Unveiling Agents' Confidence in Opinion Dynamics Models via Graph Neural Networks

Víctor A. Vargas-Pérez, Jesús Giráldez-Cru, Pablo Mesejo, and Oscar Cordón

***Abstract*—Opinion Dynamics models in social networks are a valuable tool to study how opinions evolve within a population. However, these models often rely on agent-level parameters that are difficult to measure in a real population. This is the case of the confidence threshold in opinion dynamics models based on bounded confidence, where agents are only influenced by other agents having a similar opinion (given by this confidence threshold). Consequently, a common practice is to apply a universal threshold to the entire population and calibrate its value to match observed real-world data, despite being an unrealistic assumption. In this work, we propose an alternative approach using graph neural networks to infer agent-level confidence thresholds in the opinion dynamics of the Hegselmann-Krause model of bounded confidence. This eliminates the need for additional simulations when faced with new case studies. To this end, we construct a comprehensive synthetic training dataset that includes different network topologies and configurations of thresholds and opinions. Through multiple training runs utilizing different architectures, we identify GraphSAGE as the most effective solution, achieving a coefficient of determination R^2 above 0.7 in test datasets derived from real-world topologies. Remarkably, this performance holds even when the test topologies differ in size from those considered during training.**

***Index Terms*—Opinion Dynamics, Bounded Confidence, Hegselmann-Krause Model, Agent-Based Modeling, Graph Neural Networks**

I. Introduction

AGENT-based models (ABM) [1], [2] are a leading technology to study complex systems. ABMs rely on the definition of agents' behavioral micro-rules [3], and the complex behavior of the system is inferred from the aggregation of those agents' behaviors in a bottom-up manner [4]. In many contexts, this methodology is more accurate than deriving a global definition of the system by top-down approaches. ABMs have been extensively used in many contexts, including modeling emergency evacuations [5], the analysis of political rallies [6], and the study of artificial societies [7], among others.

A straightforward application of ABM is Opinion Dynamics (OD) [8], [9], [10], [11]. OD models aim to study the evolution of opinions in a population [12]. In this scenario, agents' opinions evolve as a consequence of interactions between them [13], as well as other endogenous and exogenous factors, such as stubbornness [14] or mass communication [15], for instance. OD models are commonly used to study whether the opinion fusion rule (i.e., the agents' micro-rule that defines how opinions are updated after agents' interactions) is able to reach a consensus, a polarization, or a fragmentation of opinions [16]. Consensus is the state where all opinions have the same value, while polarization and fragmentation are cases where several clusters of opinions co-exist: two clusters in the former case, and more than two clusters in the latter [13].

A very well-studied OD model is Bounded Confidence (BC) [17], [18], [19]. The main principle of the BC model is that agents are only influenced by other agents having a similar opinion to them, where a confidence threshold determines the area of social influence for each agent. The rationale behind this principle is social confirmation bias [20], i.e., the interactions that influence individuals' own opinions are mostly those occurring with other individuals similar to them. In this work, we focus on the Hegselmann-Krause (HK) model [18], where an agent is randomly selected at each time step, and updates its opinion by averaging the agents' opinions in its confidence area (i.e., the neighbors of its social network whose opinions are within the confidence threshold). It is well known that, for a given threshold, a low value produces opinion consensus, whereas a high value results in a fragmentation of opinions [21], [22]. However, it remains unclear how to determine the confidence threshold for a given population, and this is the problem addressed in this work.

The task of determining confidence thresholds that yield specific final opinions can be framed as a model parameter calibration problem. Heuristic search methods such as evolutionary algorithms (EAs) can be used to address this problem [23], [24], but require separate calibration for each new case study. In this work, we aim to develop a method that generalizes the knowledge from training. This will allow us to effectively determine, in negligible time, the confidence thresholds for case studies that differ from those observed during the training phase, even with varying population sizes.

It is well established the interplay between Machine Learning (ML) techniques and ABMs, and how the former can help to improve the performance of the latter [25], [26], including learning micro-agent level situational awareness, micro-agent

Received 27 May 2024; revised 10 October 2024 and 21 November 2024; accepted 26 November 2024. This work was supported in part by MCIN/AEI/10.13039/501100011033 and ERDF "A way of making Europe" under Grant CONFIA PID2021-122916NB-I00, in part by the FPU Program under Grant FPU20/02441, and in part by Grant RYC2022-036395-I funded by MICIU/AEI/10.13039/501100011033 and ESF+. Funding for open access charge: Universidad de Granada/CBUA. (Corresponding author: Víctor A. Vargas-Pérez.)

Víctor A. Vargas-Pérez, Pablo Mesejo, and Oscar Cordón are with the Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada (UGR), 18071 Granada, Spain, and also with Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), 18071 Granada, Spain (e-mail: victorvp@ugr.es; pmesejo@decsai.ugr.es; ocordon@decsai.ugr.es).

Jesús Giráldez-Cru is with the University of Seville (US), 41044 Seville, Spain, and also with Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), 18071 Granada, Spain (e-mail: jgiraldez@us.es).

level behavior intervention, macro-ABM level emulation, and sequential decision-making, among other tasks. Among the diversity of existing ML techniques, Deep Learning (DL) algorithms stand out nowadays because of the massive amount of available data [27]. In this context, Graph Neural Networks (GNN) are DL models especially designed to deal with non-Euclidean data that can be represented in the form of graphs [28]. They are able to perform most ML tasks, including classification and regression tasks at the level of nodes, edges, and the whole graph [29]. Notice that ABM in general, and OD models in particular, are good examples of such graph-structured data, and the aforementioned open question of determining the confidence threshold of a given population can be seen as a suitable learning task for GNNs.

The main contribution of this work is a thorough analysis of predicting the confidence threshold of BC models by means of GNNs. To the best of our knowledge, this is the first work studying OD models by means of GNNs. In particular, we analyze the performance of several GNN layers extensively used in the literature, including Graph Convolutional Network (GCN) [30], GraphSAGE [31], and Graph Attention Network (GAT) [32], in order to predict the confidence thresholds of a population in several synthetic social network topologies. These trained models are further validated in real-world social networks. Our experimental evaluation shows that GraphSAGE and GAT are able to obtain remarkably accurate results on this problem, exhibiting notable generalization capabilities to unseen social networks. Specifically, GraphSAGE is able to achieve consistent results even when the agent population has heterogeneous confidence thresholds. This means that the challenging question of determining the confidence threshold of BC OD models can be effectively addressed by GNNs.

The rest of this work is organized as follows. Section II presents the preliminary background on OD models and GNNs. The methodology of our proposal is presented in Section III, while Section IV is devoted to the empirical validation of it. Finally, we conclude and discuss potential direction of future research in Section V.

II. Background

This section describes some preliminaries on OD models and GNN frameworks, required to properly understand the developments made in the current contribution.

A. Opinion Dynamics Models

OD models are usually classified into discrete and continuous models, according to the representation of opinions into discrete (usually binary) and continuous values, respectively [13]. Some examples of discrete OD models are the voter model [9] and the majority rule [10], among others. Unfortunately, discrete models are usually unable to represent the complex nature of opinions in most contexts. In contrast, continuous OD models generally provide a more realistic representation of opinions. Therefore, in the rest of this work we focus on continuous OD.

Let $o_i(t)$ be the opinion of agent $i \in \{1, \dots, N\}$ at time step t. For simplicity, we assume $o_i(t) \in [0, 1]$, although any other real interval could be used instead. One of first continuous OD model proposed in the literature is the DeGroot [8], where opinions are updated as per the following rule:

$$o_i(t+1) = \sum_{j=1}^{N} w_{ij} o_j(t) \tag{1}$$

where w_{ij} is the weight agent i gives to agent j. In this simple linear model, the sufficient and necessary conditions to reach a consensus are well-known [33].

A first step to introduce non-linearity in continuous OD model is BC, including the Deffuant–Weisbuch (DW) [17] and the HK [18] models. Although both models rely on only considering opinions within a bounded confidence area (given by a confidence threshold), the DW model only considers random pairwise encounters while the HK model incorporates all the opinions in the neighborhood of an agent. Therefore, the HK model is more suitable for modeling interactions in large groups, such as formal meetings, whereas the DW is better suited for pairwise interactions within a large population [21]. For this reason, we focus on the HK model to study the confidence threshold of a population by means of GNNs.

Let us consider a population of N agents interacting in a social network, which is represented by a graph $G(V, A)$, being V the set of nodes (with $|V| = N$), and A its $N \times N$ adjacency matrix (i.e., $A_{ij} = 1$ iff. there is an edge between nodes i and j in G, $A_{ij} = 0$ otherwise).[1] Each agent is represented by a distinct node i in the graph, and interacts with other agents within its neighborhood $\mathcal{N}(i)$, i.e., with some agent j in the set $\mathcal{N}(i) = \{j \in V \,|\, A_{ij} = 1\}$.

In the HK model, at each time step $t = \{1, \dots T\}$[2], an agent $i \in \{1, \dots, N\}$ is randomly selected and updates its own opinion $o_i(t)$ as:

$$o_i(t+1) = \frac{\sum_{j \in C(i)} o_j(t)}{|C(i)|} \tag{2}$$

where $C(i) = \{j \in \mathcal{N}(i) \cup \{i\} \mid |o_i(t) - o_j(t)| < \varepsilon_i\}$ is the set of node i's neighbors in its confidence area, with ε_i being the confidence threshold of agent i. Notice that this confidence threshold ε_i is agent-specific, i.e., it may differ from one agent to another.

