Message Passing Neural Network Versus Message Passing Algorithm for Cooperative Positioning

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Mattia Brambilla, *Member, IEEE*, Monica Nicoli, *Senior Member, IEEE*

***Abstract*—Cooperative Positioning (CP) relies on a network of connected agents equipped with sensing and communication technologies to improve the positioning performance of standalone solutions. In this paper, we develop a completely data-driven model combining Long Short-Term Memory (LSTM) and Message Passing Neural Network (MPNN) for CP, where agents estimate their state from inter-agent and state-dependent measurements. The proposed LSTM-MPNN model is derived from a parallelism with the probability-based Message Passing Algorithm (MPA) for CP, from which the graph-based structure of the problem and message passing scheme is inherited. In our solution, the LSTM block predicts the motion of the agents, while the MPNN elaborates the node and edge embeddings for an effective inference of the agent's state. We present numerical evidence that our approach can enhance position estimation, while being at the same time an order of magnitude less complex than typical particle-based implementations of MPA for non-linear problems. In particular, the presented LSTM-MPNN model can reduce the error on agents' positioning to one third compared to MPA-based CP, it holds a higher convergence speed and better exploits cooperation among agents.**

***Index Terms*—Message passing neural network, message passing algorithm, belief propagation, cooperative positioning, LSTM, message passing.**

I. Introduction

A. Contextualization and background

SIGNAL processing techniques operating over centralized or distributed network architectures have been largely studied in the past, especially for Situation Awareness (SA) applications [1]–[4]. The main application domains include Internet of Things (IoT) [5], Connected Autonomous Vehicles (CAVs) [6], [7] and Maritime Situational Awareness (MSA) [8], [9]. These applications are critical as they require sensors (hereafter generally referred as agents) monitoring and perceiving their surroundings and making informed decisions based on the perceived information. The key aspect is the cooperation among agents which enables Cooperative Positioning (CP) techniques and enhances the perception of the environment.

The Message Passing Algorithm (MPA), also known as Belief Propagation (BP) or Sum-Product Algorithm (SPA) [10], [11], is a probabilistic iterative technique which has gain a lot of interest in the field of CP [12] given its ability of linearly scaling with the number of agents [13]. MPA has been largely employed in a different number of SA frameworks, mainly addressing the Multiple Object Tracking (MOT) problem with static or mobile sensing agents [14]–[24], embedding or not the measurement to target association problem [25]–[28].

B. Camajori Tedeschini and M. Brambilla are with Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it, mattia.brambilla@polimi.it).

M. Nicoli is with Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20133 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Corresponding author: B. Camajori Tedeschini.

B. Related works

MPA attains optimal performances in case linear models and Gaussian processes, where the marginal posterior belief converges to the exact marginal posterior distribution. When the conditions of linearity and Gaussianity are not met, particle-based MPA can be employed, although this typically results in a notable increase of computational and communication expenses (i.e., due to particles' sharing and aggregation). Some works tried to improve performances of particle-based MPA implementations by reducing the particle degeneracy in dense and large networks [29], [30] or by auto-tuning the parameters of time-varying system models [31]. However, they did not resolve the main issue of MPA, which is related to the convergence of the beliefs.

Since MPA involves a repeated exchange of information (i.e., an iterative message passing) over a graph that is representative of the considered problem, the intrinsic cyclic structure of graphs leads the MPA's outcome to be only an approximation of the true marginal posterior distribution as the algorithm converges to a local optimum [32]–[35]. Specifically, the approximation of beliefs can be considered satisfactory if the optimization problem is locally convex. To improve the performances, Neural Enhanced Belief Propagation (NEBP) have been recently proposed [36]–[39], wherein MPA and Message Passing Neural Network (MPNN) are combined to rectify errors caused by cycles and model mismatch.

The MPNN [40], [41] is an extension of Neural Network (NN) customized to work on graph structures. Indeed, in conventional MPNN, a NN is present in each node and edge of the graph, elaborating the input features through an iterative message passing. The elaborated features, i.e., node and edge embeddings, are usually taken as input to perform a specific task, like node/edge regression or classification. Given their similarity with the message passing in MPA, they have been used within the NEBP framework to address the problems of Data Association (DA) [39], CP [37] and also MOT [38], as well as with the implicit cooperative positioning framework [42]. However, NEBP approaches require performing both iterations of MPNN and MPA, increasing the already high computational time of particle-based methods. Furthermore, it has been demonstrated that in cases where sufficient training data is available, MPNN exhibit superior performance to MPA on cyclic graphs [43], while at the same time being scalable and able to learn non-linear dependencies.

C. Contribution

Given the higher complexity of real-world problems, we deem it to be convenient to completely replace MPA with an equivalent MPNN version. A first attempt of using the MPNN as an alternative to MPA was done in [44], in which we focused on solving the DA task for pairing lidar detections in a network of connected vehicles. Here, we address the CP task, which requires intra-temporal characterization of the network dynamics and cannot be fulfilled by a stand-alone MPNN. In CP, a model of the temporal evolution (e.g., motion) of the agents is required to enable tracking. Thus, in this paper, we propose a joint architecture composed of a Long Short-Term Memory (LSTM) and an MPNN model. The LSTM learns the motion model of agents in time, while the iterative update of estimates based on measurements is obtained with the MPNN.

The main contributions of this paper are as follows:

- definition of a theoretical framework based on the analogy between MPA and MPNN, with focus on the definition of exchanged messages, iterative processing steps and inference prediction;
- proposal of an LSTM-MPNN model which completely replaces MPA for the task of CP. The model is trained using a centralized approach, while it is able to perform a completely distributed inference after deployment;
- comparison with the conventional particle-based MPA, with particular focus on positioning performances and generalization properties.

D. Paper organization

This paper is organized as follows. Sec. II is devoted to the description of the adopted system model. Sec. III first describes the MPA for CP, giving the main steps of the algorithm, and then defines the proposed LSTM-MPNN model with a one-to-one parallelism with MPA. Lastly, it provides insights on distributed inference and centralized training procedures. Sec. IV first presents the simulation scenario and implementation details, followed by simulation results. Lastly, Section V draws the conclusions.

II. System model

We denote with $\mathcal{I}_n = \{1, \dots, I_n\}$ a set of connected agents at timestep n. The connectivity graph between agents is denoted with $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$, where each node $i \in \mathcal{V}_n$ corresponds to an agent, while the edge (i, j), with $i \neq j$, indicates the presence of a communication link from agent i to agent j. Note that the graph is directed, i.e., edges (i, j) and (j, i) differ, and might not necessarily be contemporary present. Each agent $i \in \mathcal{I}_n$ communicates with the set $\mathcal{N}_{i,n}$ of its neighbors and it is described by the state $\mathbf{x}_{i,n}$, including kinematic parameters such as position and velocity. The motion model of agent i from time $n-1$ to time n is described by:

$$\mathbf{x}_{i,n} = f^{(\mathbf{x})}(\mathbf{x}_{i,n-1}, \mathbf{w}^{(\mathbf{x})}_{i,n-1}) , \tag{1}$$

where $\mathbf{w}^{(\mathbf{x})}_{i,n-1}$ is the driving noise process that accounts for motion uncertainty. The derived state-transition probability density function (pdf) is indicated with $p(\mathbf{x}_{i,n}|\mathbf{x}_{i,n-1})$, which, at time $n = 0$, coincides with the prior pdf $p(\mathbf{x}_{i,0})$.

