

Don't Stop the Curious Neighbor: From Local Beliefs to Network-Wide Inference

Malek Khammassi

`malek.khammassi@epfl.ch`

École Polytechnique Fédérale de Lausanne

Vincenzo Matta

University of Salerno

Ali H. Sayed

École Polytechnique Fédérale de Lausanne

Article

Keywords:

Posted Date: October 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11195862/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Don't Stop the Curious Neighbor: From Local Beliefs to Network-Wide Inference

Malek Khammassi^{1*}, Vincenzo Matta² and Ali H. Sayed¹

¹Institute of Electrical and Micro Engineering, École Polytechnique Fédérale de Lausanne, Lausanne, 1015, Switzerland.

²Department of Information and Electrical Engineering and Applied Mathematics, University of Salerno, Salerno, I-84084, Italy.

*Corresponding author(s). E-mail(s): malek.khammassi@epfl.ch;

Contributing authors: vmatta@unisa.it; ali.sayed@epfl.ch;

Abstract

Many sensing networks operate in heterogeneous environments, where different agents are called to act and make decisions under varied individual conditions. In these settings, it may not be enough for an agent to learn only its local state: reliable decisions can also depend on knowing the states present elsewhere in the network, especially when centralized processing is costly, unreliable, or unavailable. To address this problem, we propose cluster-aware social learning, a decentralized method that enables each agent to recover all hidden states across the network using only local measurements and neighbor-to-neighbor exchanges. The method allows agents to identify connected groups that share the same hidden state and to recover a single state estimate per group. This group-based representation reduces the amount of information exchanged over the network while preserving each agent's ability to recover the network-wide state configuration. Experiments on synthetic and real sensor data demonstrate accurate decentralized learning, highlighting the potential of cluster-aware social learning for scalable sensing and monitoring applications.

1 Introduction

Networked sensing systems are widely used to monitor complex environments for disparate applications, including industrial emissions containment, air-quality monitoring, infrastructure, health, disaster response, and public safety. In these systems, sensing agents collect local measurements that provide only partial and noisy information about the environment. Therefore, reliable decisions often require combining information distributed across the network. A central processor could aggregate all measurements and infer the underlying state of the monitored system. In practice, however, centralized processing may be costly, fragile, or unavailable, especially when communication bandwidth is limited, infrastructure is damaged, or decisions must be made locally. These limitations have motivated a broad literature on decentralized estimation, detection, and inference, where agents use local measurements and peer-to-peer communication to recover hidden quantities without relying on a central processor.

The objective of decentralized inference depends on how the hidden quantities are distributed across the network. In some settings, all agents observe different aspects of the same underlying phenomenon, and the goal is to infer the single common state. In other settings, different agents are associated with different local states, and the goal is usually for each agent to infer its own local state. Nevertheless, in many sensing and monitoring applications, neither objective is enough. The common-state objective is too restrictive when the monitored environment is heterogeneous. For example, sensors deployed over a spatial region may be exposed to different pollutant levels, gas sources, materials, structural conditions, or environmental regimes, as commonly observed in air-quality monitoring and spatial-field reconstruction applications [1–4]. In these cases, different agents may correspond to different hidden states, so a single common hypothesis cannot represent the inference problem.

On the other hand, a purely local objective may also be insufficient. An agent may need to know not only what is happening at its own location, but also what is happening elsewhere in the network. For example, consider an urban area monitored after an accident, fire, gas leak, or structural failure. A sensor located in one part of the region may detect a high hazard level. However, this local information alone is not sufficient to decide which

zones should be evacuated, which routes remain safe, where emergency teams should be deployed, or whether the hazard is localized or spreading [5, 6]. These decisions depend on how the hidden states are distributed across the entire monitored area. When the central communication infrastructure is damaged or overloaded, this network-wide information must be inferred through neighbor-to-neighbor communication. This requires a broader inference objective: Each agent should recover not only its own hidden state but also those of the other agents in the network. We refer to this collection of local hidden states as the *network-wide state configuration*.

Most existing decentralized inference methods do not address this network-wide objective. A large body of decentralized inference literature focuses on the homogeneous-state setting, where agents seek to infer a single common quantity from measurements distributed across the network. In distributed estimation and filtering, this quantity is typically a continuous-valued parameter or state [7–10], whereas in distributed detection and hypothesis testing, it is a discrete-valued hypothesis [11–14]. The social learning literature addresses a closely related formulation from the perspective of belief formation: agents maintain beliefs over a set of plausible hypotheses and update them using local observations together with beliefs received from neighboring agents [15–21]. Under appropriate connectivity and identifiability conditions, these beliefs concentrate on the common hypothesis that best explains the observations collected across the network. Therefore, these models are well-suited to settings where all agents are associated with the same underlying state.

Several related lines of work have considered decentralized inference in heterogeneous settings. In distributed estimation, node-specific and multitask formulations allow different agents to estimate different continuous-valued quantities. Some parameters may be local to one node, some may be shared by subsets of nodes, and others may be common to the entire network [22, 23]. Node-specific signal estimation methods allow each node to recover its own desired signal from observations collected across the network [24, 25]. Heterogeneous-state settings have also been studied in distributed hypothesis testing and social learning. Self-aware social learning considers a stationary setting in which agents may have different fixed hidden states and adapt their communication patterns to rely primarily on neighbors that appear to share the same state [26]. Social learning over community-structured graphs studies a related form of heterogeneity in which different communities may be associated with distinct true hypotheses [27]. These works relax the classical common-state assumption. However, the objective remains local: each agent seeks to recover its own parameter, signal, task, or hidden state, while cooperation is used mainly to improve this local inference task.

The network-wide objective considered here is different. Each agent aims to infer the hidden states of all agents in the network, not just its own. A preliminary formulation of this objective was introduced in our earlier work [28]. In that approach, each agent maintains and exchanges a multivariate belief over all possible network-wide state configurations. This allows agents to learn the hidden states of all agents, but it also introduces a major scalability limitation: the belief dimension grows exponentially with the network size. As the number of agents increases, the amount of information that must be stored, updated, and exchanged becomes substantial. However, in many heterogeneous sensing environments, the network-wide state configuration has a repeated structure where many agents may share the same hidden state. For example, although a network may contain hundreds of sensors, several of them may be exposed to the same material, gas source, hazard level, or environmental regime. The number of distinct hidden states present in the network can therefore be much smaller than the number of agents. This observation motivates a group-based representation where, instead of treating every agent separately or maintaining a belief over all possible network-wide assignments, agents can identify groups of agents that share the same hidden state and maintain one belief per group.

We refer to these same-state groups as *clusters*, and to the process of discovering them as *clustering*. This notion differs from classical clustering and community detection in networks, where nodes are often grouped according to dense connectivity patterns or stochastic cluster models [29–33]. In contrast, in our problem the clusters represent which agents share the same hidden state. Therefore, the clusters are statistical groupings defined by the hidden states of the agents, and need not coincide with densely connected regions of the graph. In this work, clustering is used as a preliminary step that maps the network-wide inference problem to a lower-dimensional representation.

Building on this idea, we propose *cluster-aware social learning* for decentralized network-wide state inference. The method allows each agent to use local measurements and neighbor-to-neighbor exchanges to identify clusters of agents that share the same hidden state. Each agent then maintains one belief, and hence one state estimate, per discovered cluster. Because agents keep track of the members of each cluster, the estimated state of a cluster can be assigned back to all agents in that cluster. In this way, every agent can recover the network-wide state configuration while exchanging only cluster-level information. Compared with the multivariate belief formulation in [28], the proposed approach avoids maintaining beliefs over the full space of network-wide state assignments. Instead, it exploits the shared hidden-state structure across the network to substantially reduce the dimension of the beliefs exchanged among the agents. We validate the method on controlled synthetic experiments and on a gas-classification task using real sensor measurements. The results show that cluster-aware social learning enables agents to recover the network-wide state configuration through local communication while reducing the amount of information exchanged across the network.

2 Results

We evaluate cluster-aware social learning in a heterogeneous-state setting where different agents may be associated with different hidden states. Each agent observes only a local measurement stream, which is informative about its own state, and exchanges information only with neighboring agents connected to it, as illustrated in Fig. 1. The objective is network-wide: every agent aims to recover the hidden states of all agents in the network, rather than only its own state.

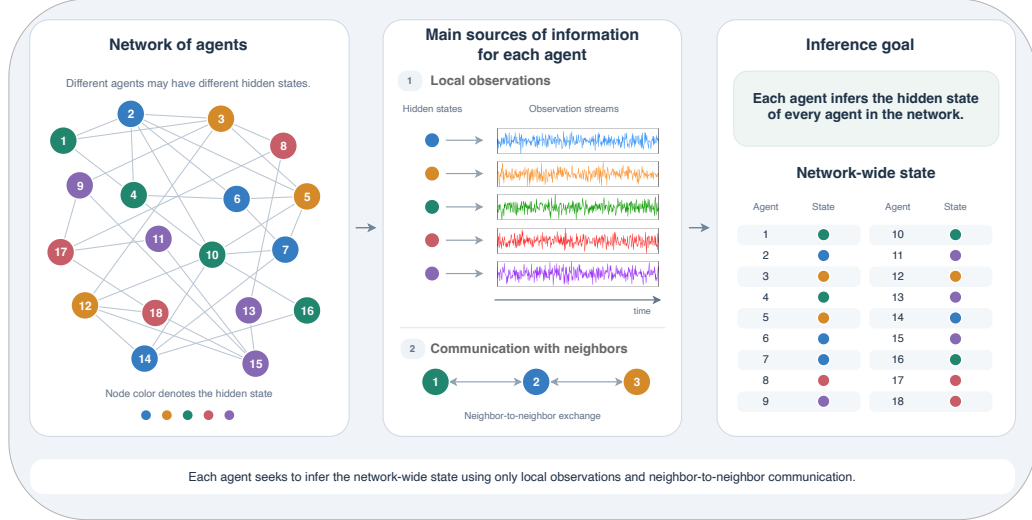


Fig. 1: Decentralized network-wide state inference problem. A network of agents is associated with unknown local hidden states. Each agent observes only its own local measurement stream and exchanges information with neighboring agents. The goal is for every agent to reconstruct the hidden states of all agents in the network using only local observations and neighbor-to-neighbor communication.

Cluster-aware social learning addresses this task through two stages, summarized in Fig. 2. The first stage is a *cluster-discovery* phase, in which each agent identifies the connected groups of agents that share the same hidden state (i.e., the clusters). The cluster-discovery procedure is fully decentralized. Thus, each agent maintains its own view of the clusters in the network.

We now introduce the main quantities used in this section, with bold symbols denoting random quantities. Consider a network of K agents indexed by $\mathcal{K} \triangleq \{1, \dots, K\}$. At time i , agent k represents its current view of the network clustering through a partition $\mathcal{P}_{k,i}$ of $\mathcal{K}$. Specifically, $\mathcal{P}_{k,i}$ is a collection of nonempty, pairwise disjoint subsets of $\mathcal{K}$ whose union is $\mathcal{K}$. We refer to these subsets as *clusters*. Accordingly, two agents that belong to the same cluster of $\mathcal{P}_{k,i}$ are regarded by agent k as sharing the same hidden state at time i . These local partitions are refined over time using local evidence and neighbor-to-neighbor exchanges, as detailed in Section 4.

The cluster-discovery phase terminates after T iterations and each agent freezes its resulting partition. We denote the partition of agent k at the end of the cluster-discovery phase by

$$\mathcal{P}_k \triangleq \mathcal{P}_{k,T}, \quad (1)$$

where T denotes the *clustering horizon*.

To conveniently index the clusters it identified, agent k labels each cluster in $\mathcal{P}_k$ by the smallest-indexed agent it contains, as illustrated in Fig. 3. Specifically, for each cluster $C \in \mathcal{P}_k$, define its label as

$$m(C) \triangleq \min_{\ell \in C} \ell. \quad (2)$$

Accordingly, the set of *cluster labels* used by agent k is defined as

$$\mathcal{S}^{(k)} \triangleq \{m(C) : C \in \mathcal{P}_k\}. \quad (3)$$

Thus, each label $r \in \mathcal{S}^{(k)}$ uniquely corresponds one cluster in the partition $\mathcal{P}_k$.

The second stage is the *network-wide state inference* phase. During this phase, each agent uses the clusters identified during cluster discovery to infer the hidden states across the network. Specifically, at each time $i > T$, agent k maintains a collection of *private cluster beliefs*,

$$\boldsymbol{\mu}_i^{(k)} \triangleq \{\boldsymbol{\mu}_{r,i}^{(k)} : r \in \mathcal{S}^{(k)}\},$$

Cluster-aware social learning

Two-phase approach for decentralized network-wide state inference

Illustrated from the perspective of agent 1. Every agent performs the same operations using its own observations and information received from its neighbors.

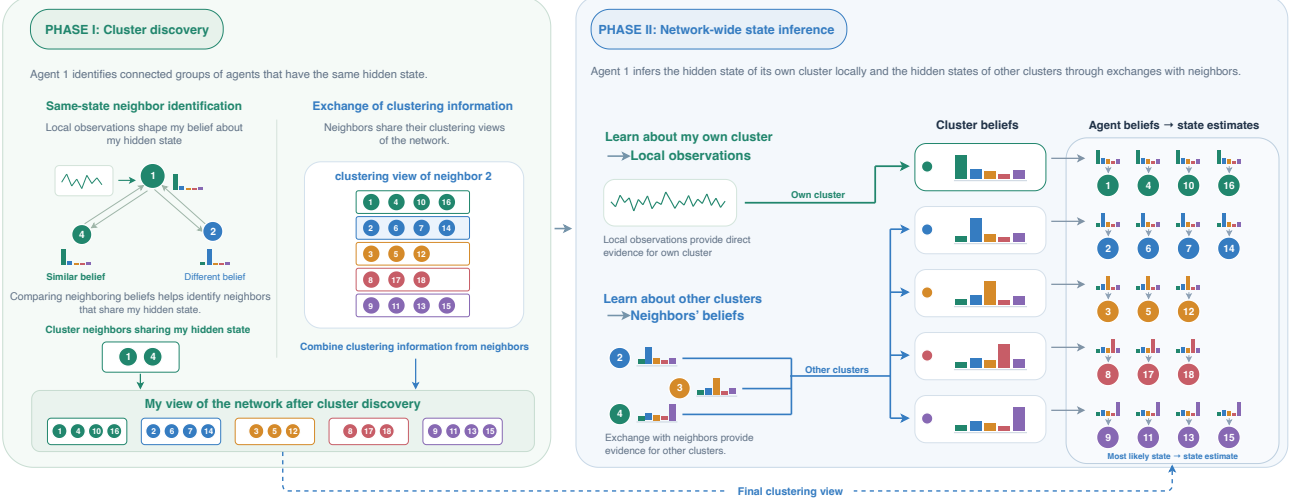


Fig. 2: Cluster-aware social learning. The proposed method follows a two-phase strategy. In Phase I, agents use local belief updates and neighbor exchanges to discover the connected groups of agents that share the same hidden state (i.e., the clusters). In Phase II, agents update one belief vector per discovered cluster and map the resulting cluster-level beliefs back to individual agents, yielding a network-wide state estimate.

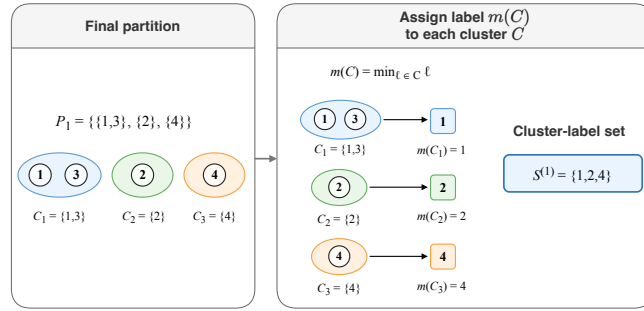


Fig. 3: Set of cluster labels. Example with $K = 4$ and final partition $\mathcal{P}_1 = \{\{1, 3\}, \{2\}, \{4\}\}$ for agent 1. Each cluster is labeled by its smallest-indexed agent, resulting in the cluster-label set $\mathcal{S}^{(1)} = \{1, 2, 4\}$.

with one belief vector $\mu_{r,i}^{(k)}$ associated with each cluster labeled by $r \in \mathcal{S}^{(k)}$. Let $\Theta \triangleq \{\theta_1, \dots, \theta_H\}$ denote the finite set of possible hidden states. Then, $\mu_{r,i}^{(k)}$ is an $H \times 1$ probability vector over Θ , whose entry $\mu_{r,i}^{(k)}(\theta)$ quantifies the confidence of agent k that the cluster represented by r has hidden state θ , at time i . These private cluster beliefs are updated using local observations and neighbor-to-neighbor belief exchanges, as detailed in Section 4.