The values of the confidence thresholds are responsible for the distribution of final opinions in an execution of the model, resulting in consensus, polarization, or fragmentation of opinions depending on such a value. Figure 1 depicts an example of this phenomenon, executing the HK model with a uniform confidence threshold ($\varepsilon_i = \varepsilon$) in a fully-mixed population of $N = 1,000$ agents during $T = 10^5$ time steps, considering three distinct confidence thresholds: (left) $\varepsilon = 0.1$, (center) $\varepsilon = 0.2$, and (right) $\varepsilon = 0.3$. It can be observed that the execution with the lowest confidence threshold $\varepsilon = 0.1$ results in a profile of final opinions with four clusters of different opinions. With an intermediate value of this confidence threshold, as in the case with $\varepsilon = 0.2$, the final opinions are just polarized into 2 clusters of distinct opinions.

[1] For completeness, we assume $A_{ii} = 0$ for every node i.
[2] T is the total number of time steps of a simulation.

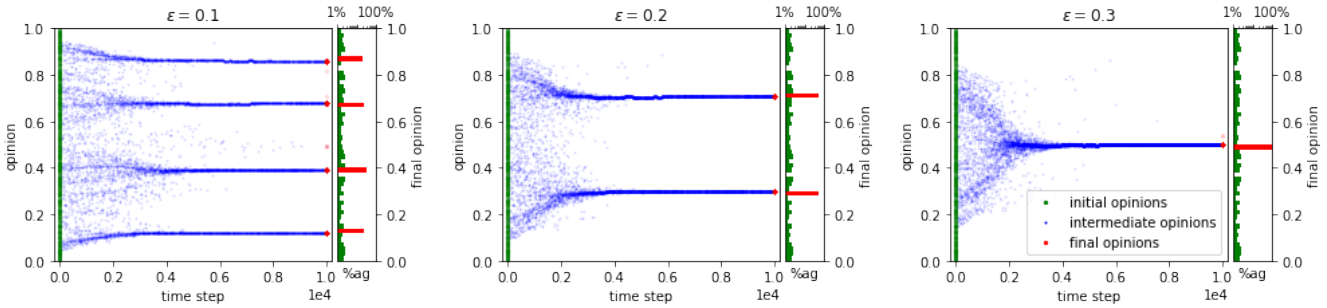

Fig. 1. Example of the execution of the HK model with several values of the confidence threshold ε. For each execution, the left subplots represent the evolution of opinions during the executions and the right subplots represent the distribution of initial and final opinions.

Finally, with the highest value of ε, as in the execution with $\varepsilon = 0.3$, the population reaches a consensus of opinions.

B. Graph Neural Networks

GNNs are a family of neural networks (NN) that operate on graph data. The key idea behind them is to generate node embeddings, which are vector representations of the nodes that encode both their individual properties (i.e., node features) and the structural information of the graph [34].

The most standardized framework for designing GNNs is Neural Message Passing, wherein nodes receive information from their neighbors (i.e., messages), aggregate it, and update their internal representations using a NN [35]. This process unfolds over a fixed number of iterations or layers L. Let $\mathbf{h}_i^{(l)}$ denote the representation of node i at layer l, and $\mathbf{x}_i = \mathbf{h}_i^{(0)}$ denote its initial feature vector. The message-passing update at layer $l \in \{1, \dots L\}$ can be expressed as shown in Equation 3:

$$\begin{aligned} \mathbf{h}_i^{(l+1)} &= \text{UPT}^{(l)}\left(\mathbf{h}_i^{(l)}, \text{AGG}^{(l)}(\{\mathbf{m}_{ij}^{(l)}, \forall j \in \mathcal{N}(i)\})\right), \\ \mathbf{m}_{ij}^{(l)} &= \text{MSG}^{(l)}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right) \end{aligned} \tag{3}$$

where $\text{MSG}^{(l)}$, $\text{AGG}^{(l)}$, and $\text{UPT}^{(l)}$ are differentiable functions that can be implemented with NNs. A foundational example within this framework is GCN [30], which introduced effective strategies for neighborhood aggregation and parameter sharing, forming the basis for many advanced architectures. GCN is defined in Equation 4 as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{\mathbf{h}_j^{(l)}}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(j)|}}\right) \tag{4}$$

where σ is a non-linear activation function and $\mathbf{W}$ is a weight matrix.[3] The aggregation includes symmetric normalization and self-loops, stabilizing training and incorporating the center node's features [34]. However, this formulation processes neighborhood and self-information uniformly, limiting its expressiveness.

[3]For simplicity, the corresponding bias term is omitted in the definition of all GNN layers.

A more advanced layer is GraphSAGE [31], which allows different ways of aggregating the neighborhood and explicitly defines the update phase. Equation 5 shows this layer with a mean aggregator. By distinguishing between the node and its neighbors, GraphSAGE is capable of learning richer and more flexible representations than GCN, albeit at the cost of requiring a higher number of parameters.

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_1^{(l)}\mathbf{h}_i^{(l)} + \mathbf{W}_2^{(l)} \frac{\sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(l)}}{|\mathcal{N}(i)|}\right) \tag{5}$$

In addition to distinguishing between the node and its neighborhood, it may be beneficial to create a distinction among the neighbors themselves, assigning different levels of importance to each of them. This is precisely what GAT [32] accomplishes: it calculates a weighted mean of the neighborhood messages by incorporating an attention mechanism, as depicted in Equations 6-8:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\alpha_{ii}^{(l)}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)}\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right), \tag{6}$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(e(\mathbf{h}_i^{(l)}, \mathbf{h}_k^{(l)}))}, \tag{7}$$

$$e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}) = \text{LeakyReLU}\left(\mathbf{a}_1^{(l)^T}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)} + \mathbf{a}_2^{(l)^T}\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right) \tag{8}$$

where $\mathbf{a}_1$ and $\mathbf{a}_2$ are learnable weight vectors, e represents the importance of node j to node i, and α_{ij} is the corresponding attention coefficient after applying softmax normalization over the neighborhood $\mathcal{N}(i)$.

Although GAT is one of the most popular GNN architectures nowadays, it presents a limitation: its attention mechanism is static, meaning that the resulting ranking of attention coefficients for every neighbor node is unconditioned by the node of origin. This issue was addressed in GATv2 [36], which achieves a dynamic attention mechanism with a small adjustment involving a reordering of internal operations within GAT and an increase in the number of parameters. Specifically, the change entails duplicating parameters in $\mathbf{W}$ by splitting it into two different matrices, $\mathbf{W}_1$ and $\mathbf{W}_2$. In turn, $\mathbf{a}_1$ and $\mathbf{a}_2$

are condensed into a single smaller weight vector $\mathbf{a}$, which is applied after the LeakyReLU activation function. Thus, Equation 8 is now rewritten as shown in Equation 9:

$$e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}) = \mathbf{a}^{(l)^T} \text{LeakyReLU}\left(\mathbf{W}_1^{(l)}\mathbf{h}_i^{(l)} + \mathbf{W}_2^{(l)}\mathbf{h}_j^{(l)}\right) \tag{9}$$

The increasing expressive power of these three layers also implies an increasing number of parameters, as shown in Table I. For illustration, this table also provides the parameter count of a perceptron layer that only considers feature information, not structure. Additionally, we include the theoretical upper bound for the time complexity of a forward pass. Although the asymptotic time complexity is the same for these GNN layers, differences in constant factors and unit operation costs (e.g., square roots in GCN or exponentials in GATv2) can significantly impact their actual runtime.

TABLE I
NUMBER OF PARAMETERS (INCLUDING BIAS TERMS) AND ASYMPTOTIC TIME COMPLEXITY OF THE DISCUSSED NN LAYERS. LET N BE THE NUMBER OF NODES OF THE GRAPH, E BE THE NUMBER OF EDGES, H^i BE THE INPUT NODE EMBEDDING SIZE, AND H^o BE THE OUTPUT NODE EMBEDDING SIZE.

Layer type	# Parameters	Time complexity
MLP	$H^i \cdot H^o + H^o$	$O(N \cdot H^i \cdot H^o)$
GCN	$H^i \cdot H^o + H^o$	$O(N \cdot H^i \cdot H^o + E \cdot H^o)$
GraphSAGE	$2 \cdot H^i \cdot H^o + H^o$	$O(N \cdot H^i \cdot H^o + E \cdot H^o)$
GATv2	$2 \cdot H^i \cdot H^o + 4 \cdot H^o$	$O(N \cdot H^i \cdot H^o + E \cdot H^o)$

Regardless the layer type considered, GNNs are useful for addressing various ML tasks, such as node-level classification (or equivalently, regression), link prediction, or graph classification [37]. When working at the node level, a distinction must be made between transductive and inductive problems, which are commonly associated with semi-supervised and supervised learning, respectively [34]. In the transductive case, the training and test node sets are connected within the same graph. This implies that test nodes participate in the training process to perform message passing operations, though their labels are neither observed nor used to compute the loss function. Alternatively, the inductive case resembles traditional supervised learning: test nodes are completely separated from the training nodes, residing in disjoint graphs, and thus do not intervene in any way in the training process. As discussed in Section III, our methodology involves solving an inductive node-level regression problem.