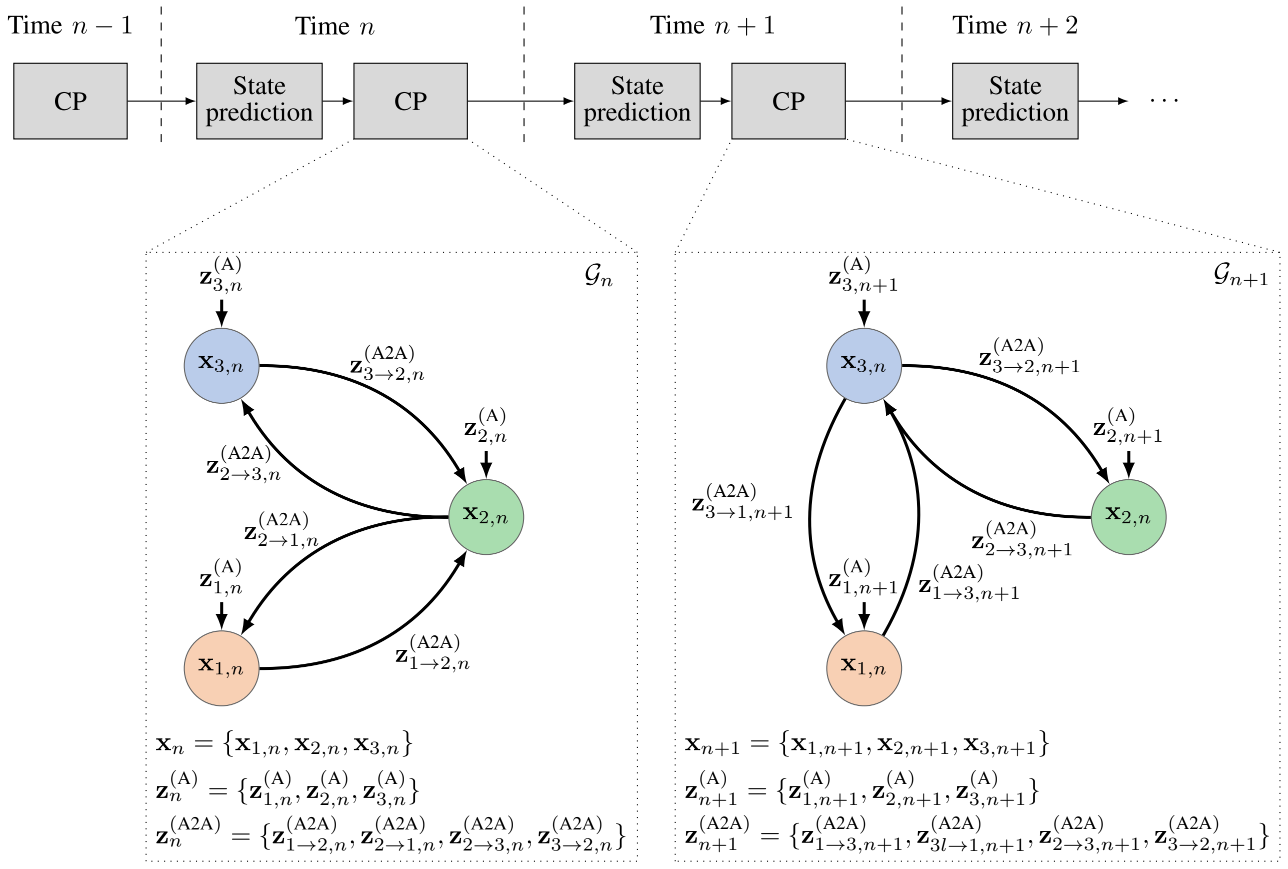
Each agent has access to two types of measurements: a partial and noisy observation $\mathbf{z}^{(\text{A})}_{i,n} = f^{(\text{A})}(\mathbf{x}_{i,n}, \mathbf{w}^{(\text{A})}_{i,n})$ of its own state vector, and an inter-agent measurement $\mathbf{z}^{(\text{A2A})}_{j\to i,n} = f^{(\text{A2A})}(\mathbf{x}_{j,n}, \mathbf{x}_{i,n}, \mathbf{w}^{(\text{A2A})}_{i,n})$, $\forall j \in \mathcal{N}_{i,n}$, where $\mathbf{w}^{(\text{A})}_{i,n}$ and $\mathbf{w}^{(\text{A2A})}_{i,n}$ are the state and inter-agent measurement noises, respectively. The functions $f^{(\text{A})}(\cdot)$ and $f^{(\text{A2A})}(\cdot)$, jointly with the statistics of noises $\mathbf{w}^{(\text{A})}_{i,n}$ and $\mathbf{w}^{(\text{A2A})}_{i,n}$, define the likelihood functions $p(\mathbf{z}^{(\text{A})}_{i,n}|\mathbf{x}_{i,n})$ and $p(\mathbf{z}^{(\text{A2A})}_{j\to i,n}|\mathbf{x}_{j,n}, \mathbf{x}_{i,n})$, respectively. The driving processes and measurement noises are assumed to be independent across agent pairs (i, j) and time n. We indicate with $\mathbf{x}_n = \{\mathbf{x}_{i,n}\}_{i=1}^{I_n}$ the set of state vectors of all agents at time n, while the two set of measurements are indicated with $\mathbf{z}^{(\text{A})}_n = \{\mathbf{z}^{(\text{A})}_{i,n}\}_{i\in\mathcal{I}_n}$ and $\mathbf{z}^{(\text{A2A})}_n = \{\mathbf{z}^{(\text{A2A})}_{j\to i,n}\}_{i\in\mathcal{I}_n, j\in\mathcal{N}_{i,n}}$. The overall set of measurements at time n is $\mathbf{z}_n = \{\mathbf{z}^{(\text{A})}_{i,n}, \mathbf{z}^{(\text{A2A})}_{i,n}\}$.

CP aims at estimating the states of agents from all the aggregated measurements up to time n, i.e., $\mathbf{z}^{(\text{A})}_{1:n}$ and $\mathbf{z}^{(\text{A2A})}_{1:n}$. The estimated state is indicated with $\widehat{\mathbf{x}}_n$. Probabilistic Bayesian methods, such as MPA, use the marginal posterior pdf $p(\mathbf{x}_{i,n}|\mathbf{z}^{(\text{A})}_{1:n}, \mathbf{z}^{(\text{A2A})}_{1:n})$ to estimate $\widehat{\mathbf{x}}_n$, e.g., through the Minimum Mean Square Error (MMSE) estimator $\widehat{\mathbf{x}}^{(\text{MMSE})}_{i,n} = \int \mathbf{x}_{i,n}\, p(\mathbf{x}_{i,n}|\mathbf{z}^{(\text{A})}_{1:n}, \mathbf{z}^{(\text{A2A})}_{1:n})\, \mathrm{d}\mathbf{x}_{i,n}$ [18]. On the other hand, discriminative probabilistic approaches, like Deep Learning (DL), directly define the posterior with a parametric model, i.e., $p(\mathbf{x}_{i,n}|\mathbf{z}^{(\text{A})}_{1:n}, \mathbf{z}^{(\text{A2A})}_{1:n}) = p(\mathbf{x}_{i,n}|\mathbf{z}^{(\text{A})}_{1:n}, \mathbf{z}^{(\text{A2A})}_{1:n}, \boldsymbol{\theta})$, and try to find the parameter vector $\boldsymbol{\theta}$ that maximizes $\widehat{\mathbf{x}}_{i,n} = \mathbb{E}_{\mathbf{x}_{i,n}}[p(\mathbf{x}_{i,n}|\mathbf{z}^{(\text{A})}_{1:n}, \mathbf{z}^{(\text{A2A})}_{1:n}, \boldsymbol{\theta})]$ [45]. This is done using as input a training dataset $\mathcal{S}^{\text{train}} = \{(\mathbf{x}_n, \mathbf{z}^{(\text{A})}_n, \mathbf{z}^{(\text{A2A})}_n)\}_{n=1}^{N_{\text{train}}}$ and minimizing the negative log-likelihood, i.e., $\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\theta}}[-\log(p(\mathbf{x}_{i,n}|\mathbf{z}^{(\text{A})}_{1:n}, \mathbf{z}^{(\text{A2A})}_{1:n}, \boldsymbol{\theta}))]$.

A compact representation of the temporal evolution of the system model is reported in Fig. 1, where two different network topologies (i.e., different measurement availability) at time n and $n+1$ are illustrated. The purpose of the figure is to highlight the temporal sequence of CP and visualize different combinations of the graph $\mathcal{G}_n$.

III. Cooperative positioning methods

In this section, we first review the MPA Bayesian solution for CP and then we perform a one-to-one comparison with our newly proposed LSTM-MPNN model. Lastly, a description of the inference and training procedure is given.



Time $n-1$
Time n
Time $n+1$
Time $n+2$
CP
State prediction
CP
State prediction
CP
State prediction
…
$\mathcal{G}_n$
$\mathbf{z}^{(A)}_{3,n}$
$\mathbf{x}_{3,n}$
$\mathbf{z}^{(A2A)}_{3\to2,n}$
$\mathbf{z}^{(A)}_{2,n}$
$\mathbf{z}^{(A2A)}_{2\to3,n}$
$\mathbf{x}_{2,n}$
$\mathbf{z}^{(A2A)}_{2\to1,n}$
$\mathbf{z}^{(A)}_{1,n}$
$\mathbf{x}_{1,n}$
$\mathbf{z}^{(A2A)}_{1\to2,n}$
$\mathbf{x}_n = \{\mathbf{x}_{1,n}, \mathbf{x}_{2,n}, \mathbf{x}_{3,n}\}$
$\mathbf{z}^{(A)}_n = \{\mathbf{z}^{(A)}_{1,n}, \mathbf{z}^{(A)}_{2,n}, \mathbf{z}^{(A)}_{3,n}\}$
$\mathbf{z}^{(A2A)}_n = \{\mathbf{z}^{(A2A)}_{1\to2,n}, \mathbf{z}^{(A2A)}_{2\to1,n}, \mathbf{z}^{(A2A)}_{2\to3,n}, \mathbf{z}^{(A2A)}_{3\to2,n}\}$
$\mathcal{G}_{n+1}$
$\mathbf{z}^{(A)}_{3,n+1}$
$\mathbf{x}_{3,n}$
$\mathbf{z}^{(A2A)}_{3\to2,n+1}$
$\mathbf{z}^{(A)}_{2,n+1}$
$\mathbf{z}^{(A2A)}_{3\to1,n+1}$
$\mathbf{x}_{2,n}$
$\mathbf{z}^{(A2A)}_{2\to3,n+1}$
$\mathbf{z}^{(A)}_{1,n+1}$
$\mathbf{z}^{(A2A)}_{1\to3,n+1}$
$\mathbf{x}_{1,n}$
$\mathbf{x}_{n+1} = \{\mathbf{x}_{1,n+1}, \mathbf{x}_{2,n+1}, \mathbf{x}_{3,n+1}\}$
$\mathbf{z}^{(A)}_{n+1} = \{\mathbf{z}^{(A)}_{1,n+1}, \mathbf{z}^{(A)}_{2,n+1}, \mathbf{z}^{(A)}_{3,n+1}\}$
$\mathbf{z}^{(A2A)}_{n+1} = \{\mathbf{z}^{(A2A)}_{1\to3,n+1}, \mathbf{z}^{(A2A)}_{3l\to1,n+1}, \mathbf{z}^{(A2A)}_{2\to3,n+1}, \mathbf{z}^{(A2A)}_{3\to2,n+1}\}$

Fig. 1. Illustration of the working principle of CP, with highlighted state vectors and measurement sets for two consecutive time instants. The figure highlights the variation of the graph $\mathcal{G}_n$ due to varied network topology and sets of measurements.