To obtain a network-wide state estimate, each agent maps its private cluster beliefs back to the agents in the network. Specifically, at time i , agent k constructs the collection of *agent-level beliefs* denoted by

$$\hat{\mu}_i^{(k)} = \left\{ \hat{\mu}_{\ell,i}^{(k)} : \ell \in \mathcal{K} \right\},$$

where $\hat{\mu}_{\ell,i}^{(k)}$ is an $H \times 1$ probability vector over the hypothesis set Θ , and its entry $\hat{\mu}_{\ell,i}^{(k)}(\theta)$ represents the belief of agent k that agent ℓ has hidden state θ , at time i . These agent-level beliefs are obtained directly from the partition $\mathcal{P}_k$. Specifically, for each cluster $C \in \mathcal{P}_k$, agent k sets the agent-level beliefs of all agents $\ell \in C$ as the private cluster belief associated with the cluster label $r = m(C)$. Thus, all agents that agent k assigns to the same discovered cluster inherit the same private cluster belief.

Finally, agent k estimates the hidden state of each agent ℓ , denoted by $\hat{s}_{\ell,i}^{(k)}$, by selecting the hypothesis with maximum belief:

$$\hat{s}_{\ell,i}^{(k)} = \arg \max_{\theta \in \Theta} \hat{\mu}_{\ell,i}^{(k)}(\theta).$$

This construction allows each agent to recover a network-wide state estimate while exchanging only beliefs at the cluster level.

The results below evaluate this two-phase strategy in two settings. First, we consider a controlled synthetic setting, where agents receive Gaussian measurements. This allows us to assess the error probability over repeated Monte Carlo simulations. Then, we evaluate the method using real gas-sensor measurements. In both settings,

we examine cluster-discovery, cluster-level belief evolution, and the final network-wide state estimates. Details on the experimental setup, dataset, likelihood models, classifiers, and metric definitions are provided in Section 4.3.

2.1 Synthetic Gaussian Experiments

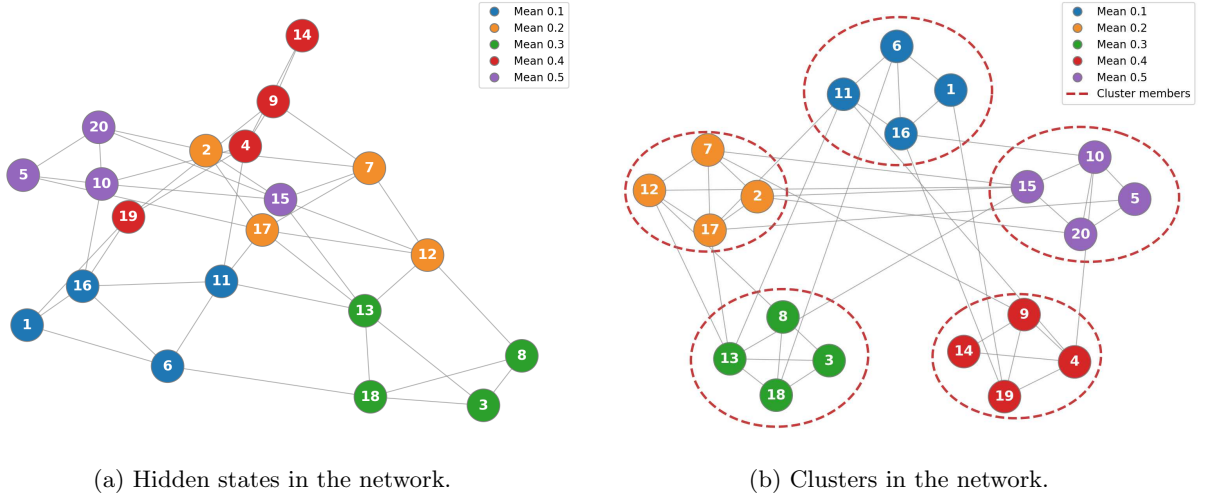


Fig. 4: Network in the synthetic-data experiment. Both panels show the same network. The graph contains $K = 20$ agents. Edges represent communication links; each agent has a self-loop, which is not displayed. Node colors indicate the hidden state of each agent. Panel (b) redraws panel (a) to highlight the cluster structure: dashed red contours marks the connected same-state components used as reference clusters.

We first evaluate the method in a controlled synthetic Gaussian setting with $K = 20$ agents. Each agent is assigned a local hidden state from a finite set of Gaussian means, $\Theta = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. At each time instant, each agent receives a local Gaussian measurement whose mean is determined by its own hidden state. Figure 4 shows the communication graph, the hidden-state assignment of the agents, and the corresponding clusters. The clusters are defined as connected components of agents that share the same hidden state.

The objective is twofold. First, each agent must identify the clusters in the network using only local observations and communication with neighboring agents. In this experiment, agents in the same cluster receive Gaussian observations with the same mean. Second, after discovering these clusters, each agent infers the Gaussian mean associated with each cluster. The inferred cluster states are then mapped back to the agents assigned to those clusters, yielding an estimate of the network-wide state configuration.

2.1.1 Cluster Discovery

We first examine the cluster-discovery phase. For visualization of the experimental results, we represent the partition $\mathcal{P}_{k,i}$ maintained by agent k at time i through the $K \times 1$ *label vector*

$$\mathbf{c}_i^{(k)} = \left[c_{1,i}^{(k)}, \dots, c_{K,i}^{(k)} \right]^\top, \quad (4)$$

where, for each cluster $C \in \mathcal{P}_{k,i}$,

$$c_{\ell,i}^{(k)} = m(C), \quad \forall \ell \in C. \quad (5)$$

Thus, the agents belonging to the same cluster are labeled by their smallest-indexed agent, and two agents belong to the same cluster if and only if they share the same label. In this experiment, the clustering procedure is run for 50 iterations; see Algorithms 1 and 4. Figure 5 shows along the horizontal direction the label vectors $\mathbf{c}_i^{(k)}$ maintained by all agents k at three time instants: $i = 0$, $i = 9$, and $i = 50$. The row index identifies the agent whose label vector is being shown; we call this agent the *observer agent*. The column index identifies the agent being assigned a label within that vector; we call this agent the *target agent*. Thus, the entry in row k and column ℓ is the discrete cluster label $c_{\ell,i}^{(k)}$ assigned by observer agent k to target agent ℓ at time i .

At initialization, agents have no information about which nodes share the same hidden state. Consequently, each agent treats every node as belonging to a distinct cluster, which results in a label vector $\mathbf{c}_0^{(k)}$ initialized as $c_{\ell,0}^{(k)} = \ell$ for every observer k and target ℓ . This initialization explains the first heatmap at time $i = 0$, where each target agent is assigned its own label.

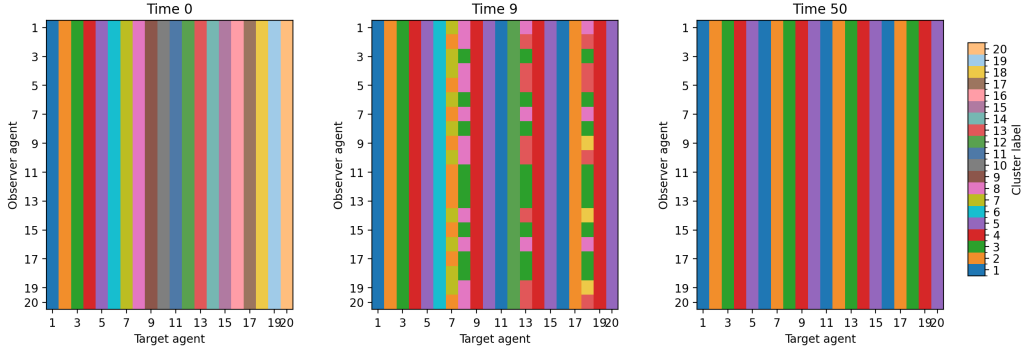


Fig. 5: Evolution of cluster labels in the synthetic-data experiment. Heatmaps show snapshots of the labels $c_{\ell,i}^{(k)}$ maintained by each observing agent k for each target agent ℓ during cluster-discovery. Rows denote observing agents and columns denote target agents; colors encode the assigned cluster label. Repeated colors within a row indicate target agents that the observing agent groups into the same cluster.

During the cluster-discovery phase, agents progressively update their clustering of the network using both their local observations and the information exchanged with their neighbors. At the intermediate time $i = 9$, agents have already started to merge target agents into common clusters, as indicated by repeated labels and the corresponding shared colors in the heatmap. For example, all agents assign the same label, namely label 4, to agents 4, 9, 14, and 19, indicating agreement that these agents belong to the same cluster. However, the label assignments are not yet fully consistent across all observer agents.

By the end of the cluster-discovery procedure, at time $i = 50$, the rows have stabilized to a common cluster-label pattern. For example, all agents assign label 1 to agents 1, 6, 11, and 16, indicating consensus that these agents belong to the same cluster. More generally, the final clustering recovered by the agents matches the connected same-state components shown in Fig. 4b. This demonstrates that, in this experiment, the cluster-discovery phase enables each agent to recover the correct cluster structure across the network. Importantly, this phase only identifies which agents share the same hidden state; it does not reveal which state in Θ is associated with each discovered cluster. The hidden states of the discovered clusters are inferred in the second phase.

2.1.2 Cluster-level Belief Evolution

After cluster-discovery, the agents freeze their recovered cluster structures and enter the network-wide state inference phase. In this phase, each agent performs social learning at the cluster level by maintaining one belief vector for each discovered cluster; see Algorithms 3 and 4. The cluster-discovery phase recovers 5 reference clusters, as shown in Fig. 5. Consequently, each agent maintains 5 private cluster beliefs during the second phase. We run the network-wide state inference phase for 50 iterations, from $i = 50$ to $i = 100$, and examine how the beliefs evolve over time.

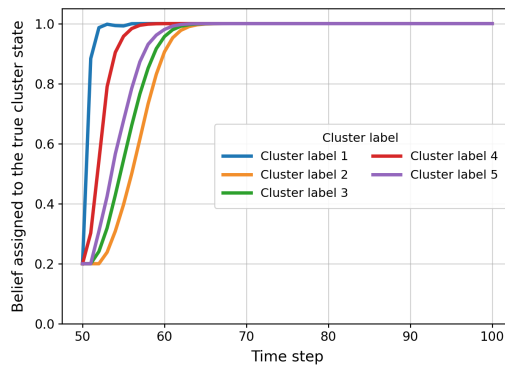


Fig. 6: Evolution of the private cluster beliefs for agent 1 in the synthetic-data experiment. Each curve corresponds to one discovered cluster and shows the belief assigned by agent 1 to its true hidden state over time.

Figure 6 shows the resulting private cluster belief trajectories for agent 1. Each curve corresponds to one discovered cluster labeled by r and reports the belief that agent 1 assigns to the true state of that cluster, namely $\mu_{r,i}^{(1)}(s_r^*)$, where s_r^* denotes the Gaussian mean associated with cluster r . The curves increase rapidly from their initial values and approach 1 for all discovered clusters. This shows that, once the cluster structure has been identified, agent 1 can correctly infer the hidden state associated with each cluster.

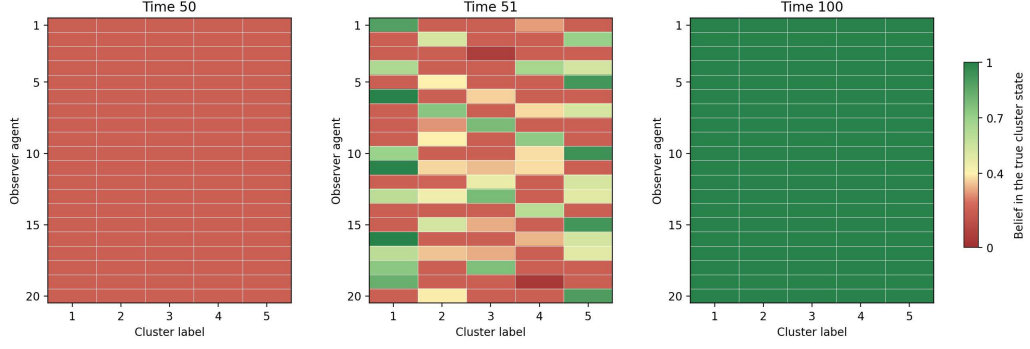


Fig. 7: Evolution of the private cluster beliefs in the synthetic-data experiment. Heatmaps show the private cluster beliefs $\mu_{r,i}^{(k)}(s_r^*)$ assigned by each agent k to the true hidden state s_r^* of each discovered cluster r . Rows denote agents and columns denote clusters; colors encode belief values between 0 and 1.

Figure 7 shows the evolution of the private cluster beliefs assigned to the correct hidden state of each cluster, across all agents, at three representative times: $i = 50$, $i = 51$, and $i = 100$. Each row corresponds to an agent k , while each column corresponds to a discovered cluster r . The entry in row k and column r is the belief that agent k assigns to the true hidden state of cluster r , namely $\mu_{r,i}^{(k)}(s_r^*)$. This quantity is a probability and therefore takes values in the interval $[0, 1]$.

At the start of the network-wide state inference phase, agents initialize their private cluster beliefs uniformly over the set of plausible hypotheses. In this experiment, the hypothesis set contains the five Gaussian means $\Theta = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. Hence, the initial belief assigned to the true state of each cluster is $1/5 = 0.2$, as shown in the first heatmap at time $i = 50$. After one learning step, at time $i = 51$, several entries have already increased above this initial value, indicating that agents are beginning to assign larger probabilities to the correct hidden states of the clusters. By the final displayed time, $i = 100$, all agents assign probability close to 1 to the true state of each discovered cluster. This shows that the network-wide state-inference phase enables all agents to concentrate their beliefs on the correct hidden state of every cluster.

