs) can take advantage of this approach since they can be considered intelligent agents, operate in fleets, and are normally managed by a central system that can run in the edge and handles the knowledge of each of them to obtain a global emerging behavioral model. Furthermore, this idea can be combined with the concept of reinforcement learning (RL). This way, the AGVs can interact with the system to learn according to the policy implemented by the RL algorithm in order to follow specified routes, and send their findings to the main system. The centralized system collects this information in a group policy to turn it over to the AGVs. In this work, a novel Federated Discrete Reinforcement Learning (FDRL) approach is implemented to control the trajectories of a fleet of AGVs. Each industrial AGV runs the modules that correspond to an RL system: a state estimator, a rewards calculator, an action selector, and a policy update algorithm. AGVs share their policy variation with the federated server, which combines them into a group policy with a learning aggregation function. To validate the proposal, simulation results of the FDRL control for five hybrid tricycle-differential AGVs and four different trajectories (ellipse, lemniscate, octagon, and a closed 16-polyline) have been obtained and compared with a Proportional Integral Derivative (PID) controller optimized with genetic algorithms. The intelligent control approach shows an average improvement of 78% in mean absolute error, 75% in root mean square error, and 73% in terms of standard deviation. It has been shown that this approach also accelerates the learning up to a 50 % depending on the trajectory, with an average of 36% speed up while allowing precise tracking. The suggested federated-learning based technique outperforms an optimized fuzzy logic controller (FLC) for all of the measured trajectories as well. In addition, different learning aggregation functions have been proposed and evaluated. The influence of the number of vehicles (from 2 to 10) on the path following performance and on network transmission has been analyzed too.

## 1. Introduction

As an alternative to driver vehicles and conveyors in the industrial sector, automated guided vehicles (AGVs) facilitate automatic intralogistics solutions. They provide high flexibility in logistic processes, increase productivity, and reduce risks, quality errors and work accidents [1]. Recent advances in 5G networks have pushed AGVs fleet management to the edge.

From the point of view of control engineering, these industrial vehicles present two coupled control problems: speed control and trajectory control. Speed control is necessary to ensure compliance with the time requirements of the logistics process, such as throughput, cadence, tag time, etc. On the other hand, the control of the trajectory followed by the vehicle is used to ensure that the AGV travels along the desired routes without entering prohibited zones and avoids collisions with obstacles or human operators. These vehicles are equipped with a set of sensors, such as safety lidars, to detect obstacles and stop before the collision [2]. But unexpected stoppages can change the rate of production or even shut down the production line. In addition, the precision in following the route is especially important when the AGV travels through narrow aisles or very close to the infrastructure, or when it must make precise stops at stations to recharge the battery, pick up pallets, containers, carts, etc.

[*] Corresponding author.
*E-mail address:* jesierra@ubu.es (J.E. Sierra-Garcia).
[1] All authors have equally contributed to the manuscript.

https://doi.org/10.1016/j.future.2023.08.021

Therefore, the control strategy is crucial for the proper functioning of the AGV and the safety in the workspace where it moves. Trajectory control is a complex task due to the nonlinearities of the AGV dynamics, its kinematic constraints and the abrupt changes in the curvature of the trajectories. For these reasons different techniques, conventional and taken from the artificial intelligence field, have been applied. Fuzzy logic [3], neural networks [4], and reinforcement learning [5] have proved successful for the AGV tracking control.

Particularly, reinforcement learning (RL) allows the AGV autonomously learn the suitable control law to follow complex trajectories. This approach saves hours spent configuring and tuning the control system, compared to classic controllers like Proportional Integral Derivative (PID) controllers. With the RL the agent learns the relationship between the state of the system and the action to be executed through the interaction with the [6] environment. Learning agents perform actions and receive rewards. In this way they tend to repeat actions with great and positive rewards. This relationship between states and actions is usually called a policy.

Recent studies have proposed the federated learning (FL) paradigm as an advanced machine learning technique for different fields applications [7]–[8]. With this approach, multiple learning agents learn in parallel and share their knowledge with each other to create a global knowledge model. Different deployments, centralized or decentralized, have been proposed, although the main classification is Horizontal FL (HFL) and Vertical FL (VFL). In horizontal FL the implementation is centralized since the agents send the information to a federated server. In the vertical FL scheme, agents locally train the model and then share the model or error gradients with each other, i.e., with the other [9] agents. In both cases, VFL and HFL, training data are not exposed and data privacy is guaranteed.

The work presented here follows the horizontal federated learning paradigm. In this case, the agents send their knowledge to a federated server that receives the information and combines it to create the global learning model. After knowledge fusion, agents download the global model, or the federated server returns it, and agents continue learning locally. This paradigm should not be confused with centralized training, where the agents send untrained information, the central system receives all this information from the agents and uses it to train the global model; then the model is downloaded to the agents. That is, in centralized training the agents are only information gateways and do not perform automatic learning. In contrast, in federated learning the agents perform machine learning calculations, learn by themselves and share this knowledge with the federated server to improve the overall model. In other words, in federated learning machine learning runs on all agents, while in a centralized approach machine learning algorithms only run on the centralized system.

AGVs can use this new federated learning approach since they usually work in fleets and there is a central system that manages all the vehicles [1] and can act as a federated server. The application of the FL in this framework consists, firstly, that the AGVs learn an RL policy by interacting with the system and send what they have learned back to the federated server. The federated server aggregates all this knowledge to create a group policy, which will speed up learning the best policy. For its implementation, this work proposes a control architecture based on federated reinforcement learning to address the trajectory control of these industrial vehicles.

Privacy is one of the main benefits of federated learning. Instead of sending the training data to the federated server, agents learn the model locally and send the trained model or weights to the federated server. This increases the privacy of the training data (which may contain private information) as this information does not travel through different machines. With industrial vehicles, privacy is important since the training data can reveal relevant information about the internal behavior of the AGV, its aging, malfunction, etc. This is especially important when there are AGVs of different brands in a facility. In addition, in this application the training data can reveal confidential information about the production and logistics processes; this fact also emphasizes the argument for using FRL. Furthermore, to provide a higher level of security, blockchain can be included in [10] communications. For instance, in this specific case the transfer of the incremental policy could be certified by blockchain. This would further improve the security and traceability of the entire process.

In addition, the federated approach helps speed up the RL learning process. In this way a better control is obtained more quickly. A possible explanation for this acceleration of learning is that with more agents more different actions can be explored in less time. This exploration leads to finding actions that grant higher rewards and improve controller performance.

In this work an RL-based control architecture is defined for each AGV. It consists of a state estimator, a rewards calculator, an action selector, a policy stored in a table, and a policy update algorithm. The policy assigns an angular velocity reference for each state of the system, based on the guidance error and its derivative. The reward calculation takes into account whether the AGV is moving towards or away from the desired trajectory. In this way the AGV learns to follow the path. Periodically each AGV sends the policy change to the federated server. Different learning aggregation functions have been proposed, tested and compared. This group policy supersedes the individual policy and AGVs continue to learn from this point.

The proposal has been evaluated in simulation with five hybrid AGVs that combine the kinematics of a tricycle and a differential robot [11], a very common vehicle in the automobile industry, with challenging trajectories: a lemniscate, an ellipse, a polyline with abrupt changes of direction and an octagon. The operation of the FL-based system has been evaluated against a PID whose tuning parameters have been optimized with genetic algorithms. In addition, during the simulations the transmission has been analyzed to determine which learning aggregation function is the most efficient from the perspective of the network, considering from 2 to 10 vehicles. The Federated Discrete Reinforcement Learning (FDRL) control provides better performance compared to PID for all tested trajectories in terms of lower guidance error and therefore follows the trajectory more accurately. Besides, results have been also compared with a fuzzy logic controller (FLC) whose membership functions have been optimized with genetic algorithms to minimize the trajectory error.

The main contributions of this work can be summarized as follows.

- A federated discrete reinforcement learning scheme is proposed. This scheme is application-independent and can be used to accelerate general discrete reinforcement learning systems.
- A control scheme for industrial AGVs which exploits the proposed FDRL is designed. To our best knowledge, this is the first work where FDRL is used to regulate the angular speed of AGVs for trajectory tracking.
- Different learning aggregation functions have been defined that combine the incremental discrete RL policies learnt by the agents. For all we know, the procedure to combine these learnings is completely original.
- It has been shown that the transfer of incremental policies can save bandwidth in the communication process.

The rest of the paper is organized as follows. In Section 2, related works are presented. The application of the federated with reinforcement learning control approach to the autonomous guided vehicle is explained in Section 3. Simulations results are discussed in Section 4 for different scenarios. Analysis of the sensibility to some configuration parameters is also presented. Conclusions and future works end the paper.

## 2. Related works

Federated learning has the intrinsic properties of protecting data privacy and decreasing the amount of data transmitted, so allowing companies to train machine learning models without transferring data from the devices where that data are created. For industrial Internet of Things (IoT) applications, which demand real-time data processing solutions, these functions are necessary. However, FL has hardly been applied to data from the real environment of AGVs, despite its evident potential in the smart industry.

In a very recent paper by [12], the detection of anomalous energy consumption that can point at an AGV component continuing deterioration or incorrect AGV use is done via federated learning. A global prediction model is iteratively created based on several local prediction models of specific AGVs, and the local models are then re-trained using fresh data to estimate the energy consumption of the AGV using IoT devices. They carry out experiments with Formica-1, an AGV developed by AIUT Ltd. The same authors had previously suggested using three virtual devices in production processes in conjunction with certain AGVs to improve forecast accuracy using FL [13]. The key contribution of that paper is a new method for data interchange between devices that enables AGVs or intelligent devices to learn from one another and improve the precision of recurrent neural networks. Unlike these papers, we use FL to improve the trajectory control of the AGVs rather than for forecasting.