A. MPA-based CP

The agent's marginal posterior probability $p(\mathbf{x}_{i,n}|\mathbf{z}^{(A)}_{1:n}, \mathbf{z}^{(A2A)}_{1:n})$ can be obtained by marginalizing the joint posterior pdf $p(\mathbf{x}_{0:n}|\mathbf{z}^{(A)}_{1:n}, \mathbf{z}^{(A2A)}_{1:n})$, where $\mathbf{x}_{0:n} = \{\mathbf{x}_{n'}\}^{n}_{n'=0}$. Assuming statistical independence across agents at timestep $n = 0$ and adopting Bayes' rule, the joint posterior pdf is:

$$p(\mathbf{x}_{0:n}|\mathbf{z}^{(A)}_{1:n},\mathbf{z}^{(A2A)}_{1:n}) \propto \prod_{i=1}^{I_n} p(\mathbf{x}_{i,0}) \prod_{n'=1}^{n} p(\mathbf{x}_{i,n'}|\mathbf{x}_{i,n'-1}) \\ p(\mathbf{z}^{(A)}_{i,n'}|\mathbf{x}_{i,n'}) \prod_{j\in\mathcal{N}_{i,n'}} p(\mathbf{z}^{(A2A)}_{j\to i,n'}|\mathbf{x}_{j,n'}, \mathbf{x}_{i,n'})\,. \quad (2)$$

Since computing the marginalization of (2) can be unfeasible or extremely complex, the MPA addresses this issue by approximating the marginal posterior with an iterative message passing scheme over a factor graph which factorizes the joint posterior pdf in (2). Denoting the beliefs of agent i at timestep n and message passing iteration $t \in \{1, \ldots, T\}$ with $\mathbf{b}^{(t)}_{i,n} \triangleq \mathbf{b}^{(t)}_{i}(\mathbf{x}_{i,n}) \approx p(\mathbf{x}_{i,n}|\mathbf{z}^{(A)}_{1:n}, \mathbf{z}^{(A2A)}_{1:n})$, the MPA-based CP performs the following operations in parallel for each agent.

1) *Prediction message*: The predicted state of agent i is represented by the message:

$$\mu_{i,\overrightarrow{n}}(\mathbf{x}_{i,n}) \propto \int p(\mathbf{x}_{i,n}|\mathbf{x}_{i,n-1})\, \mathbf{b}^{(T)}_{i,n-1} \mathrm{d}\mathbf{x}_{i,n-1}\,, \quad (3)$$

where $\mathbf{b}^{(T)}_{i,n-1}$ is the agent's belief computed at previous time $n-1$ after T message passing steps. Note that the beliefs are initialized at time $n = 0$ as $\mathbf{b}^{(T)}_{i,0} \triangleq p(\mathbf{x}_{i,0})$.

2) *Beliefs exchange*: During message passing iteration $t \in \{1, \ldots, T\}$, each agent i broadcasts $\mathbf{b}^{(t-1)}_{i,n}$ and receives $\mathbf{b}^{(t-1)}_{j,n}$ from its neighbors $j \in \mathcal{N}_{i,n}$. At $t = 1$, the exchanged beliefs are $\mathbf{b}^{(0)}_{i,n} = \mu_{i,\overrightarrow{n}}(\mathbf{x}_{i,n})$.

3) *Measurement messages computation*: During message passing iteration $t \in \{1, \ldots, T\}$, each agent i computes two measurements messages (one for each type of

measurement) as:

$$\mu_{i,n}^{(t)(\mathrm{A})}(\mathbf{x}_{i,n}) \triangleq p(\mathbf{z}_{i,n}^{(\mathrm{A})}|\mathbf{x}_{i,n}) \,, \tag{4}$$

$$\mu_{j\to i,n}^{(t)(\mathrm{A2A})}(\mathbf{x}_{i,n}) \propto \int p(\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}|\mathbf{x}_{j,n}, \mathbf{x}_{i,n})\, \mathbf{b}_{j,n}^{(t-1)} \mathrm{d}\mathbf{x}_{j,n} \quad \forall j \in \mathcal{N}_{i,n}. \tag{5}$$

4) *Beliefs update*: At message passing iteration $t \in \{1, \ldots, T\}$, the beliefs are updated as:

$$\mathbf{b}_{i,n}^{(t)} \propto \mu_{i,\overrightarrow{n}}(\mathbf{x}_{i,n})\, \mu_{i,n}^{(t)(\mathrm{A})}(\mathbf{x}_{i,n}) \prod_{j\in\mathcal{N}_{i,n}} \mu_{j\to i,n}^{(t)(\mathrm{A2A})}(\mathbf{x}_{i,n}) \,. \tag{6}$$

5) *State inference*: Lastly, after T message passing steps, the state of agent i is estimated with the MMSE estimator as:

$$\widehat{\mathbf{x}}_{i,n} = \mathbb{E}\left[\mathbf{b}_{i,n}^{(t)}\right] \,. \tag{7}$$

Step 1) is indicated as *prediction step* and it is computed once per timestep n. On the contrary, steps 2), 3) and 4) are called *update steps* as they involve the measurements available at current timestep n and they are performed for all T message passing iterations per each timestep n.

For graphs with a tree structure, the MPA provides exact approximation of the beliefs, which coincide with the true marginal posterior pdf [10]. However, for cyclic graphs, MPA only provides a reasonably accurate approximation of the marginal posterior with a computational complexity that linearly scales with the number of agents I_n and message passing iterations T. Moreover, in case of non-linear motion or measurement models, particle-based methods can be exploited, despite incurring in a significant increase of communication and computational costs.

In comparison, MPNN holds the same time scalability [46], it has fewer parameters and it is able to catch any linear or non-linear relationships between input-output data, outperforming BP on loopy graphs if there is a sufficient amount of training data [43]. However, MPNN does not have the knowledge of features relation between time instants, i.e., each message passing iteration t at timestep n is completely independent with respect to the previous timestep $n-1$. To solve this issue, we propose an LSTM-MPNN model which combines the time-dependent capabilities of the recurrent network as well as the flexibility and scalability of the message passing over NNs.

B. *LSTM-MPNN-based CP*

The idea behind the proposed model is to build an equivalent DL-based model of the MPA-based CP described in Sec. III-A. We start describing the overall model structure, shown in Fig. 2, and then we analyze each single model block. The proposed architecture is composed of two main components, an LSTM block and an MPNN block. Adopting the same logic of the MPA at prediction step, the LSTM at time n receives in input the output of the MPNN $\widehat{\mathbf{x}}_{i,n-1}$ and predicts the most likely change of feature state according to the learned motion

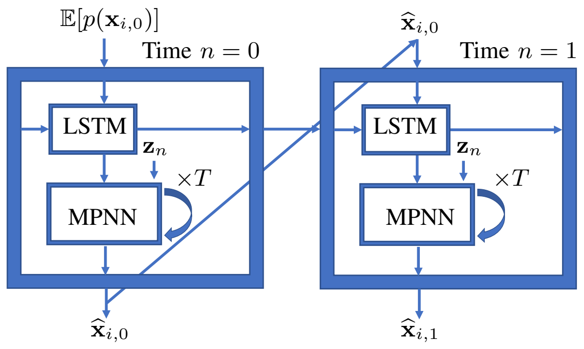

$\mathbb{E}[p(\mathbf{x}_{i,0})]$
Time $n=0$
LSTM
$\mathbf{z}_n$
$\times T$
MPNN
$\widehat{\mathbf{x}}_{i,0}$
$\widehat{\mathbf{x}}_{i,0}$
Time $n=1$
LSTM
$\mathbf{z}_n$
$\times T$
MPNN
$\widehat{\mathbf{x}}_{i,1}$

Fig. 2. Block representation of the proposed LSTM-MPNN model.

model of the agent. This is done by forwarding the hidden states of the LSTM throughout the timesteps. Therefore, the LSTM represents the equivalent block of (3) in the MPA. On the other hand, the MPNN block is performed over T message passing steps, exactly as the message passing in the MPA, and, at last iteration T, it returns the update of feature states, i.e., $\widehat{\mathbf{x}}_{i,n}$. We remark that, by analogy with MPA, we adopt the MPNN at the place of a Graph Neural Network (GNN) since the final prediction in the inference step (7) is a direct function of only the beliefs.