III. METHODOLOGY

Our methodology aims to determine the individual confidence threshold ε_i of each agent i within a population whose interactions are governed by the HK model. To accomplish this task, we propose training a ML model to learn the function that maps the initial and final opinions of each agent, along with the social network connecting them, to the corresponding confidence threshold for each agent. Given the inherent graph structure of this input data, a GNN emerges as the most natural choice for the ML model.

A standard approach that closely resembles our objective is ABM calibration. In this methodology, the confidence thresholds are treated as free parameters of the OD model, and the goal is to find a configuration of values that yields specific simulation outputs, namely, the desired set of final opinions. When the dimensionality of the parameters is high, it is appropriate to employ an automated calibration process using heuristic search methods. These methods aim to minimize an objective function that quantifies the distance between the resulting simulation output and the target output [23], [24]. EAs are among the most commonly used optimization techniques for this purpose [38], [39], [40], due to its capability to explore a wide range of parameter values and consider the non-linear interactions between them.

However, we must highlight that the objective of this work is not to calibrate the parameters of an ABM to a specific case study. Instead, we aim to develop a method that, after a single training process, is capable of inferring the agent parameters for any new BC case study, without additional ABM simulation runs. To the best of our knowledge, this is the first study that seeks to generalize OD ABM parameter fitting across scenarios that were not observed during the optimization phase and with varying population sizes. The motivation behind this novel approach lies in two key advantages with respect to traditional ABM calibration:

- A standard calibration approach is specific to each instance of the problem or case study, requiring a comprehensive set of simulations for each new scenario, without direct use of prior knowledge. In contrast, our approach benefits from learning generalization. In a new case study, the trained GNN requires only one inference step to derive the thresholds associated with each agent, without the need for retraining or conducting additional simulation runs of the ABM, which tend to be computationally expensive.
- It is common in OD to establish a uniform confidence threshold for all agents in the population [15], or to alternatively segment it. This practice aims to prevent a substantial increase in the number of parameters to calibrate, which would be equal to the population size if a fully personalized threshold were adopted. Alternatively, our approach facilitates addressing a more realistic scenario where each individual owns a distinct confidence threshold, without entailing an increase in the number of parameters of the GNN. Indeed, our approach is potentially capable of managing scenarios with varying population sizes, as the same GNN model can be applied to graphs or cases of different sizes.

Despite the potential advantages described in our proposal, it is crucial for the GNN training process to be sufficiently robust and generalizable for effective application in real-world scenarios not previously observed. Particularly, to ensure the GNN can generalize across different topologies, the training set must encompass diverse characteristics across different topologies. Fortunately, since the HK model is fully defined (see Equation 2), we can generate a synthetic training dataset representing a wide array of problem variants. This includes

diverse social network topologies and a multitude of settings of confidence thresholds and initial opinions within the range $[0, 1]$. Moreover, knowledge about the problem allows us to tailor the data generation process to encompass various scenarios while avoiding the over-representation of equivalent scenarios. Specifically, prior research [18] suggests that, despite the theoretical domain of $\varepsilon \in [0, 1]$, in practical scenarios where $\varepsilon \geq 0.25$, the population tends to reach a consensus opinion, resulting in similar final outcomes (see Figure 1). Therefore, it is appropriate to limit the generation of synthetic data with values of ε_i within a narrower domain, such as $[0, 0.5]$.[4]

Figure 2 shows a summary of the steps for training the GNN model, which are defined as follows:

1) Selection and generation of a set of topologies or adjacency matrices $\mathcal{A}$ representing different social networks of agents. It is essential to consider a range of topologies based on different network properties such as density, clustering coefficient, and degree distribution, among others, to provide the learning process with enough diversity.
2) Generation of M different random configurations of initial opinions and thresholds for the agents for each social network topology considered. We denote the resulting set of graphs as $\mathcal{G}$, and for each topology $A \in \mathcal{A}$, we define the subset $\mathcal{G}(A) = \{G \in \mathcal{G} | G(V, A)\}$, where $|\mathcal{G}(A)| = M, \forall A \in \mathcal{A}$. This ensures a comprehensive representation of diverse scenarios across all considered topologies.
3) Running the HK model on each graph $G \in \mathcal{G}$, thereby obtaining the final opinion of each agent.[5] This process allows us to construct the dataset associated with each graph, denoted as $D(G(V, A)) = ((\mathbf{X}, A), \mathbf{y})$. Here, $\mathbf{X} \in [0, 1]^{N \times 2}$ comprises the feature vector of each node i, defined as $\mathbf{X} = \{\mathbf{x}_i = (o_i(1), o_i(T)) | \forall i \in \{1, \ldots, N\}\}$. Meanwhile, the label vector $\mathbf{y} \in [0, 0.5]^N$ contains the confidence threshold of each node, i.e., $\mathbf{y} = \{y_i = \varepsilon_i | \forall i \in \{1, \ldots, N\}\}$. The number of time-steps T of the simulations should be high enough to reach a stationary state, i.e., a state beyond which the agents no longer change their opinions.
4) Training a GNN following a supervised approach to address the regression problem associated with the previous datasets. Formally, the GNN aims to approximate the function $f : (\mathbf{X}, A) \rightarrow \mathbf{y}$. Note that, while the dimensions of $\mathbf{X}$, A, and $\mathbf{y}$ must remain consistent within a specific graph (same number of nodes N), the number of parameters in the GNN model remains constant and independent of the graph size. Thus, we can combine graphs of varying sizes in both training and inference stages. The set of $|M| \times |\mathcal{A}|$ graphs or case studies is randomly divided into training $\mathcal{G}(\text{Train})$, validation $\mathcal{G}(\text{Val})$, and test $\mathcal{G}(\text{Test})$ sets. The GNN is trained using the training graphs and validated using the validation ones. Subsequently, the final performance and generalization capacity of the trained GNN are evaluated on the test graphs, which remained completely unobserved during the training process.

In addition to decisions regarding synthetic data generation and training process, it is crucial to define the specific architecture of the GNN. This architecture is contingent upon the type of GNN layer chosen, which should be flexible enough to capture the complexity of the problem at hand. Lastly, and not least, it is imperative to verify the underlying hypothesis of our proposal: that determining the confidence thresholds for a population of agents based on their initial and final opinion in an OD diffusion process is non-trivial, and that the structural information of the graph plays a crucial role. These aspects are further explored in the following Section IV.

IV. Experiments and Analysis

This section presents the experiments conducted to evaluate our methodological proposal. First, we describe the experimental setup and the generation of synthetic datasets in Section IV-A. Next, we detail the considered model architectures, training details, and hyperparameter tuning in Section IV-B. We then proceed to evaluate the best models on the synthetic test dataset $\mathcal{G}_{\text{Syn}}$ in Section IV-C. In Section IV-D, we perform a final evaluation on test datasets generated from common real-world topologies in the literature, whose union we denote as $\mathcal{G}_{\text{Real}}$. Finally, in Section IV-E, we develop an alternative EA method to determine the agents' thresholds using a more standard heuristic search approach, and compare its results with our GNN-based proposal.

A. Experimental Setup and Synthetic Datasets

All experiments related to NN training are conducted on a server with 2 × Intel Xeon CPU E5-2698, 512 GB DDR4, and 8 × Tesla V100 32Gb GPUs. On the other hand, since the experiments related to an EA run (see Section IV-E) do not take advantage of the GPU (they are CPU-oriented processes), we use a server with 2 × 2.2 GHz Intel Xeon Silver 4214 (12 cores), 256 GB DDR4. This architecture allows us to parallelize the simulation runs corresponding to the evaluation of each EA solution, thus reducing computation time. Additionally, to ensure result reproducibility, we have created a public repository containing all the code related to dataset generation and experimentation.[6] We use the Pytorch Geometric library [41] to implement the GNN models and NetworkX [42] to generate the synthetic topologies.

The first step to create the synthetic dataset $\mathcal{G}_{\text{Syn}}$ is to generate the set of base topologies $\mathcal{A}$. To achieve this, we consider three classic algorithms for generating artificial social networks, each with distinct characteristics. The first is the Ërdos-Rényi (ER) algorithm to create random networks [43], where each pair of nodes is connected with a uniform

[4]The upper limit of 0.25 is an indicative empirical value, justifying our choice of a higher upper bound. In fact, these results are made considering a fully connected social network, without considering the impact of the social network topology.

[5]The HK model has a stochastic component, wherein an agent is randomly chosen at each time-step. Thus, the final opinion $o_i(T)$ recorded in the dataset for each agent is actually its average final opinion after 10 Monte Carlo simulation runs.

