Learning Data Dependency with Communication Cost

Hyeryung Jang
Dept. of Informatics,
King's College London, UK
hyeryung.jang@kcl.ac.uk

HyungSeok Song
Dept. of Electrical Engineering,
KAIST, South Korea
hssong@lanada.kaist.ac.kr

Yung Yi
Dept. of Electrical Engineering,
KAIST, South Korea
yiyung@kaist.edu

ABSTRACT

In this paper, we consider the problem of recovering a graph that represents the statistical data dependency among nodes for a set of data samples generated by nodes, which provides the basic structure to perform an inference task, such as MAP (maximum a posteriori). This problem is referred to as structure learning. When nodes are spatially separated in different locations, running an inference algorithm requires a non-negligible amount of message passing, incurring some communication cost. We inevitably have the trade-off between the accuracy of structure learning and the cost we need to pay to perform a given message-passing based inference task because the learnt edge structures of data dependency and physical connectivity graph are often highly different. In this paper, we formalize this trade-off in an optimization problem which outputs the data dependency graph that jointly considers learning accuracy and message-passing cost. We focus on a distributed MAP as the target inference task due to its popularity, and consider two different implementations, **ASYNC-MAP** and **SYNC-MAP** that have different message-passing mechanisms and thus different cost structures. In **ASYNC-MAP**, we propose a polynomial time learning algorithm that is optimal, motivated by the problem of finding a maximum weight spanning tree. In **SYNC-MAP**, we first prove that it is NP-hard and propose a greedy heuristic. For both implementations, we then quantify how the probability that the resulting data graphs from those learning algorithms differ from the ideal data graph decays as the number of data samples grows, using the large deviation principle, where the decaying rate is characterized by some topological structures of both original data dependency and physical connectivity graphs as well as the degree of the trade-off, which provides some guideline on how many samples are necessary to obtain a certain learning accuracy. We validate our theoretical findings through extensive simulations, which confirm that it has a good match.

CCS CONCEPTS

• **Computing methodologies** → **Learning in probabilistic graphical models**; • **Mathematics of computing** → *Probabilistic inference problems*; • **Theory of computation** → *Sample complexity and generalization bounds*;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Mobihoc '18, June 26–29, 2018, Los Angeles, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5770-8/18/06…$15.00
https://doi.org/10.1145/3209582.3209600

KEYWORDS

Graph structure learning, Distributed inference, Sample complexity

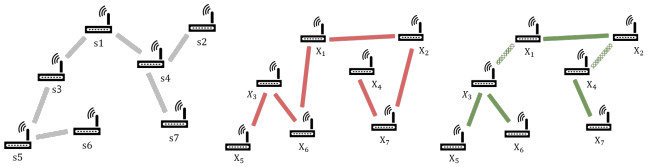
ACM Reference Format:
Hyeryung Jang, HyungSeok Song, and Yung Yi. 2018. Learning Data Dependency with Communication Cost . In *Mobihoc '18: The Nineteenth International Symposium on Mobile Ad Hoc Networking and Computing, June 26–29, 2018, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209582.3209600

1 INTRODUCTION

In many online/offline systems with spatially-separated agents (or nodes), a variety of applications involve distributed in-network statistical inference tasks, which have been widely studied, exploiting given knowledge of statistical dependencies among agents. As one example, in sensor networks with multiple targets, each sensor node measures the target-specific information in its coverage area (*e.g.*, position, direction, distance), which further has a correlation among sensors. One well-recognized inference problem is a data association which determines the correct match between measurements of sensors and target tracks by maximum a posteriori (MAP) estimation that is executed in a distributed fashion by exchanging some information messages. Other examples include target tracking, and detection/estimation in sensor networks [23, 27, 29, 34] and de-anonymization, rumor/infection propagation in social networks [2, 11, 14, 19].

To solve these distributed in-network inference problems, it is of crucial importance to understand how data from nodes are interdependent. To that end, a notion of the *graphical model* has been one of the powerful frameworks in machine learning for a succinct modeling of the statistical uncertainty, where each node in the graphical models corresponds to a random variable and each edge specifies the statistical dependency between random variables. A wide variety of scalable inference algorithms on graphical models via message-passing have been developed, of which examples include belief propagation (BP) or max-product with a certain degree of convergence and accuracy guarantees [15, 24, 31, 32]. This graphical model, which we also call data dependency graph or simply data graph throughout this paper, is not given a priori, and it should be learnt only by using a given set of data samples from nodes. This problem, referred to as *graph learning* or *structure learning* [1, 13, 21, 25], has been an active research topic in statistical machine learning.