The MPNN runs on the same physical graph of the agent network, i.e., $\mathcal{G}_n$. It does not create a different graph abstraction, thus it can be computed among the physically connected agents. An MPNN considers two types of features: node embeddings, i.e., $\mathbf{v}_{i,n}^{(t)}$, and edge embeddings, i.e., $\mathbf{e}_{j\to i,n}^{(t)}$. The embeddings, also called attributes, contain elaborated latent information that propagates throughout $\mathcal{G}_n$ at every message passing step t. We can see an analogy between MPA update step and MPNN if we consider the node embeddings $\mathbf{v}_{i,n}^{(t)}$ as a elaborated versions of the beliefs $\mathbf{b}_{i,n}^{(t)}$, and the edge embeddings $\mathbf{e}_{j\to i,n}^{(t)}$ as the corresponding measurement messages between agents $\mu_{j\to i,n}^{(t)(\mathrm{A2A})}$.

The proposed MPNN model is composed of NNs for three different functions, encoding of input features ($g_v^{(\mathrm{A})}(\cdot)$ and $g_e^{(\mathrm{A2A})}(\cdot)$), update of node and edge embeddings ($g_v(\cdot)$ and $g_e(\cdot)$) and inference regression ($g_v^{(\mathrm{regres})}$). The encoding of input features is used to extract the most effective representation of measurements $\mathbf{z}_{i,n}^{(\mathrm{A})}$ and $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}$ to accomplish the regression task, i.e., agent state estimation. The update of the node and edge embeddings takes the role of (4), (5) and (6), preparing the node embeddings $\mathbf{v}_{i,n}^{(t)}$ for the inference prediction computed by the regressor $g_v^{(\mathrm{regres})}$.

The complete proposed LSTM-MPNN algorithm is shown in Fig. 3 and it is computed by each agent i in parallel.

1) *Prediction LSTM*: The LSTM model in agent i predicts the node embeddings $\mathbf{v}_{i,n}^{(t)}$ at time n as

$$\mathbf{v}_{i,n}^{(0)} = g_v^{(\mathrm{LSTM})}(\widehat{\mathbf{x}}_{i,n-1}) \,, \tag{8}$$

where $g_v^{(\mathrm{LSTM})}$ is the LSTM model. At $n=0$, the inference is initialized as $\widehat{\mathbf{x}}_{i,n-1} \triangleq \mathbb{E}[p(\mathbf{x}_{i,0})]$. Note that

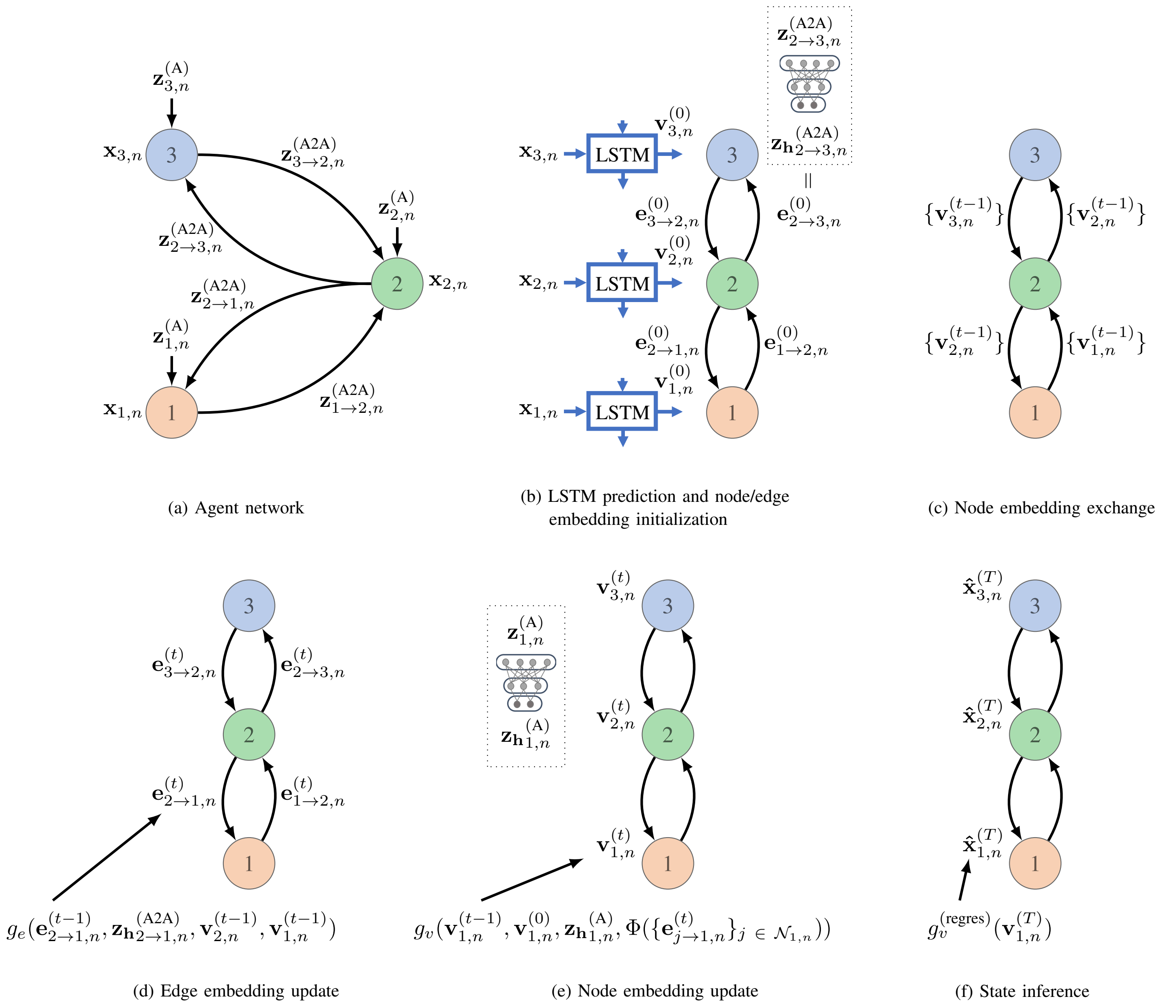

(a) Agent network
(b) LSTM prediction and node/edge embedding initialization
(c) Node embedding exchange
(d) Edge embedding update
(e) Node embedding update
(f) State inference
LSTM
$g_e(\mathbf{e}^{(t-1)}_{2\to1,n}, \mathbf{z_h}^{(\mathrm{A2A})}_{2\to1,n}, \mathbf{v}^{(t-1)}_{2,n}, \mathbf{v}^{(t-1)}_{1,n})$
$g_v(\mathbf{v}^{(t-1)}_{1,n}, \mathbf{v}^{(0)}_{1,n}, \mathbf{z_h}^{(\mathrm{A})}_{1,n}, \Phi(\{\mathbf{e}^{(t)}_{j\to1,n}\}_{j\in\mathcal{N}_{1,n}}))$
$g_v^{(\mathrm{regres})}(\mathbf{v}^{(T)}_{1,n})$

Fig. 3. LSTM-MPNN algorithm for CP. (a) graph representation of the agent network with agent states and measurements. (b) LSTM prediction at time n and initialization of node and edge embeddings at message passing iteration $t = 0$. (c) exchange of node embeddings among agents. (d) update of edge embeddings according to (11). (e) update of node embeddings according to (12). (f) state inference at time n after T message passing iteration according to (13).

the output of the LSTM coincides with the initialization of the node embeddings at message passing iteration $t = 0$. Observing the parallelism with MPA, the belief estimate $\mathbf{b}^{(T)}_{i,n-1}$ is replaced by the state estimate $\widehat{\mathbf{x}}_{i,n-1}$, while the state-transition probability pdf $p(\mathbf{x}_{i,n}|\mathbf{x}_{i,n-1})$ is learned by the LSTM.

2) *Measurements encoding*: At each time n, before starting the message passing, the agent and inter-agent measurements are encoded as:

$$\mathbf{z_h}^{(\mathrm{A})}_{i,n} = g_v^{(\mathrm{A})}(\mathbf{z}^{(\mathrm{A})}_{i,n}), \tag{9}$$

$$\mathbf{z_h}^{(\mathrm{A2A})}_{j\to i,n} = g_e^{(\mathrm{A2A})}(\mathbf{z}^{(\mathrm{A2A})}_{j\to i,n}), \quad \forall j \in \mathcal{N}_{i,n}. \tag{10}$$

The encoding is necessary to elaborate the input features, it transforms the input measurements into a hidden representation. This is important since all features within the message passing should not belong to the original feature space, but to the hidden space for data privacy reasons. At message passing iteration $t = 1$, the edge embeddings are initialized as: $\mathbf{e}^{(0)}_{j\to i,n} = \mathbf{z_h}^{(\mathrm{A2A})}_{j\to i,n}$.

3) *Node embeddings exchange*: At message passing iteration $t \in \{1, \ldots, T\}$, each agent i broadcasts $\mathbf{v}^{(t-1)}_{i,n}$ and receives $\mathbf{v}^{(t-1)}_{j,n}$ from its neighbors $j \in \mathcal{N}_{i,n}$. Here, the analogy with MPA is straightforward if we compare the beliefs exchange with the node embeddings exchange.