[6]https://github.com/vvarper/GNN4BCPrediction

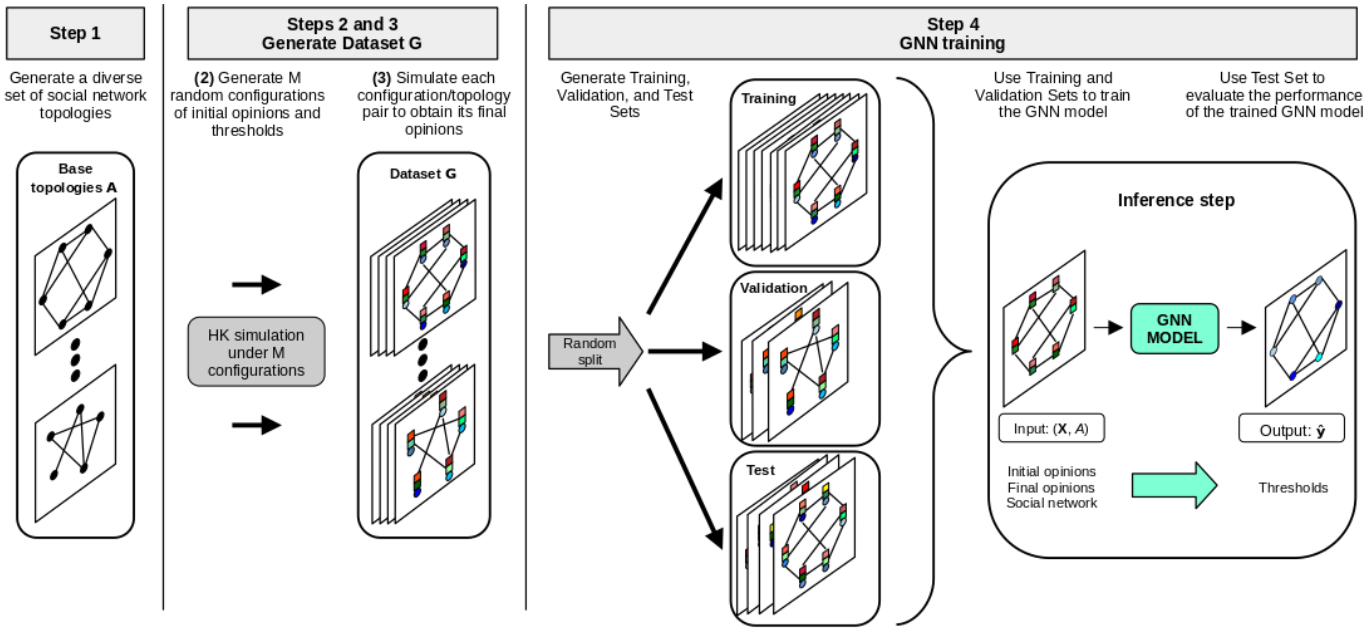

Fig. 2. Overview of the methodology for training a GNN to predict the confidence threshold of agents in a population governed by the HK OD model. The process involves generating a set of initial topologies $\mathcal{A}$ (Step 1), creating a set of M random configurations of initial opinions and thresholds for the agents (Step 2), running the HK model on each graph to obtain the final opinions (Step 3), and constructing the complete dataset $\mathcal{G}$, and splitting it into training ($\mathcal{G}$(Train)), validation ($\mathcal{G}$(Val)), and test ($\mathcal{G}$(Test)) subsets (Step 4). The GNN receives as input the initial and final opinions of each agent ($\mathbf{X}$) along with the social network connecting them (A), and returns as output the predicted confidence threshold of each agent ($\hat{\mathbf{y}}$). The colors in the nodes from Step 2 indicate the different configurations of initial and final opinions, as well as confidence threshold values.

probability p. As a result, these graphs exhibit a binomial degree distribution. The second algorithm is the Newman-Watts-Strogatz, which generates small-world networks (SW) [44]. This algorithm starts with a regular ring lattice where each node has k neighbors, and then creates shortcuts by randomly adding new connections with a probability p. Thus, the resulting graph has both small paths (similar to a random network) and a high clustering coefficient. Finally, the third algorithm considered is the Barabási-Albert (BA) preferential attachment [45]. This method begins with a fully connected graph and iteratively adds new nodes with m connections to previous ones, such that each node receives a new connection with a probability proportional to its degree. This results in a graph with a power-law degree distribution. In other words, most nodes have a low degree, while a few hubs receive most of the connections. These graphs are also known as scale-free because the distribution is independent of both the number of nodes and the average degree, and many real networks follow this distribution [45].

For each network generation algorithm, we apply three different configurations, resulting in a total of $|\mathcal{A}| = 9$ base topologies. In the case of ER, we consider three values of the link probability p. For SW, we set $p = 0.3$ and vary the value of the initial degree k. Lastly, for BA we use different values for the number of links m added per iteration. In all cases, we set $N = 1,000$, as we consider this to be a sufficient number of nodes to create a realistic scenario while not dramatically increasing the dataset size. Table II shows the 9 configurations and the properties of the resulting networks. This table also includes the results of applying the Louvain community detection algorithm [46] to each network, which will be relevant in the next step of the dataset generation.

The second step in our methodology is the creation of the M configurations of initial and final opinions, as well as confidence thresholds. To cover a wide variety of scenarios, we consider $M = 20$ configurations. In all of them, the initial opinion of each agent is randomly generated according to a uniform distribution $[0, 1]$. The difference between each of the M configurations lies in the values of the confidence thresholds. To address problem scenarios with incremental difficulty, we consider two cases:

- Graphs with homogeneous thresholds $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$. In this scenario, we generate M evenly spaced thresholds in the range [0.1, 0.5] and assign each of them to all nodes of the corresponding graph configuration.
- Graphs with heterogeneous thresholds by communities $\mathcal{G}_{\text{Syn}}^{\text{Com}}$. The idea behind this scenario is that an individual has similar characteristics to those close to them, differing from the rest. For every topology, we consider the community partition from Table II and assign a random confidence threshold to each community drawn from a uniform distribution $[0, 0.5]$. The generation of thresholds for each community is repeated M times per topology to obtain the desired number of configurations. The community partitions and their corresponding modularity values are presented in Table II.

The final opinions of each graph are obtained by averaging the results of 10 MC simulations of the HK model. Every simulation runs for $T = 10^6$ time-steps, ensuring that a stationary state is reached in all cases. At this point, both $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$

TABLE II
CHARACTERISTICS OF THE SYNTHETIC BASE TOPOLOGIES $\mathcal{A}$ ALONG WITH THE COMMUNITY DETECTION RESULTS.

Topology	Nodes	Edges	Density	Average Degree	Diameter	Average Clustering Coefficient	#Communities	Modularity
ER(0.1)	1,000	49,925	9.99e-02	9.99e+01	3	9.95e-02	10	8.04e-02
ER(0.2)	1,000	100,129	2.00e-01	2.00e+02	2	2.01e-01	8	5.41e-02
ER(0.3)	1,000	149,559	2.99e-01	2.99e+02	2	2.99e-01	7	3.98e-02
SW(0.3, 3)	1,000	1,311	2.62e-03	2.62e+00	20	0.00e+00	27	7.36e-01
SW(0.3, 5)	1,000	2,646	5.30e-03	5.29e+00	9	3.05e-01	26	7.01e-01
SW(0.3, 7)	1,000	3,882	7.77e-03	7.76e+00	7	3.71e-01	22	7.08e-01
BA(2)	1,000	1,996	4.00e-03	3.99e+00	7	3.04e-02	19	5.22e-01
BA(4)	1,000	3,984	7.98e-03	7.97e+00	5	3.53e-02	16	3.11e-01
BA(6)	1,000	5,964	1.19e-02	1.19e+01	4	4.42e-02	9	2.58e-01

and $\mathcal{G}_{\text{Syn}}^{\text{Com}}$ have a total of $|\mathcal{A}| \times M = 9 \times 20 = 180$ graphs each. [7] We randomly split each dataset, assigning 64% of the graphs for training (i.e., 115 graphs), 16% for validation (29 graphs), and 20% for testing (36 graphs).

B. Models' Architecture and Hyperparameter Tuning

The three GNN architectures previously discussed in Section II-B are considered: GCN, GraphSAGE, and GATv2. Additionally, with the aim of having a base benchmark and verifying that structural information is crucial in our problem, we also consider a multi-layer perceptron (MLP) that treats each graph as a dataset in which each node is an independent sample. MLP is a structure-agnostic architecture that aims to predict the confidence thresholds solely based on feature information.

We define each model as a concatenation of L layers selected from the four types defined earlier, such that $|\mathbf{h}_i^{(0)}| = 2$, $|\mathbf{h}_i^{(L)}| = 1$, and $|\mathbf{h}_i^{(j)}| = H$ for $j \in \{1, \dots, L-1\}$ and $i \in \{1, \dots, N\}$. All layers are followed by a ReLU activation function, except the last one, which uses a sigmoid activation function to ensure that the predicted threshold $\hat{\varepsilon}_i$ is in the range $[0, 0.5]$. Architectures such as Convolutional NNs usually benefit from lager number of layers, as they facilitate hierarchical feature extraction. However, in the case of GNNs, the number of layers is typically more limited due to the over-smoothing problem [47].