With a control-oriented approach, Wang et al. 2022, present a hierarchical architecture with federated control, which includes a federated control center, a network layer, and a control node [13]. A reliable localization model is developed using this foundation. The correctness of the localization has been tested by simulation experiments. In their article from 2022, Liu et al. address the issue of several AGVs reaching edge servers (ES) and causing offloading and processing delays as a result of competition for resources [14]. The Multi-AGV Cyclical Offloading Optimization (MCOO) technique is used by the authors to reduce scheduling conflicts for cyclical workloads. For the purpose of optimizing the offloading strategy, the reinforcement learning-based A3C algorithm is used.

Without using a federated approach, RL-based path following has been recently applied to autonomous vehicles, even if mainly to marine or aerial ones. In [15] the authors propose a deep interactive reinforcement learning strategy for trajectory tracking of autonomous underwater vehicles by combining deep reinforcement learning and interactive RL. Both human and environmental rewards feed the learning system. In [16], using deep reinforcement learning with interactive RL, the authors develop a deep interactive reinforcement learning system for route following of autonomous underwater vehicles. The learning strategy draws its knowledge from both environmental and human incentives simultaneously. In [17], based on deep reinforcement learning theory, the authors solve the quadrotor vehicle path following problem. They propose three different ways to build the Deep Deterministic Policy Gradient algorithm. In [18], three kernel strategies are applied for a path-integral-based RL algorithm that is developed for a mobile robot path-following strategy. Unlike these works, we have enhanced the learning including the federated paradigm in the RL scheme.

Fleet management of AGVs has been also benefited from RL techniques. Among other researches, [19] proposes a self-adaptive traffic model that combines behavior trees with RL. The RL model is further improved based on these behavior trees that are used to enumerate all potential states in AGV traffic control. The architecture of the proposed cyber–physical system employs multiagent system technology, and some parts like AGVs and traffic controllers are characterized as agents that interact with one another autonomously. In [20], it is suggested to use an adaptive deep reinforcement learning based approach for real-time AGV scheduling with mixed rule. The makespan and delay ratio must be kept to a minimum. In [21] the multi-AGV flow-shop scheduling issue is dealt with a reinforcement learning approach. The goal of this work is to create an AGV timetable that reduces the typical task delay and the overall makespan. The issue of work scheduling in automated warehouses using heterogeneous autonomous robotic systems is addressed in the study by Ho et al. 2022 [22]. They present a deep reinforcement learning (DRL) based task scheduling algorithm that uses the proximal policy optimization approach to identify the best task scheduling policy, in order to handle the stochastic nature of the goods/tasks flow and a high number of robots in the system. They use a proximal weighted federated learning-based method to create the decentralized optimal proximal policy.

As shown, there have been several recent studies on federated learning for a number of AGV-related subjects, including RL-based path planning and RL-based fleet management. However, as far as we know, there have not been any prior efforts on AGV path-following control based on federated RL.

## 3. Intelligent federated control architecture

In federated learning, agents individually learn and regularly send their learnings to a federated server. The federated server aggregates these findings, builds a common knowledge database, and then this global knowledge is downloaded to the agents. This process is repeated periodically. To have a global picture of the system we will describe first the learning modules embedded in the AGVs, and then the federated server.

### 3.1. Individual RL control architecture of each AGV

Fig. 1 shows the control architecture embedded in each AGV. The state estimator based on the guide error, $err_{gui}$, and its derivative, $\dot{err}_{gui}$, calculates the current state $s_t \in S$, where $S$ is the set of discrete states and subindex $t$ means current time. The guide error is the deviation (in this case, in cm) of the AGV from the desired path, measured by the AGV position sensor. The action selector selects the action $a_t \in A$ that maximizes the expected reward, where $A$ is the set of discrete actions. To do it, this module considers the previous rewards accumulated in the table of expected rewards $T : S \times A \rightarrow R$. Then, the discrete action is translated into a reference angular speed $w_{ref}$ that feeds the input of the AGV. It also receives a reference for the longitudinal speed, $v_{ref}$. Both references are combined to calculate the desired speed for each wheel [5].

In reinforcement learning with continuous policies it is quite common to use neural networks to store the policy. In these cases, where different neural network architectures can be used, it is a good option to exploit deep learning techniques to handle large data [23] models. However, in our case we are using discrete policies. The policy is stored in a table, with as many rows as states and as many columns as actions, and the use of neural networks is usually not necessary. The size of the table only depends on the number of states and the number of actions, where both the states and the action sets have a constant size,

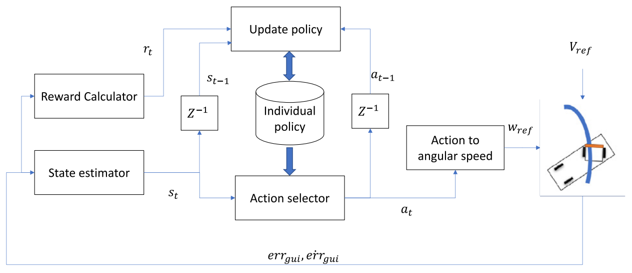

**Fig. 1.** Individual RL control architecture of an AGV.

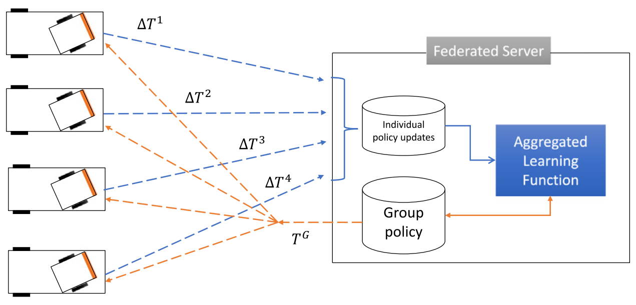

**Fig. 2.** Learning aggregation in the federated server.

as well as the total number of cells. Furthermore, the size of the table does not grow with the number of agents, so the proposal is easily scalable and large fleets of AGVs can be handled.

The action $a_t$ produces a change in the system and the AGV moves away or closer to the desired trajectory. Thus, the guide error and its derivative changes every time $t_{(i+1)}$. These values are used by the reward calculator to obtain the recompense $r_{(t+1)}$. Finally, this reward is used to update the cell associated to the (row, column)= $(s_t, a_t)$ of table $T$. This way, the actions which receive higher rewards will be selected with a larger probability.

The state estimator calculates the state $s_t$ by (1)–(3).

$$err_D(t_i) = DIV\left(MIN\left(e_{max}, MAX\left(err_{gui}(t_i), e_{min}\right)\right) - e_{min}, n_e\right) \tag{1}$$

$$derr_D(t_i) = DIV\left(MIN\left(de_{max}, MAX\left(\dot{err}_{gui}(t_i), de_{min}\right)\right) - de_{min}, n_{de}\right) \tag{2}$$

$$s_t = err_D(t_i) \cdot n_{de} + derr_D(t_i) \tag{3}$$

where $DIV$ is the integer division operator, $MIN$ and $MAX$ are the minimum and maximum operators, and $[e_{min}, e_{max}, de_{min}, de_{max}, n_e, n_{de}]$ can be manually adjusted to set the number of states and their size. The size of the set $S$ is $|S| = n_e \cdot n_{de}$.

The action selector implements an $\epsilon$-greedy exploration strategy. This means that an action is randomly selected with probability $\epsilon$, and the actions that maximize the expected rewards are selected with probability $(1-\epsilon)$. This strategy searches a balance between the exploitation of previous learnings and the exploration of new alternatives. When $\epsilon = 0$ the system only relies on previous experiences, and actions with large rewards may be not used due to lack of exploration. On the other hand, when $\epsilon = 1$ the actions are purely random, and the systems does not learn anything. Therefore, it is important to find a correct trade-off between exploration and exploitation of previous knowledge. The execution of the action selector and the corresponding angular speed can be formalized by (4)–(5).

$$a_t = \begin{cases} argMAX_a\left[Q^{\pi}_{(s_t,a_t)}\right] & rand() \geqslant \epsilon \\ n_{act} \cdot rand() & rand() < \epsilon \end{cases} \tag{4}$$

$$w_{REF}(t_i) = w_{r_{min}} + a_t(w_{r_{max}} - w_{r_{min}})/n_{act} \tag{5}$$

where $\epsilon$ is the chance of choosing a random action (value between 0 and 1), $n_{act}$ denotes the number of actions, and $\left[w_{r_{min}}, w_{r_{max}}\right]$ are used to adjust the range of the possible discrete actions.