4) *Edge and node embeddings update*: At message passing iteration $t \in \{1, \ldots, T\}$, the edge embeddings are

updated as:

$$\mathbf{e}_{j\to i,n}^{(t)} = g_e(\mathbf{e}_{j\to i,n}^{(t-1)}, \mathbf{z_h}_{j\to i,n}^{(\text{A2A})}, \mathbf{v}_{j,n}^{(t-1)}, \mathbf{v}_{i,n}^{(t-1)}), \quad \forall j \in \mathcal{N}_{i,n}\,. \tag{11}$$

Note that (11) is the analogous of (5). Subsequently, the node embeddings are updated as:

$$\mathbf{v}_{i,n}^{(t)} = g_v(\mathbf{v}_{i,n}^{(t-1)}, \mathbf{v}_{i,n}^{(0)}, \mathbf{z_h}_{i,n}^{(\text{A})}, \Phi(\{\mathbf{e}_{j\to i,n}^{(t)}\}_{j\,\in\,\mathcal{N}_{i,n}}))\,, \tag{12}$$

where $\Phi(\cdot)$ is called aggregation function, i.e., a function invariant to permutations of its inputs (e.g., element-wise summation, mean, maximum). In the node embeddings update, exactly as in the beliefs update in (6), the initial node embeddings $\mathbf{v}_{i,n}^{(0)}$ are used as a short-connection from the output of the LSTM, i.e., prediction step.

5) *State inference*: Lastly, after T message passing steps, the regressor NN predicts the state of agent i as:

$$\widehat{\mathbf{x}}_{i,n} = \widehat{\mathbf{x}}_{i,n}^{(T)} = g_v^{(\text{regres})}(\mathbf{v}_{i,n}^{(T)})\,. \tag{13}$$

The MMSE estimator in (7) is substituted here by the node regressor $g_v^{(\text{regres})}(\cdot)$ which has the objective of extracting the state prediction from the compact node embeddings.

An interesting fact to point out is that the dimension of the node and edge embeddings, as well as the dimension of the encoded measurements, can be changed according to the problem. As an example, for the case of a state vector described in terms of 2D position and 2D velocity, we need a dimension of eight for the encoding of the node vector, i.e., corresponding to a propagation of a Gaussian belief distribution which holds only two parameters (mean and variance). Increasing the latent feature size leads to a higher complexity of the model which becomes able to learn more complex non-linear dependencies. On the contrary, in particle-based BP, each agent has to exchange a number of parameters equal to the number of adopted particles, each of them with a dimension of the state space, which overall is order of magnitudes higher than the dimension of the latent features in MPNN.

C. Inference and training procedure

The proposed LSTM-MPNN model for CP, exactly as the MPA-based CP, is suited for distributed inference. This is due to the fact that each agent i has its own NNs, i.e., $g_v^{(\text{LSTM})}(\cdot)$, $g_v^{(\text{A})}(\cdot)$, $g_e^{(\text{A2A})}(\cdot)$, $g_v(\cdot)$, $g_e(\cdot)$ and $g_v^{(\text{regres})}(\cdot)$. The physical exchange of embeddings only happens at step 3) of the algorithm at each message passing iteration t and each agent predicts its own state update according to (13). However, in order to have a convergence of the method, each NN at each agent should retain the same parameters, as in classical MPNN. This permits a scalable solution to a non-predetermined number of edges, i.e., measurements, and nodes, i.e., agents.

To this aim, we propose a centralized training procedure in which the NNs are firstly trained to learn the CP task and then deployed in an agent network. To compute the training loss and perform back-propagation, we employ the Residual Sum of Squares (RSS) that is estimated at each timestep n and at the end of each message passing iteration t after the regressor prediction $\widehat{\mathbf{x}}_{i,n}^{(t)}$ as:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|\mathcal{V}_n|}\sum_{t=1}^{T}\sum_{i\,\in\,\mathcal{V}_n}\|\widehat{\mathbf{x}}_{i,n}^{(t)} - \mathbf{x}_{i,n}^{(t)}\|_2^2, \tag{14}$$

where N is the time sequence length on which the LSTM is trained for tracking. For performance evaluation, we analyze the Root Mean Square Error (RMSE) on the position and velocity of agents.

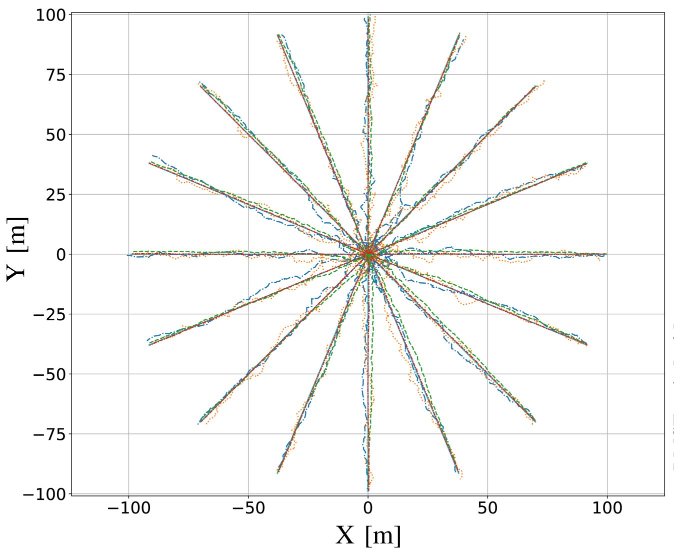
IV. Simulation experiments

A. Dataset

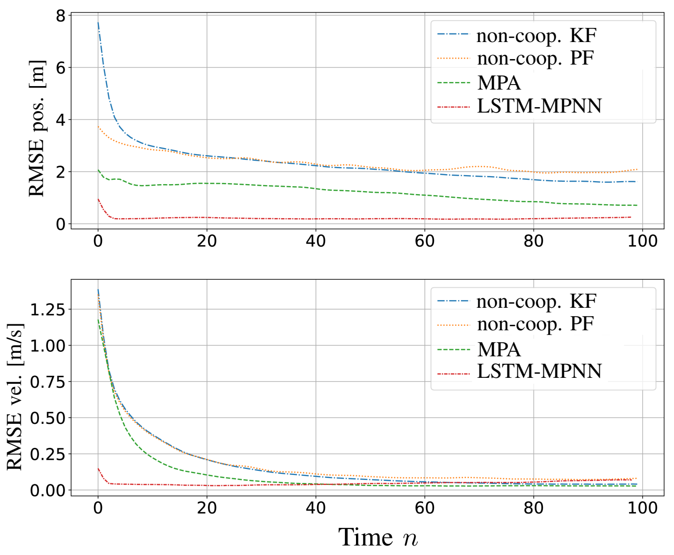
We consider a 2D scenario in which $I_n = 16$ connected agents move in an area of 200×200 m for 100 timesteps sampled at 1 s. The agent trajectories create a star shape starting from the origin and moving towards the limits of the area (see Fig. 4a), and the graph $\mathcal{G}_n$ is fully connected. The state of the agents is $\mathbf{x}_{i,n} = [\mathbf{p}_{i,n}^{\mathrm{T}}\dot{\mathbf{p}}_{i,n}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{p}_{i,n} \in \mathbb{R}^2$ and $\dot{\mathbf{p}}_{i,n} \in \mathbb{R}^2$ are the 2D position and velocity, respectively. The measurements are defined as $\mathbf{z}_{i,n}^{(\text{A})} = \mathbf{x}_{i,n} + \mathbf{w}_{i,n}^{(\text{A})}$ and $\mathbf{z}_{j\to i,n}^{(\text{A2A})} = \|\mathbf{p}_{j,n} - \mathbf{p}_{i,n}\|_2 + \mathbf{w}_{i,n}^{(\text{A2A})}$. Unless otherwise specified, we model the kinematics with a constant velocity model, while the state measurements and inter-agent measurements are zero-mean Gaussian distributed, i.e., $\mathbf{w}_{i,n}^{(\text{A})} \sim \mathcal{N}(\mathbf{0}_4, \mathbf{C}_{\mathbf{w}^{(\text{A})}})$, with $\mathbf{C}_{\mathbf{w}^{(\text{A})}} = \mathrm{diag}(\sigma^2_{\mathbf{p},\mathbf{w}^{(\text{A})}}, \sigma^2_{\mathbf{p},\mathbf{w}^{(\text{A})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\text{A})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\text{A})}})$, and $\mathbf{w}_{i,n}^{(\text{A2A})} \sim \mathcal{N}(0, \sigma^2_{\mathbf{w}^{(\text{A2A})}})$, with standard deviations $\sigma_{\mathbf{p},\mathbf{w}^{(\text{A})}} = 5$ m, $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\text{A})}} = 1$ m/s and $\sigma_{\mathbf{w}^{(\text{A2A})}} = 2$ m.

For both MPA and MPNN, we consider $T = 10$ message passing iterations. The proposed LSTM-MPNN model has been trained on 10000 instances of constant velocity trajectories, varying $\dot{\mathbf{p}}_{i,n} \in [-10, 10]$ m/s. In order to enhance model convergence and prevent biases, we standardized all the samples by performing a min-max scaler so that each feature lies in $[0, 1]$. This is done by having a prior knowledge on the agent position, i.e., $\mathbf{p}_{i,n} \in [-100, 100]$ m, and velocity, i.e., $\dot{\mathbf{p}}_{i,n} \in [-10, 10]$ m/s. We highlight that this is not a strong assumption, since to cover higher areas, we just need to enlarge the maximum range of the state features. We trained the LSTM-MPNN model for a total of 300 epochs, using a batch size of 32 samples and randomizing the order of the dataset at the beginning of each epoch. Here a sample refers to an instance of trajectories composed of $N = 10$ timesteps, i.e., the training sequence length of the LSTM model.