2.1.3 Agent-level State Estimation

The ultimate objective is for each agent to infer the hidden state of every agent in the network, not only the hidden state of each discovered cluster. To obtain agent-level estimates, each agent maps its cluster beliefs back to individual agents to construct agent-level beliefs. Figure 8 shows the evolution of the agent-level beliefs

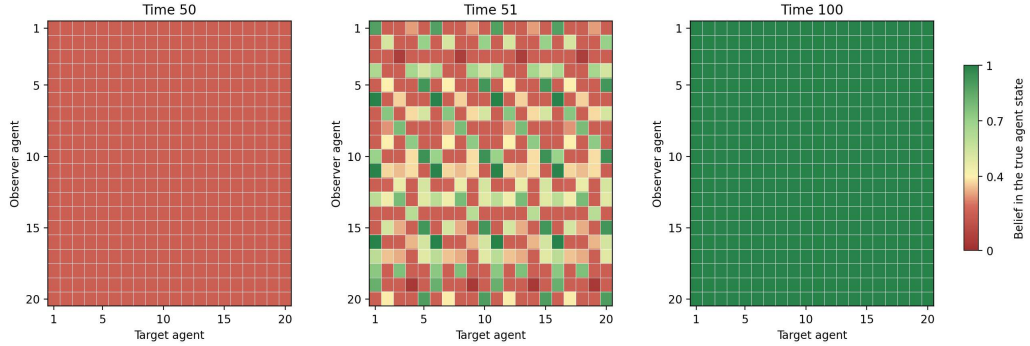


Fig. 8: Evolution of the agent-level beliefs in the synthetic-data experiment. Heatmaps show the agent-level beliefs $\hat{\mu}_{\ell,i}^{(k)}(s_\ell^*)$ assigned by each agent k to the true hidden state s_ℓ^* of each agent ℓ after mapping private cluster beliefs back to individual agents. Rows denote observer agents and columns denote target agents; colors encode belief values between 0 and 1.

assigned to the correct hidden state of each agent at three representative times: $i = 50$, $i = 51$, and $i = 100$. Each row corresponds to an observer agent k , while each column corresponds to a target agent ℓ . Thus, the entry in row k and column ℓ is the belief $\hat{\mu}_{\ell,i}^{(k)}(s_\ell^*)$ that agent k assigns to the true hidden state of agent ℓ . By the final displayed time, the heatmap is uniformly close to one, indicating that all observer agents assign high probability to the correct hidden state of every target agent. Each agent then forms a network-wide state estimate by selecting, for every target agent, the hypothesis with maximum agent-level belief. Figure 9 shows the network-wide state configuration inferred by agent 1 at $i = 50$, $i = 51$, and $i = 100$. At the beginning of the inference phase, the hidden states are incorrectly assigned because the cluster-level beliefs are uniform. As the

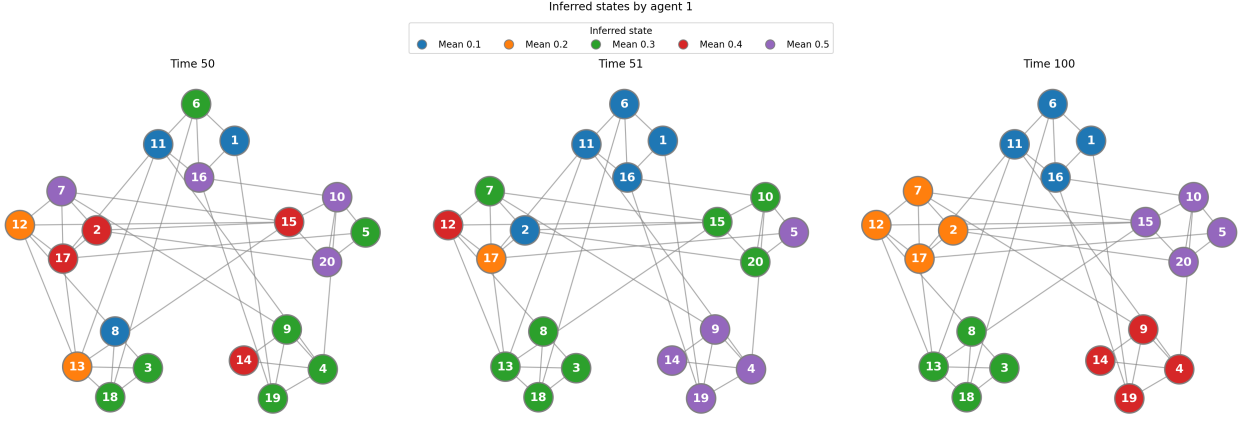


Fig. 9: Network-wide state estimates inferred by agent 1 in the synthetic-data experiment. Snapshots show the hidden-state configuration inferred by agent 1 at selected times during the network-wide state-inference phase. Node colors indicate the state estimated for each agent.

cluster-level beliefs concentrate on the correct hidden states, the inferred configuration progressively approaches the ground-truth assignment shown in Fig. 4a. By $i = 100$, the state configuration inferred by agent 1 matches the true hidden-state assignment.

2.1.4 Accuracy of Cluster-aware Social Learning

Cluster-aware social learning depends on two parameters that govern the cluster-discovery phase: the clustering horizon T and the clustering threshold ε ; see Algorithm 1. The parameter T specifies the number of iterations allocated to the cluster-discovery phase. The threshold $\varepsilon \in (0, 1)$ controls how strict the local clustering decision is. Each agent maintains a belief about its own hidden state, updated only from its private observations. Because neighboring agents exchange these beliefs, agent k can compare its belief with the belief of each neighbor. The threshold ε specifies how close a neighbor's belief must be to that of agent k before the neighbor is treated as sharing the same hidden state. A larger ε makes the decision less strict: agent k may include a neighbor in its cluster even when their beliefs are not very close. A smaller ε makes the decision more strict: a neighbor is included only when its belief is highly similar to that of agent k . Thus, ε determines how strong the local similarity between beliefs must be before agents are merged into the same cluster.

Both parameters affect the accuracy of the recovered clusters and the representation size of the second phase, since the number of inferred clusters determines how many cluster-level beliefs each agent maintains and exchanges during network-wide state inference.

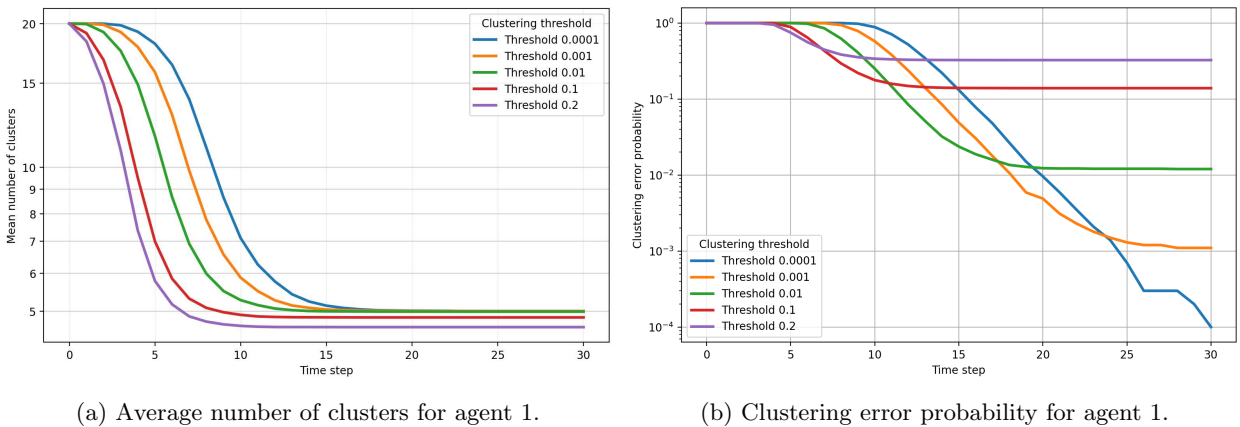
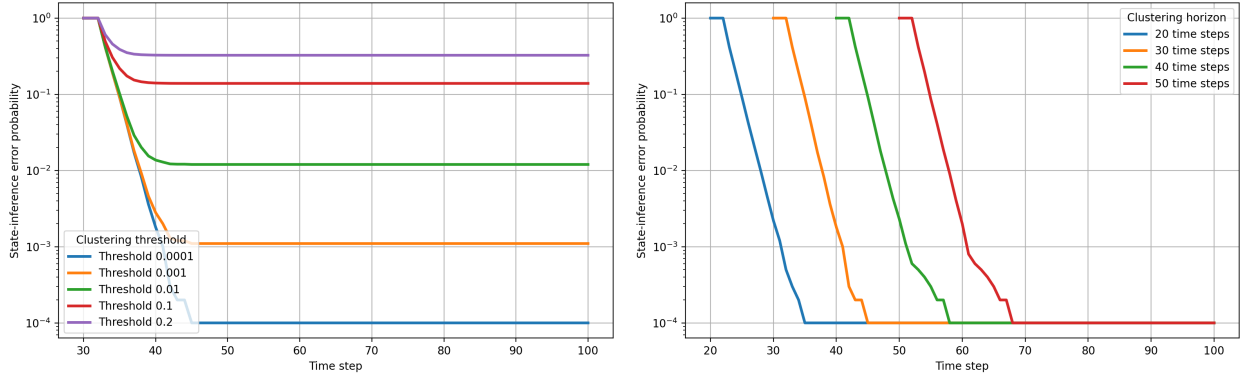


Fig. 10: Effect of the clustering parameters on the cluster-discovery phase. The cluster-discovery phase is run for $T = 30$ iterations for different values of the threshold ε .

We first examine how these parameters affect the cluster-discovery stage. Using the synthetic setup described above, we run the cluster-discovery procedure for 30 iterations over repeated Monte Carlo trials, while varying the clustering threshold as $\varepsilon \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.2\}$. For each value of ε , Fig. 10 reports two quantities for agent 1: the average number of inferred clusters and the clustering error probability. The corresponding metric definitions are provided in Section 4.3.



(a) Varying clustering threshold ε and fixed clustering horizon $T = 30$. (b) Varying clustering horizon T and fixed clustering threshold $\varepsilon = 0.0001$.

Fig. 11: Effect of the clustering parameters on network-wide state inference. State-inference error probability for agent 1.

Figure 10a shows that the number of inferred clusters is initialized to $K = 20$, since each agent initially treats every target node as a separate cluster. As cluster-discovery progresses, agents assign connected nodes that appear to share the same hidden state to the same cluster, and the number of inferred clusters decreases. For most threshold values, this number stabilizes close to the correct number of reference clusters, which is 5 in this experiment. However, for $\varepsilon = 0.1$ and $\varepsilon = 0.2$, the number of inferred clusters stabilizes below 5, indicating that some distinct reference clusters have been incorrectly merged.

This behavior is explained by the role of the threshold ε . The threshold controls how easily clustering decisions are made: a larger value of ε allows agents to group nodes together more easily, whereas a smaller value requires more evidence before nodes are assigned to the same cluster. Accordingly, larger thresholds lead to a faster decrease in the number of inferred clusters, while smaller thresholds lead to a slower decrease. However, when ε is too large, the cluster-discovery phase becomes too permissive, and agents may merge nodes before there is enough evidence that they belong to the same connected same-state component.

The clustering error probability in Fig. 10b confirms this behavior. Smaller thresholds lead to a more conservative clustering process: the error probability decreases more slowly, but reaches lower final levels. Larger thresholds lead to a faster initial decrease because clusters are merged more quickly, but the error may stabilize at a higher value when early merges are incorrect.

Overall, Fig. 10 shows that ε trades faster reduction in the number of inferred clusters against clustering accuracy: larger values reduce the number of clusters more quickly, while smaller values reduce the risk of incorrect merges.

The trajectories of Fig. 10 can also be interpreted in terms of the clustering horizon T : choosing a larger T means freezing the cluster structure later along these curves. If T is too small, cluster discovery may stop before the number of inferred clusters and the clustering error probability have stabilized. If T is too large, agents continue cluster discovery after the curves have already plateaued, which provides little additional benefit and only delays the start of the network-wide state-inference phase.

We next examine how these clustering decisions affect the network-wide state-inference phase. Figure 11 reports the state-inference error probability for agent 1. In Fig. 11a, we vary $\varepsilon \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.2\}$ while fixing the clustering horizon to $T = 30$. Smaller thresholds yield lower final error probabilities, whereas larger thresholds yield higher error floors. This shows that the quality of the cluster structure recovered in the first phase directly determines the accuracy achievable in the second phase. If ε is too large, agents with different hidden states may be assigned to the same discovered cluster, and the state inference phase cannot undo this incorrect grouping once the cluster structure is frozen.

In Fig. 11b, we vary the clustering horizon $T \in \{20, 30, 40, 50, 60\}$ while fixing $\varepsilon = 10^{-4}$. Each curve starts at a different time because the network-wide state inference phase begins only after cluster discovery has ended. The curves have nearly the same shape and reach the same final error level, but larger values of T shift the learning curve to the right. Thus, for this conservative threshold, increasing T beyond the time needed to recover a reliable cluster structure does not improve final inference accuracy; it mainly delays the start of the second phase.

Overall, Figs. 10 and 11 show that ε and T must be selected jointly. The threshold ε determines how aggressively clusters are formed, while T determines when this structure is passed to the state-inference phase. A good choice uses a threshold small enough to avoid incorrect merges and a horizon long enough for the cluster structure to stabilize.

2.2 Real Gas-sensor Experiment

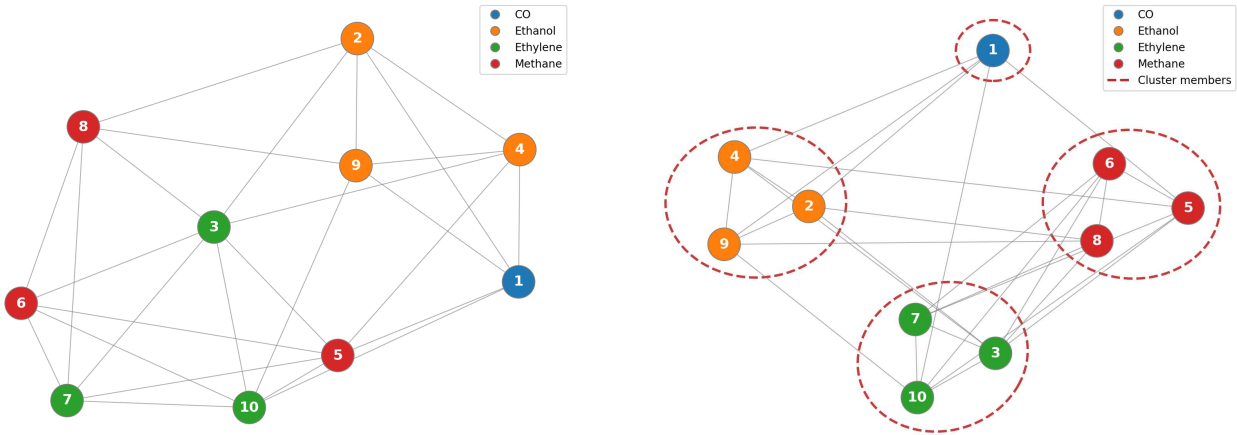


Fig. 12: Network used for real-data validation. Both panels show the same network. The graph contains $K = 10$ agents. Edges represent communication links; each agent also has a self-loop, which is not displayed. Node colors indicate the hidden state of each agent. Panel (b) redraws panel (a) to highlight the cluster structure: dashed red contours marks the connected same-state components used as reference clusters.

We next evaluate cluster-aware social learning using real chemical gas-sensor measurements. This experiment follows the same structure as the synthetic experiment, but replaces the Gaussian observations with empirical sensor recordings. The communication graph is synthetically generated, whereas the local measurements processed by the agents are obtained from real gas-sensor recordings. Details on the dataset, preprocessing, likelihood construction, and classifiers are provided in Section 4.3.

We consider a network of $K = 10$ agents, shown in Fig. 12. The hidden state of each agent corresponds to one of four gas types: CO, ethanol, ethylene, and methane. Each agent is equipped with a four-channel chemical sensor and, at each time instant, receives a four-dimensional real-valued conductance measurement associated with the gas type assigned to that agent. Thus, the experiment tests whether the proposed cluster-aware social learning procedure can recover a heterogeneous network-wide state when the local statistical evidence comes from empirical sensor data rather than from a synthetic observation model.

Figure 12 shows the hidden gas assignment and the clusters over the network. As in the synthetic experiment, agents do not initially know this cluster structure. They only observe their own measurement streams and communicate locally with their neighbors. In the first phase of cluster-aware social learning, each agent aims to discover the clusters in the network. In the second phase, they perform social learning at the cluster level and map the resulting cluster beliefs back to individual agents.

2.2.1 Cluster Discovery

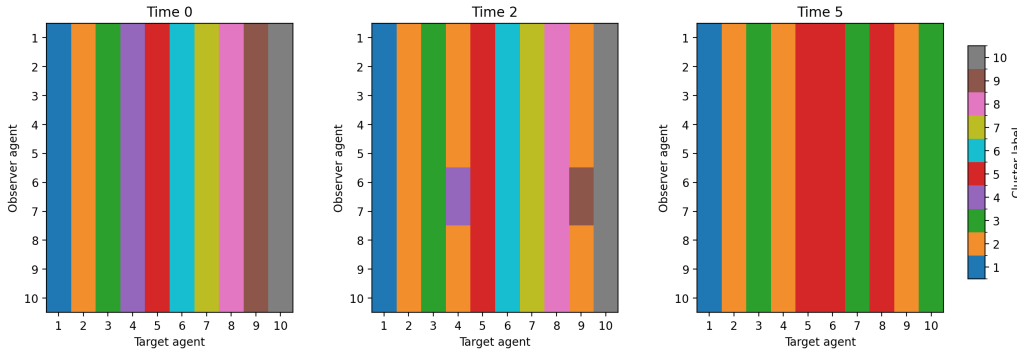


Fig. 13: Evolution of cluster labels in the real-data experiment. Heatmaps show snapshots of the labels $c_{\ell,i}^{(k)}$ maintained by each observing agent k for each target agent ℓ during cluster-discovery. Rows denote observing agents and columns denote target agents; colors encode the assigned cluster label. Repeated colors within a row indicate target agents that the observing agent groups into the same cluster.