The reward calculator checks if the AGV is moving closer to or moving away from the trajectory, in order to reward or punish the previous action. Rewards and punishments $r_t$ are positive and negative values, respectively. To do so, the guide error and its derivative are evaluated. When the $err_{gui}$ is zero, the AGV is in the center of the trajectory. In this case, if $\dot{err}_{gui}$ is also zero the action is rewarded as the AGV will stay within the trajectory; otherwise, the action is punished as the AGV is moving away from it (6).

$$r_t = \begin{cases} -\dot{err}_{gui}(t_i) & err_{gui}(t_i) > 0 \\ -\left|\dot{err}_{gui}(t_i)\right| & err_{gui}(t_i) = 0 \wedge \dot{err}_{gui}(t_i) \neq 0 \\ de_{max} & err_{gui}(t_i) = 0 \wedge \dot{err}_{gui}(t_i) = 0 \\ \dot{err}_{gui}(t_i) & err_{gui}(t_i) < 0 \end{cases} \tag{6}$$

This reward $r_t$ is used to update the cell associated to the previous state and action of table T. That is, when the system is at state $s_{t-1}$ and action $a_{t-1}$ is carried out, it receives the reward $r_t$. There are different alternatives to update table $T$ [24]–[25]. In this work we have used the accumulation and the mean of the previous rewards (7)–(8). These techniques are also called SAR (Summatory of All Rewards) and MAR (Mean of All Rewards). They are only recommended if the rewards take positive and negative values, as in other case they may not converge.

$$SAR: T_{(s_{t-1},s_{t-1})}(t_i) = T_{(s_{t-1},s_{t-1})}(t_{i-1}) + r_t \tag{7}$$

In case of MAR updating strategy, the calculation is based on two tables: table R that accumulates the rewards and table $N$ that counts the times a cell has been selected. This way, $T$ is calculated as the accumulated reward divided by the number of times each option has been selected (8).

$$MAR: \begin{cases} R_{(s_{t-1},a_{t-1})}(t_i) = R_{(s_{t-1},a_{t-1})}(t_{i-1}) + r_t \\ N_{(s_{t-1},a_{t-1})}(t_i) = N_{(s_{t-1},a_{t-1})}(t_{i-1}) + 1 \\ T_{(s_{t-1},a_{t-1})}(t_i) = R_{(s_{t-1},a_{t-1})}(t_i)/N_{(s_{t-1},a_{t-1})}(t_i) \end{cases} \tag{8}$$

### 3.2. Aggregation of learning

Periodically, each AGV $i$ sends its table $T^i$ (or $N^i$ and $R^i$ in case of MAR strategy) to the federated server. The server fuses these individual tables with a learning aggregation function, $f_{la}$ : $\mathbb{R}^{|S||A|n_{AGV}} \rightarrow \mathbb{R}^{|S||A|}$, to obtain a table which represents the global policy $T^G$ (or $N^G$ and $R^G$ in case of MAR policy). Once the fusion is completed, the AGVs download the global table and substitute its own individual table by it. This way the AGVs can exploit the knowledge learned by the others.

Instead of sending the whole table $T^i$, it is possible to send the variations of the table during the last learning period, $\triangle T^i$. This allows to save bandwidth of the channel. Fig. 2 illustrates the global architecture when incremental tables are used and the policy update strategy is SAR. When MAR strategy is used, instead of sending $\triangle T^i$ the tables $\triangle R^i$ and $\triangle N^i$ are sent. The incremental tables are stored in the federated server. Once all incremental tables have been received, the table with the group policy is generated by the learning aggregation function.

Fig. 3 shows the communication process for two AGVs. Periodically, each $t_f$ period, the AGVs send the incremental table to the federated server. When all incremental tables have been received, the group policy $T^G$ is generated and then it is passed

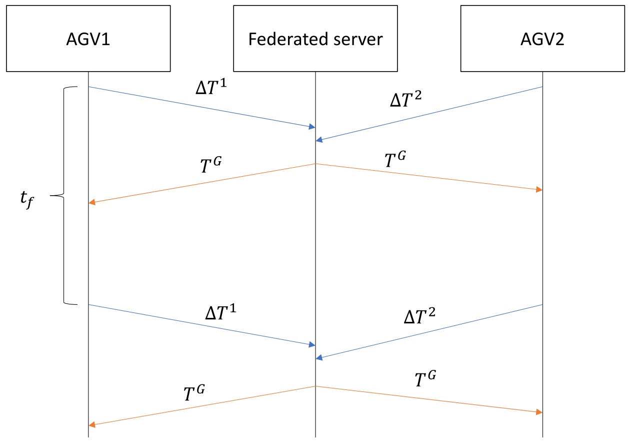

**Fig. 3.** Communication process between two AGVs and the federated server.

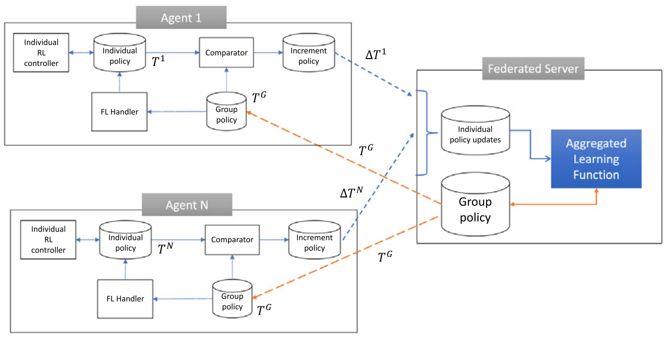

**Fig. 4.** Update of the individual policy in the learning agents.

simultaneously to every AGVs. It is worthy to remark that the incremental table $\triangle T^i$ takes up less space than the standard table $T^i$, thus it is possible to save bandwidth on the uplink channel.

An example of how this bandwidth can be reduced is as follows. Let us assume than each cell of the table is stored with 4 bytes, thus the size of table $T^i$ will be $4|S||A|$. However, for incremental tables only cells with changes must be passed on, that is, cells with non-zero values. Since not all cells are transmitted, it is necessary to send the address of each cell to avoid cluttering the table. Assuming the address of a cell requires 2 bytes, and that the number of cells with non-zero value is $n_z$, the number of bytes to be transmitted is $6n_z$. Therefore, whenever $n_z < (2/3)|S||A|$ it is possible to save bandwidth in the uplink channel. This condition can be handled at the sender, within the AGV, and decide when it is convenient to send the complete incremental table or only the incremental table with the address information.

The federated discrete reinforcement learning scheme proposed in this work is divided into two main blocks. The first, called the individual RL control architecture, runs inside each agent. The second block contains the learning aggregation function that combines the individual incremental policies to obtain the global one. Fig. 1 shows the individual RL control architecture running inside each agent (AGV), without considering any federated mechanism. Fig. 4 shows how the learning aggregation function calculates the group policy from the individual policies.

On the left side of Fig. 4, it is possible to see how the federation mechanism integrates with the individual RL. The Individual RL block represents the scheme shown in Fig. 2. The federated learning handler monitors the reception of the group policy and replaces the individual policy with the group one. The individual policy is compared with the group one to obtain the incremental policy; this operation is mathematically formalized in (9). This scheme is repeated for each agent.

The right side of Fig. 4 shows the aggregation of learning as in Fig. 2. The dotted arrows connect the agents with the federated server. Agents send the incremental policy $\triangle T^i$ and receive the group policy $T^G$.

The process to obtain the incremental table for $n$-AGV can be expressed by (9), where $n_{AGV}$ denotes the number of AGVs involved.

$$\triangle T^n_{(j,k)}(t_i) = T^n_{(j,k)}(t_i) - T^n_{(j,k)}(t_i - t_f) \quad \forall (j,k,n) \in S \times A \times [\mathbb{N} \le n_{AGV}] \tag{9}$$

Different strategies can be used to aggregate the learning. We have proposed and tested one function that uses the individual tables $T^i$ and different functions that combine the incremental tables $\triangle T^i$. As it will be shown in the results section, the aggregation function that uses the incremental tables requires less network requirements. The first proposal is to add up all individual tables, a strategy that we have called tSAR (10).

$$tSAR : T^G_{(j,k)}(t_i) = \sum_n^{n_{AGV}} T^n_{(j,k)}(t_i) \quad \forall (j,k) \in S \times A. \tag{10}$$

The sum of the individual tables is the iSAR strategy. In this case, the previous value of the group table must be added so as not to lose the previously learned knowledge. It is also possible to calculate the average of the rewards received by all the AGVs with the imSAR strategy. Furthermore, instead of averaging the rewards of the AGVs, it is possible to select the maximum expected reward among all the AGVs using the iMAX strategy. These strategies are formalized as follows (11)–(13).

$$iSAR : T^G_{(j,k)}(t_i) = T^G_{(j,k)}(t_{i-1}) + \sum_n^{n_{AGV}} \Delta T^n_{(j,k)}(t_i) \quad \forall (j,k) \in S \times A \tag{11}$$

$$imSAR : T^G_{(j,k)}(t_i) = T^G_{(j,k)}(t_{i-1}) + \frac{1}{n_{AGV}} \sum_n^{n_{AGV}} \Delta T^n_{(j,k)}(t_i) \quad \forall (j,k) \in S \times A \tag{12}$$

$$iMAX : T^G_{(j,k)}(t_i) = T^G_{(j,k)}(t_{i-1}) + \max_{n \le n_{AGV}} \Delta T^n_{(j,k)}(t_i) \quad \forall (j,k) \in S \times A \tag{13}$$

All these aggregation functions are related to the policy update algorithm SAR, as they can be seen as an extension of SAR for multiple learning agents. On the other hand, the natural extension of MAR with multiple learning agents is iMAR (14). In this case, the federated server receives the incremental tables, $\triangle N^G$ and $\triangle R^G$, and combines them to obtain the group tables $N^G$ and $R^G$. Once these tables are generated, the policy table $T^G$ is obtained as the division of $R^G$ by $N^G$. This way, the expected reward is the average reward considering the times that each cell is selected.

$$iMAR : \begin{cases} N^G_{(j,k)}(t_i) = N^G_{(j,k)}(t_{i-1}) + \sum_n^{n_{AGV}} \Delta N^G_{(j,k)}(t_i) & \forall (j,k) \in S \times A \\ R^G_{(j,k)}(t_i) = R^G_{(j,k)}(t_{i-1}) + \sum_n^{n_{AGV}} \Delta R^G_{(j,k)}(t_i) & \forall (j,k) \in S \times A \\ T^G_{(j,k)}(t_i) = R^G_{(j,k)}(t_i) / N^G_{(j,k)}(t_i) & \forall (j,k) \in S \times A \end{cases} \tag{14}$$

## 4. Simulation results and discussion

### 4.1. Simulation environment and performance metrics

In this section, the simulation results obtained with the federated proposal for different roads are presented and analyzed.