For the training and testing phases of the model, we used PyTorch version 1.12 and Python version 3.7.11. These operations were conducted on a workstation equipped with an Intel(R) Xeon(R) Silver 4210R CPU, which operates at a frequency of 2.40 GHz. The workstation was also supported by 96 GB of RAM and a Quadro RTX 6000 GPU with 24 GB of memory. For what concerns the optimizer, we used the Adam optimization algorithm [47] with an initial learning rate of 0.0001, and momentum values of 0.9 and 0.999 for β_1 and β_2, respectively.



(a) Agent network scenario



(b) Position and velocity RMSE

Fig. 4. Performance evaluation of the proposed LSTM-MPNN for CP. (a) scenario with 16 moving agents. (b) RMSE of position and velocity over time for the non-cooperative Kalman and particle filters, and the cooperative MPA and the proposed LSTM-MPNN.

B. Model and implementation details

The LSTM architecture has been inspired by [48], but here we reduced the complexity such that it is constituted by two LSTM layers and a hidden output dimension, i.e., node embeddings, of 16. The complexity reduction is motivated by considering that the state estimation in CP comprises two steps (i.e., prediction and update). For the measurement encoding, update of node and edge embeddings, and state inference, we use Multi-Layer Perceptrons (MLPs) with linear layers and Gaussian Error Linear Unitss (GELUs) activation functions [49]. The complete LSTM and MLPs model structures are reported in Table I.

The selected final architecture of our model was derived upon experimentation, including varying the number of layers and neurons. However, the main rationale behind the general structures is the following. First, the NN encoders $g_v^{(\mathrm{A})}(\cdot)$ and $g_e^{(\mathrm{A2A})}(\cdot)$, despite their small input sizes of 1×1 and 4×1, are characterized by a higher computational complexity when normalized by input size in comparison to the node and edge embedding updates. Second, between $g_v(\cdot)$ and $g_e(\cdot)$, the latter is more complex given its primary role at the initial step of each iteration and the need of processing non-linear inter-agent measurements $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}$. Finally, the state inference regressor $g_v^{(\mathrm{regres})}(\cdot)$ is the most challenging task and thus it requires an additional linear layer (4 in total) to effectively predict the state.

C. Simulation results

1) Tracking performances: The first test aims at assessing the performances of the proposed LSTM-MPNN model to highlight the advantages of adopting a data-driven solution. The comparison includes two non cooperative algorithms, i.e., a Kalman Filter (KF) and a Particle Filter (PF), which only use the agent state measurements $\mathbf{z}_{i,n}^{(\mathrm{A})}$, and the cooperative MPA described in Sec. III-A, which uses the agent state measurements $\mathbf{z}_{i,n}^{(\mathrm{A})}$ and the inter-agent ones $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}$, and it is implemented following a particle based approach.

For the particle-based methods, the number of particles was set to $N_{\mathrm{PF}} = 1000$. We would like to point out that the KF represents the optimal non-cooperative case since all noises are Gaussian and all models, i.e., motion and agent state measurements, are linear. On the contrary, the MPA results to be sub-optimal given the non-linearity of inter-agent measurements and the full connectivity of the agent graph.

The results of the comparison are reported in Fig. 4, where we plot a realization of the scenario (Fig. 4a) and the RMSE of the position and velocity for each timestep (Fig. 4b) (averaged over 30 simulations). Starting from non-cooperative methods, we notice that the KF is well approximated by the particle-based MPA and reaches a positioning error of 1.62 m while tracking. The cooperative MPA permits to increase furthermore the performances by reaching 89 cm at convergence. Lastly, the proposed LSTM-MPNN method outperforms all the other methods, achieving an RMSE of 21 cm on the position. Concerning the velocities, all the methods converge at about 0.05 m/s of RMSE. Apart from regime performances, an additional important aspect to consider is the model convergence. Indeed, the LSTM-MPNN method is able to converge after few timesteps, while BP-based algorithms require more time. This feature allows the LSTM-MPNN model to fast react in case of track initialization and recovery after a sudden trajectory variation as it rapidly forgets the previous estimates, updating the state knowledge through LSTM hidden states.

2) Generalization capabilities: This experiment compares the performances of MPA and LSTM-MPNN under different validation conditions. In particular, we test different intensities of driving process and state-measurement noises. The MPA retains inside the true value of the motion and measurement

TABLE I
DETAILS ABOUT THE LAYER STRUCTURE OF LSTM AND MLPS MODELS.

LSTM	
Type	**Output**
Input	4×1
LSTM layer	128×1
LSTM layer	256×1
Maxout	128×1
Linear	64×1
Linear	16×1

$g_v^{(\mathrm{A})}$		$g_v^{(\mathrm{A2A})}$		g_v		g_e		$g_v^{(\mathrm{regres})}$	
Type	**Output**	**Type**	**Output**	**Type**	**Output**	**Type**	**Output**	**Type**	**Output**
Input	4×1	Input	1×1	Input	64×1	Input	64×1	Input	16×1
Linear+GELU	72×1	Linear+GELU	18×1	Linear+GELU	18×1	Linear+GELU	80×1	Linear+GELU	16×1
Linear+GELU	16×1	Linear+GELU	18×1			Linear+GELU	16×1	Linear+GELU	256×1
		Linear+GELU	16×1			Linear+GELU	16×1	Linear+GELU	128×1
								Linear	4×1

noises, while the LSTM-MPNN has been trained with noise-free driving and measurement models. This is done in order to prove the efficacy of the method with a full-calibrated MPA and a completely miscalibrated LSTM-MPNN.

In a first test, we consider a zero-mean Gaussian-distributed driving noise, i.e., $\mathbf{w}_{i,n}^{(\mathbf{x})} \sim \mathcal{N}(\mathbf{0}_4, \mathbf{C}_{\mathbf{w}^{(\mathbf{x})}})$, with $\mathbf{C}_{\mathbf{w}^{(\mathbf{x})}} = \mathrm{diag}(\sigma^2_{\mathbf{p},\mathbf{w}^{(\mathbf{x})}}, \sigma^2_{\mathbf{p},\mathbf{w}^{(\mathbf{x})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\mathbf{x})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\mathbf{x})}})$. In Fig. 5, we compare the MPA and LSTM-MPNN in terms of RMSE on position, with $\sigma_{\mathbf{p},\mathbf{w}^{(\mathbf{x})}} = 0$ m and varying $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\mathbf{x})}} \in [0, 10]$ m/s. From the results, we notice that when $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\mathbf{x})}} < 0.5$ m/s, the proposed LSTM-MPNN outperforms the particle-based MPA. On the contrary, increasing the noise intensity leads to a faster degradation of performances with respect to the MPA. This is justified by two main factors. Firstly, the model has been trained using error-free trajectories, leading it to anticipate motion models that adhere to the distribution of the training trajectories. Secondly, the increased noise raises the likelihood of encountering an agent with a speed beyond the training range of $[-10, 10]$ m/s, potentially leading to inaccurate predictions.

In a second test, we consider a constant motion model and a varying state-measurement noise, i.e., $\sigma_{\mathbf{p},\mathbf{w}^{(\mathrm{A})}} \in [0, 10]$ m. This time, analyzing the results in Fig. 6, we observe that the LSTM-MPNN achieves a lower RMSE across all considered values of state measurement noise. This confirms the trend that on peak performances, i.e., with same noises and within the same area of cooperation, the proposed LSTM-MPNN model outperforms the cooperative MPA method by reducing the error to one third. Moreover, even with unfavorable conditions, i.e., training on absence of noise, the LSTM-MPNN model better generalizes against noisy state-measurements.

3) Impact on different number of agents: For this last assessment, we evaluate how the different number of cooperative agents affects the performances of the two methods. To this aim, in Fig. 7, we plot the RMSE on the position varying the number of connected agents $I_n \in [2, 22]$. As expected, we observe that, for a low number of agents, the two methods tend to converge to the RMSE achieved for the non-cooperative case, i.e., about 1.5 m. This confirms that with a decreasing number of agents, the LSTM-MPNN

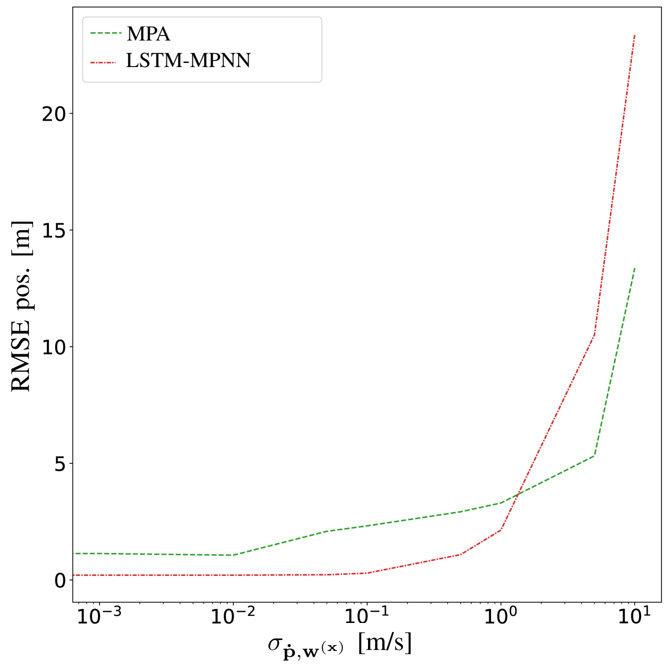


Fig. 5. Comparison of the impact of driving noise error in terms of RMSE of the position between MPA and LSTM-MPNN.

model converges to the optimal case of single-agent KFing. Increasing I_n, the cooperation plays a crucial role in improving CP, especially for the proposed LSTM-MPNN model. As a matter of fact, in LSTM-MPNN with only 6 cooperative agents the same RMSE of 20 agents for the MPA method is achieved.