We first examine cluster-discovery. Here, we also use the label vector defined in (4)-(5) to visualize the results. To this end, we run the cluster-discovery phase for 5 iterations; see Algorithms 1 and 4. Figure 13 shows the cluster label vectors $\mathbf{c}_i^{(k)}$ maintained by all agents k at three time instants: $i = 0$, $i = 2$, and $i = 5$. Each row corresponds to an observing agent k , and each column corresponds to a target agent ℓ whose cluster label is estimated by agent k . The entry at row k and column ℓ is the label $\mathbf{c}_{\ell,i}^{(k)}$ assigned by observer k to target ℓ at time i .

At initialization, every agent treats every target node as belonging to a separate cluster, which corresponds to the heatmap at $i = 0$. After two iterations, the agents begin to merge labels associated with connected agents whose observations correspond to the same gas type. By time $i = 5$, the label vectors have stabilized and agree with the reference clusters shown in Fig. 12. This demonstrates that, in this real-data experiment, the cluster-discovery phase enables each agent to recover the correct cluster structure. However, this phase only identifies the clusters; it does not reveal which state is associated with each discovered cluster. This state-inference task is performed in the second phase.

2.2.2 Cluster-level Belief Evolution

After cluster-discovery, the agents freeze their recovered cluster structures and enter the network-wide state inference phase. We run this phase for 16 iterations, from $i = 5$ to $i = 20$; see Algorithms 3 and 4.

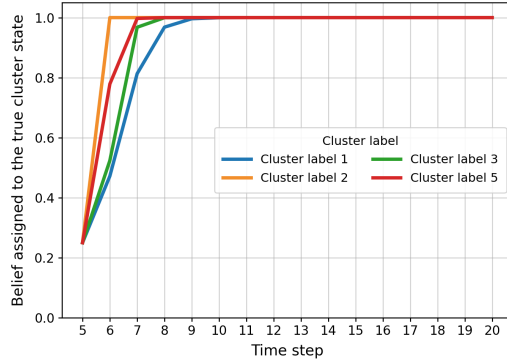


Fig. 14: Evolution of the private cluster beliefs for agent 4 in the real-data experiment. Each curve corresponds to one discovered cluster and shows the belief assigned by agent 4 to its true hidden state over time.

Figure 14 reports the evolution of the private cluster beliefs maintained by agent 4, for the correct hidden state of each cluster. For each discovered cluster, the belief assigned to the true gas state increases over time and eventually approaches one. Thus, the private cluster beliefs concentrate on the correct gas identity, showing that agent 4 can recover the hidden state of every discovered cluster.

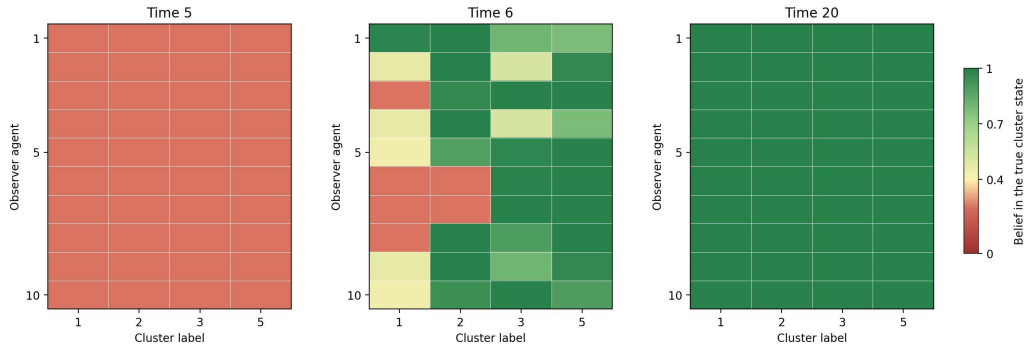


Fig. 15: Evolution of the private cluster beliefs in the real-data experiment. Heatmaps show the private cluster beliefs $\mu_{r,i}^{(k)}(s_r^*)$ assigned by each agent k to the true hidden state s_r^* of each discovered cluster r . Rows denote agents and columns denote clusters; colors encode belief values between 0 and 1.

Figure 15 reports the belief assigned by each agent to the true gas type of each discovered cluster at three time instants: $i = 5$, $i = 6$, and $i = 20$. At the beginning of the inference phase, the private cluster beliefs are still uninformative, as they are initialized to a uniform distribution. After one iteration, the agents start assigning larger probability mass to the correct gas type of each cluster. By time $i = 20$, the belief assigned to the true state of each cluster is close to 1 across the network. This confirms that, once the cluster structure has

been recovered, the agents can learn the gas identity of every discovered cluster using only local measurements and neighbor-to-neighbor communication.

2.2.3 Agent-level State Estimation

In order to recover the network-wide state, each agent maps the private cluster beliefs back to individual agents. Specifically, for every target agent ℓ , observer agent k assigns to ℓ the belief of the cluster to which ℓ belongs. This produces the collection of agent-level beliefs.

Figure 16 shows the resulting agent-level beliefs for all observing agents at the correct hidden states of all target agents. At $i = 5$, the beliefs are still uniform because the private cluster beliefs have just been initialized. At $i = 6$, agents begin to assign higher belief to the correct gas states of the target agents. By $i = 20$, the belief assigned to the true gas state of each target agent is close to one. This demonstrates that cluster-aware social learning enables the recovery of the hidden states of all agents across the network.

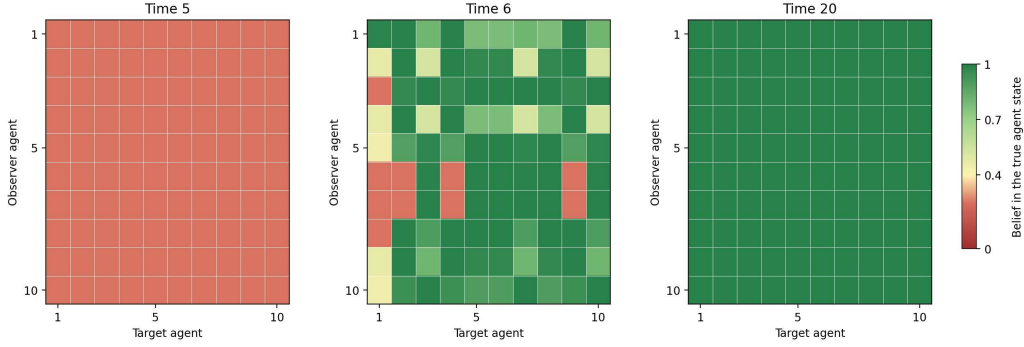


Fig. 16: Evolution of the agent-level beliefs in the real-data experiment. Heatmaps show the agent-level beliefs $\hat{\mu}_{\ell,i}^{(k)}(s_\ell^*)$ assigned by each agent k to the true hidden state s_ℓ^* of each agent ℓ after mapping private cluster beliefs back to individual agents. Rows denote observer agents and columns denote target agents; colors encode belief values between 0 and 1.

Figure 17 provides a direct visualization of the network-wide state estimate formed by agent 4. At the beginning of the inference phase, the gas states inferred by agent 4 are not all correct. As the private cluster beliefs concentrate on the correct gas states, the inferred network-wide state becomes increasingly consistent with the true hidden gas assignment shown in Fig. 12. By the final displayed time, agent 1 correctly recovers the network-wide gas-state configuration.

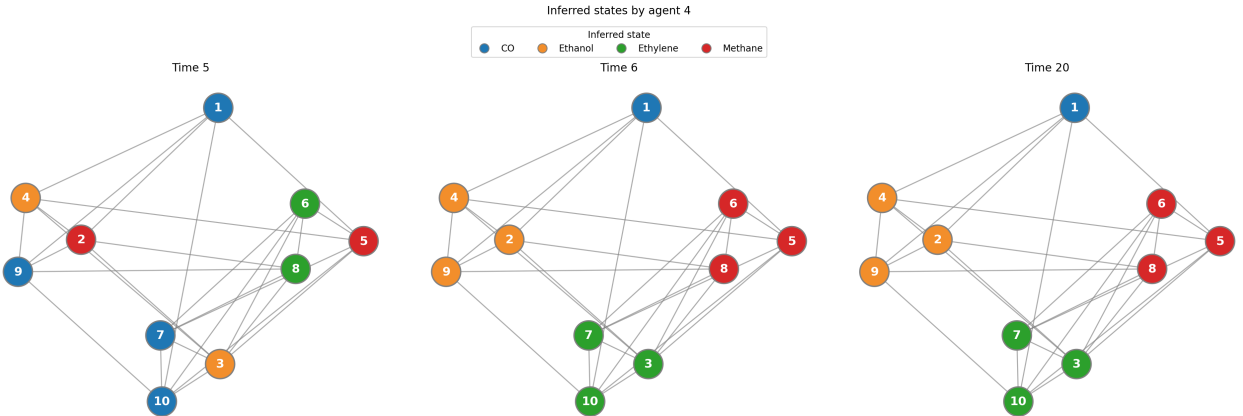


Fig. 17: Network-wide state estimates inferred by agent 1 in the real-data experiment. Snapshots show the hidden-state configuration inferred by agent 1 at selected times during the network-wide state-inference phase. Node colors indicate the state estimated for each agent.

Overall, the real gas-sensor experiment confirms the behavior observed in the synthetic setting. Once the agents recover the correct clusters, they learn the gas state associated with each discovered cluster, and successfully recover the network-wide state configuration using only local measurements and neighbor-to-neighbor communication.

3 Discussion

In this work, we consider a setting that departs from standard social learning formulations. In many existing works, agents either collaborate to infer a single hidden state shared by the entire network or, when hidden states are agent-specific, each agent focuses only on recovering its own state. Here, the objective is network-wide: each agent aims to recover the hidden state of all agents in the network. This is relevant in sensing applications where the useful information is not only whether an event is present locally, but also how different events are distributed across the network.

We introduced cluster-aware social learning as a decentralized method for network-wide state inference in heterogeneous-state environments. The central idea is to exploit the fact that, although agents may have different hidden states, several agents may share the same state. Rather than maintaining and exchanging beliefs for every possible network-wide configuration, agents first identify groups of connected agents that appear to share the same hidden state (i.e., clusters) and then perform social learning within these clusters. This allows each agent to recover an estimate of the hidden state of every agent in the network while exchanging one belief per discovered cluster.

The results show that the proposed two-phase structure can recover the network-wide state configuration in both synthetic and real-data experiments. In the synthetic experiment, once the agents correctly recover the clusters, they can successfully infer the hidden state associated with each cluster and obtain an estimate of the hidden state of all agents. The real gas-sensor experiment shows the same behavior when the observations come from empirical sensor recordings rather than from a synthetic Gaussian model. This suggests that cluster discovery is a useful way to reduce the complexity of the network-wide inference problem.

The main advantage of the approach is that the number of beliefs maintained and exchanged by each agent depends on the number of discovered clusters, rather than on the number of agents or on the number of possible network-wide configurations. Thus, when the network contains only a small number of clusters, each agent can recover a network-wide state while exchanging only cluster-level information. This substantially reduces the representation used during the second phase.

The experiments also show that this reduction comes with a trade-off. The clustering threshold ε controls how easily agents merge clusters, while the clustering horizon T represents the stopping time of the cluster-discovery phase. A larger threshold can reduce the number of inferred clusters faster, but it can also lead to incorrect merges between agents with different hidden states. A lower threshold is more conservative and can improve the accuracy of recovered clusters, but may require a longer cluster-discovery phase. Similarly, increasing T gives agents more time to stabilize their cluster labels, but it also delays the start of network-wide state inference. Thus, the clustering parameters must be chosen so that the cluster structure is accurate enough before it is used in the second phase.

A limitation of the current method is that cluster discovery and network-wide state inference are decoupled. Agents must first complete the cluster-discovery phase before starting the main inference task. A natural extension would be to design a coupled version of cluster-aware social learning, where agents update their cluster structure and their cluster-level beliefs in parallel.

Such a coupled approach could also address another limitation of the current method. If two agents with different hidden states are incorrectly merged during cluster discovery, they share the same cluster belief in the second phase. As a result, the state-inference phase cannot fully correct this type of error once the cluster structure has been frozen. Allowing agents to continue revising their cluster structure during inference could make the method more robust to early clustering mistakes. It would also remove the need to choose a fixed clustering horizon T , since the clustering and inference phases would no longer be separated.

Several other extensions are possible. The present work focuses on stationary hidden states and fixed communication graphs. In many monitoring problems, however, the underlying states may change over time, and communication links may be unreliable or time-varying. Extending the method to such settings would make it more suitable for dynamic sensing systems.

Overall, cluster-aware social learning provides a decentralized approach to network-wide state inference when agents may have different, yet possibly shared, hidden states. By first identifying clusters and then learning one belief per cluster, the method reduces the number of beliefs exchanged during inference while still allowing agents to estimate the hidden state of every agent in the network.

4 Methods

4.1 Problem formulation

We consider a network of K spatially distributed agents, indexed by the set $\mathcal{K} = \{1, \dots, K\}$. Communication among agents is peer-to-peer: each agent interacts only with its immediate neighbors. We denote by $\mathcal{N}_k \subseteq \mathcal{K}$ the neighborhood of agent k , including k itself.

Information exchange over the network is described by a left-stochastic combination matrix $A = [a_{\ell k}]$. That is, the entries on each column of A add up to 1:

$$\sum_{\ell=1}^K a_{\ell k} = 1, \quad \forall k \in \mathcal{K}, \quad (6)$$

with $a_{\ell k} > 0$ for all $\ell \in \mathcal{N}_k$ and 0 otherwise. An entry $a_{\ell k}$ represents the relative weight assigned by agent k to the information received from agent ℓ . Thus, each agent forms a convex combination of the information available in its neighborhood. We impose the following standard connectivity condition on the combination matrix.

Assumption 1 (Primitive matrix) *The combination matrix A is assumed to be primitive [34].* ■

A sufficient condition for the combination matrix A to be primitive is that undirected paths with nonzero weights exist between every pair of distinct agents, and that at least one agent k has a self-loop (i.e., $a_{kk} > 0$).

Each agent observes a stream of noisy measurements governed by a fixed but unknown local state. This state represents an unknown property associated with the agent and takes values in a finite hypothesis set $\Theta = \{\theta_1, \dots, \theta_H\}$. We consider a general *heterogeneous setting* in which the hidden states of the agents are not necessarily identical. We denote the true state of agent k by $s_k^* \in \Theta$ and the collection of all agents' true states by

$$s^* = (s_1^*, \dots, s_K^*) \in \Theta^K, \quad (7)$$

where we refer to s^* as the *network-wide state*.

At each time instant $i = 1, 2, \dots$, agent k records a local measurement $\mathbf{x}_{k,i}$ generated according to the likelihood model $L_k(\cdot | s_k^*)$, independently over time. Each agent knows the family of local likelihoods $\{L_k(\cdot | \theta)\}_{\theta \in \Theta}$. These likelihoods allow each agent to assess how consistent its observations are with the competing local hypotheses in the set Θ . We consider the following assumption on the statistical models of the agents.

Assumption 2 (Statistical model) *Let $\mathbf{x}_i \triangleq \{\mathbf{x}_{k,i}\}_{k=1}^K$ collect all observations from the agents at time i , and let $s = (s_1, \dots, s_K) \in \Theta^K$ be a plausible network-wide state. The joint likelihood at time i satisfies*

$$L(\mathbf{x}_i | s) = \prod_{k=1}^K L_k(\mathbf{x}_{k,i} | s_k). \quad (8)$$

■

We also require the log-likelihood ratios between competing hypotheses to be well behaved, which is represented by the following assumption.