For the experimental configuration described in [26], the Matlab/Simulink software has been used to simulate the intelligent controller and the AGVs. Each simulation lasts for 100 s; however, for safety reasons, if the AGV deviates from the defined path before the simulation ends, the experiment stops. To reduce discretization errors and speed up simulation execution, the sample time, $T_s$, is not constant but varies during the simulation. A control period of 25 ms has been specified.

The most common trajectory controller used for these industrial AGVs is the Proportional–Integral–Derivative (PID) regulator, widely applied in many industrial applications. Thus, we have compared the here proposed intelligent control strategy with a PID that is described by the following equation based on the guide error [5].

$$w_{REF}(t_i) = K_p \cdot err_{gui}(t_i) + K_d \cdot \dot{err}_{gui}(t_i) + K_i \int_0^{t_i} err_{gui}(t)dt \tag{15}$$

where $[K_p, K_d, K_i]$ are the PID tuning gains. For each trajectory the PID parameters have been optimized using genetic algorithms to minimize the error.

The AGV has been simulated with Eqs. (16)–(27). These equations, the specific configuration parameters and the values used in the simulations are explained in [26]–[27] with more detail.

$$M_{er} = M_r - F_{SWR} \cdot sign(\dot{\theta}_R),\ M_{el} = M_l - F_{SWL} \cdot sign(\dot{\theta}_L), \tag{16}$$

$$\begin{bmatrix} m_T \cdot R_h/2 & m_T \cdot R_h/2 \\ \frac{I_h \cdot R_h}{L_h} & -\frac{I_h \cdot R_h}{L_h} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_R \\ \ddot{\theta}_L \end{bmatrix} = \begin{bmatrix} \frac{M_{er}+M_{el}}{2R_h} - f_{rT} \\ \frac{(M_{er}+M_{el})L_h}{2R_h} - f_{rR} \end{bmatrix}, \tag{17}$$

$$f_{rT} = 0.5 \cdot \delta_{air} \cdot S_{AGV} \cdot C_{aero} \cdot v_h^2 sign(v_h) + 9.8 \cdot m_T \cdot C_{roll} \cdot sign(v_h), \tag{18}$$

$$f_{rR} = F_{vh} \cdot \dot{\Phi}_h + F_{sh} \cdot sign(\dot{\Phi}_h), \tag{19}$$

$$V_L = R_h \cdot \dot{\theta}_L,\ V_R = R_h \cdot \dot{\theta}_R, \tag{20}$$

$$\begin{bmatrix} \dot{x}_h \\ \dot{y}_h \\ \dot{\Phi}_h \end{bmatrix} = \begin{bmatrix} \frac{V_L+V_R}{2} cos(\Phi_h) \\ \frac{V_L+V_R}{2} sin(\Phi_h) \\ \frac{V_R-V_L}{L_h} \end{bmatrix}, \tag{21}$$

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{\Phi}_b \end{bmatrix} = \begin{bmatrix} v_h cos(\gamma) cos(\Phi_b) \\ v_h cos(\gamma) sin(\Phi_b) \\ \frac{v_h}{l_b} sin(\gamma) \end{bmatrix}, \tag{22}$$

$$v_h = \sqrt{\dot{x}_h^2 + \dot{y}_h^2} = \frac{v_L + v_R}{2}, \tag{23}$$

$$\gamma = \Phi_h - \Phi_b,\ \gamma_{min} \leqslant \gamma \leqslant \gamma_{max}, \tag{24}$$

$$\dot{x}_b sin(\Phi_b) - \dot{x}_b cos(\Phi_b) = 0, \tag{25}$$

$$\dot{x}_b sin(\Phi_b + \gamma) - \dot{x}_b cos(\Phi_b + \gamma) - \dot{\Phi}_b L_b cos(\Phi - b) = 0, \tag{26}$$

$$err_{gui} = f_{sen}(x_h, y_h, \Phi_h, path) \tag{27}$$

The Mean Absolute Error (MAE), Root Mean Square Error (RMS), and STandard Deviation (STD) calculated by (28)–(30) have been used as Key Performance Indicators (KPIs) [28].

$$MAE = \frac{1}{T_{sim}} \sum_i \left| err_{gui}(t_i) \right| T_s(t_i) \tag{28}$$

$$RMSE = \sqrt{\frac{1}{T_{sim}} \sum_i err_{gui}(t_i)^2 \cdot T_s(t_i)} \tag{29}$$

Table 1
Tuning parameters of the PID optimized by genetic algorithms.

| Trajectory | KP | KD | KI |
|---|---|---|---|
| Ellipse | 12.4902 | 2.2661 | 1.4421 |
| Octagon | 7.0893 | 9.1489 | 0.9107 |
| 16-S Polyline | 8.7382 | 7.0594 | 1.3822 |
| Lemniscate | 25.4700 | 2.2300 | 0.1201 |

$$STD = \sqrt{\frac{1}{T_{sim}} \sum_i \left[ err_{gui}(t_i) - \frac{1}{T_{sim}} \sum_i err_{gui}(t_i) T_s(t_i) \right]^2 \cdot T_s(t_i)} \tag{30}$$

Four different trajectories have been proposed to validate the controller: a small ellipse, a small lemniscate, an octagon, and a closed polyline with 16 segments. Considering that the octagon can be formalized as a polyline, the expressions of these trajectories are the following [5],

Ellipse:

$$x(t) = a_{eli} \cos(t) \tag{31}$$

$$y(t) = b_{eli} \sin(t) \tag{32}$$

Lemniscate:

$$x(t) = \sqrt{2} \cdot a_{lem} \frac{\cos(t)}{1 + \sin(t)^2} \tag{33}$$

$$y(t) = \sqrt{2} \cdot a_{lem} \frac{\cos(t)\sin(t)}{1 + \sin(t)^2} \tag{34}$$

Polyline:

$$x(t) = t \qquad i = 1 \ldots M, P_{x_{i-1}} < t < P_{x_i} \tag{35}$$

$$y(t) = \frac{P_{y_i} - P_{y_{i-1}}}{P_{x_i} - P_{x_{i-1}}}(t - P_{x_{i-1}}) + P_{y_{i-1}} \qquad i = 1 \ldots M, P_{x_{i-1}} < t < P_{x_i} \tag{36}$$

where $[P_x, P_y] \in \mathbb{R}^{2M}$ is the set of points of the polyline, $a_{eli}$ and $b_{eli}$ are the semi-axes of the ellipse, and $a_{lem}$ is the width of the lemniscate. Due to the kinematic limitations of the AGV, the route becomes more difficult when $a_{eli}$, $b_{eli}$, or $a_{lem}$ shrink. Indeed, too low values can cause the AGV to leave the circuit. Finding the parameters that are appropriate to make trajectory tracking challenging but not impossible has required running a series of simulations. Finally, these values have been set to $[a_{eli}, b_{eli}, a_{lem}] = [2, 0.66, 0.75]$. The ellipse and the lemniscate can be seen in Figs. 5 and 8.

The octagon has the following points $\{(Px, Py)\} = \{(0, 0), (0.75,0), (1.5, 0.75), (1.5, 1.5), (0.75, 2.25), (-0.75, 2.25), (-1.5, 1.5), (-1.5, 0.75),(-0.75, 0), (0, 0)\}$, and the polyline with 16 segments is defined by the set of points: $\{(Px, Py)\} = \{(0, 0), (1.25, 0), (2.5, 1.25), (3.75, 1.25), (5, 0), (6.25, 0), (7.5, 1.25), (7.5, 1.875), (6.25, 2.5), (5, 2.5), (3.75, 3.75), (2.5, 3.75), (1.25, 2.5), (0, 2.5), (-1.25, 1.875), (-1.25, 1.25), (0, 0)\}$. The octagon and the polyline are shown in Figs. 6–7.

As said, in order to make a fair comparison, the gains of the PID regulator have been optimized with genetic algorithms. The optimum parameters obtained with this heuristic technique are show in Table 1.