In this paper, for a collection of n data sample vectors generated by nodes, we study a problem of graph learning, which also considers the communication cost incurred by the distributed in-network inference algorithm being applied to the learnt data graph. Physical communication cost often becomes a critical issue, for example, exerting a significant impact on the lifetime of networked sensors. Clearly, there exists a trade-off between the amount of incurred cost



(a) Physical graph (b) Data graph (c) Cost-efficient data graph

Figure 1: Network graphs with 7 sensors. (a) physical connectivity, (b) exact statistical dependency graph called data graph, (c) Data graph considering communication cost between nodes.

and the learning accuracy of the data graph. Figure 1(a) illustrates the physical connection among 7 sensors, which differs from the exact data dependency graph in Figure 1(b). The sensor nodes $s1$ and $s6$ have non-negligible data dependency, requiring message-passing when performing inference, but they are three hops away from each other, incurring a large amount of communication cost. In this case, one may want to sacrifice the estimation accuracy a little bit and reduce communication cost by utilizing the data graph as shown in Figure 1(c). As done in many prior works on graph learning [3, 8, 9, 21], we restrict our attention to tree-structured data graphs due to its simplicity, yet a large degree of expressive powers and other benefits, *e.g.*, some inference algorithms such as BP over tree-structured data graphs become optimal.

We now summarize our contributions in what follows:

(a) We first formulate an optimization problem of learning data graph, having as an objective function the weighted sum of learning accuracy and the amount of cost that will be incurred by a distributed inference algorithm. Out of many possible inference algorithms, we consider the maximum a posteriori (MAP) estimator that is popular for many inference tasks, and two versions for the MAP implementation: (i) asynchronous and (ii) synchronous, which we call **ASYNC-MAP** and **SYNC-MAP**. These implementations have different patterns of passing messages, thus leading to different forms of communication costs, being useful to understand how distributed algorithms' cost affect the resulting data dependency graph.

(b) Next, for **ASYNC-MAP** we develop a polynomial-time algorithm to find an optimal (cost-efficient) data graph that corresponds to simply finding a maximum weight spanning tree. This simplicity stems from the cost structure of **ASYNC-MAP** that is characterized only by the sum of all 'localized' edge costs. Being in sharp contrast to **ASYNC-MAP**, for **SYNC-MAP** we first prove that it is computationally intractable (*i.e.*, NP-hard) in terms of the number of nodes, by reducing it to the problem of the Exact Cover by 3-sets. The hardness is due to the fact that the cost structure of **SYNC-MAP** depends on the diameter of the resulting tree which is the 'global' information involving the entire topology. As a practical solution, we propose a polynomial-time greedy heuristic to recover a sub-optimal, but cost-efficient data graph.

(c) Finally, for both **ASYNC-MAP** and **SYNC-MAP**, we quantify how the probability that the resulting (cost-efficient) data graph for a finite number of n samples differs from the ideal data graph decays as n increases, using the large deviation principle (LDP), as a form of $\exp(-n \cdot K)$. The error exponent K is characterized for each of **ASYNC-MAP** and **SYNC-MAP** by some topological information of physical/data graphs, cost structure for both inference mechanisms, and the degree of the trade-off. We validate our theoretical findings through simulations over a 20-node graph for a variety of scenarios and show their good match with the simulation results.

To validate our theoretical results, we perform numerical simulations a pair of physical and data graphs with 20 nodes, where we quantitatively analyze (i) how estimating a data graph considering communication cost affects the resulting estimation for various values of trade-off parameters between inference accuracy and cost, (ii) how the estimation error decays as the same size increases.

1.1 Related Work

A variety of applications which involve distributed in-network statistical inference tasks among spatially inter-connected agents or sensors have been widely studied in many online/offline systems. In sensor networks, where the knowledge of statistical dependencies among sensed data is given, the tasks of target tracking [7, 22, 33], detection [6], parameter estimation [16, 27] are the examples, see [29] for a survey. In social networks, where the underlying social phenomenon of interest such as voting models, rumor/opinion propagation [2] evolves over a given social interaction graph, the inference tasks of distributed consensus-based estimation [19], de-anonymization of community-structured social network [14] and distributed observability [11] are studied.Message-passing has manifested as an efficient procedure for inference over graphical models that provide the framework of succinct model of the statistical uncertainty of multi-agents. Examples include belief propagation (BP) [24], max-product [15, 32] and references therein. They are known to be exact and efficient when the underlying graphical model is a tree [24, 31]. Recent research progress has been made for scalable message-passing for general graphs, *e.g.*, junction tree [30] and graphs with loops [17].