Assumption 3 (Finite divergence) *For all $\theta, \theta' \in \Theta$ with $\theta \neq \theta'$, and for all $k \in \mathcal{K}$, the Kullback-Leibler (KL) divergence [15] between $L_k(\cdot | \theta)$ and $L_k(\cdot | \theta')$ is finite:*

$$\mathbb{E}_\theta \left[\log \frac{L_k(\mathbf{x} | \theta)}{L_k(\mathbf{x} | \theta')} \right] < \infty, \quad (9)$$

where the subscript on the expectation operator means that the expectation is computed under the model $L_k(\cdot | \theta)$. ■

Finally, we assume that each local state is statistically distinguishable from the other plausible states from the point of view of each agent in the network.

Assumption 4 (Identifiability) *For all $\theta, \theta' \in \Theta$ with $\theta \neq \theta'$, and for all $k \in \mathcal{K}$, the KL divergence [15] between $L_k(\cdot | \theta)$ and $L_k(\cdot | \theta')$ is positive:*

$$\mathbb{E}_\theta \left[\log \frac{L_k(\mathbf{x} | \theta)}{L_k(\mathbf{x} | \theta')} \right] > 0, \quad (10)$$

where the subscript on the expectation operator means that the expectation is computed under the model $L_k(\cdot | \theta)$. ■

Assumption 4 is a feasibility condition for network-wide inference. Since agent k can be the only agent that receives observations directly informative about its own hidden state s_k^* , it must be able to distinguish s_k^* from all competing hypotheses in Θ . If this local ambiguity cannot be resolved by agent k , then no other agent can recover the state of agent k , and the uncertainty persists at the network level.

However, Assumption 4 is only local: it requires each agent to distinguish its own hidden state using its private observations. It does not imply that any single agent can distinguish the true network-wide state s^* from

all alternatives in Θ^K . Thus, each agent initially has only partial information about the network-wide state, and this information must be shared across the network. A straightforward way to do so would be for agents to propagate separate decision statistics for every agent in the network. However, this agent-level statistics exchange would make the communication cost grow with the network size.

The next subsection introduces cluster-aware social learning as a way to avoid this agent-level statistics exchange. The key idea is to exploit shared hidden states: agents first identify groups that can be represented by a common state, and then exchange beliefs about these groups rather than about individual agents.

4.2 Cluster-aware Social Learning

In principle, since each agent seeks to infer the states of all agents in the network, it would need to keep track of one state estimate for every agent. This means that the amount of information stored and exchanged by each agent would grow with the network size K . Such a representation becomes costly in large networks, both in terms of communication and computation.

However, this agent-level representation can be highly redundant when several agents share the same hidden state. In that case, it is unnecessary to infer and communicate separate state estimates for agents that actually have the same state. The key idea is therefore to exploit this redundancy. If each agent can identify subsets of the network that share a common hidden state, it can group these agents into clusters and infer one state per cluster instead of one state per agent. As a result, the information that each agent needs to store and exchange scales with the number of discovered clusters, rather than with the total number of agents in the network.

Motivated by this observation, we propose *cluster-aware social learning* (CASL), a two-phase inference strategy where in the first phase, agents use local observations and peer-to-peer exchanges to identify groups of agents that share the same hidden state and in the second phase, agents cooperatively infer the hidden state associated with each discovered group. This two phase procedure allows each agent to discover all the hidden states in the network by assigning every agent the state inferred for its corresponding cluster.

In what follows, we present the two phases of the proposed strategy: i) cluster discovery and ii) network-wide state inference. Algorithm 1 describes a single iteration of the cluster discovery procedure, and Algorithm 3 describes a single iteration of the state-inference procedure. Algorithm 4 presents the complete CASL strategy, where cluster discovery is repeated until its stopping criterion is satisfied, after which the network-wide state-inference phase is carried out.

4.2.1 Phase I: cluster-discovery

We recall that a cluster refers to a group of agents that share the same hidden state and can therefore be represented by a single state estimate. Since agents do not initially know these groups, the goal of this phase is to enable each agent to partition the network into groups of agents sharing the same state.

To implement cluster discovery, each agent k represents its clustering of the network at time i by a *private partition* $\mathcal{P}_{k,i}$ of the set of agents $\mathcal{K}$. Specifically, $\mathcal{P}_{k,i}$ represents a collection of subsets of $\mathcal{K}$ that are nonempty, mutually disjoint, and collectively contain all agents:

$$\bigcup_{C \in \mathcal{P}_{k,i}} C = \mathcal{K}, \quad \text{and} \quad C \cap C' = \emptyset, \quad \forall C, C' \in \mathcal{P}_{k,i}, C \neq C'. \quad (11)$$

Each block of $\mathcal{P}_{k,i}$ represents one cluster inferred by agent k at time i . Hence, two agents ℓ and ℓ' are regarded by agent k as belonging to the same cluster if and only if they belong to the same cluster of $\mathcal{P}_{k,i}$.

At the beginning of the cluster-discovery phase, agent k has no information about which agents share the same hidden state. It therefore starts from the finest possible partition by assigning each agent to a distinct cluster:

$$\mathcal{P}_{k,0} = \{\{1\}, \{2\}, \dots, \{K\}\}. \quad (12)$$

Then, at each iteration $i \geq 1$, each agent k updates its private partitions using local observations and neighbor exchanges by repeatedly applying the updates in Algorithm 1. To this end, each agent k first constructs a *local belief* over its own hidden state,

$$\boldsymbol{\pi}_{k,i} = [\pi_{k,i}(\theta_1), \dots, \pi_{k,i}(\theta_H)]^\top.$$

The vector $\boldsymbol{\pi}_{k,i}$ is a probability distribution over Θ , where $\pi_{k,i}(\theta)$ represents the confidence of agent k that its hidden state is θ at time i . The local belief is initialized uniformly so that all hypotheses are initially considered equally possible. After observing $\mathbf{x}_{k,i}$, agent k updates its local belief using the Bayesian rule in line 2 of Algorithm 1. The symbol $\propto$ indicates that the entries in $\boldsymbol{\pi}_{k,i}$ are normalized so that they sum up to 1. This update increases the belief assigned to hypotheses that better explain the local measurement $\mathbf{x}_{k,i}$.

After the local Bayesian update, agent k transmits its belief $\boldsymbol{\pi}_{k,i}$ to its neighbors and receives their corresponding local beliefs, as described in lines 3–4 of Algorithm 1. Agent k then uses these exchanged beliefs to identify the neighboring agents that currently appear to share its hidden state. Specifically, in line 5, agent k constructs the set $\mathcal{N}_{k,i}$ by retaining those neighbors $\ell \in \mathcal{N}_k$ whose beliefs $\boldsymbol{\pi}_{\ell,i}$ are sufficiently concentrated on

Algorithm 1 Cluster-discovery step for agent k

- 1: **function** CLUSTERDISCOVERYSTEP($\pi_{k,i-1}, \mathcal{P}_{k,i-1}, \mathbf{x}_{k,i}$)
 - 2: **Local Bayesian update:**

$$\pi_{k,i}(\theta) \propto \pi_{k,i-1}(\theta) L_k(\mathbf{x}_{k,i}|\theta) \quad \forall \theta \in \Theta.$$
 - 3: Transmit $\pi_{k,i}$ to all neighbors $\ell \in \mathcal{N}_k \setminus \{k\}$.
 - 4: Receive $\pi_{\ell,i}$ from all neighbors $\ell \in \mathcal{N}_k \setminus \{k\}$.
 - 5: **Same-state neighbor identification:**

$$\mathcal{N}_{k,i} = \left\{ \ell \in \mathcal{N}_k : \exists \theta \in \Theta \text{ such that } \max_{\theta' \neq \theta} \pi_{k,i}(\theta') \leq \varepsilon \text{ and } \max_{\theta' \neq \theta} \pi_{\ell,i}(\theta') \leq \varepsilon \right\} \cup \{k\}.$$
 - 6: **Intermediate partition update:**

$$\tilde{\mathcal{P}}_{k,i} = \text{Merge}(\mathcal{P}_{k,i-1}, \mathcal{N}_{k,i}).$$
 - 7: Transmit $\tilde{\mathcal{P}}_{k,i}$ to all neighbors $\ell \in \mathcal{N}_k \setminus \{k\}$.
 - 8: Receive $\tilde{\mathcal{P}}_{\ell,i}$ from all neighbors $\ell \in \mathcal{N}_k \setminus \{k\}$.
 - 9: **Private partition update:**

$$\mathcal{P}_{k,i} = \bigvee_{\ell \in \mathcal{N}_k} \tilde{\mathcal{P}}_{\ell,i}.$$
 - 10: **return** $\pi_{k,i}, \tilde{\mathcal{P}}_{k,i}, \mathcal{P}_{k,i}$
 - 11: **end function**
-

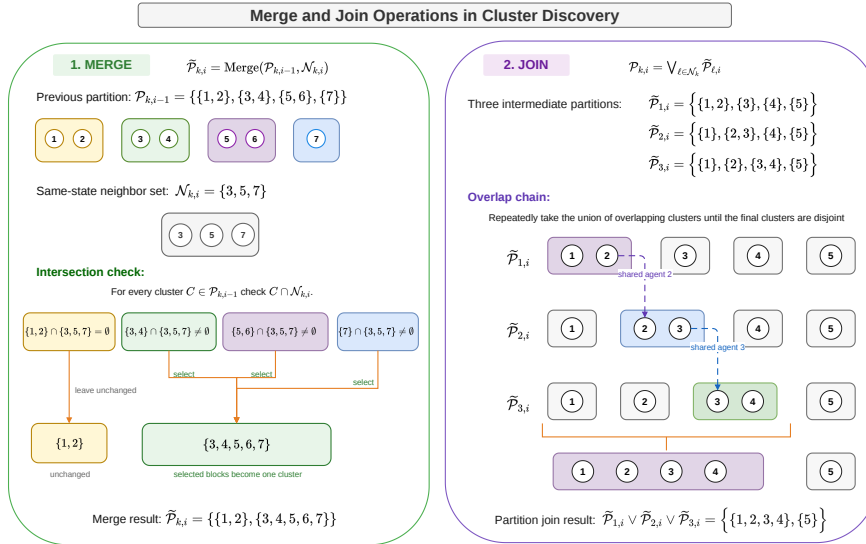


Fig. 18: Illustrative examples of the merge and partition join operations in Algorithm 1.

the same hypothesis as $\pi_{k,i}$. More precisely, agent k treats ℓ as belonging to its cluster if there is a common hypothesis θ such that both agents assign only a small probability, less than ε , to every alternative hypothesis $\theta' \neq \theta$. The parameter $\varepsilon \in (0, 1)$ is referred to as the *clustering threshold*. If no neighbor satisfies this condition, then $\mathcal{N}_{k,i}$ contains only agent k .

Agent k then incorporates the set $\mathcal{N}_{k,i}$ into its previous private partition $\mathcal{P}_{k,i-1}$ to construct the *intermediate partition* $\tilde{\mathcal{P}}_{k,i}$, as specified in line 6 of Algorithm 1. The update is performed through the operator Merge, represented in Algorithm 2. Specifically, this operator identifies all clusters of $\mathcal{P}_{k,i-1}$ that contain at least one agent in $\mathcal{N}_{k,i}$ and merges these clusters into a single cluster. All remaining clusters are left unchanged (see Fig. 18 for an illustrative example).

Since $k \in \mathcal{N}_{k,i}$, the cluster containing agent k always participates in the merge. Therefore, if agent k identifies a neighbor ℓ as sharing its hidden state, the entire cluster to which ℓ was previously assigned is merged with the cluster containing k . This allows agent k to preserve clustering information accumulated over earlier iterations while incorporating the new evidence available at iteration i . For example, if we have

$$\mathcal{P}_{2,i-1} = \{\{1, 2\}, \{3\}, \{4, 5\}\}, \quad \mathcal{N}_{2,i} = \{2, 4\},$$

Algorithm 2 Merge operation at agent k

```

1: function Merge( $\mathcal{P}_{k,i-1}, \mathcal{N}_{k,i}$ )
2:    $\tilde{\mathcal{P}}_{k,i} \leftarrow \mathcal{P}_{k,i-1}$ 
3:    $\mathcal{C}_{k,i} \leftarrow \emptyset$ 
4:   for each cluster  $C \in \mathcal{P}_{k,i-1}$  do
5:     if  $C \cap \mathcal{N}_{k,i} \neq \emptyset$  then
6:        $\mathcal{C}_{k,i} \leftarrow \mathcal{C}_{k,i} \cup C$ 
7:        $\tilde{\mathcal{P}}_{k,i} \leftarrow \tilde{\mathcal{P}}_{k,i} \setminus \{C\}$ 
8:     end if
9:   end for
10:   $\tilde{\mathcal{P}}_{k,i} \leftarrow \tilde{\mathcal{P}}_{k,i} \cup \{\mathcal{C}_{k,i}\}$ 
11: end function

```

then

$$\tilde{\mathcal{P}}_{2,i} = \text{Merge}(\mathcal{P}_{2,i-1}, \mathcal{N}_{2,i}) = \{\{1, 2, 4, 5\}, \{3\}\}.$$

Indeed, when agent 2 identifies agent 4 as sharing its hidden state, it merges their previously inferred clusters.

Thus, the steps in lines 2–6 primarily update the estimate by agent k for its own cluster. To also update its view of the remaining clusters in the network, agent k exchanges clustering information with its neighbors. Specifically, it transmits its intermediate partition and receives the corresponding intermediate partitions from its neighbors, as described in lines 7–8. Then, agent k combines these intermediate partitions using the *partition join* $\vee$, a standard operation on set partitions [35], to obtain its updated private partition:

$$\mathcal{P}_{k,i} = \bigvee_{\ell \in \mathcal{N}_k} \tilde{\mathcal{P}}_{\ell,i}, \quad (13)$$

as specified in line 9 of Algorithm 1. Intuitively, the partition join combines the clustering relationships identified by the different agents. Whenever two clusters from different intermediate partitions overlap, they are merged, and this merging is repeated until all resulting clusters are disjoint; See Fig. 18. For example, consider

$$\tilde{\mathcal{P}}_{1,i} = \{\{1, 2\}, \{3\}, \{4\}\}, \quad \tilde{\mathcal{P}}_{2,i} = \{\{1\}, \{2, 3\}, \{4\}\}.$$

Since the clusters $\{1, 2\}$ and $\{2, 3\}$ share agent 2, the join combines them into a single cluster, yielding

$$\tilde{\mathcal{P}}_{1,i} \vee \tilde{\mathcal{P}}_{2,i} = \{\{1, 2, 3\}, \{4\}\}.$$

Hence, information that agents 1 and 2 belong to the same cluster in one partition, together with information that agents 2 and 3 belong to the same cluster in another, implies that all three agents belong to the same cluster in the joined partition.

Equivalently, the partition join can be viewed through a graph representation. The agents form the vertices of a graph, and agents that appear in the same cluster of at least one intermediate partition are connected. The clusters of the resulting join are then simply the connected components of this graph [36]. This interpretation leads to efficient implementations using standard procedures such disjoint-set union (DSU) [37] or depth-first search (DFS) [38]. Specifically, at agent k , combining the $|\mathcal{N}_k|$ intermediate partitions requires $\mathcal{O}(|\mathcal{N}_k|K\alpha(K))$ operations with DSU, where $\alpha(\cdot)$ is the inverse Ackermann function and is practically constant for all realistic network sizes, or $\mathcal{O}(|\mathcal{N}_k|K)$ operations with DFS.

In summary, during the cluster-discovery phase, each agent repeatedly identifies neighboring agents that appear to share its hidden state and merges this information to its previous private partition to form an intermediate partition. The agents then exchange these intermediate partitions with their neighbors and combine them to update their private partitions. These steps are repeated for T iterations. At the end of the cluster-discovery phase, each agent freezes its resulting partition, which is subsequently used during the network-wide state-inference phase. To understand the structure of the clusters discovered at the end of the cluster-discovery phase, we present the following theorem.