### 4.2. Testing the performance of the federated learning controller

We have also analytically and graphically evaluate the performance of the FDRL controller and compared it with the PID

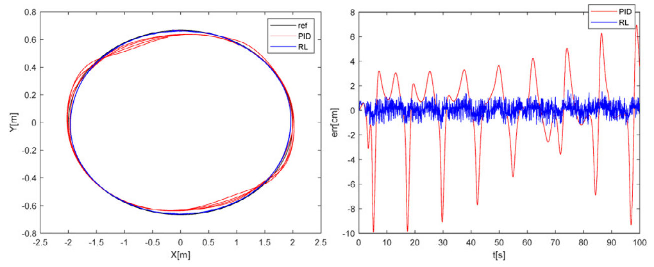

**Fig. 5.** Elliptical trajectory (left) and the guide error (right).

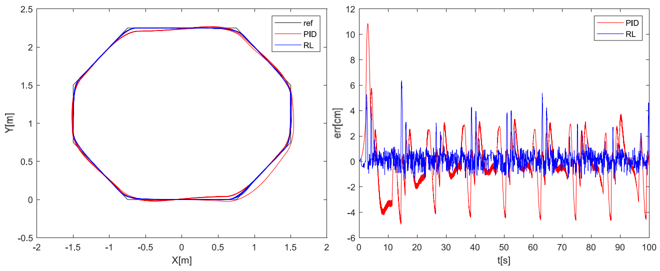

**Fig. 6.** Octagonal trajectory (left) and the guide error (right).

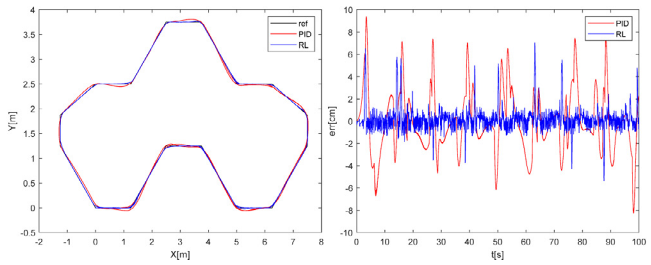

**Fig. 7.** A 16-segment polyline trajectory (left) and the guide error (right).

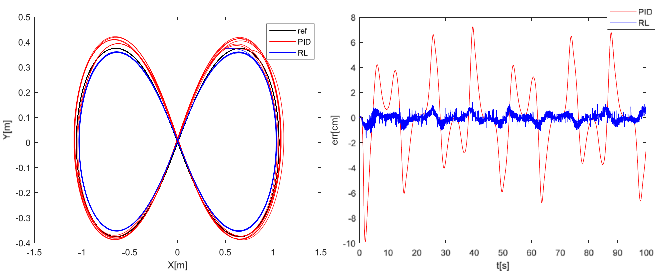

**Fig. 8.** A lemniscate trajectory (left) and corresponding guide error (right).

regulator for the four previously defined trajectories: the ellipse, lemniscate, octagon, and 16-segment polyline. In each experiment, 5 AGVs work in parallel following the same trajectory but on different circuits. They are supposed to be far enough apart that they cannot collide with each other. At the end of each episode (i.e., when the AGV goes out-of-circuit or the 100 s simulation time expires), each AGV sends its policy table variation to the federated server and the global policy table is generated. Then, the global policy table is downloaded to the AGVs, and the next simulation episode starts. The reference for the longitudinal AGV speed is a sinusoidal signal with an average value of 0.35 m/s and an amplitude of 0.35 m/s. Only lengthy straight lines allow these AGVs to travel at their top speed of 1 m/s. As a result, the sinusoidal reference signal requirement for 0.7 m/s corresponds to the recommended speed for these applications.

Twenty-five episodes are carried out for each trajectory. Figs. 5–8 show the episode guidance error (left) and the trajectory followed by one of the initial AGV traction units (right) at the last episode. These graphs compare the PID trajectory (red line) with the FDRL trajectory (blue line) and the reference (black line). The guide error with the FDRL-based control is the blue line, whereas the guide error with the PID controller is shown in red on the right side image.

In particular, Fig. 5 shows the tracking of the elliptical trajectory. It is noticeable how the AGV with FDRL follows much more accurately the ellipse. Indeed, the blue and black lines seem overlapped. The larger errors appear at the top and the bottom of the ellipse, as can be seen in the error graph (Fig. 5, right). In the case of the FDRL, the error is always smaller than 1 cm. However, when the PID is applied, larger error peaks appear that correspond to the edges of the ellipse.

For the octagon, the difference in the tracking with the two controllers is not so large (Fig. 6). Both controllers follow quite well the trajectory, but the errors when the FDRL is applied are still smaller. As expected, the errors tend to be larger at the edges of the polygon, and they become smaller in the straight parts of the trajectory. The largest peak error with the PID appears at the beginning of the trajectory, with a value around 10 cm; after this moment the peaks are smaller and more repetitive.

The 16-segment polyline tracking is similar to the octagon path tracking but with bigger errors as it is a more complex trajectory (Fig. 7). Indeed, the peaks are larger, and the mean error is also bigger. In the case of the FDRL, the error is still smaller than 1 cm in the straight parts of the trajectory, with peaks of few centimeters in the edges of the segments of the polyline. However, with the PID the error values oscillate between values of several cm. Furthermore, when the FDRL is applied there are fewer and smaller peaks than with the PID.

Fig. 8 shows the tracking of the lemniscate. With both controllers the trajectory described by the AGV has that specific shape, but the size is different. In the case of the FDRL control, the lemniscate is slightly smaller than that of the reference, and when the PID is applied it is larger. Observing the error, the performance of the FDRL control is much better: although it is a bit noisy, the amplitude is much smaller, less than 1 cm. In both cases, the largest errors usually appear in the upper and lower part of the lemniscate, but in the case of the PID their values are much higher, up to 7 cm. The biggest error appears at the beginning of the trajectory because the direction of the AGV is aligned with the *x*-axis and it needs a large correction in orientation to align with the trajectory.

Along with the graphical data, quantitative values of both controllers performance have also been obtained. Table 2 shows the RMSE, MAE, and STD of the guidance error for each trajectory and for the two controllers, the PID and the FDRL. The values correspond to the metric average values when 5 AGVs operate concurrently. The average value for each column is listed in the last row. Results highlighted in bold are the best ones.

Table 2 allows us to confirm the graphical results: every KPI is better when the FDRL-based controller is used. The greatest improvement is obtained for the lemniscate trajectory, with 87% in MAE, 85% in RMSE and 88% in STD improvement with the intelligent control. The average improvement is very high, 78% in MAE, 75% in RMSE and 73% in STD. With the PID, the largest errors appear for the lemniscate path and the smallest ones for the octagon trajectory. In the case of the FDRL, the largest errors appear with the 16-segment polyline as this trajectory is more abrupt.

Furthermore, the FDRL has also been compared to a fuzzy logic controller whose membership functions have been optimized with genetic algorithms. In this case the FDRL also gave better

**Table 2**
Comparison of KPIs for different trajectories with both controllers.

| Trajectory | MAE | | RMSE | | STD | |
|---|---|---|---|---|---|---|
| | PID | FDRL | PID | FDRL | PID | FDRL |
| Ellipse | 2.91 | 0.47 | 2.11 | 0.37 | 2.91 | 0.49 |
| Octagon | 2.21 | 0.71 | 1.61 | 0.48 | 2.20 | 0.80 |
| 16-S Polyline | 2.84 | 1.14 | 2.17 | 0.65 | 2.85 | 1.29 |
| Lemniscate | 3.13 | 0.46 | 2.31 | 0.29 | 3.13 | 0.35 |
| Average | 2.77 | 0.70 | 2.05 | 0.45 | 2.77 | 0.73 |

results. As an example, for the lemniscate the RMSE is 1.49 cm, and with the FDRL it is 80% smaller. For the octagon, the reduction is 64%, and the average reduction considering all trajectories is 65%.

These findings allow us to confirm that for these trajectories, the FDRL controller performs significantly better than the PID and the fuzzy logic controller.

### 4.3. Federated learning vs monolithic learning

In order to evaluate the improvement obtained by the federated learning approach, the evolution of the RMSE is analyzed for each trajectory. Like in the previous section, 5 AGVs are working in parallel and these agents send the variation of its individual policy table to the federated server at the end of every iteration. For each trajectory, the results are compared with the ones obtained when RL is used without the federated approach, and with the RMSE given by the PID regulator. The learning aggregation function used is iSAR (11).

Fig. 9 shows this comparison for the ellipse, Fig. 10 for the octagon, Fig. 11 for the 16-segment polyline and finally, Fig. 12 for the lemniscate trajectories. Blue lines represent the evolution of the RMSE when the federated approach is not applied. The dashed black lines indicate the trajectories obtained when the PID is used. The rest of the lines represent values related to the federated learning strategy. As the federated approach is applied with 5 AGVs, for each iteration there are 5 different values of the RMSE, one per AGV. Instead of representing the evolution of these 5 values, we show the average value (red line), the minimum (yellow line), and the maximum (purple line).

For all tested trajectories, federated and monolithic RL give a better performance than the PID regulator once the intelligent controller has learned, around episode 6. During the first episodes, the RMSE for the RL approaches is larger than for the PID, but this value decreases with the training as the controller learns the control law. Another interesting result is that with the federated approach, the RMSE decreases faster as the AGVs are able to exploit the knowledge obtained by the others.

As expected, at the first episode both, federated and monolithic RL give similar results as the learnings have not been combined yet. The effects of the learning aggregation function are only visible from episode 2 onwards. For all the trajectories the largest reduction of RMSE takes place between episodes 1–3; as the system goes on learning, this reduction gets smaller and smaller. Thus, when the number of episodes grows both learning approaches seem to converge to the same value. A possible explanation is that once all possible actions have been explored, all AGVs tend to select the same actions, those that have received greatest rewards. It is noteworthy to remark that the federated approach improves the learning in the sense that it speeds it up markedly, 50% in the best case (ellipse) and an average of 36% for all trajectories.