In order to find the best possible model, our experimentation involves hyperparameter tuning for each layer type and dataset. Specifically, we use a grid search, which is one of the most common techniques in the literature for this task [48]. It offers a comprehensive and straightforward approach to explore the parameter space, ensuring that no region of the predefined spaces is overlooked [49]. We adjust the number of layers L, the dimension of the hidden layers H, the learning rate, and the batch size. Table III shows the candidate values considered for each hyperparameter, resulting in a total of 36 possible settings, and thus, a total of 288 training runs. In all these training processes, we use the Adam optimizer with the Mean Squared Error (MSE) loss function with a maximum of $10,000$ epochs. Furthermore, to prevent over-fitting to training data, we apply early stopping if the validation loss does not improve for $1,000$ consecutive epochs.

[7]Note that among these graphs, there are repeated topologies but with different node features. In this work, we distinguish between topology (the structure of the graph) and the graph as a whole, which includes the node information.

TABLE III
HYPERPARAMETERS TO BE TUNED AND THEIR RESPECTIVE CANDIDATE VALUES.

Hyperparameter	Description	Candidate Values
L	Number of layers of the NN.	{4, 5}
H	Node embedding dimension in hidden layers.	{16, 32}
Learning Rate	Step size on each SGD update.	{1.00e-02, 1.00e-03, 1.00e-04}
Batch Size	Number of graphs used on each SGD update.	{2, 4, 8}

The results of the hyperparameter tuning are shown in Table IV, where we indicate the best configuration found according to validation loss for each layer type and dataset, along with the corresponding MSE on the training and validation sets. We can observe that the settings found vary in each case, although in the case with homogeneous thresholds, the architecture with $L = 5$ layers and a hidden layer size of $H = 32$ is consistently chosen as the best one. Regarding the quality of the models, we observe that in both problem scenarios, and in both training and validation datasets, the lowest MSE found corresponds to GraphSAGE, followed by GATv2. The worst results are obtained with MLP and GCN, with the ranking varying among them depending on the threshold scenario. Another issue to note is that, in the case with homogeneous thresholds, the MSE of the GraphSAGE and GATv2 models is an order of magnitude lower than that of MLP and GCN, a trend also observed in the community scenario but only with GraphSAGE. Furthermore, although the MSE in training is generally lower than in validation, the difference is not wide, with the magnitude order and the ranking of the models remaining consistent. Therefore, there do not seem to be over-fitting problems in either case.

Table IV also includes the number of parameters of each configuration (corresponding to the counts provided in Table I) and the training times. A direct comparison based on the layer type may be misleading, as both values depend on additional factors: while the number of parameters is affected by the number of layers and node embedding dimension, the training time is influenced by the batch size and the number of epochs before early-stopping. Notably, focusing on the most

TABLE IV
BEST HYPERPARAMETER CONFIGURATION FOUND PER LAYER TYPE AND PROBLEM ACCORDING TO THE MSE IN THE VALIDATION SET, INCLUDING THE NUMBER OF PARAMETERS, THE NUMBER OF EPOCHS UNTIL EARLY-STOPPING, AND THE TOTAL TRAINING TIME. THE LOWEST MSE VALUES ARE HIGHLIGHTED IN BOLD.

Dataset type	Layer	Learning Rate	L	H	Batch Size	# Parameters	Epochs	Training time (s)	MSE (Train)	MSE (Val)
Homogeneous Thresholds $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$	**MLP**	1.00e-03	5	32	8	3,297	4,557	7.28e+02	7.33e-03	6.49e-03
	GCN	1.00e-03	5	32	2	3,297	10,000	8.92e+03	2.78e-03	3.06e-03
	GraphSAGE	1.00e-03	5	32	4	6,465	6,361	2.28e+03	**6.32e-04**	**4.42e-04**
	GATv2	1.00e-04	5	32	2	6,852	10,000	1.29e+04	7.44e-04	8.15e-04
Thresholds by Community $\mathcal{G}_{\text{Syn}}^{\text{Com}}$	**MLP**	1.00e-02	4	16	4	609	1,911	5.17e+02	1.38e-02	1.36e-02
	GCN	1.00e-03	4	32	8	2,241	7,754	1.90e+03	1.53e-02	1.68e-02
	GraphSAGE	1.00e-03	5	32	4	6,465	2,953	1.04e+03	**5.22e-03**	**6.49e-03**
	GATv2	1.00e-02	4	32	8	4,676	3,233	1.47e+03	9.55e-03	1.05e-02

promising layers, the training time for GraphSAGE ranged from 17 to 38 minutes, whereas GATv2 varied significantly, taking between 24 minutes (in the community scenario) and 4 hours (in the homogeneous scenario). The latter extended training time is due to reaching the maximum number of epochs allowed.

C. Evaluation on Synthetic Topologies

We continue to delve into the performance of the best models found in the previous section, evaluating them on the synthetic test datasets. The results are shown in Table V, where, in addition to MSE, Mean Average Error (MAE), Mean Absolute Percentage Error (MAPE), and the coefficient of determination R^2 are included. Note that while MSE, MAE, and MAPE are error metrics to minimize, R^2 is a goodness-of-fit measure where $R^2 = 1$ indicates a perfect fit and $R^2 = 0$ indicates that the model is not better than predicting the mean of the target variable. The results remain consistent with those observed previously in Table IV, although in the case of homogeneous thresholds, GraphSAGE and GATv2 show an MSE of one order of magnitude higher than in training and validation datasets. Nevertheless, we should note that R^2 is above 0.79 in both cases. As could be inferred from the MSE, the GATv2 model exhibits notably lower performance in $\mathcal{G}_{\text{Syn}}^{\text{Com}}$ (R^2 = 4.59e-01), while GraphSAGE maintains remarkable performance ($R^2 = 7.08$e-01).

The underlying hypothesis of our methodology was that structural information plays a crucial role in this problem, and therefore MLP should serve as a baseline model to measure whether the performance of a GNN model is indeed positive. The results seen so far seem to confirm this, with the only case of discrepancy being GCN, the most basic GNN architecture. To discuss this issue further, we visualize in Figure 3 the predictions of each model against the labels of each node in $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$, the easiest variant of the problem that also presents fully balanced and evenly spaced thresholds values. The visualization for a perfect model should show a diagonal line. MLP is unable to adequately capture the relationship between node features and their thresholds, as it makes noisy predictions around the mean. In this scenario, the visualization for the three GNNs shows a gradual increasing trend in predictions according to the real thresholds, which is particularly noticeable in the GraphSAGE case. Therefore, we can conclude that indeed, the error values for MLP serve as a reference point that must be widely surpassed to claim that a GNN model is effective.

Finally, we conduct statistical ranking tests [50] by evaluating each model on the 36 test graphs from each scenario separately. First, the Friedman test is applied to determine if there are significant differences between the models' performance. The result of this test is $\chi_F^2 = 4.52$e+01 (p-value = 8.39e-10) for $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$ and $\chi_F^2 = 9.14$e+01 (p-value = 1.08e-19) for $\mathcal{G}_{\text{Syn}}^{\text{Com}}$. Thus, setting a confidence level of 99%, we can affirm that there are significant differences in both scenarios. Subsequently, it makes sense to examine whether there are significant differences between each pair of models with the Nemenyi post-hoc test, whose results are shown in Table VI. We can see that in the scenario with homogeneous thresholds, GraphSAGE and GATv2 are significantly better than MLP and GCN, while in the community scenario, GraphSAGE is also significantly superior to GATv2. However, it cannot be stated that there are significant differences between GCN and MLP in either case. The fact that GCN does not outperform MLP may seem surprising, but there are already previous studies indicating that GCN is a very basic model with limited expressive capacity. In fact, there are cases where it does not provide advantages over structure-agnostic models such as MLP [51]. The intuition behind the insufficient power of GCN can be derived from its mathematical definition. In Equation 4, we observe that GCN uses a singles weight matrix and performs aggregation without distinguishing between the center node and its neighbors. This is a key difference from GraphSAGE (Equation 5), which also applies to GATv2: in these layers, different weight matrices are used for the target node and its neighbors, applying different linear transformations to their features. Given the results, this distinction seems to be crucial when addressing this threshold prediction problem.

D. Evaluation on Real Topologies

We conclude the evaluation of the models with additional test datasets generated from real-world topologies commonly used in the literature [52]. The characteristics of these topologies are shown in Table VII, and it can be seen that they differ from the synthetic topologies (Table II) even in their larger size, ranging from networks of $2,000$ to $20,000$ nodes. For each of these topologies, we create a dataset with homogeneous thresholds and another one with thresholds by communities, following the same procedure as in Section

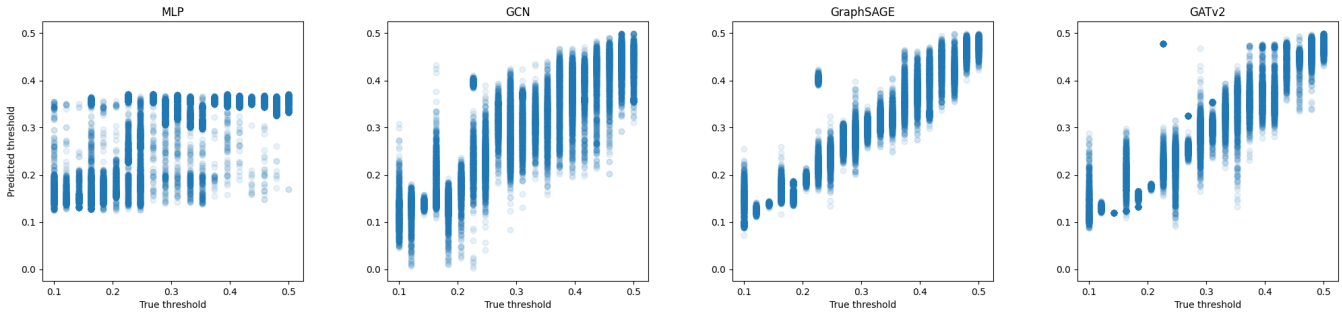

Fig. 3. Inference results of best models on the synthetic test dataset with homogeneous thresholds $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$. The x-axis represents the real threshold ε_i of each node i, while the y-axis represents the pr edicted threshold $\hat{\varepsilon}_i$.