Theorem 1 (Correct cluster discovery) *For each agent $k \in \mathcal{K}$, let*

$$\tilde{\mathcal{N}}_k \triangleq \{\ell \in \mathcal{N}_k : s_\ell^* = s_k^*\} \quad (14)$$

*denote the set of neighbors that share the true state of agent k . Suppose that the neighbor-identification rule in line 5 of Algorithm 1 selects **only** same-state neighbors at all times, and selects **all** same-state neighbors after some finite time $\tau \geq 1$. That is,*

$$\mathcal{N}_{k,i} \subseteq \tilde{\mathcal{N}}_k, \quad \forall k \in \mathcal{K}, \quad \forall i \geq 0, \quad (15)$$

$$\mathcal{N}_{k,i} = \tilde{\mathcal{N}}_k, \quad \forall k \in \mathcal{K}, \quad \forall i \geq \tau. \quad (16)$$

Let $d_{\ell k}$ denote the minimum number of hops between agents ℓ and k , and let

$$D \triangleq \max_{\ell, k \in \mathcal{K}} d_{\ell k} \quad (17)$$

be the diameter of the communication graph. Under Assumption 1, repeated application of the cluster-discovery updates in Algorithm 1 ensures that, within at most D iterations after time τ , all agents agree on a common partition. Formally, there exists a partition $\bar{\mathcal{P}}$ such that

$$\mathcal{P}_{k,i} = \tilde{\mathcal{P}}_{k,i+1} = \bar{\mathcal{P}}, \quad \forall k \in \mathcal{K}, \quad \forall i \geq \tau + D. \quad (18)$$

The clusters of $\bar{\mathcal{P}}$ are exactly the connected components of the graph obtained by retaining only links between neighboring agents with identical true states, i.e., the graph in which each agent k is connected to only the agents in $\bar{\mathcal{N}}_k$.

Proof According to (16), the neighbor-identification rule in line 5 of Algorithm 1 selects exactly the same-state neighbors starting from time τ . Hence, at time τ , the merge step performed by agent k places agent k and all agents in $\bar{\mathcal{N}}_k$ in the same cluster of $\mathcal{P}_{k,\tau}$. Since the algorithm only merges clusters and never splits them, these agents remain in the same cluster at all later times. Therefore, after time τ , the intermediate partition update at agent k introduces no further change:

$$\tilde{\mathcal{P}}_{k,i} = \mathcal{P}_{k,i-1}, \quad \forall i \geq \tau + 1. \quad (19)$$

Consequently, for all $i \geq \tau + 1$,

$$\mathcal{P}_{k,i} = \bigvee_{\ell \in \bar{\mathcal{N}}_k} \mathcal{P}_{\ell,i-1}. \quad (20)$$

First, we show that after $m \geq 1$ iterations beyond time τ , the private partition held by agent k is the join of the private partitions held at time τ by all agents within m hops of k . Specifically, we prove by induction that

$$\mathcal{P}_{k,\tau+m} = \bigvee_{\ell: d_{\ell k} \leq m} \mathcal{P}_{\ell,\tau}, \quad \forall m \geq 1. \quad (21)$$

We see that (20) implies that the claim holds for $m = 1$. Suppose it holds for $m - 1$. Then, we have

$$\mathcal{P}_{k,\tau+m} \stackrel{(a)}{=} \bigvee_{q \in \bar{\mathcal{N}}_k} \mathcal{P}_{q,\tau+m-1} \stackrel{(b)}{=} \bigvee_{q \in \bar{\mathcal{N}}_k} \bigvee_{\ell: d_{\ell q} \leq m-1} \mathcal{P}_{\ell,\tau} \stackrel{(c)}{=} \bigvee_{\ell: d_{\ell k} \leq m} \mathcal{P}_{\ell,\tau}, \quad (22)$$

where (a) follows from (20) and (b) follows from the induction hypothesis. Moreover, (c) follows from the associativity of the partition join [35] and the fact that the agents within m hops of k are precisely those within $m - 1$ hops of some neighbor of k . This proves (21) by induction. Since D is the graph diameter, every agent lies within D hops of every other agent. Hence, (21) implies that for all $i \geq \tau + D$,

$$\mathcal{P}_{k,i} = \bigvee_{\ell \in \mathcal{K}} \mathcal{P}_{\ell,\tau}, \quad \forall k \in \mathcal{K}. \quad (23)$$

Thus, all agents agree on the same partition within at most D iterations after τ . It remains to identify the partition in (23). To this end, we equip the set of partitions of $\mathcal{K}$ with the standard refinement order $\preceq$ [35]: for two partitions $\mathcal{P}$ and $\mathcal{Q}$, we write

$$\mathcal{P} \preceq \mathcal{Q},$$

if every cluster of $\mathcal{P}$ is contained in some cluster of $\mathcal{Q}$. Thus, we say that $\mathcal{P}$ is finer than $\mathcal{Q}$, or equivalently, $\mathcal{Q}$ is coarser than $\mathcal{P}$. Under this order, the join $\mathcal{P} \vee \mathcal{Q}$ is the least upper bound of $\mathcal{P}$ and $\mathcal{Q}$. That is, among all partitions that are coarser than both $\mathcal{P}$ and $\mathcal{Q}$, the join $\mathcal{P} \vee \mathcal{Q}$ is the finest one [35]. This is analogous to the subset order for sets. Under $\subseteq$, the union $A \cup B$ is the smallest set containing both A and B . Under $\preceq$, the join $\mathcal{P} \vee \mathcal{Q}$ is the finest partition that is coarser than both $\mathcal{P}$ and $\mathcal{Q}$.

Let $\bar{\mathcal{G}}$ be the same-state subgraph, that is the graph obtained by retaining only links between neighboring agents with the same true state, and let $\bar{\mathcal{P}}$ be the partition of $\mathcal{K}$ representing the connected components of $\bar{\mathcal{G}}$. We first show that

$$\bigvee_{\ell \in \mathcal{K}} \mathcal{P}_{\ell,\tau} \preceq \bar{\mathcal{P}}. \quad (24)$$

Starting from the singleton initialization in (12), it is straightforward to verify by induction that, under condition (15), every cluster generated by the cluster-discovery procedure remains contained in a connected component of $\bar{\mathcal{G}}$. Specifically, the updates of Algorithm 1 only expand clusters in each step. Condition (15) ensures that a cluster can be expanded only by adding neighbors with the same true state. Hence, in every iteration, clusters may expand within a same-state connected component, but they can never merge two distinct same-state connected components. Therefore, for all $\ell \in \mathcal{K}$,

$$\mathcal{P}_{\ell,\tau} \preceq \bar{\mathcal{P}}, \quad (25)$$

which makes $\bar{\mathcal{P}}$ a common upper bound of the partitions $\mathcal{P}_{\ell,\tau}$ for $\ell \in \mathcal{K}$. Since their join is the least upper bound under the refinement order, it must be finer than any other common upper bound, which proves (24). Conversely, we now show that

$$\bar{\mathcal{P}} \preceq \bigvee_{\ell \in \mathcal{K}} \mathcal{P}_{\ell,\tau}. \quad (26)$$

Under condition (16), from time τ onward the neighbor-identification rule in line 5 of Algorithm 1 selects all same-state neighbors. Therefore, for every edge (q, r) of $\bar{\mathcal{G}}$, agents q and r are placed in the same cluster of the private partition $\mathcal{P}_{q,\tau}$, and hence also in the same cluster of the join

$$\bigvee_{\ell \in \mathcal{K}} \mathcal{P}_{\ell,\tau}. \quad (27)$$

Now consider any two agents in the same connected component of $\bar{\mathcal{G}}$. They are connected by a path of consecutive agents that are same-state neighbors. Since each two agents forming an edge along this path are contained in a cluster of the join in (27) and the join must place all agents on the path in one cluster, every connected component of $\bar{\mathcal{G}}$ is contained in a cluster of

$$\bigvee_{\ell \in \mathcal{K}} \mathcal{P}_{\ell, \tau},$$

which proves (26). Combining (24) and (26) gives

$$\bigvee_{\ell \in \mathcal{K}} \mathcal{P}_{\ell, \tau} = \bar{\mathcal{P}}, \quad (28)$$

and completes the the proof. $\square$

Theorem 1 is stated under the deterministic conditions in (15) and (16). These conditions can be justified probabilistically from the asymptotic behavior of the beliefs. In particular, a refined analysis shows that, for sufficiently small ε , there exists a large enough finite time τ such that the conditions in (15) and (16) hold with high probability.

The intuition follows from the same-neighbor identification step in line 5 of Algorithm 1. Agent k adds a neighbor ℓ to $\mathcal{N}_{k,i}$ only if both their local beliefs concentrate on the same hypothesis. If $s_k^* = s_\ell^*$, then both beliefs concentrate on the same true hypothesis, so this test is eventually satisfied. If $s_k^* \neq s_\ell^*$, then the two beliefs concentrate on different hypotheses, so for small ε the test is unlikely to be satisfied. Thus, a smaller ε reduces the probability of including neighbors with different true states, as required by (15), while a sufficiently long time horizon allows all same-state neighbors to be included, as required by (16).

Consequently, the recovered partition is only as reliable as the local sets $\mathcal{N}_{k,i}$. When these sets do not include neighbors with different states and eventually contain all same-state neighbors, Theorem 1 guarantees the convergence of the cluster-discovery procedure: all agents partition the network into same-state connected components.

4.2.2 Phase II: Network-wide State Inference.

The cluster-discovery phase stops after T iterations. Then, the *network-wide state inference* phase begins, during which each agent infers the hidden state of each cluster in its final partition. From Theorem 1, if one could identify a time τ that satisfies conditions (15) and (16), then choosing $T = \tau + D$ would guarantee that all agents have reached the same partition $\bar{\mathcal{P}}$ by the end of the clustering phase. In this ideal case, each agent can maintain one belief vector for each cluster in $\bar{\mathcal{P}}$. Then, in each network-wide state-inference iteration, the agents update their beliefs about the clusters using local observations and beliefs received from neighbors. Since all the agents agree on the same partition $\bar{\mathcal{P}}$, their cluster beliefs are consistent: each cluster belief held by an agent has a matching belief, for the same cluster, in the belief collection of every neighbor. Hence, each agent can combine beliefs received from neighbors, for each cluster, without ambiguity.

Technically, conditions (15) and (16) hold with high probability, for sufficiently small ε and sufficiently large time τ . In practice, agents may still hold different private partitions when the cluster-discovery phase is stopped at time T , and the subsequent network-wide state-inference phase must be well-defined in this scenario.

When neighboring agents stop clustering with different partitions, a mismatch can arise after beliefs are exchanged. An agent may transmit a belief associated with one of its clusters, but that belief may not match any cluster maintained by its neighbor. Discarding all such unmatched beliefs avoids ambiguity, but it is too conservative because useful information may be ignored.

OCA-GSL resolves the mismatch between cluster beliefs by separating the partition used for transmission from the partition used for belief updates. Specifically, before entering the network-wide state inference phase, agent k executes lines 7–8 of Algorithm 4 and freezes the two partitions obtained at the clustering horizon T :

$$\tilde{\mathcal{P}}_k \triangleq \tilde{\mathcal{P}}_{k,T}, \quad \mathcal{P}_k \triangleq \mathcal{P}_{k,T}.$$

The intermediate partition $\tilde{\mathcal{P}}_k$ determines what agent k transmits: one belief vector for each cluster in $\tilde{\mathcal{P}}_k$. The private partition $\mathcal{P}_k$ determines what agent k updates: one belief vector for each private cluster.

This strategy ensures that each transmitted belief can be assigned to a unique cluster in the receiving agent's partition. Specifically, consider two neighboring agents k and ℓ . During the final clustering update, agent k has already received $\tilde{\mathcal{P}}_\ell$ and used it, together with the intermediate partitions of its other neighbors, to form $\mathcal{P}_k$. Therefore, every cluster in $\tilde{\mathcal{P}}_\ell$ is contained in exactly one cluster of $\mathcal{P}_k$. To highlight the relevance of the intermediate partition $\tilde{\mathcal{P}}_\ell$, note that if agent ℓ sends a belief associated with a cluster in $\tilde{\mathcal{P}}_\ell$, then agent k can assign it to one cluster in $\mathcal{P}_k$, without ambiguity. This lets neighboring agents exchange cluster-level beliefs and combine them consistently, even when their final private partitions differ.

We now introduce the belief collections associated with these two partitions. Each cluster is labeled by its smallest-indexed agent. For any cluster $C \subseteq \mathcal{K}$, let

$$m(C) \triangleq \min_{\ell \in C} \ell. \quad (29)$$

Algorithm 3 Belief update step for agent k

- 1: **function** BELIEFUPDATESTEP($\tilde{\mathcal{P}}^{(k)}, \mathcal{S}^{(k)}, \mathcal{T}^{(k)}, \{\mu_{r,i-1}^{(k)}\}_{r \in \mathcal{S}^{(k)}}, \mathbf{x}_{k,i}$)
 - 2: **Local Bayesian update:**

$$\eta_{r,i}^{(k)}(\theta) \propto \begin{cases} \mu_{r,i-1}^{(k)}(\theta) L_k(\mathbf{x}_{k,i} | \theta), & r \text{ and } k \text{ belong to the same cluster of } \mathcal{P}_k, \\ \mu_{r,i-1}^{(k)}(\theta), & \text{otherwise,} \end{cases} \quad \forall \theta \in \Theta, r \in \mathcal{S}^{(k)}.$$
 - 3: **Prepare intermediate cluster beliefs for transmission:**

For each $r \in \mathcal{T}^{(k)}$, identify the unique cluster $C \in \mathcal{P}_k$ containing r , and set

$$\psi_{r,i}^{(k)}(\theta) = \eta_{m(C),i}^{(k)}(\theta), \quad \forall \theta \in \Theta.$$
 - 4: Transmit $\{\psi_{r,i}^{(k)} : r \in \mathcal{T}^{(k)}\}$ to all neighbors $\ell \in \mathcal{N}_k \setminus \{k\}$.
 - 5: Receive $\{\psi_{r,i}^{(\ell)} : r \in \mathcal{T}^{(\ell)}\}$ from all neighbors $\ell \in \mathcal{N}_k \setminus \{k\}$.
 - 6: **Private cluster belief update:**

$$\mu_{r,i}^{(k)}(\theta) \propto \begin{cases} \eta_{r,i}^{(k)}(\theta), & r \text{ and } k \text{ belong to the same cluster of } \mathcal{P}_k, \\ \prod_{\ell \in \mathcal{N}_k} [\psi_{r,i}^{(\ell)}(\theta)]^{a_{\ell k}}, & \text{otherwise,} \end{cases} \quad \forall \theta \in \Theta, r \in \mathcal{S}^{(k)}.$$
 - 7: **Agent-level belief mapping:**

For each $\ell \in \mathcal{K}$, identify the unique cluster $C \in \mathcal{P}_k$ containing ℓ , and set

$$\hat{\mu}_{\ell,i}^{(k)}(\theta) = \mu_{m(C),i}^{(k)}(\theta), \quad \forall \theta \in \Theta.$$
 - 8: **return** $\{\mu_{r,i}^{(k)}\}_{r \in \mathcal{S}^{(k)}}, \{\hat{\mu}_{\ell,i}^{(k)}\}_{\ell \in \mathcal{K}}$.
 - 9: **end function**
-

Agent k maintains one *intermediate cluster belief* for each cluster in its intermediate partition $\tilde{\mathcal{P}}_k$. The collection of intermediate cluster beliefs maintained by agent k at time i is denoted by

$$\psi_i^{(k)} \triangleq \{\psi_{r,i}^{(k)} : r \in \mathcal{T}^{(k)}\}, \quad (30)$$

where $\mathcal{T}^{(k)}$ is the set of labels of the clusters in $\tilde{\mathcal{P}}_k$, namely

$$\mathcal{T}^{(k)} \triangleq \{m(C) : C \in \tilde{\mathcal{P}}_k\}. \quad (31)$$

For every $r \in \mathcal{T}^{(k)}$, the corresponding intermediate cluster belief

$$\psi_{r,i}^{(k)} = [\psi_{r,i}^{(k)}(\theta_1), \dots, \psi_{r,i}^{(k)}(\theta_H)]^\top \quad (32)$$

is a probability distribution over Θ , where the entry $\psi_{r,i}^{(k)}(\theta)$ represents the confidence of agent k at time i that the cluster labeled by r has hidden state θ .