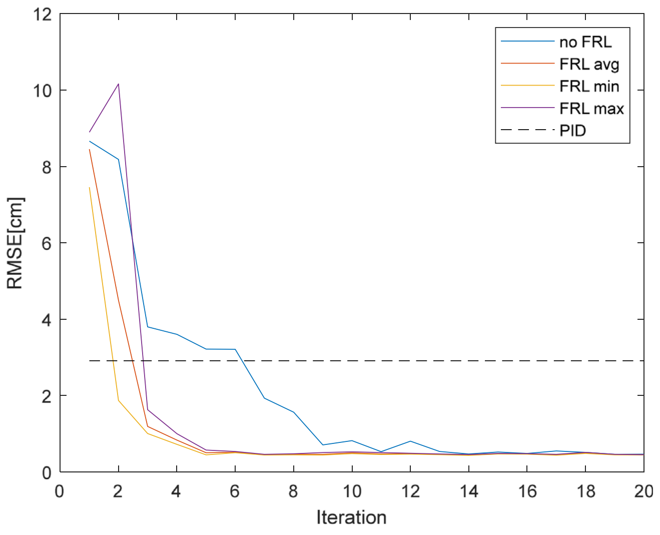

**Fig. 9.** Evolution of the RMSE for elliptical trajectory.

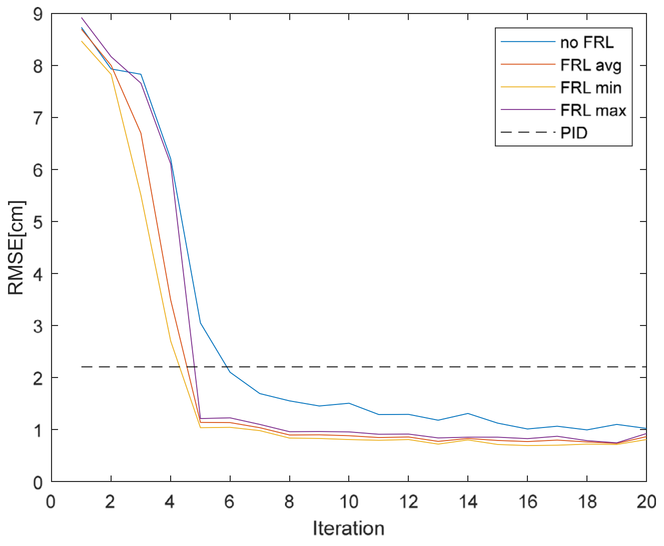

**Fig. 10.** Evolution of the RMSE for octagonal trajectory.

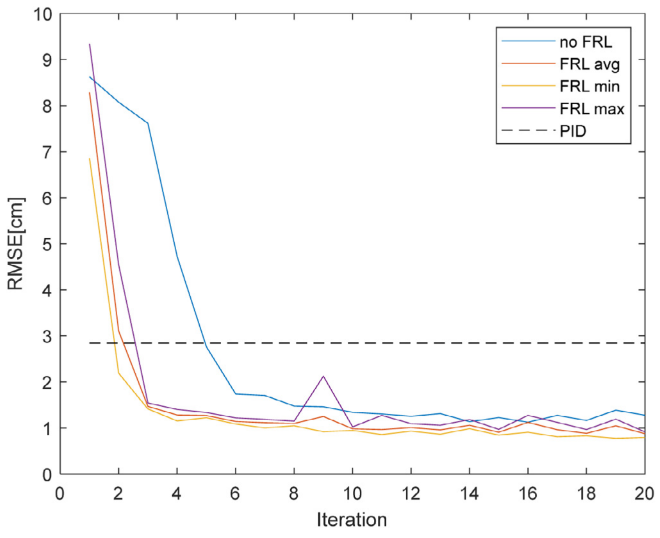

**Fig. 11.** Evolution of the RMSE for 16-segment polyline trajectory.

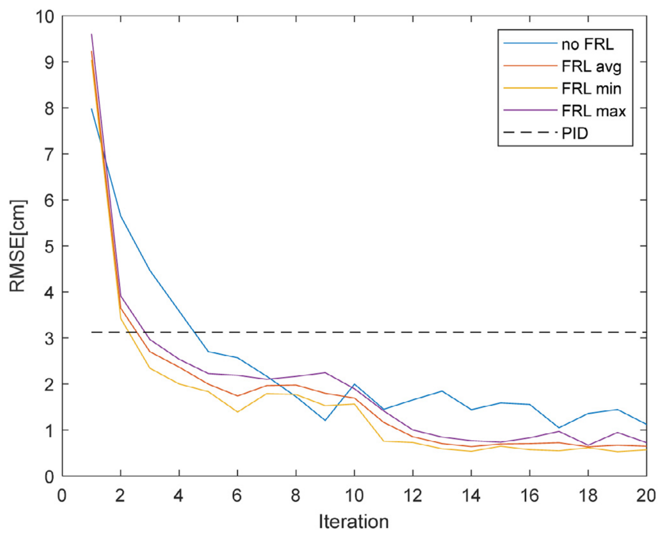

**Fig. 12.** Evolution of the RMSE for lemniscate trajectory.

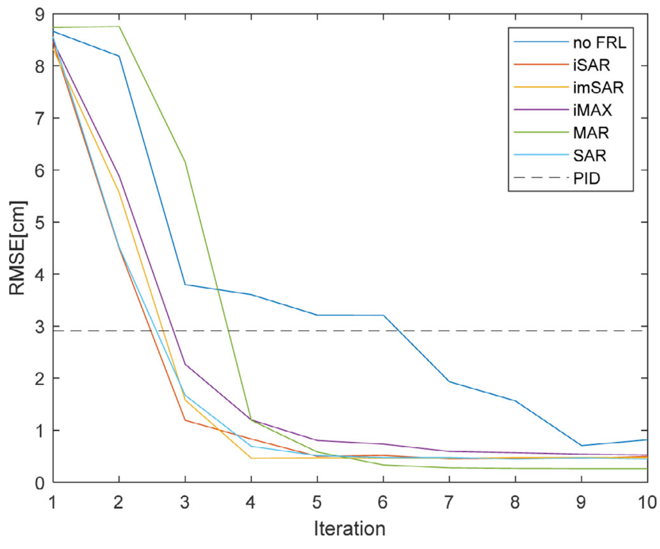

**Fig. 13.** Evolution of RMSE for elliptical trajectory and different learning aggregation functions.

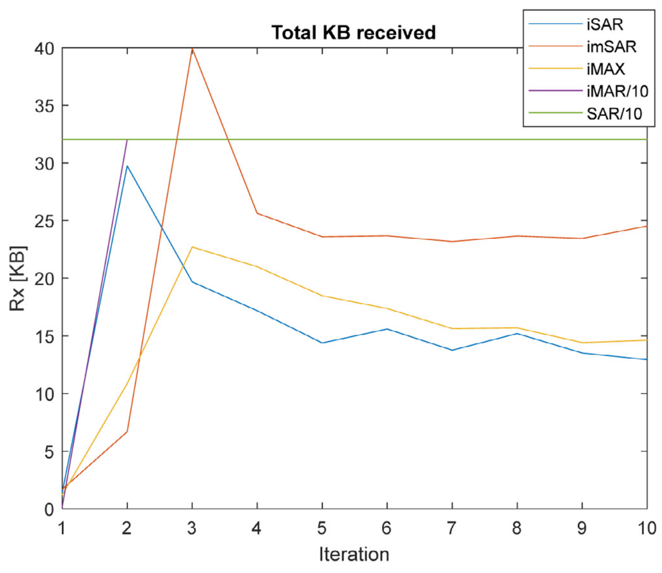

**Fig. 14.** Total KB received by the federated server for the different aggregation functions.

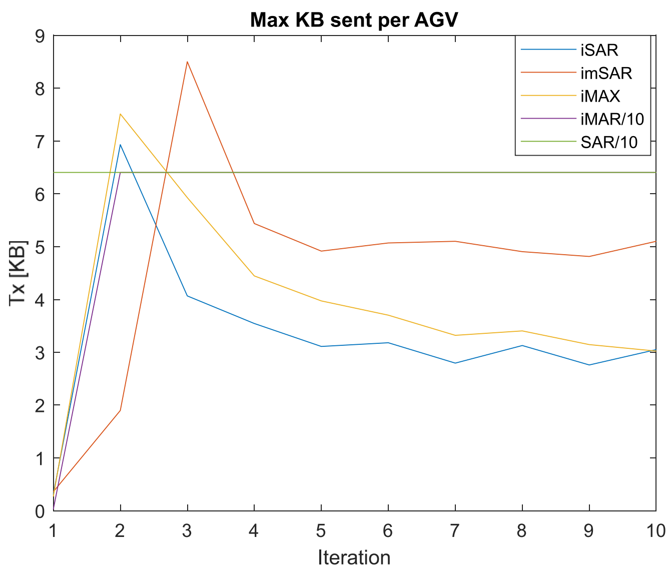

**Fig. 15.** Maximum KB sent by an AGV for the different aggregation functions.

## *4.4. Influence of the learning aggregation function*

The influence of the defined learning aggregation function has been also analyzed. To this end, several experiments with different learning aggregation functions have been carried out. In all cases the trajectory used is the ellipse and the number of AGVs is 5. Fig. 13 shows the results of this analysis, where the legend indicates the aggregation function; "no FRL" means that learning aggregation is not applied. As in the previous section, for each episode we represent the average value of all AGVs.

As expected, the error value for the first episode is the same for all the aggregation functions. Then, with the MAR function the RMSE decreases slower than with the other functions, but it converges to a lowest value, 0.26 cm, when with the others the error is around 0.44–0.52 cm. The other functions decrease at similar speed, although it is possible to see that iMAX is the second slowest. For the MAR function, the largest decrement appears to be between episodes 3 and 4; however, for the other functions the largest decrement is located between 1 and 3.

The transmission is also different depending on the learning aggregation function that is used. That is, the number of bytes sent by the AGVs and received by the federated server varies depending on this function. Fig. 14, compares the total KB received by the federated server at each iteration with the different aggregation functions. Each AGV sends a different number of bytes, so in order to estimate the required capacity of the uplink network channel we need to consider the maximum number of bytes sent by a device. Fig. 15, shows the comparison of the maximum number of KB sent by an AGV, considering all AGVs, depending on the aggregation function. The legend of the figure correspond to the aggregation function. It is noteworthy to remark that the KB of iMAR and SAR have been divided by 10 to be able to graphically compare them with the rest in the same figure.