4) Computational complexity: Given the same graph structure and same number of message passing iterations between MPA and LSTM-MPNN models, the major difference in computational complexity lies in the computation of the prediction and update steps. In order to compare one-to-one the two methods, we define with N_{PF} and N_h the number of particles in MPA and the dimension of the node and edge embeddings, respectively. These variables drive the computational complexity since they tune the trade-off

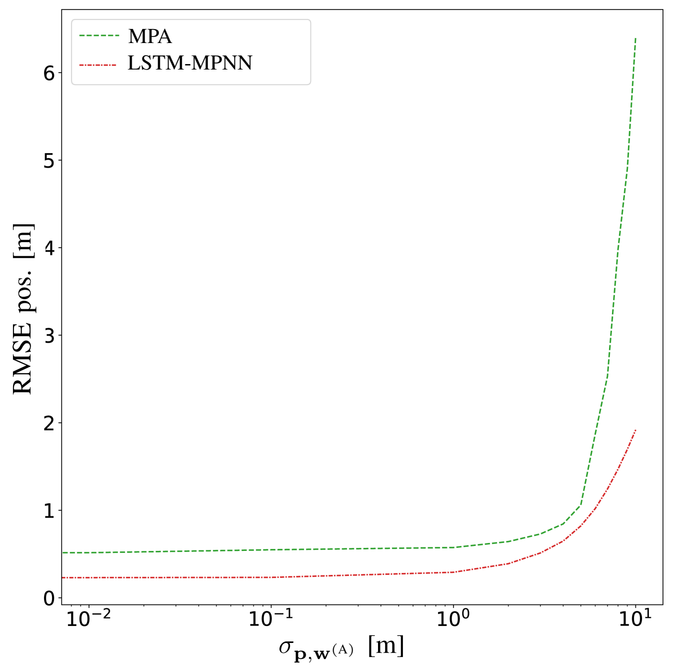


Fig. 6. Comparison of the impact of state-measurement noise error in terms of RMSE of the position between MPA and LSTM-MPNN.

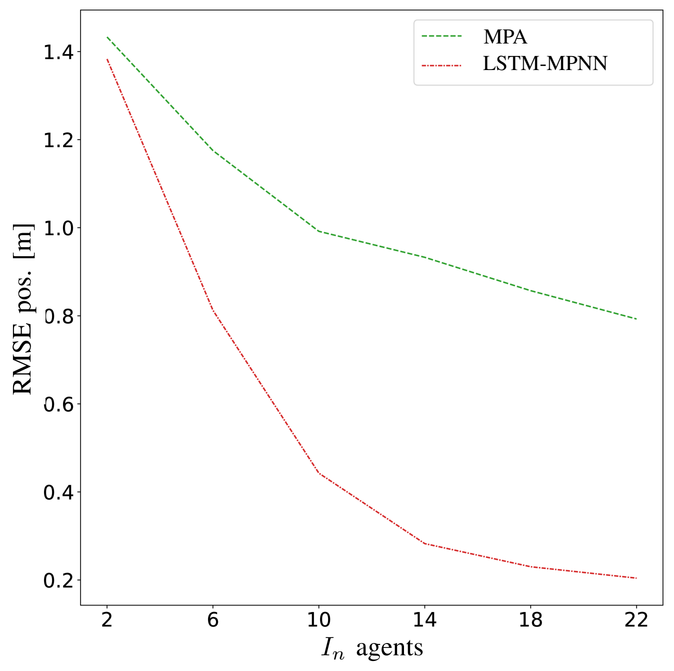


Fig. 7. Comparison of the impact of varying number of cooperative agents in terms of RMSE of the position between MPA and LSTM-MPNN.

between performances and efficiency. Indeed, N_{PF} and N_h are the dimension of the messages exchanged during each message passing step. Moreover, in MPA, N_{PF} regulates the capability of the model of approximating the distributions according to the importance sampling principle. In LSTM-MPNN, N_h has the same function of N_{PF} in MPA, but with the fundamental difference that here the exchanged vector, i.e.,

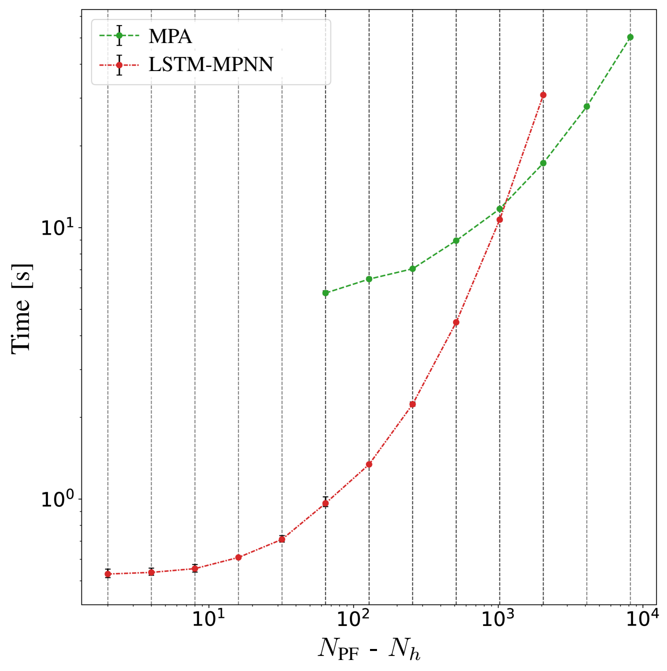


Fig. 8. Comparison of the impact of varying number of particles N_{PF} and node embedding dimension N_h in terms of inference time between MPA and LSTM-MPNN.

node embedding, does not represent an approximation of the distributions using a sampling mechanism. On the contrary, it represents an effective combination of distribution parameters, e.g., moments, in order to accomplish the CP task.

To this aim, in Fig. 8 we show the whole prediction time of an instance of agent trajectories, i.e., 16 agents moving as shown in Fig. 4a, varying N_{PF} or N_h according to the model. Note that here the time required to exchange the particles and the node embeddings are not considered. Moreover, for a fair comparison, all agent predictions are computed on CPU and in a sequential manner. Observing the results, we notice that the LSTM-MPNN is very efficient for a number of latent dimension $N_h < 100$, performing the whole inference in less than 1 s. On the contrary, the MPA is slower even with $N_{\text{PF}} = 100$ particles. Comparing the two methods for $N_{\text{PF}} = N_h$, we note that, from a pure inference time point of view, it is more convenient to adopt the LSTM-MPNN if $N_{\text{PF}} = N_h < 1000$. However, we would like to point out that, comparing the two methods with the previously adopted $N_{\text{PF}} = 1000$ and $N_h = 16$, we obtain an inference time of 600 ms and 11 s for the LSTM-MPNN and MPA, respectively. Thus, with the proposed method, we reach one third of the error at $1/18$ of the time.

V. Conclusion

This paper addressed the problem of CP by proposing an innovative LSTM-MPNN model that can be considered as a promising alternative to conventional probabilistic MPA. Besides providing for the first time a one-to-one parallelism with respect to MPA, we demonstrated the improved performance of a fully DL-based model. We detailed each part of the proposed model, starting from the need of temporal-dependence solved using an LSTM block, up to the message passing structure. The MPNN runs on the same physical graph created by the network of connected agents and it is able to perform inference in a completely distributed way. Mirroring the MPA, the messages, i.e., node embeddings, are exchanged between agents until convergence. Finally, as opposed to the MMSE estimator in MPA, the state inference is carried out through a NN at the node.

We validated the proposed approach in a synthetic network of cooperative agents moving in a scenario over straight trajectories. Numerical results showed that the proposed approach is able to address the problem of CP in an efficient and effective way by outperforming particle-based MPA in a different number of aspects. First, under peak performances point of view, the LSTM-MPNN model reaches a lower RMSE on the position by a factor of 3. Second, the LSTM-MPNN model holds a much higher speed of convergence, an order of magnitude lower computational complexity. As an example, in our experiments, the dimension of the messages exchanged by the MPNN is 16, while the number of particles exchanged by the BP is 1000. Moreover, the proposed model better handles different state-measurement noises, as well as driving noises if trained on all ranges of state feature values. Finally, the LSTM-MPNN model better exploits the power of cooperation, giving a huge improvement even with small number of cooperating agents.