In the area of structure learning, several algorithms have been proposed in the literatures to recover the statistical dependencies from a set of data samples [1, 13, 21, 25]. It is known that the exact structure learning for general graphical models is NP-hard. The research of structure learning for special graphical models includes: maximum likelihood estimation (MLE) [8, 21] for tree graphs, ℓ_1 regularized MLE for binary undirected graphs [25], convexified MLE for Gaussian graphical models, known as Lasso [13]. Theoretical guarantees for the learning accuracy have been established as the number of data samples, *e.g.*, on tree graph [28], on binary undirected graphs [25], on a class of Ising model [4], or on Bayesian network [36]. Our work differs from all of the above works in that we consider physical communication cost incurred by some target inference algorithms when learning the data dependency graph.

There exists an array of work that addresses the trade-off between inference quality and cost in running distributed in-network inference on the known data graph, which are summarized as two directions: *(i)* developing novel inference algorithms with less communication of messages or *(ii)* constructing a new graphical model upon which the existing distributed in-network inference algorithms are performed with less communication resources. In *(i)*,

the need of conserving resources requires to propose new message-passing schemes where the messages are compressed by allowing some approximation error in message values [10, 17, 20, 22], and/or some messages are censored (*i.e.*, not to be transmitted) [7]. In *(ii)*, most of the related works focused on constructing a junction tree that minimizes the inference cost [23], building a data dependency structure upon which message-passing is run energy-efficiently, where the communications among all agents are assumed to be done in one-hop [34], or optimizing the data dependency structure formulated by a multi-objective problem of inference quality and energy, assuming that the exact statistical dependencies are given as a complete graph [35]. While the main interest of this area has been focused on characterizing the desirable dependency structure for given complete knowledge of accurate data dependencies, our work is motivated by the practical situation where one can just be able to observe a finite number of data sample vectors of nodes, which do not provide such a complete knowledge. Therefore, our interest lies in *learning* the desirable data dependencies from a finite number of data samples.

2 MODEL AND PRELIMINARY

2.1 Model

Physical graph. We consider a (connected) physical network $G = (V, E_P)$ with a set of d nodes V and links E_P, where each node corresponds to an agent such as a sensor or an individual, and each link corresponds to a physical connectivity between two nodes. For example, in sensor networks, when nodes have wireless radios, then each link is established when two corresponding nodes over the link reach each other within each radio's communication range.

Data samples. Each node $i \in V$ generates a binary data, denoted by $x_i \in \mathcal{X} := \{0, 1\}$[1], where we denote by $\boldsymbol{x} = [x_i]_{i \in V}$ the data vector of all nodes, or simply a *sample*, *e.g.*, target locations measured by all sensors. The underlying statistical uncertainty of samples can be represented by a joint distribution $P(\boldsymbol{x})$ of a random vector $X := [X_i]_{i \in V} \in \mathcal{X}^d$, called *data distribution,* where each random variable X_i is associated to each node $i \in V$. We often collect n multiple samples ($\boldsymbol{x}^{1:n} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n\}$) in order to infer what happens in the network by understanding the inter-dependence of data generated by nodes. For instance, when sensors measure the target location, then we infer the underlying statistical correlation among sensors from the observed samples, to estimate the true target location.

Data graph via graphical model. The underlying statistical dependency is often understood by the framework of *graphical model*, which has been a popular tool for modeling uncertainty by a graph structure, where each node corresponds to a random variable and each edge captures the probabilistic interaction between nodes. In particular, we model the data distribution $P(\boldsymbol{x})$ as an undirected graph $T = (V, E_D)$, which we call *data graph*, which consists of the same set V of nodes as that in the physical graph and nodes' statistical dependencies captured by an edge structure E_D as: any two non-adjacent random variables are conditionally independent given all other variables, *i.e.*, for any $(i, j) \notin E_D$,

$$P\left(x_i, x_j \mid x_{V\setminus\{i,j\}}\right) = P\left(x_i \mid x_{V\setminus\{i,j\}}\right) \cdot P\left(x_j \mid x_{V\setminus\{i,j\}}\right). \quad (1)$$