TABLE V
EVALUATION OF THE BEST MODEL FOR EACH LAYER TYPE IN THE SYNTHETIC TEST DATASETS ACCORDING TO MSE, MAE, MAPE AND R^2 METRICS.

Layer	Homogeneous Thresholds $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$				Thresholds by Community $\mathcal{G}_{\text{Syn}}^{\text{Com}}$			
	MSE	**MAE**	**MAPE**	**R^2**	**MSE**	**MAE**	**MAPE**	**R^2**
MLP	6.09e-03	6.17e-02	2.33e-01	5.72e-01	1.60e-02	1.07e-01	1.55e+00	3.08e-01
GCN	3.71e-03	4.62e-02	1.76e-01	7.40e-01	1.97e-02	1.20e-01	2.14e+00	1.50e-01
GraphSAGE	**1.69e-03**	**2.53e-02**	**9.84e-02**	**8.81e-01**	**6.76e-03**	**6.33e-02**	**6.07e-01**	**7.08e-01**
GATv2	2.89e-03	3.30e-02	1.44e-01	7.97e-01	1.25e-02	9.08e-02	1.07e+00	4.59e-01

TABLE VI
NEMENYI POST-HOC TEST RESULTS (P-VALUES) ON THE 36 SYNTHETIC TEST GRAPHS. P-VALUES LOWER THAN 1.00E-02 ARE HIGHLIGHTED IN BOLD.

Dataset type	Layer	GCN	GraphSAGE	GATv2
Homogeneous Thresholds $\mathcal{G}_{\text{Syn}}^{\text{Hom}}$	**MLP**	7.73e-01	**1.00e-03**	**1.00e-03**
	GCN		**1.00e-03**	**5.60e-03**
	GraphSAGE			3.55e-01
Thresholds by Community $\mathcal{G}_{\text{Syn}}^{\text{Com}}$	**MLP**	8.24e-01	**1.00e-03**	**1.00e-03**
	GCN		**1.00e-03**	**1.00e-03**
	GraphSAGE			**4.08e-03**

IV-A. Again, we consider $M = 20$ configurations per case. The only difference is that this time the topologies are kept in separate datasets and that these are not split into training, validation, and test subsets, but are used directly to evaluate the models previously trained on synthetic topologies.

Table VIII shows the results on these real-topology test datasets, which follow a similar trend to those observed in Table V. GraphSAGE and GATv2 remain the models with the best performance, although this time GATv2 outperforms GraphSAGE in the scenarios with homogeneous thresholds. On the other hand, GCN shows even worse results than in the synthetic datasets, being surpassed by MLP in all cases.

Once again, we conduct statistical tests to determine if there are significant differences. This time, we combine the datasets of the five real-world topologies into a single test set per case ($\mathcal{G}_{\text{Real}}^{\text{Hom}}$ and $\mathcal{G}_{\text{Real}}^{\text{Com}}$), resulting in a total of 100 graphs per scenario. The Friedman test result for $\mathcal{G}_{\text{Real}}^{\text{Hom}}$ is $\chi_F^2 = 2.15\text{e+}02$ (p-value = 2.89e-46), while for $\mathcal{G}_{\text{Real}}^{\text{Com}}$ is $\chi_F^2 = 2.76\text{e+}02$ (p-value = 1.31e-59). The results of the Nemenyi post-hoc test are shown in Table IX, and they indicate that the differences between the models are even more significant than in the synthetic datasets. The main difference is that now GCN is significantly worse than MLP in the scenario with homogeneous thresholds, while the rest of the comparisons remain consistent with the previous datasets (see Table VI).

Therefore, we can confidently assert that GraphSAGE and GATv2 are the most suitable models for our task. The significant outperformance of GraphSAGE over GATv2 in one of the scenarios, despite its simpler architecture, may seem surprising. Our intuition is that GraphSAGE already has sufficient expressiveness to capture the relevant information for our problem, while the slightly higher number of parameters in GATv2 does not offer any substantial advantage for this specific task.

We must note that the best GATv2 model we achieved for the community scenario has fewer parameters than the best GraphSAGE model found. However, this specific GATv2 configuration, derived from hyperparameter tuning, outperformed other alternative configurations with a higher number of parameters (exceeding those of any GraphSAGE model considered). Therefore, this does not invalidate our current discussion. Even if extended training allowed some GATv2 configurations to improve their performance, they would still not surpass GraphSAGE as a better alternative, since the latter already provides strong performance with lower computational cost (see Table IV).

The performance of GraphSAGE is always notably superior to that of MLP, consistently achieving a R^2 value above 0.7. This outcome remains true even when acting on real-world topologies not seen during training time and with networks of different sizes. These findings confirm that structural information is crucial in this problem. Thus, by judiciously selecting the appropriate GNN model, our proposed methodology suc-

TABLE VII
CHARACTERISTICS OF THE REAL-WORLD TEST TOPOLOGIES ALONG WITH THE COMMUNITY DETECTION RESULTS.

Topology	Nodes	Edges	Density	Average Degree	Diameter	Average Clustering Coefficient	Communities	Modularity
CORA	18,800	62,685	3.55e-04	6.67e+00	23	2.67e-01	41	8.12e-01
CORA-ML	2,810	7,981	2.02e-03	5.68e+00	17	2.78e-01	22	7.62e-01
CiteSeer	1,681	2,902	2.06e-03	3.45e+00	23	1.53e-01	27	8.18e-01
PubMed	19,717	44,324	2.28e-04	4.50e+00	18	6.02e-02	36	7.70e-01
DBLP	16,191	51,913	3.96e-04	6.41e+00	34	1.45e-01	37	7.58e-01

TABLE VIII
EVALUATION OF THE BEST MODEL FOR EACH LAYER TYPE IN THE REAL-TOPOLOGY TEST DATASETS ACCORDING TO MSE, MAE, MAPE AND R^2 METRICS.

Dataset	Layer	Homogeneous thresholds				Thresholds by Community			
		MSE	**MAE**	**MAPE**	**R^2**	**MSE**	**MAE**	**MAPE**	**R^2**
CORA	**MLP**	7.89e-03	7.15e-02	2.65e-01	4.65e-01	1.15e-02	8.88e-02	1.57e+00	4.61e-01
	GCN	2.10e-02	1.16e-01	3.52e-01	-4.26e-01	1.39e-02	9.10e-02	7.84e-01	3.50e-01
	GraphSAGE	3.15e-03	4.31e-02	1.44e-01	7.87e-01	**4.21e-03**	**4.92e-02**	**5.03e-01**	**8.03e-01**
	GATv2	**2.06e-03**	**3.47e-02**	**1.25e-01**	**8.60e-01**	6.90e-03	6.53e-02	6.80e-01	6.77e-01
CORA-ML	**MLP**	8.69e-03	7.75e-02	3.04e-01	4.10e-01	1.29e-02	9.48e-02	1.30e+00	3.61e-01
	GCN	1.77e-02	1.05e-01	3.27e-01	-2.00e-01	1.22e-02	8.55e-02	6.27e-01	3.95e-01
	GraphSAGE	2.65e-03	3.87e-02	1.34e-01	8.20e-01	**4.29e-03**	**5.04e-02**	**4.55e-01**	**7.88e-01**
	GATv2	**1.88e-03**	**3.33e-02**	**1.25e-01**	**8.72e-01**	7.77e-03	7.02e-02	5.82e-01	6.16e-01
CiteSeer	**MLP**	7.11e-03	6.72e-02	2.40e-01	5.18e-01	1.35e-02	9.45e-02	7.77e-01	3.23e-01
	GCN	2.14e-02	1.18e-01	3.95e-01	-4.50e-01	1.62e-02	1.01e-01	5.28e-01	1.89e-01
	GraphSAGE	3.15e-03	4.38e-02	1.63e-01	7.86e-01	**4.80e-03**	**5.38e-02**	**3.61e-01**	**7.59e-01**
	GATv2	**2.08e-03**	**3.38e-02**	**1.39e-01**	**8.59e-01**	8.90e-03	7.41e-02	4.50e-01	5.54e-01
PubMed	**MLP**	7.92e-03	7.15e-02	2.61e-01	4.63e-01	1.20e-02	9.00e-02	1.20e+00	3.70e-01
	GCN	3.48e-02	1.54e-01	4.95e-01	-1.36e+00	1.61e-02	1.01e-01	6.98e-01	1.58e-01
	GraphSAGE	2.66e-03	3.89e-02	1.42e-01	8.19e-01	**4.71e-03**	**5.27e-02**	**5.09e-01**	**7.53e-01**
	GATv2	**2.33e-03**	**3.81e-02**	**1.42e-01**	**8.42e-01**	7.93e-03	7.03e-02	6.46e-01	5.85e-01
DBLP	**MLP**	7.93e-03	7.17e-02	2.64e-01	4.62e-01	1.11e-02	8.69e-02	1.57e+00	4.23e-01
	GCN	2.27e-02	1.20e-01	3.69e-01	-5.43e-01	1.21e-02	8.52e-02	7.77e-01	3.70e-01
	GraphSAGE	2.83e-03	4.04e-02	1.43e-01	8.08e-01	**4.55e-03**	**5.07e-02**	**6.63e-01**	**7.63e-01**
	GATv2	**2.34e-03**	**3.67e-02**	**1.35e-01**	**8.41e-01**	7.01e-03	6.53e-02	7.98e-01	6.35e-01

TABLE IX
NEMENYI POST-HOC TEST RESULTS (P-VALUES) ON THE 100 REAL-TOPOLOGY GRAPHS. P-VALUES LOWER THAN 1.00E-02 ARE HIGHLIGHTED IN BOLD.