The value of cooperative positioning is foreseen to dramatically grow over the next several years, especially in the context of automated and connected mobility, where dense networks of agents have to handle complex and dynamic environments. It results that an effective data-driven approach is of paramount importance to enhance positioning capabilities. Our method makes a step toward this direction, by enabling distributed and efficient cooperative inference. Future developments could be implementing not only a distributed inference but also a distributed training, maintaining at the same time the agent's local data privacy. Moreover, applications of fully DL-based methods are foreseen for the major fields of target detection and tracking.

Code availability statement

The GitHub repository with the dataset and the Python code for the model, training and inference is available upon request to the corresponding author.

References

[1] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proceedings of the IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018.

[2] M. Z. Win, W. Dai, Y. Shen, G. Chrisikos *et al.*, "Network operation strategies for efficient localization and navigation," *Proceedings of the IEEE*, vol. 106, no. 7, pp. 1224–1254, Jul. 2018.

[3] M. Win, A. Conti, S. Mazuelas, Y. Shen *et al.*, "Network localization and navigation via cooperation," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 56–62, May 2011.

[4] D. Gaglione, G. Soldi, F. Meyer, F. Hlawatsch *et al.*, "Bayesian information fusion and multitarget tracking for maritime situational awareness," *IET Radar, Sonar & Navigation*, vol. 14, no. 12, pp. 1845–1857, Dec. 2020.

[5] A. A. Saucan and M. Z. Win, "Information-seeking sensor selection for ocean-of-things," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 072–10 088, Oct. 2020.

[6] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020.

[7] S. Zhang, E. Staudinger, T. Jost, W. Wang *et al.*, "Distributed direct localization suitable for dense networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 2, pp. 1209–1227, Apr. 2020.

[8] G. Ferri, A. Munafo, A. Tesei, P. Braca *et al.*, "Cooperative robotic networks for underwater surveillance: an overview," *IET Radar, Sonar & Navigation*, vol. 11, no. 12, pp. 1740–1761, Dec. 2017.

[9] P. Braca, P. Willett, K. LePage, S. Marano *et al.*, "Bayesian tracking in underwater wireless sensor networks with port-starboard ambiguity," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1864–1878, Apr. 2014.

[10] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[11] D. Bickson, O. Shental, and D. Dolev, "Distributed Kalman filter via Gaussian belief propagation," in *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, Sep. 2008, pp. 628–635.

[12] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.

[13] R. Sánchez-Cauce, I. París, and F. J. Díez, "Sum-product networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3821–3839, Jul. 2022.

[14] W. Zhang and F. Meyer, "Multisensor multiobject tracking with high-dimensional object states," Dec. 2022, arXiv:2212.14556.

[15] D. Gaglione, P. Braca, G. Soldi, F. Meyer *et al.*, "Fusion of sensor measurements and target-provided information in multitarget tracking," *IEEE Transactions on Signal Processing*, vol. 70, pp. 322–336, Dec. 2022.

[16] F. Meyer and J. Williams, "Scalable detection and tracking of geometric extended objects," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6283–6298, Oct. 2021.

[17] R. Mendrzik, M. Brambilla, C. Allmann, M. Nicoli *et al.*, "Joint multitarget tracking and dynamic network localization in the underwater domain," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 4890–4894.

[18] F. Meyer, T. Kropfreiter, J. L. Williams, R. Lau *et al.*, "Message passing algorithms for scalable multitarget tracking," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018.

[19] F. Meyer, P. Braca, P. Willett, and F. Hlawatsch, "A scalable algorithm for tracking an unknown number of targets using multiple sensors," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3478–3493, Jul. 2017.

[20] M. Brambilla, D. Gaglione, G. Soldi, R. Mendrzik *et al.*, "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 169–195, Mar. 2022.

[21] B. Teague, Z. Liu, F. Meyer, A. Conti *et al.*, "Network localization and navigation with scalable inference and efficient operation," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2072–2087, Jun. 2022.

[22] F. Meyer and M. Z. Win, "Joint navigation and multitarget tracking in networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. Kansas City, MO: IEEE, May 2018, pp. 1–6.

[23] F. Meyer, E. Riegler, O. Hlinka, and F. Hlawatsch, "Simultaneous distributed sensor self-localization and target tracking using belief propagation and likelihood consensus," in *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov. 2012, pp. 1212–1216.

[24] E. Leitinger, S. Grebien, and K. Witrisal, "Multipath-based SLAM using belief propagation with interacting multiple dynamic models," in *2021 15th European Conference on Antennas and Propagation (EuCAP)*. Dusseldorf, Germany: IEEE, Mar. 2021, pp. 1–5.

[25] F. Meyer and M. Z. Win, "Scalable data association for extended object tracking," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 491–507, May 2020.

[26] D. Gaglione, G. Soldi, P. Braca, G. De Magistris *et al.*, "Classification-aided multitarget tracking using the sum-product algorithm," *IEEE Signal Processing Letters*, vol. 27, pp. 1710–1714, Sep. 2020.

[27] F. Meyer and M. Z. Win, "Data association for tracking extended targets," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*. Norfolk, VA, USA: IEEE, Nov. 2019, pp. 337–342.

[28] F. Meyer, Z. Liu, and M. Z. Win, "Scalable probabilistic data association with extended objects," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. Shanghai, China: IEEE, May 2019, pp. 1–6.

[29] L. Wielandner, E. Leitinger, F. Meyer, B. Teague *et al.*, "Message passing-based cooperative localization with embedded particle flow," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Singapore, Singapore: IEEE, May 2022, pp. 5652–5656.

[30] W. Zhang and F. Meyer, "Graph-based multiobject tracking with embedded particle flow," in *2021 IEEE Radar Conference (RadarConf21)*. Atlanta, GA, USA: IEEE, May 2021, pp. 1–6.

[31] G. Soldi, F. Meyer, P. Braca, and F. Hlawatsch, "Self-tuning algorithms for multisensor-multitarget tracking using belief propagation," *IEEE Transactions on Signal Processing*, vol. 67, no. 15, pp. 3922–3937, Aug. 2019.

[32] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, Oct. 2001.

[33] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.

[34] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, Nov. 2007.

[35] E. Riegler, G. E. Kirkelund, C. N. Manchon, M.-A. Badiu *et al.*, "Merging belief propagation and the mean field approximation: A free energy approach," *IEEE Transactions on Information Theory*, vol. 59, no. 1, pp. 588–602, Jan. 2013.

[36] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 685–693.

[37] M. Liang and F. Meyer, "Neural enhanced belief propagation for cooperative localization," in *2021 IEEE Statistical Signal Processing Workshop (SSP)*. Rio de Janeiro, Brazil: IEEE, Jul. 2021, pp. 326–330.

[38] ——, "Neural enhanced belief propagation for multiobject tracking," Dec. 2022, arXiv:2212.08340.

[39] ——, "Neural enhanced belief propagation for data association in multiobject tracking," in *2022 25th International Conference on Information Fusion (FUSION)*, Jul. 2022, pp. 1–7.

[40] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner *et al.*, "Computational capabilities of graph neural networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 81–102, Jan. 2009.

[41] ——, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[42] L. Barbieri, B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative lidar sensing," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2023.

[43] K. Yoon, R. Liao, Y. Xiong, L. Zhang *et al.*, "Inference in probabilistic graphical models by graph neural networks," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. Pacific Grove, CA, USA: IEEE, Nov. 2019, pp. 868–875.

[44] B. Camajori Tedeschini, M. Brambilla, L. Barbieri, G. Balducci *et al.*, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Transactions on Signal Processing*, pp. 1–15, Aug. 2023.

[45] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4, no. 4.

[46] J. Zhou, G. Cui, S. Hu, Z. Zhang *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Apr. 2020.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017, arXiv:1412.6980.

[48] J. Liu, Z. Wang, and M. Xu, "Deepmtt: A deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network," *Information Fusion*, vol. 53, pp. 289–304, Jan. 2020.

[49] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," Jul. 2020, arXiv:1606.08415.

Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) received the B.Sc. (Hons.) in Computer Science and M.Sc. (Hons.) degrees in Telecommunications Engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively. From November 2021 he started as PhD fellow in Information Technology at Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. He is currently a visiting researcher with the Laboratory for Information & Decision Systems at the Massachusetts Institute of Technology (MIT), Cambridge, MA.

His research interests include federated learning, machine learning and localization methods. He was a recipient of the Ph.D. grant from the ministry of the Italian government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship granted by MIT and Politecnico di Milano.

Mattia Brambilla (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering and the Ph.D. degree (cum laude) in information technology from the Politecnico di Milano, in 2015, 2017, and 2021, respectively. He was a Visiting Researcher with the NATO Centre for Maritime Research and Experimentation (CMRE), La Spezia, Italy, in 2019. In 2021 he joined the faculty of Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) at the Politecnico di Milano as Research Fellow. His research interests include signal processing, statistical learning, and data fusion for cooperative localization and communication. He was the recipient of the Best Student Paper Award at the 2018 IEEE Statistical Signal Processing Workshop.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT). She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the EuMA Mediterranean Microwave Symposium, in 2022, the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the IET Intelligent Transport Systems journal, in 2014. She is an Associate Editor of the IEEE Transactions on Intelligent Transportation Systems. She has also served as an Associate Editor for the EURASIP Journal on Wireless Communications and Networking, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.