Some conclusions can be drawn from these results. Using the SAR function, each AGV sends the full policy table, that is, 400 × 41 cells. Each cell is stored with 4 bytes, so each AGV sends 64 KB as shown in Fig. 15, with a green straight line, and the federated server receives 320 KB, Fig. 14. This can be considered as the maximum limit. The iMAR aggregation function tends to explore all possible actions and produces many more policy table changes, so the number of bytes with the iMAR function tends to approximate the number of bytes of SAR. However, it should be noted that the actual iMAR transmission is double to what is shown in Figs. 14–15, since it is necessary to send two tables (N and R) instead of one. Therefore, this aggregation function is the most demanding regarding network requirements. On the contrary, the iSAR function and its variations, imSAR and iMAX, reduce the exploration and therefore, they demand less network capacity. Specifically, iSAR is the function which needs lowest uplink capacity, and iMAX is the function that requires less capacity in the federated server.

The time required to send the information to the federated server depends on the available bandwidth of the uplink channel. For instance, Low Range (LoRa) stablishes a bandwidth of 300 kbps, which means 245 ms to send 9 KB (the maximum in Fig. 15 without considering SAR or iMAR). With Narrow Band IOT (NB-IOT) release 13 this time goes up to 294 ms, as it has a 250 kbps bandwidth, and with LTE-Cat −1 this time decreases to 15 ms [29]. The time needed to download the global policy table depends on the available bandwidth of the downlink channel. As the global policy table size is 64 KB, 1.74 s are needed with LoRa, 2.09 s with NB-IOT release 13, and only 52 ms with LTE-Cat −1 [29]. Depending on the time requirements of the application, we must consider one communication technology or another. In any case, the implementation is feasible.

### *4.5. Influence of the number of agents*

In this section we have studied the influence of the number of learning agents. Several experiments have been carried out changing the number of AGVs, from 2 to 10. The learning aggregation function is set to iSAR and the selected trajectory is the ellipse. Fig. 16 compares the evolution of the RMSE for different number of AGVs. The number of AGVs is represented by a different color, as indicated in the legend, and "no FRL" means that learning aggregation is not considered.

It is possible to observe (Fig. 16) how increasing the number AGVs accelerates the learning as the reduction of the RMSE is faster. A reasonable explanation is that more AGVs explore more actions in the same period of time and thus, the actions with best rewards are found quicker. However, all lines converge at the same value, independently of the number of AGVs. That may be because once the best policy has been found, adding more AGVs does not give additional better policies.

It is also interesting to analyze the influence of the number of AGVs on the network transmission. Figs. 17–19 show the KB sent by the AGVs and received by the federated server as a function of the number of learning agents. Fig. 17 shows the total number of KB received by the federated server with na = 2 to na = 10 AGVs. As expected, this number grows with the number of AGVs, but this growing is not linear. To better understand it, it is convenient to divide this quantity by the number of AGVs (Fig. 18). It is noticeable how the average number of KBs sent by the AGVs decreases with the number of AGVs. There are a couple of reasons that may explain this behavior. On the one hand, the exploration is shared among the AGVs and correspondingly, so does the number of cells with variations in the individual policy table that must be sent to the federated server. On the other hand, increasing the number of AGVs accelerates the learning; when the learning converges less explorations in new cells are carried out, what results in fewer cells with variations in the policy table to be transmitted.

Fig. 19, shows the capacity required by the uplink channel. It is possible to observe how it increases during the first iterations up to a maximum, and then it decreases to reach a stationary value.

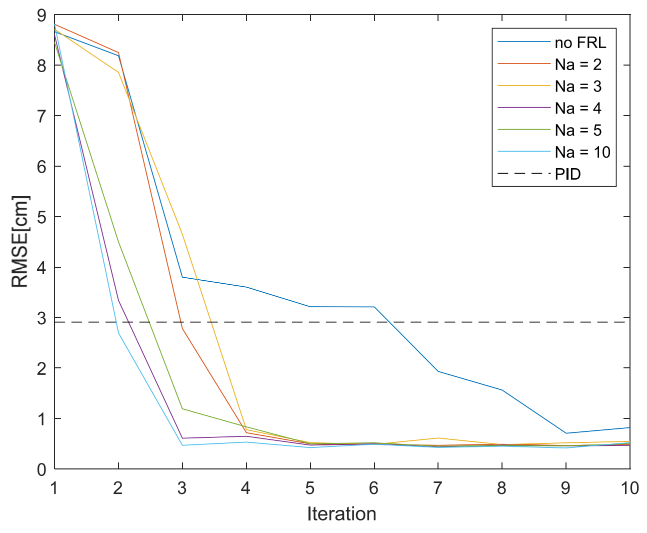

**Fig. 16.** Evolution of RMSE for elliptical trajectory and different number of AGVs.

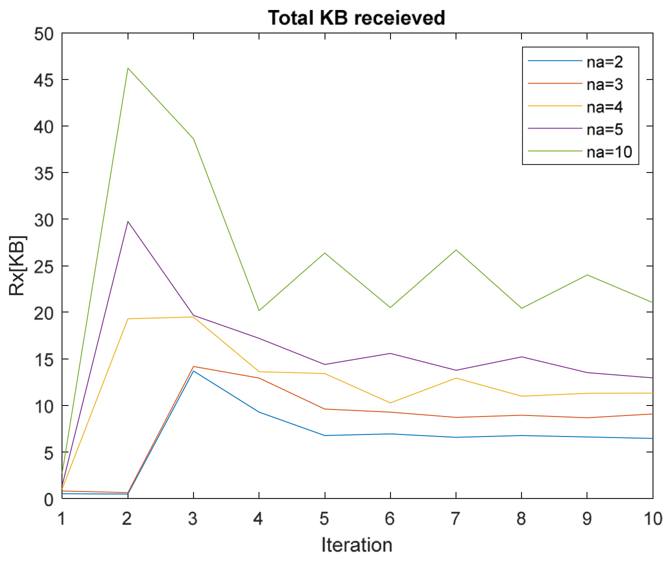

**Fig. 17.** Total KB received by the federated server.

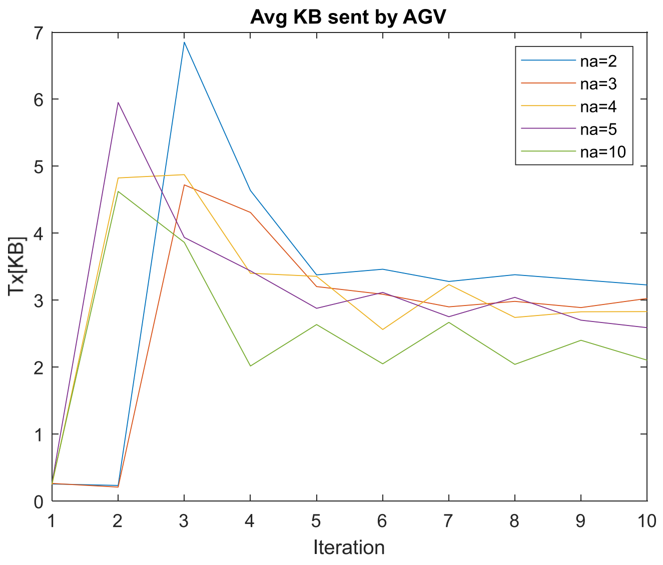

**Fig. 18.** Average KB sent by an AGV.

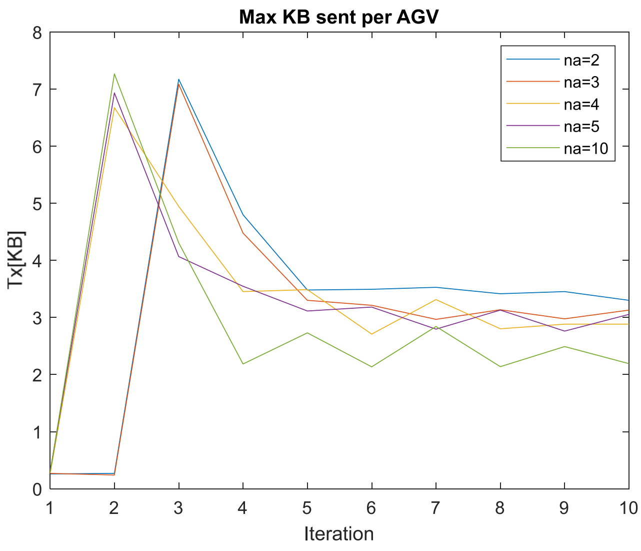

**Fig. 19.** Max KB sent by an AGV.

For a small number of AGVs (2–3), this peak appears at iteration 3, but for more AGVs the peak moves to iteration 2. This may be explained by the acceleration in the learning. However, it is interesting to see how the amplitude of this peak changes very little with the number of AGVs. Therefore, although the uplink capacity at iteration 10 is reduced we cannot extend the number of AGVs to reduce the uplink capacity as the amplitude of the peak remains practically unaltered.

## 5. Conclusions and future works

Industrial AGVs pose at least two interesting problems for the field of control engineering: speed control and trajectory control. When an AGV must circulate in confined spaces or very close to walls, machines, etc., or when it must make stops at stations to charge its battery, or pick up pallets, containers, carts, etc., precise trajectory control is crucial. This, together with the need to obtain good performance even in complex non-linear trajectories with abrupt changes, has directed the exploration of control solutions towards intelligent control techniques such as reinforcement learning.