Dataset type	Layer	GCN	GraphSAGE	GATv2
Homogeneous Threshold $\mathcal{G}^{\text{Hom}}_{\text{Real}}$	**MLP**	**1.00e-03**	**1.00e-03**	**1.00e-03**
	GCN		**1.00e-03**	**1.00e-03**
	GraphSAGE			2.21e-01
Threshold by Community $\mathcal{G}^{\text{Com}}_{\text{Real}}$	**MLP**	5.70e-02	**1.00e-03**	**1.00e-03**
	GCN		**1.00e-03**	**1.00e-03**
	GraphSAGE			**1.00e-03**

cessfully infers the confidence thresholds of agents under an HK model from the network topology and the initial and final opinions of each agent.

E. Comparison with an Evolutionary Algorithm Approach for Confidence Threshold Estimation

As discussed in Section III, determining individual-level confidence thresholds in an OD model to match specific simulation outputs can be framed as a parameter calibration problem in a high-dimensional space, for which heuristic techniques like EAs are typically used [38], [39]. However, this work pursues a different goal: to develop an ML method that, after training on a synthetically generated dataset with diverse case studies, can automatically infer the confidence thresholds for any new case study, hence being able to generalize to unseen data. This inference step takes negligible time and avoids the need for additional ABM simulations, which are often the bottleneck in calibration processes. The computationally intensive part of our approach lies in generating the synthetic datasets (9 topologies and 180 ABM simulations per scenario) and training the GNN model. As we showed in Table IV, the training part, in particular for the best GraphSAGE configurations, takes less than one hour.

In this section, we develop better why traditional heuristic calibration algorithms are not well-suited for our specific generalization-oriented goal and why the GNN-based proposal is more appropriate. To this end, we designed a genetic algorithm (GA) for comparison in which each solution is represented as a vector of real numbers encoding agents' confidence threshold. Since we aim for generalization across case studies, the objective function minimizes the average MSE between simulation outputs and final opinions across all training cases. This allows us to apply the best threshold configuration found by the GA directly to new cases without additional simulations.

The GA operates as follows. We use a population of 100

solutions, with each solution encoded as a real-valued vector where each gene represents an agent's confidence threshold, $\hat{\epsilon}_i \in [0, 1]$. In each iteration, 100 offspring solutions are generated through a binary tournament selection process and a BLX-α crossover (with $\alpha = 0.3$ and a crossover probability of 0.6). Random uniform perturbations within $[\hat{\epsilon}_i - 0.3, \hat{\epsilon}_i + 0.3]$ are applied to each gene with a probability of 0.1. The replacement mechanism selects the 100 best solutions from the combined parent and offspring populations. The GA stops after evaluating 5,000 solutions (50 generations). Each evaluation in the GA requires running $|\mathcal{G}(\text{Train})| = 115$ ABM simulations, which is equivalent to the number of simulations involved in the training set ($\mathcal{G}(\text{Train})$) generation.[8]

The GA execution took 1.59e+06 and 1.61e+06 seconds (around 18.5 days) in the homogeneous and community-based scenarios, respectively. Table X provides the error in confidence threshold estimation for each synthetic split, which can be compared to the results of our GNN-based approach (see Tables IV and V). We can see that this GA method performs much worse than the GNN-based approaches, even when compared to models with poorer performance, such as MLP and GCN layers. For example, the test MSE of the GA is one order of magnitude higher than any NN model in the homogeneous scenario, and between one and two orders of magnitude higher in the community-based scenario.

TABLE X
EVALUATION OF THE THRESHOLD CONFIGURATIONS OBTAINED BY THE GA FOR EACH SPLIT OF EACH SYNTHETIC DATASET, ASSESSED USING MSE, MAE, MAPE, AND R^2 METRICS.

Dataset type	Split	MSE	MAE	MAPE	R2
Homogeneous Thresholds $\mathcal{G}^{\text{Hom}}_{\text{Syn}}$	**Train**	9.22e-02	2.49e-01	1.17e+00	-5.48e+00
	Validation	1.04e-01	2.67e-01	1.43e+00	-5.13e+00
	Test	9.62e-02	2.56e-01	1.23e+00	-5.76e+00
Thresholds by Community $\mathcal{G}^{\text{Com}}_{\text{Syn}}$	**Train**	1.46e-01	3.16e-01	9.39e+00	-6.12e+00
	Validation	1.42e-01	3.11e-01	3.34e+01	-6.12e+00
	Test	1.45e-01	3.14e-01	6.91e+00	-5.27e+00

This experiment highlights three issues of the heuristic search method, which further emphasizes the advantages of our proposal. First, when provided with the same training data as our GNN-based method, the GA requires significantly more computation time, increasing from hours to weeks. Second, the heuristic search struggles to adequately address our generalization goal across different case studies. It must search for a fixed threshold configuration that performs well on average, leading to significantly poorer results than our method, which infer case-specific thresholds. Third, since the GA solution size depends on the graph size (i.e., the number of agents in the case study), its applicability is restricted to case studies on the same size as the training data. This, for instance, prevents us from testing it on the real topologies test set discussed in Section IV-D.

[8]For time constraints, we use only one MC run for each GA evaluation. Even with this computational reduction and parallelizing all simulations for each evaluation, the evaluation time of each chromosome is around 320 seconds.

V. CONCLUSIONS AND FUTURE WORKS

In this research, we have addressed the challenge of determining agent-level confidence thresholds in a social network of individuals following the HK model of OD. We approached this task by reframing it as a regression problem, wherein the input data consists of the social network topology and the initial and final opinions of each agent. Given this formulation, GNNs emerge as the most suitable ML model family for the problem solving of this learning challenge. The theoretical foundation of the OD model enabled the generation of large synthetic datasets, facilitating the proper training of the models.

After conducting a comprehensive experimentation involving two threshold scenarios of increasing difficulty and evaluating various GNN architectures along with an MLP model as a baseline and a GA-based method as benchmark, we have identified GATv2 and GraphSAGE layers as the most successful to address the problem at hand, achieving remarkably good results in both cases. Notably, GraphSAGE stands out for obtaining the best results in scenarios involving heterogeneous thresholds by communities, which represents the most challenging variant of the problem. It is noteworthy that the superior results achieved in synthetic datasets extend to test datasets comprising real-world topologies, with GraphSAGE consistently achieving a coefficient of determination R^2 exceeding 0.7. This consistency across different topologies suggests that the GraphSAGE model effectively generalizes its learning to previously unseen social networks, underscoring its potential for direct application to new real-world case studies. Consequently, these findings affirm the success of the proposed methodology.

While our work has yielded positive results, several limitations warrant consideration for future research. Firstly, our exploration of GNN architectures has focused on three of the most prevalent layers in the literature. Examining alternative options could potentially enhance the results obtained. Additionally, it would be valuable to deploy our trained models in a fully real-world case study involving a population with known initial and final opinions. Such an application could provide further insights into the practical utility and effectiveness of the proposed methodology. Moreover, to increase the realism of the OD model, future works could explore an extension of this methodology to handle dynamic social networks that evolve over time [53]. Finally, we propose investigating the application of this approach to predict agent-level attributes in other ABMs. Like the HK model, these models should depend solely on individual agent properties and an underlying social network for their operation.

REFERENCES

[1] J. M. Epstein, *Generative social science: Studies in agent-based computational modeling*. Princeton University Press, 2006.
[2] J. D. Farmer and D. Foley, "The economy needs agent-based modelling," *Nature*, vol. 460, pp. 685–686, 2009.
[3] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation," in *Proceedings of the 37th Conference on Winter Simulation*, 2000, pp. 86–98.
[4] M. A. Janssen and E. Ostrom, "Empirically based, agent-based models," *Ecology and society*, vol. 11, no. 2, 2006.

[5] A. Trivedi and S. Rao, "Agent-based modeling of emergency evacuations considering human panic behavior," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 1, pp. 277–288, 2018.