In this paper, we limit our focus on the tree-structured data graph (thus simply data tree), for which let $\mathcal{T}$ and $\mathcal{P}(\mathcal{X}^d)$ be set of all spanning trees and set of all tree data distributions over V, respectively, *i.e.*, we assume $T \in \mathcal{T}$ and $P \in \mathcal{P}(\mathcal{X}^d)$. Tree data graph is a class of graphical models that has received considerable attention in literatures [3, 8], since it possesses the following factorization property:

$$P(\boldsymbol{x}) = \prod_{i \in V} P_i(x_i) \prod_{(i,j) \in E_D} \frac{P_{i,j}(x_i, x_j)}{P_i(x_i) P_j(x_j)}, \quad (2)$$

where P_i and $P_{i,j}$ are the marginals on node $i \in V$ and edge $(i, j) \in E_D$, respectively. Tree-structured data graph is known to strike a good balance between the expressive power and the computational tractability. In particular, the distribution P in (2) is completely specified only by the set of edges E_D and their pairwise marginals. Thus, if P has the factorization property as in (2), in other words, if $P \in \mathcal{P}(\mathcal{X}^d)$, there exists a unique tree $T = T(P)$ corresponding to P. To abuse the notation, we henceforth denote by $T(P)$ the unique data tree of a tree distribution P. Figure 1 shows an example of the physical graph and two data graphs with 7 nodes.

2.2 Goal: Cost-efficient Learning of Data Graph

Learning data graph: What and why? To understand the underlying data dependency (2), it is enough to learn the structure of data graph E_D from the observed samples, which is known as the problem of *(data graph) structure learning*. Formally, when we are given a set of i.i.d. n samples $\boldsymbol{x}^{1:n}$ generated from an unknown (tree) data distribution $P \in \mathcal{P}(\mathcal{X}^d)$ on a data tree T, a *structure learning* algorithm is a (possibly randomized) map ϕ defined by:

$$\phi : (\mathcal{X}^d)^n \mapsto \mathcal{T}.$$

The quality of this algorithm $\hat{T} = \phi(\boldsymbol{x}^{1:n})$ is evaluated by how "close" $\hat{T}$ is to the original data graph T.

Distributed inference on data graph. One of the practical goals of estimating the data tree given a set of data samples is to perform an inference task based on T. Thus, in many applications, primary interests are not focused on data itself but rather on how to exploit the data dependency for reliable decision making, such as target tracking, detection, estimation in sensor networks and/or social networks, which involves statistical inference about the networks described by a data graph. One example of inference tasks is the MAP (maximum a posteriori) based estimation. Distributed in-network inference has been widely studied with the help of various distributed algorithms on graphical models using *message-passing*. In particular, for a specific inference problem, a message between two nodes contains the information on influence that one node exerts on another, which is obtained based on the value contained in neighboring messages over an estimated data graph $\hat{T}$. One critical issue of message-passing based inference algorithm is that *messages are often passed along the multi-hop path on the physical graph G*, which incurs some amount of *communication cost*. Then, assuming that some inference algorithm would be run for the estimated data graph $\hat{T}$, such a data graph learning must have the trade-off

[1] We assume a binary data for simplicity, and our results are readily extended to any finite set $\mathcal{X}$.

between the accuracy of the learnt graph (*i.e.*, how close the learnt graph is to the original data graph) and the communication cost generated by performing the distributed inference.

Goal: Cost-efficient data graph learning. Given an observed samples $\boldsymbol{x}^{1:n}$ from the unknown data distribution P, our objective is to estimate a cost-efficient data tree, which captures the trade-off between *(i) inference accuracy* and *(ii) communication cost for inference*. For tree distributions, finding a distribution naturally gives rise to the corresponding data tree, as mentioned earlier. Thus, it is natural to find the tree distribution $\hat{Q}^\star(n) = \hat{Q}^\star(\boldsymbol{x}^{1:n}, G, \Pi, \gamma)$ that is the solution of the following optimization problem: for a constant parameter $\gamma \geq 0$ and a fixed inference algorithm Π,

$$\textbf{CDG}(\boldsymbol{n})\ : \hat{Q}^\star(n) = \operatorname*{arg\,min}_{Q \in \mathcal{P}(\mathcal{X}^d)} D(\hat{P}(\boldsymbol{x}^{1:n}) \parallel Q) + \gamma C\big(T(Q); G, \Pi\big), \tag{3}$$

where $\hat{P}(\boldsymbol{x}^{1:n}) := \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}\{\boldsymbol{x}^k = \boldsymbol{x}\}$ is the empirical distribution of $\boldsymbol{x}^{1:n}$, $D(\cdot \parallel \cdot)$ is some distance metric between two distributions, and $C(T(Q); G, \Pi)$ is the communication cost paid by running an inference algorithm Π with respect to the data tree $T(Q)$ over the physical graph G. Recall that $T(Q)$ is the data tree for the tree distribution Q. The value of γ parameterizes how much we prioritize the communication cost compared to the inference accuracy $D(\hat{P}(\boldsymbol{x}^{1:n}) \parallel Q)$. Note that as $n \to \infty$, $\hat{P}(\boldsymbol{x}^{1:n})$ converges to the original data distribution P, which requires to solve **CDG(∞)**.