Similarly, agent k maintains one *private cluster belief* for each cluster in its private partition $\mathcal{P}_k$. The collection of private cluster beliefs maintained by agent k at time i is denoted by

$$\mu_i^{(k)} \triangleq \{\mu_{r,i}^{(k)} : r \in \mathcal{S}^{(k)}\}. \quad (33)$$

where $\mathcal{S}^{(k)}$ is the set of labels of the clusters in $\mathcal{P}_k$, namely

$$\mathcal{S}^{(k)} \triangleq \{m(C) : C \in \mathcal{P}_k\}. \quad (34)$$

For each $r \in \mathcal{S}^{(k)}$, the vector

$$\mu_{r,i}^{(k)} = [\mu_{r,i}^{(k)}(\theta_1), \dots, \mu_{r,i}^{(k)}(\theta_H)]^\top \quad (35)$$

is likewise a probability distribution over Θ . Its entry $\mu_{r,i}^{(k)}(\theta)$ quantifies the confidence of agent k at time i that the cluster labeled by r has hidden state θ .

The private cluster beliefs $\mu_0^{(k)}$ are initialized uniformly for all agents $k \in \mathcal{K}$. Then, these collections of beliefs are updated by repeatedly applying the steps in Algorithm 3. At each iteration $i \geq 1$, agent k proceeds in four main steps.

First, agent k incorporates its new observation $\mathbf{x}_{k,i}$. Since this observation is generated at agent k , it is informative only about the hidden state of the cluster containing agent k in $\mathcal{P}_k$. Therefore, agent k applies the Bayesian update in line 2 only to the private cluster belief associated with its own cluster. The beliefs associated with all other clusters in $\mathcal{P}_k$ are simply carried over from the previous iteration. The resulting beliefs are denoted by $\eta_{r,i}^{(k)}$, for $r \in \mathcal{S}^{(k)}$.

Second, agent k prepares the beliefs to transmit to its neighbors. At this point, agent k has one updated belief for each cluster in $\mathcal{P}_k$. However, it must transmit beliefs indexed by the clusters of $\tilde{\mathcal{P}}_k$. To do this, agent k identifies, for each cluster in $\tilde{\mathcal{P}}_k$, the cluster in $\mathcal{P}_k$ that contains it. Then, it assigns the updated belief of that cluster in $\mathcal{P}_k$ to the corresponding cluster in $\tilde{\mathcal{P}}_k$. This produces the intermediate cluster beliefs $\psi_{r,i}^{(k)}$ in line 3, one for each $r \in \mathcal{T}^{(k)}$. Agent k then transmits these intermediate beliefs to its neighbors and receives their corresponding intermediate beliefs, as specified in lines 4–5.

Third, after this belief exchange, agent k updates its private cluster beliefs. For its own cluster, agent k keeps the locally updated belief obtained from the Bayesian update step. For every other cluster in $\mathcal{P}_k$, agent k combines the corresponding intermediate beliefs received from its neighbors using the geometric pooling rule in line 6. This update is well-defined because each received belief is indexed by an intermediate partition that was used to form $\mathcal{P}_k$, so it can be associated with a unique cluster in $\mathcal{P}_k$.

Finally, agent k maps its private cluster beliefs back to individual agents, as specified in line 7 of Algorithm 3. Each agent is assigned the private belief associated with the cluster of $\mathcal{P}_k$ to which it belongs. Hence, all agents that agent k places in the same cluster inherit the same private cluster belief.

Algorithm 4 Cluster-aware social learning (CASL-executed by all agents $k \in \mathcal{K}$ in parallel)

Require: Hypothesis set Θ ; local likelihoods $\{L_k(\cdot \mid \theta)\}_{\theta \in \Theta}$; neighborhood $\mathcal{N}_k$; combination weights $\{a_{\ell k}\}_{\ell \in \mathcal{N}_k}$; clustering threshold ε ; clustering horizon T .

Initialization for cluster-discovery:

- 1: Set $\pi_{k,0}(\theta) \leftarrow \frac{1}{H}$ for all $\theta \in \Theta$.
- 2: Set $\mathcal{P}_{k,0} \leftarrow \{\{1\}, \{2\}, \dots, \{K\}\}$.

Phase I: cluster-discovery

- 3: **for** $i = 1, \dots, T$ **do**
- 4: Observe local measurement $\mathbf{x}_{k,i}$.
- 5: $(\pi_{k,i}, \tilde{\mathcal{P}}_{k,i}, \mathcal{P}_{k,i}) \leftarrow \text{CLUSTERDISCOVERYSTEP}(\pi_{k,i-1}, \mathcal{P}_{k,i-1}, \mathbf{x}_{k,i})$.
- 6: **end for**

Freeze discovered clusters:

- 7: Set $\tilde{\mathcal{P}}_k \leftarrow \tilde{\mathcal{P}}_{k,T}$; $\mathcal{P}_k \leftarrow \mathcal{P}_{k,T}$.
- 8: Set $\mathcal{T}^{(k)} \leftarrow \{m(C) : C \in \tilde{\mathcal{P}}_k\}$; $\mathcal{S}^{(k)} \leftarrow \{m(C) : C \in \mathcal{P}_k\}$.

Initialization for network-wide state inference:

- 9: Set $\mu_{r,0}^{(k)}(\theta) \leftarrow \frac{1}{H}$ for all $\theta \in \Theta$ and $r \in \mathcal{S}^{(k)}$.

Phase II: Network-wide state inference

- 11: **for** $i = 1, 2, \dots$ **do**
- 12: Observe local measurement $\mathbf{x}_{k,i}$.
- 13: $\{\mu_{r,i}^{(k)}\}_{r \in \mathcal{S}^{(k)}}, \{\hat{\mu}_{\ell,i}^{(k)}\}_{\ell \in \mathcal{K}} \leftarrow \text{BELIEFUPDATESTEP}(\tilde{\mathcal{P}}^{(k)}, \mathcal{P}^{(k)}, \mathcal{S}^{(k)}, \mathcal{T}^{(k)}, \{\mu_{r,i-1}^{(k)}\}_{r \in \mathcal{S}^{(k)}}, \mathbf{x}_{k,i})$

Network-wide state estimate:

$$\hat{s}_{\ell,i}^{(k)} = \arg \max_{\theta \in \Theta} \hat{\mu}_{\ell,i}^{(k)}(\theta), \quad \text{for all } \ell \in \mathcal{K}.$$

- 15: **end for**
-

The complete procedure is summarized in Algorithm 4. All agents run the algorithm in parallel and infer the network-wide state as in line 14: agent k estimates the hidden state of agent ℓ by maximizing $\hat{\mu}_{\ell,i}^{(k)}$, and collecting these estimates over all $\ell \in \mathcal{K}$ yields its estimate of the network-wide state s^* .

Theorem 2 (Correct network-wide state inference) *Suppose that, after cluster-discovery, no agent places two agents with different hidden states in the same cluster. Equivalently, for every agent $k \in \mathcal{K}$, every cluster C in its final*

private partition $\mathcal{P}_k$ contains agents with one common hidden state:

$$s_p^* = s_q^*, \quad \forall k \in \mathcal{K}, \forall C \in \mathcal{P}_k, \forall p, q \in C. \quad (36)$$

Then, under Assumptions 1-4, every agent recovers the correct network-wide state almost surely. That is, for all $k, \ell \in \mathcal{K}$,

$$\lim_{i \rightarrow \infty} \hat{\mu}_{\ell,i}^{(k)}(s_\ell^*) = 1 \quad \text{almost surely.} \quad (37)$$

Proof Fix any agents k and ℓ , and let r be the cluster label that agent k assigns to agent ℓ . By assumption, this cluster contains only agents with the same hidden state as ℓ . By following similar steps to the proof of [15, Lemma 2.2], we can show that this cluster belief converges almost surely to the truth. Since the agent-level belief $\hat{\mu}_{\ell,i}^{(k)}$ is exactly the belief of that cluster, it also converges almost surely to s_ℓ^* . $\square$

Theorem 2 states that, under condition (36), all agents recover the correct network-wide state. Notice that this condition holds for the partition $\bar{\mathcal{P}}$ of Theorem 1: if all agents recover the connected same-state components as the clusters, then no cluster contains agents with different hidden states. However, condition (36) does not hold only for $\bar{\mathcal{P}}$: an agent may have a partition that splits a connected same-state component into several smaller clusters and still satisfy condition (36).

This distinction is useful because the cluster-discovery phase stops at a prescribed finite horizon T . At that time, agents may not yet have reached the partition $\bar{\mathcal{P}}$, especially if information has not fully propagated through the network. Theorem 2 shows that this does not necessarily prevent correct network-wide state inference. Correct inference only requires that no final cluster mixes agents with different hidden states. The critical error is not splitting a true same-state component into several smaller clusters. Such splits increase the number of beliefs that must be maintained and exchanged, but each belief still represents a single hidden state. The critical error is placing agents with different hidden states in the same cluster, because then one belief vector is forced to represent incompatible states.

4.3 Experimental setup

We evaluate cluster-aware social learning in two settings. The first is a synthetic Gaussian experiment where the data is Gaussian and the observation model is known. The second uses real chemical gas-sensor measurements, with an unknown observation model. In both cases, agents use the cluster-aware social learning procedure described in Algorithms 1, 3, and 4.

In all experiments, the combination matrix $A = [a_{\ell k}]$ is chosen to assign equal weights to all agents in the neighborhood of agent k , including agent k itself.

4.3.1 Synthetic Gaussian Experiment

The synthetic experiment uses $K = 20$ agents and $H = 5$ possible hidden states, where the set of hypotheses

$$\Theta = \{0.1, 0.2, 0.3, 0.4, 0.5\}. \quad (38)$$

Each agent k has a fixed hidden state $s_k^* \in \Theta$. The hidden-state assignment is fixed as

$$(s_1^*, \dots, s_{20}^*) = (0.1, 0.2, 0.3, 0.4, 0.5, 0.1, 0.2, 0.3, 0.4, 0.5, \\ 0.1, 0.2, 0.3, 0.4, 0.5, 0.1, 0.2, 0.3, 0.4, 0.5). \quad (39)$$

The clusters identified by cluster-aware social learning are the connected components of the communication graph, that have the same hidden state. In the synthetic experiment, we consider a simple case in which all agents with the same hidden state form a single connected component, and therefore a single cluster. Hence, each cluster corresponds to exactly one hidden state, which simplifies the visualization and interpretation of the results. Specifically, we consider the 5 clusters

$$\{1, 6, 11, 16\}, \quad \{2, 7, 12, 17\}, \quad \{3, 8, 13, 18\}, \quad \{4, 9, 14, 19\}, \quad \{5, 10, 15, 20\}. \quad (40)$$

This one-to-one correspondence between clusters and hidden states is used solely to simplify the illustrations. In general, if agents with the same hidden state are disconnected, they form multiple connected components with the same state, each treated as a separate cluster.

The communication graph is a fixed-seed, random, undirected graph. To ensure that all agents with the same hidden state form a single connected component, we first connect agents within each group with the same hidden state, and then add links between the groups to connect the full network. Self-loops are included at every agent. The resulting graph is strongly connected and represented in Fig. 4.

At each time instant i , each agent k observes a scalar measurement

$$\mathbf{x}_{k,i} = \mathbf{s}_k^* + \boldsymbol{\xi}_{k,i}, \quad \boldsymbol{\xi}_{k,i} \sim \mathcal{N}(0, \sigma^2), \quad (41)$$

with $\sigma = 0.05$. The variables $\boldsymbol{\xi}_{k,i}$ are independent across agents, time, and Monte Carlo trials. Each agent k uses the Gaussian likelihood

$$L_k(x_{k,i} | \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{(x_{k,i} - \theta)^2}{2\sigma^2} \right\}, \quad \text{for all } \theta \in \Theta. \quad (42)$$

For every agent k , the initial local beliefs are uniform over the hypotheses,

$$\pi_{k,0}(\theta) = \frac{1}{H}, \quad \text{for all } \theta \in \Theta, \quad (43)$$

and the initial partitions are set to

$$\mathcal{P}_{k,0} = \{\{1\}, \{2\}, \dots, \{K\}\}. \quad (44)$$

To generate Figs. 5–9, we run Algorithm 4 on the network shown in Fig. 4 for $T = 100$ iterations. The cluster-discovery phase is run for $T = 50$ iterations with threshold $\varepsilon = 10^{-4}$. The discovered clusters are then frozen, and the network-wide state-inference phase is run until time T .

For the clustering threshold sweep in Figs. 10 and 11a, we use

$$T = 100, \quad T = 30, \quad \varepsilon \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.2\}. \quad (45)$$

For the clustering horizon sweep in Fig. 11b, we use

$$T = 100, \quad \varepsilon = 10^{-4}, \quad T \in \{20, 30, 40, 50\}. \quad (46)$$

The probability of error and average cluster count curves are estimated using $N_{\text{MC}} = 10000$ independent Monte Carlo trials. The graph, hidden-state assignment, and reference clusters are kept fixed across trials, while the observation streams are resampled independently.

4.3.2 Real Gas-sensor Experiment

The real-data experiment uses the UCI Twin Gas Sensor Arrays dataset [39], associated with the chemical sensor array measurements reported in [40]. The dataset contains measurements from five independent chemical sensing units, each equipped with eight metal-oxide gas sensors. To construct the agents, we split each sensing unit into two composite agents. The first agent uses the measurements from sensor channels $\{0, 2, 5, 7\}$, while the second agent uses the measurements from sensor channels $\{1, 3, 4, 6\}$. Thus, each agent receives observations from four sensor channels of a single sensing unit. Since there are five sensing units in total, this split produces $K = 5 \times 2 = 10$ agents.

Each sensing unit was exposed separately to each of the four gases: carbon monoxide, ethanol, ethylene, and methane. During each exposure, the sensor conductivities were recorded continuously. In our experiment, we therefore take the hypothesis set Θ to be the set of gas classes in the dataset:

$$\Theta = \{\text{CO, ethanol, ethylene, methane}\}. \quad (47)$$

In [40], the responses of the five chemical sensing units were measured using the same experimental protocol over a period of 22 days. On each testing day, one unit was exposed to the four gas types, and the sensor conductivities were recorded. Each unit was tested up to four times on different days. In our experiment, we use only the measurements from the first testing day available for each unit: day 4 for unit 1, day 1 for unit 2, day 2 for unit 3, day 3 for unit 4, and day 18 for unit 5.

Unlike in the synthetic setting, the likelihood models of the measurements are not available in closed form. We therefore need to construct local likelihood values from data. To this end, for each agent, we first transform its raw four-channel conductivity measurements into a stream of feature vectors, which we split into training and test sets. Then, we train a local classifier on the labeled feature vectors of each agent. During cluster-aware social learning, the outputs of the classifier on test feature vectors serve as a proxy for local likelihood values.

For each raw measurement trajectory, the exposure protocol in [40] consisted of three stages: 50 s of clean air to establish the baseline, 100 s of exposure to one gas at one concentration level, and 450 s of clean-air recovery. This gives a total duration of 600 s per trajectory. Sensor conductivities were recorded continuously at 100 Hz, resulting in 60000 raw conductivity samples per sensor channel.

In our experiment, we only use the response interval [50, 150] s, during which the sensors are exposed to the target gas. The initial interval [0, 50] s is used to estimate the baseline response of each sensor channel, while

the recovery interval after 150 s is discarded because it mainly reflects the return of the sensors to clean-air conditions. Thus, the observations used for learning are extracted from the 100 s gas-exposure interval.