In this work, a new federated discrete reinforcement learning approach for AGV trajectory control has been proposed. These industrial vehicles act as learning agents: they interact with the environment to learn the proper control law to follow the desired path. Embedded in each AGV, a set of modules oversees the individual reinforcement learning process. A state estimator identifies the state of the system based on the guidance error; a reward calculator assigns a reward to the previous action performed; an action selector implements a greedy strategy $\epsilon$ to select the best action; the policy is stored in a table since we are working with discrete reinforcement learning; and finally, a policy update algorithm adjusts the table to modify the policy based on the rewards received. Each AGV detects its individual policy changes and sends them to the federated server. The federated server collects all the learning from the AGVs and runs a learning aggregation function to get a group policy, which is downloaded to the AGVs for further learning.

To our knowledge, this is the first work using federated discrete reinforcement learning to directly control the angular velocity of industrial automated guided vehicles. In addition, the novel proposed federated reinforcement learning scheme (Fig. 4) can be used to speed up reinforcement learning in any application. To achieve this, different learning aggregation functions ((9)–(14) equations) have been designed to combine the individual learning policies ((7)–(8) equations). In addition, the control performance and the efficiency of the network communication process have been studied. The presented control strategy works significantly better than the conventional PID, and than an intelligent controller based on fuzzy logic, both optimized with genetic algorithms. Averaging all the trajectories, according to the results obtained by the simulation, the intelligent strategy improves the MAE by 78%, the RMSE by 75% and STD by 73%.

Regarding the speed of learning, the results for the different trajectories that have been tested demonstrate how the federated approach accelerates learning with respect to individual reinforcement learning. Specifically, an acceleration of learning of 50% in the best of cases and an average of 36% has been observed. An interesting result is that increasing the number of agents does speed up learning, but the value of RMSE to which the system converges does not depend on it. In addition, it has also been validated how the transmission of incremental tables, instead of total tables, to share the policy, allows significant savings in bandwidth. This result is especially interesting for selecting wireless network technology.

Different future works remain open, such as the definition of other aggregation functions and other reward strategies, and the application of this control approach to other engineering systems that act in groups, such as wind turbines in wind farms. It would also be desirable to test the approach with a fleet of AGV prototypes.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References

[1] F. Espinosa, C. Santos, J. Sierra-García, Transporte multi-AGV de una carga: estado del arte y propuesta centralizada, Rev. Iberoamericana Autom. e Inf. Ind. 18 (1) (2020) 82–91.
[2] L. Zamora-Cadenas, I. Velez, J.E. Sierra-Garcia, UWB-based safety system for autonomous guided vehicles without hardware on the infrastructure, IEEE Access 9 (2021) 96430–96443.
[3] X. Zhou, T. Chen, Y. Zhang, Research on intelligent AGV control system, in: 2018 Chinese Automation Congress, CAC, IEEE, 2018, pp. 58–61.
[4] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, C.P. Chen, Design and implementation of deep neural network-based control for automatic parking maneuver process, IEEE Trans. Neural Netw. Learn. Syst. 33 (4) (2020) 1400–1413.
[5] J.E. Sierra-Garcia, M. Santos, Combining reinforcement learning and conventional control to improve automatic guided vehicles tracking of complex trajectories, Expert Syst. (2022) e13076.
[6] R. Sutton, A. Barto, Reinforcement learning: An introduction, in: Google Scholar, MIT Press, 2018, pp. 329–331.
[7] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.
[8] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, et al., Towards federated learning at scale: System design, Proc. Mach. Learn. Syst. 1 (2019) 374–388.
[9] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.
[10] J. Leng, M. Zhou, J.L. Zhao, Y. Huang, Y. Bian, Blockchain security: A survey of techniques and research directions, IEEE Trans. Serv. Comput. 15 (4) (2020) 2490–2510.

[11] R. Sánchez, J.E. Sierra-García, M. Santos, Modelado de un AGV híbrido triciclo-diferencial, Rev. Iberoamericana Autom. e Inf. Ind. 19 (1) (2022) 84–95.
[12] B. Shubyn, D. Kostrzewa, P. Grzesik, P. Benecki, T. Maksymyuk, V. Sunderam, J.-H. Syu, J.C.-W. Lin, D. Mrozek, Federated learning for improved prediction of failures in autonomous guided vehicles, J. Comput. Sci. 68 (2023) 101956.
[13] B. Shubyn, D. Mrozek, T. Maksymyuk, V. Sunderam, D. Kostrzewa, P. Grzesik, P. Benecki, Federated learning for anomaly detection in industrial IoT-enabled production environment supported by autonomous guided vehicles, in: Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part IV, Springer, 2022, pp. 409–421.
[14] P. Liu, Z. Liu, J. Wang, Z. Wu, P. Li, H. Lu, Reinforcement learning empowered multi-AGV offloading scheduling in edge-cloud IIoT, J. Cloud Comput. 11 (1) (2022) 1–14.
[15] Q. Zhang, J. Lin, Q. Sha, B. He, G. Li, Deep interactive reinforcement learning for path following of autonomous underwater vehicle, IEEE Access 8 (2020) 24258–24268.
[16] A.B. Martinsen, A.M. Lekkas, Straight-path following for underactuated marine vessels using deep reinforcement learning, IFAC-PapersOnLine 51 (29) (2018) 329–334.
[17] B. Rubí, B. Morcego, R. Pérez, Deep reinforcement learning for quadrotor path following with adaptive velocity, Auton. Robots 45 (1) (2021) 119–134.
[18] W. Zhu, X. Guo, Y. Fang, X. Zhang, A path-integral-based reinforcement learning algorithm for path following of an autoassembly mobile robot, IEEE Trans. Neural Netw. Learn. Syst. 31 (11) (2019) 4487–4499.
[19] H. Hu, X. Jia, K. Liu, B. Sun, Self-adaptive traffic control model with behavior trees and reinforcement learning for AGV in industry 4.0, IEEE Trans. Ind. Inform. 17 (12) (2021) 7968–7979.
[20] H. Hu, X. Jia, Q. He, S. Fu, K. Liu, Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0, Comput. Ind. Eng. 149 (2020) 106749.
[21] T. Xue, P. Zeng, H. Yu, A reinforcement learning method for multi-AGV scheduling in manufacturing, in: 2018 IEEE International Conference on Industrial Technology, ICIT, IEEE, 2018, pp. 1557–1561.
[22] T.M. Ho, K.-K. Nguyen, M. Cheriet, Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic system, IEEE Trans. Autom. Sci. Eng. (2022).
[23] J. Leng, G. Ruan, Y. Song, Q. Liu, Y. Fu, K. Ding, X. Chen, A loosely-coupled deep reinforcement learning approach for order acceptance decision of mass-individualized printed circuit board manufacturing in industry 4.0, J. Clean. Prod. 280 (2021) 124405.
[24] J.E. Sierra-García, M. Santos, Exploring reward strategies for wind turbine pitch control by reinforcement learning, Appl. Sci. 10 (21) (2020) 7462.
[25] J.E. Sierra-Garcia, M. Santos, R. Pandit, Wind turbine pitch reinforcement learning control improved by PID regulator and learning observer, Eng. Appl. Artif. Intell. 111 (2022) 104769.
[26] J.E. Sierra-García, M. Santos, Mechatronic modelling of industrial AGVs: a complex system architecture, Complexity 2020 (2020) 1–21.
[27] R. Sánchez-Martinez, J.E. Sierra-García, M. Santos, Performance and extreme conditions analysis based on iterative modelling algorithm for multi-trailer AGVs, Mathematics 10 (24) (2022) 4783.
[28] A. Ghasempour, M. Martinez-Ramon, Short-term electric load prediction in smart grid using multi-output Gaussian processes regression, in: IEEE Kansas Power and Energy Conference, IEEE KPEC, 2023.
[29] J. Lee, J. Lee, Prediction-based energy saving mechanism in 3GPP NB-IoT networks, Sensors 17 (9) (2017) 2008.

**Sierra-Garcia J.E.** was born in Burgos, Spain. He received his M.Sc. degrees in Electronics and Telecommunications from the University of Valladolid (UVA) in 2007 and 2015 respectively, his M.Sc degree in Control Engineering from the National University for Distance Education (UNED) in 2014, and his Ph.D in Computer Science in 2019. Since 2012 he has been with the University of Burgos, where he is currently a Senior Lecturer in System Engineering and Automatic Control in the Department of Digitalization. He is the founder of the Joint Research Unit ASTI-UBU on Autonomous Vehicles, Mobile Robotics and AGVs. He has been principal researcher in more than 15 research projects related with mobile robotics. In addition, his professional experience includes 15 years working as researcher and software & hardware engineer in several companies. His major research interests are: Intelligent Control, Robotics, Autonomous Guided Vehicles, Modeling, Simulation, and Wind energy.

**Santos. M** was born in Madrid, Spain. She received her B.Sc. and M.Sc. degrees in Physics (Computer Engineering) and her Ph.D in Physics from the University Complutense of Madrid (UCM). She is currently Full Professor in System Engineering and Automatic Control. She is member of the European Academy of Sciences and Arts. She has published many papers in international scientific journals and several books and book chapters. She has supervised more than 10 Ph.Ds. She has worked on several national, European and international research projects, leading some of them. She currently serves as member of the editorial board of prestigious journals and she is editor-in-chief assistant of one of them. Her major research interests are: Intelligent Control (fuzzy and neuro-fuzzy), Pattern Recognition, Modeling and Simulation, Wind energy.