Then, this paper aims at answering the following two questions:

(a) What are good data-tree learning algorithms that compute $T(\hat{Q}^\star(n))$ by solving **CDG(*n*)**? In Section 3, we consider the MAP estimator as an applied inference algorithm, and their two implementations having different cost functions, for which we propose two cost-efficient learning algorithms.

(b) How fast does $\hat{Q}^\star(n)$ converge to $\hat{Q}^\star(\infty)$ as the number of samples n grows? We use the large deviation principle (LDP) to characterize the decaying rate of the probability that $T(\hat{Q}^\star(n)) \neq T(\hat{Q}^\star(\infty))$ for two different MAP implementations in Section 3.

In this paper, we use the popular KL divergence as a distance metric $D(\cdot||\cdot)$ for inference accuracy, denoted by D_{KL}, where for two distributions P and Q, $D_{\mathrm{KL}}(P \parallel Q) := \sum_{\boldsymbol{x} \in \mathcal{X}^d} P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}$. For notional simplicity, we simply denote by $Q^\star := \hat{Q}^\star(\infty)$ the solution of **CDG(∞)** throughout this paper.

3 COST-EFFICIENT DATA GRAPH LEARNING ALGORITHMS

In this paper, out of many possible inference tasks, we consider the maximum a posteriori (MAP) estimation, which is popularly applied in many applications such as data association for a multi-target tracking problem in sensor networks, community-structured social network de-anonymization problem in social networks [14].

3.1 Distributed MAP and Cost

Distributed MAP on tree-structured data graph. The MAP estimator of some tree distribution $Q \in \mathcal{P}(\mathcal{X}^d)$ on its associated data tree $T(Q) = (V, E_Q)$ is given by:

$$\boldsymbol{x}^{\mathrm{MAP}} := \operatorname*{arg\,max}_{\boldsymbol{x} \in \mathcal{X}^d} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E_Q} \psi_{i,j}(x_i, x_j), \tag{4}$$

where we use $\psi_i(x_i) = Q_i(x_i)$ and $\psi_{i,j}(x_i, x_j) = \frac{Q_{i,j}(x_i, x_j)}{Q_i(x_i)Q_j(x_j)}$ for simplicity. A standard message-passing algorithm for the distributed MAP is a max-product algorithm, which defines a message $m_{i\to j}^{(t)}(\cdot)$ from node i to j at t-th iteration with $(i,j) \in E_Q$. Each node exchanges messages with their neighbors on the data tree $T(Q)$, and these messages are updated over time in an iterative fashion by the following rule: at t-th iteration,

$$m_{i\to j}^{(t+1)}(x_j) := \kappa \cdot \max_{x_i \in \mathcal{X}} \Big[\psi_i(x_i)\psi_{i,j}(x_i, x_j) \prod_{k \in \mathcal{N}(i)\setminus\{j\}} m_{k\to i}^{(t)}(x_i)\Big], \tag{5}$$

with the normalizing constant κ to make the sum of all message values be 1, and $\mathcal{N}(i)$ denotes the neighboring nodes of i.

Communication cost of distributed MAP. The communication cost of MAP is paid, depending on the actual protocol that specifies how to schedule message-passing procedures. Two natural message-passing protocols studied in literatures are: (a) asynchronous depth-first (unicast) update [12] and (b) synchronous (broadcast) parallel update [24]. Both protocols for a tree distribution Q with its data tree $T(Q)$ have been shown to be consistent in that the message update (5) converges to a unique fixed point $\{m_{i\to j}^*, m_{j\to i}^*\}_{(i,j)\in E_Q}$, which defines the exact MAP assignment in (4) as $x_i^{\mathrm{MAP}} = \kappa \cdot \psi_i(x_i) \prod_{k \in \mathcal{N}(i)} m_{k\to i}^*(x_i)$ for each $i \in V$. We denote the the cost of a single message-passing over an edge $e = (i,j)$, under a given physical graph G, as c_e or $c_{i,j}$. Recall that the message passing over $e = (i,j)$ may need to be done over a multi-hop path on the physical graph G. One simple example of $c_{i,j}$ is the shortest path distance from node i to j in G. Then, both protocols incur the communication cost as elaborated in what follows:

(a) *Asynchronous*: In the *asynchronous* protocol (simply **ASYNC-MAP**), one node is arbitrarily picked as a root, and messages are passed from the leaves upwards to the root, then back downwards to the leaves. It involves a total $|E_Q|$ number of messages upon termination. Thus, the communication cost would be:

$$C(T(Q); G, \textbf{ASYNC-MAP}) = \sum_{(i,j)\in E_Q} 2c_{i,j}. \tag{6}$$

(b) *Synchronous:* In the *synchronous* protocol (simply **SYNC-MAP**), at each iteration, every node sends messages to all of its neighbors. Then, since the diameter $\mathtt{diam}(T(Q))$[2] is the minimum amount of time required for a message to pass between two most distant nodes in $T(Q)$, this protocol involves at most $\mathtt{diam}(T(Q))$ iterations with total $2|E_Q| \cdot \mathtt{diam}(T(Q))$ number of messages. Thus, we have the following cost:

$$C(T(Q); G, \textbf{SYNC-MAP}) = \sum_{(i,j)\in E_Q} 2c_{i,j} \cdot \mathtt{diam}(T(Q)). \tag{7}$$

In the next subsection, we will use the above two cost functions for two different learning algorithms for **CDG(*n*)** in (3) to estimate two cost-efficient data trees.

[2] For a tree T with d nodes, $2 \leq \mathtt{diam}(T) \leq d-1$.

Algorithm 1: ASYNC-ALGO

Input: $\boldsymbol{x}^{1:n}$: a set of n samples, γ: the trade-off parameter, a physical graph $G = (V, E_P)$
Output: Estimated tree $T = (V, E)$.

S0. $E = \emptyset$ and for each possible edge $e = (i, j) \in V \times V$, we initialize its weight by:

$$w_e = I_e(\hat{P}) - 2\gamma \cdot c_e, \tag{9}$$

where $I_e(\mu)$ is the mutual information between two end-points of edge e with respect to a given joint distribution μ.

S1. Run a maximum weight spanning tree algorithm for H, and save its resulting spanning tree at $T = (V, E)$.

S2. Return T.

3.2 Algorithm for Asynchronous MAP

Using the cost function for the asynchronous MAP in (6), the original optimization problem **CDG($\boldsymbol{n}$)** is re-cast into:

$$\textbf{CDG-A}(\boldsymbol{n}): \ \hat{Q}^\star(n) = \arg\min_{Q \in \mathcal{P}(\mathcal{X}^d)} D_{\mathrm{KL}}(\hat{P}(\boldsymbol{x}^{1:n}) \parallel Q) + \gamma \sum_{e \in E_Q} 2c_e. \tag{8}$$

We now describe **ASYNC-ALGO** that computes $\hat{Q}^\star(n)$ in (8) and thus estimates the cost-efficient data tree $T(\hat{Q}^\star(n))$ in Algorithm 1. As we see, the algorithm is remarkably simple. Using given n data samples, we construct a weighted complete graph, where the weight for each edge is assigned some combination of the mutual information of nodes i and j with respect to the empirical distribution $\hat{P}$ obtained from the data samples and the per-message cost, as in (9). Then, we run an algorithm that computes the maximum weight spanning tree, *e.g.*, Prim's algorithm or Kruskal's algorithm, and the resulting spanning tree is the output of this algorithm.

Correctness of ASYNC-ALGO. We now present the correctness of the above algorithm in the sense that we can obtain the data tree corresponding to the optimal distribution formulated in (8), as explained in what follows: For some tree distribution Q (thus, satisfying the factorization property in (2)), we have:

$$\begin{aligned} D_{\mathrm{KL}}(\hat{P} \parallel Q) &= -H(\hat{P}) - \sum_{\boldsymbol{x} \in \mathcal{X}^d} \hat{P}(\boldsymbol{x}) \log Q(\boldsymbol{x}) \\ &\geq -H(\hat{P}) + \sum_{i \in V} H(\hat{P}_i) - \sum_{(i,j) \in E_Q} I(\hat{P}_{i,j}), \end{aligned} \tag{10}$$

where $H(\cdot)$ is the entropy, and the inequality holds when the pairwise marginals over the edges of a fixed E_Q are set to that of $\hat{P}$, *i.e.*, $Q_{i,j}(x_i, x_j) = \hat{P}_{i,j}(x_i, x_j)$ for all $(i, j) \in E_Q$. Since the entropy terms are constant w.r.t. Q, it is straightforward that the structure of the estimator $\hat{Q}^\star(n)$ of **CDG-A($\boldsymbol{n}$)** in (8) is given by:

$$\hat{E}^\star(n) := \arg\max_{E_Q : Q \in \mathcal{P}(\mathcal{X}^d)} \sum_{e \in E_Q} I_e(\hat{P}) - 2\gamma \cdot c_e, \tag{11}$$

$$\hat{Q}^\star_{i,j}(n) = \hat{P}_{i,j}, \quad \forall (i, j) \in \hat{E}^\star(n). \tag{12}$$