Instead of using the raw conductivity samples directly, we convert each response trajectory of a sensor channel into a sequence of feature vectors. The response interval is divided into 20 non-overlapping windows of duration 5 s each. From each window, we extract a set of features that describe the sensor response. Specifically, we construct 22 features based on summary statistics, including the mean, standard deviation, minimum, maximum, median, first, and third quartiles. Since each composite agent contains four sensor channels, each window is represented by a feature vector of $4 \times 22 = 88$ components. Therefore, each raw measurement trajectory of an agent provides a sequence of $T = 20$ observations, with each observation being an 88-dimensional feature vector. This window-based representation reduces the size of the raw signal and minimizes the temporal dependence between consecutive measurements.

The dataset contains measurements for each gas type at ten concentration levels. Following the split used in [40], we train and test on disjoint concentration levels. The training set uses concentration levels $\{2, 4, 6, 8, 10\}$, while the test set uses concentration levels $\{1, 3, 5, 7, 9\}$. Within the training set, concentration level 6 is used for validation, and levels $\{2, 4, 8, 10\}$ are used for fitting the classifier. This split ensures that the likelihood streams used during social learning are generated from concentration levels that were not used to fit the local classifiers.

For each agent, we train a separate feedforward neural-network classifier to distinguish the four gas classes: CO, ethanol, ethylene, and methane. The classifier takes the agent’s 88-dimensional feature vector as input and outputs a probability vector over the four gas classes. The network has two hidden layers of sizes 32 and 16, ReLU nonlinearities, dropout probability 0.2, and a final softmax layer. The input features are standardized using a `StandardScaler` fitted only on the local fitting data; the same transformation is then applied to the validation and test samples. The network is trained using the AdamW optimizer with learning rate 5×10^{-4} , weight decay 10^{-3} , batch size 32, a maximum of 250 epochs, and early stopping with patience 25.

For each agent, the training set contains four gas types, five training concentration levels, and 20 feature vectors per gas-concentration pair. This gives $4 \times 5 \times 20 = 400$ training feature vectors before the validation split. The validation set contains $4 \times 20 = 80$ feature vectors from concentration level 6, and the remaining $4 \times 4 \times 20 = 320$ feature vectors are used for fitting. The test set contains four gas types, five test concentration levels, and 20 feature vectors per gas-concentration pair, giving $4 \times 5 \times 20 = 400$ test feature vectors per agent. After training, the classifier scores are used to construct the likelihood ratios for cluster-aware social learning as in [15, Chapter 12] and [41].

To validate cluster-aware social learning on this real gas-sensor data, each agent is assigned one gas type and one test concentration. The gas type determines the true local state s_k^* , while the test concentration determines which test trajectory provides the feature vector stream. The true gas-state assignment for agents $1, \dots, 10$ is

$$(s_1^*, \dots, s_{10}^*) = (\text{CO, ethanol, ethylene, ethanol, methane, methane, ethylene, methane, ethanol, ethylene}). \quad (48)$$

The selected test concentration levels for agents $1, \dots, 10$ are

$$(3, 5, 5, 5, 1, 1, 9, 7, 7, 9). \quad (49)$$

Similarly to the synthetic scenario, we consider a simple case in which all agents with the same hidden state form a single connected component, and therefore a single cluster. This is guaranteed by the communication graph in Fig. 12. Thus, the 4 clusters corresponding to the 4 gas types are

$$\{1\}, \quad \{2, 4, 9\}, \quad \{3, 7, 10\}, \quad \{5, 6, 8\}. \quad (50)$$

To generate Figs. 13–17, we run Algorithm 4 on the communication graph shown in Fig. 12. Each agent has a test observation stream of $T = 20$ feature vectors, one for each 5 s window of the gas-response interval. The cluster-discovery phase uses the first $T = 5$ feature vectors and threshold $\varepsilon = 10^{-3}$. After this phase, the discovered clusters are frozen. The remaining observations are then used to update the private cluster beliefs and recover the network-wide state estimates until the final time $T = 20$.

4.3.3 Evaluation Metrics

Cluster labels and beliefs.

During the cluster-discovery, we visualize the private partition of each agent k at time i through the label vector $\mathbf{c}_i^{(k)}$, defined in (4) and (5). We display these label vectors as heatmaps, where each entry at row k and column ℓ represents the value of the label $c_{\ell,i}^{(k)}$. These visualizations are shown in Fig. 5 for the synthetic experiment and Fig. 13 for the real gas-sensor experiment.

During the network-wide state inference phase, we visualize the private cluster beliefs maintained by each agent. For a discovered cluster label r , we plot the belief mass assigned by agent k to the correct hidden state

of that cluster, namely $\mu_{r,i}^{(k)}(s_r^*)$. These cluster-level belief trajectories and snapshots are shown in Figs. 6 and 7 for the synthetic experiment, and in Figs. 14 and 15 for the real gas-sensor experiment.

We also visualize the agent-level beliefs obtained by mapping the private cluster beliefs back to individual agents. For each observing agent k and target agent ℓ , we plot the belief assigned to the correct hidden state of agent ℓ , namely $\hat{\mu}_{\ell,i}^{(k)}(s_\ell^*)$. These agent-level belief snapshots are shown in Fig. 8 for the synthetic experiment and Fig. 16 for the real gas-sensor experiment.

Cluster count.

The number of clusters inferred by agent k at time i is given by $|\mathcal{P}_{k,i}|$. This quantity counts the number of distinct sets (i.e., clusters) in the private partition of agent k at time i . Thus, it determines the size of the cluster-level representation that would be used in the second phase if the cluster-discovery phase were stopped at time $T = i$. Figure 10a reports the Monte Carlo average of $|\mathcal{P}_{k,i}|$ for agent $k = 1$ as the time index i increases.

Clustering error probability.

We evaluate the clustering performance directly in terms of the partitions inferred by the agents. Let $\bar{\mathcal{P}}$ denote the target partition characterized in Theorem 1, whose clusters are the connected same-state components of the communication graph. The clustering error probability of agent k at time i is defined as

$$p_{k,i}^{\text{clust}} \triangleq \mathbb{P}[\mathcal{P}_{k,i} \neq \bar{\mathcal{P}}]. \quad (51)$$

Thus, a realization is counted as erroneous whenever the partition inferred by agent k differs from $\bar{\mathcal{P}}$ in any way, whether because a true cluster is split or because distinct clusters are incorrectly merged. Figure 10b reports the clustering probability $p_{k,i}^{\text{clust}}$, estimated using Monte Carlo simulations, for agent $k = 1$, and as the time index i progresses.

State-inference error probability.

The state-inference error probability for agent k at time i is defined as

$$p_{k,i}^{\text{state}} = \mathbb{P} \left[\exists \ell \in \{1, \dots, K\} : \arg \max_{\theta \in \Theta} \hat{\mu}_{\ell,i}^{(k)}(\theta) \neq s_\ell^* \right]. \quad (52)$$

This metric evaluates network-wide state recovery. It is stricter than local accuracy, since a realization is counted as erroneous if any agent in the network is assigned an incorrect state. Figure 11 reports the state-inference probability $p_{k,i}^{\text{state}}$, estimated using Monte Carlo simulations, for agent $k = 1$, and as the time index i increases.

4.4 Implementation and Reproducibility

All simulations were implemented in Python using the standalone project `npj.simulations`. The main notebooks were `simulation_synthetic_data.ipynb`, `simulation_real_data.ipynb`, and `Plots.ipynb`. The corresponding scripts are `main_synthetic_data.py` and `main_real_data.py`. The software environment used Python 3.12.2, NumPy 1.26.4, pandas 2.2.2, Matplotlib 3.9.2, NetworkX 3.3, scikit-learn 1.5.1, PyTorch 2.11.0, SciPy 1.13.1, IPython 8.37.0, and Jupyter Core 5.7.2.

The synthetic random seeds are: graph seed 777 and Monte Carlo observation seed 20260601. The real-data random seeds are: graph seed 123, hidden gas-state assignment seed 123, test-concentration assignment seed 456, classifier seed $123 + k$ for agent k .

Data availability

The synthetic data used in this study were generated from the Gaussian observation model described in the Methods section. The real-data experiments use the publicly available Twin Gas Sensor Arrays dataset associated with the chemical sensing measurements reported in [40]. The dataset is available from the UCI Machine Learning Repository at <https://archive.ics.uci.edu/dataset/361/twin+gas+sensor+arrays>.

Code availability

The Python code used to run the synthetic and real-data experiments, train the local classifiers, and generate the figures will be made publicly available at [\(repository-url\)](#) upon publication.

Competing interests

The authors declare no competing interests.

References

- [1] He, H., Schäfer, B. & Beck, C. Spatial heterogeneity of air pollution statistics in europe. *Scientific Reports* **12**, 12215 (2022).
- [2] Feinberg, S. N. *et al.* Examining spatiotemporal variability of urban particulate matter and application of high-time resolution data from a network of low-cost air pollution sensors. *Atmospheric Environment* **213**, 579–584 (2019).
- [3] Mukherjee, A. *et al.* Measuring spatial and temporal PM_{2.5} variations in sacramento, california, communities using a network of low-cost sensors. *Sensors* **19**, 4701 (2019).
- [4] Zhang, P., Nevat, I., Peters, G. W., Septier, F. & Osborne, M. A. Spatial field reconstruction and sensor selection in heterogeneous sensor networks with stochastic energy harvesting. *IEEE Transactions on Signal Processing* **66**, 2245–2257 (2018).
- [5] Filippopolitis, A., Hey, L., Loukas, G., Gelenbe, E. & Timotheou, S. Emergency response simulation using wireless sensor networks. *International Conference on Ambient Media and Systems* 1–7 (2008).
- [6] George, S. M. *et al.* Distressnet: A wireless ad hoc and sensor network architecture for situation management in disaster response. *IEEE Communications Magazine* **48**, 128–136 (2010).
- [7] Cattivelli, F. S. & Sayed, A. H. Diffusion LMS strategies for distributed estimation. *IEEE Transactions on Signal Processing* **58**, 1035–1048 (2010).
- [8] Olfati-Saber, R. Distributed kalman filtering for sensor networks. *Proc. IEEE Conference on Decision and Control* 5492–5498 (2007).
- [9] Kar, S. & Moura, J. M. Consensus + innovations distributed inference over networks: cooperation and sensing in networked systems. *IEEE Signal Processing Magazine* **30**, 99–109 (2013).
- [10] Sayed, A. H. Adaptation, learning, and optimization over networks. *Found. Trends Mach. Learn.* **7**, 311–801 (2014).
- [11] Viswanathan, R. & Varshney, P. Distributed detection with multiple sensors part i. fundamentals. *Proceedings of the IEEE* **85**, 54–63 (1997).
- [12] Zhao, X. & Sayed, A. H. Learning over social networks via diffusion adaptation. *Proc. Asilomar Conf. Signals, Syst. and Comput.* 709–713 (2012).
- [13] Lalitha, A., Javidi, T. & Sarwate, A. D. Social learning and distributed hypothesis testing. *IEEE Trans. Inf. Theory* **64**, 6161–6179 (2018).
- [14] Nedić, A., Olshevsky, A. & Uribe, C. A. Fast convergence rates for distributed non-bayesian learning. *IEEE Trans. Autom. Control* **62**, 5538–5553 (2017).
- [15] Matta, V., Bordignon, V. & Sayed, A. H. *Social Learning: Opinion Formation and Decision-Making over Graphs* (NOW Book Series on Information and Learning Sciences, 2024).
- [16] Jadbabaie, A., Molavi, P., Sandroni, A. & Tahbaz-Salehi, A. Non-Bayesian social learning. *Games Econ. Behav.* **76**, 210–225 (2012).
- [17] Pooya, M., Alireza, T.-S. & Ali, J. A theory of non-Bayesian social learning. *Econometrica* **86**, 445–490 (2018).
- [18] Salami, H., Ying, B. & Sayed, A. H. Social learning over weakly connected graphs. *IEEE Trans. Signal and Inf. Process. over Networks* **3**, 222–238 (2017).
- [19] Molavi, P., Rad, K. R., Tahbaz-Salehi, A. & Jadbabaie, A. On consensus and exponentially fast social learning. *Proc. Amer. Control Conf. (ACC)* 2165–2170 (2012).
- [20] Bordignon, V., Matta, V. & Sayed, A. H. Socially intelligent networks: A framework for decision making over graphs. *IEEE Signal Processing Magazine* **41**, 20–39 (2024).

- [21] Krishnamurthy, V. & Poor, H. V. Social learning and Bayesian games in multiagent signal processing: How do local and global decision makers interact? *IEEE Signal Process. Mag.* **30**, 43–57 (2013).
- [22] Plata-Chaves, J., Bogdanović, N. & Berberidis, K. Distributed diffusion-based LMS for node-specific adaptive parameter estimation. *IEEE Transactions on Signal Processing* **63**, 3448–3460 (2015).
- [23] Plata-Chaves, J., Bertrand, A., Moonen, M., Theodoridis, S. & Zoubir, A. M. Heterogeneous and multi-task wireless sensor networks: Algorithms, applications, and challenges. *IEEE Journal of Selected Topics in Signal Processing* **11**, 450–465 (2017).
- [24] Bertrand, A. & Moonen, M. Distributed adaptive node-specific signal estimation in fully connected sensor networks—part I: Sequential node updating. *IEEE Transactions on Signal Processing* **58**, 5277–5291 (2010).
- [25] Bertrand, A. & Moonen, M. Distributed adaptive node-specific signal estimation in fully connected sensor networks—part II: Simultaneous and asynchronous node updating. *IEEE Transactions on Signal Processing* **58**, 5292–5306 (2010).
- [26] Ntemos, K., Bordignon, V., Vlaski, S. & Sayed, A. H. Self-aware social learning over graphs. *IEEE Trans. Inf. Theory* **69**, 5299–5317 (2023).
- [27] Shumovskaia, V., Kayaalp, M. & Sayed, A. H. Social learning in community structured graphs. *IEEE Trans. Signal Process.* **72**, 2812–2826 (2024).
- [28] Khammassi, M., Matta, V. & Sayed, A. H. Multivariate belief formation for distributed multitask learning. *European Signal Processing Conference (EUSIPCO)* 1173–1177 (2025).
- [29] Newman, M. E. J. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* **103**, 8577–8582 (2006).
- [30] Fortunato, S. Community detection in graphs. *Physics Reports* **486**, 75–174 (2010).
- [31] Holland, P. W., Laskey, K. B. & Leinhardt, S. Stochastic blockmodels: First steps. *Social Networks* **5**, 109–137 (1983).
- [32] Lee, C. & Wilkinson, D. J. A review of stochastic block models and extensions for graph clustering. *Applied Network Science* **4**, 122 (2019).
- [33] Rohe, K., Chatterjee, S. & Yu, B. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics* **39**, 1878–1915 (2011).
- [34] Sayed, A. H. *Inference and Learning from Data* (Cambridge University Press, 2022).
- [35] Grätzer, G. *Lattice Theory: Foundation* (Birkhäuser Basel, 2011).
- [36] Abiad, A., Hojny, C. & Zeijlemaker, S. Characterizing and computing weight-equitable partitions of graphs. *Linear Algebra and its Applications* **645**, 30–51 (2022).
- [37] Tarjan, R. E. Efficiency of a good but not linear set union algorithm. *Journal of the ACM* **22**, 215–225 (1975).
- [38] Tarjan, R. Depth-first search and linear graph algorithms. *SIAM Journal on Computing* **1**, 146–160 (1972).
- [39] Fonollosa, J. Twin gas sensor arrays. UCI Machine Learning Repository (2016). DOI: <https://doi.org/10.24432/C5MW3K>.
- [40] Fonollosa, J., Fernández, L., Gutiérrez-Gálvez, A., Huerta, R. & Marco, S. Calibration transfer and drift counteraction in chemical sensor arrays using direct standardization. *Sensors and Actuators B: Chemical* **236**, 1044–1053 (2016).
- [41] Hu, P., Bordignon, V., Kayaalp, M. & Sayed, A. H. Non-asymptotic performance of social machine learning under limited data. *Signal Processing* **230**, 109849 (2025).