[6] P. Lv, Z. Zhang, C. Li, Y. Guo, B. Zhou, and M. Xu, "Crowd behavior evolution with emotional contagion in political rallies," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 2, pp. 377–386, 2019.

[7] P. Ye, S. Wang, and F.-Y. Wang, "A general cognitive architecture for agent-based modeling in artificial societies," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 1, pp. 176–185, 2018.

[8] M. H. Degroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[9] R. A. Holley and T. M. Liggett, "Ergodic theorems for weakly interacting infinite systems and the voter model," *Annals of Probability*, vol. 3, no. 4, pp. 643–663, 1975.

[10] S. Galam, "Minority opinion spreading in random geometry," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 25, no. 4, pp. 403–406, 2002.

[11] H. Xia, H. Wang, and Z. Xuan, "Opinion dynamics: A multidisciplinary review and perspective on future research," *International Journal of Knowledge and Systems Science*, vol. 2, no. 4, p. 72–91, 2011.

[12] J.-H. Cho, "Dynamics of uncertain and conflicting opinions in social networks," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 2, pp. 518–531, 2018.

[13] Y. Dong, M. Zhan, G. Kou, Z. Ding, and H. Liang, "A survey on the fusion process in opinion dynamics," *Information Fusion*, vol. 43, pp. 57–65, 2018.

[14] N. E. Friedkin and E. C. Johnsen, "Social influence and opinions," *The Journal of Mathematical Sociology*, vol. 15, no. 3-4, p. 193 – 206, 1990.

[15] J. Giráldez-Cru, A. Suárez-Vázquez, C. Zarco, and O. Cordón, "Modeling the opinion dynamics of superstars in the film industry," *Expert Systems with Applications*, vol. 250, p. 123750, 2024.

[16] D.-S. Lee, C.-S. Chang, and Y. Liu, "Consensus and polarization of binary opinions in structurally balanced networks," *IEEE Transactions on Computational Social Systems*, vol. 3, no. 4, pp. 141–150, 2016.

[17] G. Deffuant, D. Neau, F. Amblard, and G. Weisbuch, "Mixing beliefs among interacting agents," *Advances in Complex Systems*, vol. 3, no. 01n04, pp. 87–98, 2000.

[18] R. Hegselmann and U. Krause, "Opinion dynamics and bounded confidence models, analysis, and simulation," *Journal of Artificial Societies and Social Simulation*, vol. 5, no. 3, 2002.

[19] Q. Zha, H. Liang, G. Kou, Y. Dong, and S. Yu, "A feedback mechanism with bounded confidence- based optimization approach for consensus reaching in multiple attribute large-scale group decision-making," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 994–1006, 2019.

[20] R. Axelrod, "The dissemination of culture: A model with local convergence and global polarization," *Journal of Conflict Resolution*, vol. 41, no. 2, pp. 203–226, 1997.

[21] C. Castellano, S. Fortunato, and V. Loreto, "Statistical physics of social dynamics," *Rev. Mod. Phys.*, vol. 81, pp. 591–646, 2009.

[22] J. Lorenz, "Heterogeneous bounds of confidence: Meet, discuss and find consensus!" *Complexity*, vol. 15, no. 4, pp. 43–52, 2010.

[23] F. Stonedahl, D. Anderson, and W. Rand, "When does simulated data match real data?" in *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, 2011, pp. 231–232.

[24] A.-T. Nguyen, S. Reiter, and P. Rigo, "A review on simulation-based optimization methods applied to building performance analysis," *Applied energy*, vol. 113, pp. 1043–1058, 2014.

[25] W. Zhang, A. Valencia, and N. Chang, "Synergistic integration between machine learning and agent-based modeling: A multidisciplinary review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 5, pp. 2170–2190, 2023.

[26] M. Ale Ebrahim Dehkordi, J. Lechner, A. Ghorbani, I. Nikolic, E. Chappin, and P. Herder, "Using machine learning for agent specifications in agent-based models and simulations: A critical review and guidelines," *Journal of Artificial Societies and Social Simulation*, vol. 26, no. 1, 2023.

[27] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[28] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[29] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A gentle introduction to graph neural networks," *Distill*, 2021.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2016.

[31] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[32] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[33] R. L. Berger, "A necessary and sufficient condition for reaching a consensus using degroot's method," *Journal of the American Statistical Association*, vol. 76, no. 374, p. 415 – 418, 1981.

[34] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[35] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[36] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in *International Conference on Learning Representations*, 2021.

[37] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[38] N. Malleson, L. See, A. Evans, and A. Heppenstall, "Optimising an agent-based model to explore the behaviour of simulated burglars," *Theories and simulations of complex social systems*, pp. 179–204, 2014.

[39] J. Zhong and W. Cai, "Differential evolution with sensitivity analysis and the powell's method for crowd model calibration," *Journal of computational science*, vol. 9, pp. 26–32, 2015.

[40] J. F. Robles, E. Bermejo, M. Chica, and Ó. Cordón, "Multimodal evolutionary algorithms for easing the complexity of agent-based model calibration," *Journal of Artificial Societies and Social Simulation*, vol. 24, no. 3, 2021.

[41] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[42] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[43] P. Erdös and A. Rényi, "On random graphs i," *Publ. math. debrecen*, vol. 6, no. 290-297, p. 18, 1959.

[44] M. E. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, no. 4-6, pp. 341–346, 1999.

[45] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[46] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[47] C. Cai and Y. Wang, "A note on over-smoothing for graph neural networks," *arXiv preprint arXiv:2006.13318*, 2020.

[48] S. Simon, N. Kolyada, C. Akiki, M. Potthast, B. Stein, and N. Siegmund, "Exploring hyperparameter usage and tuning in machine learning research," in *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE, 2023, pp. 68–79.

[49] M. Ogunsanya, J. Isichei, and S. Desai, "Grid search hyperparameter tuning in additive manufacturing processes," *Manufacturing Letters*, vol. 35, pp. 1031–1042, 2023.

[50] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[51] D. Gomes, F. Ruelens, K. Efthymiadis, A. Nowe, and P. Vrancx, "When are graph neural networks better than structure-agnostic methods?" in *NeurIPS 2022*. OpenReview.net, 2022, pp. 1–10.

[52] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, 2018, pp. 1–13.

[53] Z. Wu, Q. Zhou, Y. Dong, J. Xu, A. H. Altalhi, and F. Herrera, "Mixed opinion dynamics based on degroot model and hegselmann–krause model in social networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 1, pp. 296–308, 2022.

Víctor A. Vargas-Pérez received the B.Sc. degree in Computer Science and the M.Sc. degree in Data Science from the University of Granada (Spain) in 2020 and 2021, respectively. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada (Spain). His research interests include agent-based modeling and machine learning.

Jesús Giráldez-Cru obtained his PhD in Computer Science in 2016 from the Autonomous University of Barcelona (Spain). He did his PhD in the Artificial Intelligence Research Institute of the Spanish National Research Council (IIIA-CSIC) and was a postdoctoral researcher in the Royal Institute of Technology (KTH, Stockholm, Sweden) and in the University of Granada (Spain). Currently, he is a "Ramón y Cajal" senior postdoctoral researcher at the University of Seville (Spain). He has co-authored more than 30 research works published in top international journal and conferences, such as *Artificial Intelligence*, *International Journal of Intelligent Systems*, *Knowledge-based Systems*, *Journal of Artificial Intelligence Research*, and *International Joint Conference on Artificial Intelligence (IJCAI)*, among others, with a H-index of 14 according to Google Scholar, and he is an active member of the Program Committee of top AI conferences including IJCAI and AAAI. His research interests focus on constraint satisfaction and optimization problems, complex networks, and opinion dynamics in agent-based modeling.

Pablo Mesejo is Associate Professor at the Department of Computer Science and Artificial Intelligence of the University of Granada (Spain), and Co-Founding Partner and Chief AI Officer of Panacea Cooperative Research (Spain). The main topic of his research is the analysis and design of machine learning, computer vision, and computational intelligence methods able to solve image analysis problems. He has published his research work at top venues like IEEE TPAMI, CVPR, ECCV, MICCAI, ICML, IJCAI, ECAI, IEEE TMI, and many more. Dr. Mesejo is the Vice-Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing (Chair from 2018 to 2021), Associate Member of the American Academy of Forensic Sciences (AAFS, Digital and Multimedia Sciences Section), and member of the Andalusian Research Institute on Data Science and Computational Intelligence, among others.

Oscar Cordón (Fellow, IEEE) is Professor at University of Granada, Spain. Founding Director of Virtual Learning Center (2001–2005) and VP for Digital University (2015–2019). Founding researcher of European Centre for Soft Computing (2006–2015). IEEE CIS Outstanding Early Career Award (2011), National Award on Computer Science (2014), IEEE Fellow (2018), IFSA Fellow (2019), and Recognition of the Spanish AI Association (2020), among other awards. Member of High-Level Expert Group for Spanish R&D Strategy on AI (2019). 131 JCR-indexed journal papers, 23 Ph.D. dissertations advised, 42 research projects/contracts (∼10M€) coordinated, and 4 international patents. Included in 1% of most-cited researchers in the world (Clarivate WOS) and Top 2% of the most cited researchers in the world in Artificial Intelligence (University of Stanford-Elsevier).