Then, it is easy to see that (11) requires us to find the maximum weight spanning tree using $I_e(\hat{P}) - 2\gamma \cdot c_e$ as the edge e's weight, where the standard maximum weight spanning tree (MWST) computation algorithm runs in $O(d^2 \log d)$ time, where $|V| = d$.

3.3 Algorithm for Synchronous MAP

Similarly to **ASYNC-MAP**, using the cost in (7), the original optimization problem **CDG($\boldsymbol{n}$)** is re-cast into:

$$\textbf{CDG-S}(\boldsymbol{n}): \hat{Q}^\star(n) = \min_{Q \in \mathcal{P}(\mathcal{X}^d)} D_{\mathrm{KL}}(\hat{P}(\boldsymbol{x}^{1:n}) \parallel Q) + \gamma \cdot \mathtt{diam}(T(Q)) \sum_{e \in E_Q} 2c_e. \tag{13}$$

Following the similar arguments in Section 3.2, the structure of the above estimator of **CDG-S($\boldsymbol{n}$)** in (13) is given by

$$\hat{E}^\star(n) := \arg\max_{E_Q : Q \in \mathcal{P}(\mathcal{X}^d)} \sum_{e \in E_Q} I_e(\hat{P}) - 2\gamma \mathtt{diam}(T(Q)) \cdot c_e, \tag{14}$$

$$\hat{Q}^\star_{i,j}(n) = \hat{P}_{i,j}, \quad \forall (i, j) \in \hat{E}^\star(n). \tag{15}$$

We comment that this optimization is non-trivial in that the objective function contains the diameter of the tree, which can be computed only when the solution is fully characterized.

Hardness. The key difference in the cost function of **SYNC-MAP** from **ASYNC-MAP** is simply the existence of $\mathtt{diam}(T(Q))$. However, this simple difference completely changes the hardness of learning the optimal data tree in **SYNC-MAP**, as formally stated in the next Theorem.

Theorem 3.1 (Hardness of **CDG-S($\boldsymbol{n}$)**). *For any parameter $\gamma \geq 0$, obtaining the optimal distribution $\hat{Q}^\star(n)$ in* **CDG-S($\boldsymbol{n}$)** *and thus its associated data tree $T(\hat{Q}^\star(n))$ is NP-hard with respect to the number of nodes.*

Due to space limitation, we present the full proof of Theorem 3.1 in our technical report [18]. The key step in proof is to reduce the **CDG-S($\boldsymbol{n}$)** in (13) to *Exact Cover by 3-sets* problem.

Greedy algorithm. Due to the above-mentioned hardness, we propose a greedy heuristic algorithm that outputs the tree structure denoted by $\hat{E}^{\mathrm{S}}(n)$, called **SYNC-ALGO(β)**, as we describe in Algorithm 2, where β is the algorithm parameter. The overall algorithm operates as follows:

S0. Initialize the weight of each possible edge with some initial value.

S1. Sequentially select the edge that has the maximum weight and add it to the temporary resulting tree.

S2. Update the weight of each edge whose one end-point is in the current resulting tree V_S and another end-point is not, and go to **S1** until we handle all nodes.

One of the central steps here is: first, we *dynamically* update the weight of the candidate edges (*i.e.*, the set E') that we will add and, second, which value is chosen as the weight is different from the "one-shot" weight assignment as done in **ASYNC-ALGO**. To explain this intuition, we first note that from (14) it is easy to see that the degree of contribution in terms of weight by adding an edge $e \in E'$ to the existing resulting tree would be re-expressed as:

$$I_e(\hat{P}) - 2\gamma \mathtt{diam}(T \cup \{e\})) \cdot c_e - K_e(T), \tag{20}$$

where $K_e(T)$ is defined in (18). Here, $K_e(T)$ corresponds to the change of the communication cost over the existing edges in T,

Algorithm 2: SYNC-ALGO(β)

Input: $\boldsymbol{x}^{1:n}$: a set of n samples from P, γ: the trade-off parameter, a physical graph $G = (V, E_P)$, and a tunable parameter β.
Output: Estimated tree $T = (V_S, E_S)$.

S0. $V_S = \emptyset, E_S = \emptyset$, and for each possible edge $e \in V \times V$, we initialize its weight by:

$$w_e = I_e(\hat{P}) - 2\gamma \cdot c_e, \tag{16}$$

and initialize the edge set E' by the set of all possible edges.

repeat

S1. Select an edge $e = (u, v) \in E'$ with the maximum weight, and update $V_S \leftarrow V_S \cup \{u, v\}$ and $E_S \leftarrow E_S \cup \{e\}$.

S2. Update E' as the set of all edges $e = (i, j)$, such that $i \in V_S$ and $j \in V \setminus V_S$, and set the weight of each edge $e = (i, j) \in E'$ as:

$$w_e = I_e(\hat{P}) - 2\gamma \texttt{diam}(T \cup \{e\}) \cdot c_e - \beta \frac{d}{|E_S|} \cdot K_e(T) - 2\gamma \cdot D(T) \cdot c_e, \tag{17}$$

where

$$K_e(T) = \Big(\texttt{diam}(T \cup \{e\}) - \texttt{diam}(T)\Big) \cdot \sum_{e' \in E_S} 2\gamma \cdot c_{e'}, \tag{18}$$

and

$$D(T) = \sqrt{d} \cdot \left(1 - \frac{\sqrt{|E_S|}}{\sqrt{d}}\right). \tag{19}$$

until $V_S = V$;

Return $T = (V_S, E_S)$.

under the grown tree $T \cup \{e\}$. For example, $K_e(T) = 0$ if the diameter of the grown tree does not change by adding the edge e, or $K_e(T) = \sum_{e' \in E_S} 2\gamma \cdot c_{e'}$, if the diameter of the grown tree increases by 1.

In dynamically assigning the weight of the candidate edges in E', we do not use the value of (20). Instead, as seen in (17), (i) we use the expected diameter growth of the tree, denoted by $D(T)$ in (19), and (ii) we use a tunable parameter $\beta > 0$ to compensate for the impact of the change in communication cost over the existing edges $K_e(T)$ in (18). In more detail, we use $D(T)$ in (19), which captures the expected diameter growth of the tree T via the term $\sqrt{d}(1 - \frac{\sqrt{|E_S|}}{\sqrt{d}})$, since the diameter of a uniformly random spanning tree is known to be of the order $\sqrt{d}$ in [26]. We note that this term decreases to 0 as the tree becomes to a spanning tree from the term $1 - \frac{\sqrt{|E_S|}}{\sqrt{d}}$. Second, we consider the impact of old weights over the existing edges in T, captured by $K_e(T)$ in (20), by controlling a scale of $\beta \frac{d}{|E_S|}$.

To summarize, these two modified choices of the weight are for handling a probable sacrifice of the performance when using a vanilla greedy method as in (18), since the edge weight should be modified suitably for the changed diameter on the way of tree construction. We expect that these two engineerings play an important role when the cost-efficient data graph is attained with a large diameter, where the edges chosen in the begging phase of the procedure (*i.e.*, with a small diameter value) could exert much impact of communication cost at the end of the procedure. Our greedy algorithm runs in $O(d^4)$ times.

4 ESTIMATION ERROR FOR INCREASING SAMPLE SIZE

In this section, we provide the analysis of how the estimation error probability decays with the growing number of samples n, using the large deviation principle (LDP).

4.1 Estimation Error of ASYNC-ALGO

Clearly, when we use more and more data samples, $\hat{E}^\star(n)$ approaches to $\hat{E}^\star(\infty)$ that is the optimal edge structure solving **CDG-A(∞)**. We are interested in characterizing the following error probability of the event $\mathcal{A}_n$:

$$\mathbb{P}\Big[\mathcal{A}_n(\boldsymbol{x}^{1:n}) := \big\{\hat{E}^\star(n) \neq \hat{E}^\star(\infty)\big\}\Big]. \tag{21}$$

To characterize the probability in (21) that is one of the rare events, we use LDP that *rare events occurs in the most probable way*. To this end, we aim at studying the following rate function $K = K(\gamma)$:

$$K(\gamma) := \lim_{n\to\infty} -\frac{1}{n} \log \mathbb{P}(\mathcal{A}_n(\boldsymbol{x}^{1:n})), \tag{22}$$

whenever the limit exists.

We now consider a simple event, called *crossover event*, as defined in what follows: Recall that **ASYNC-ALGO** uses, for each edge e, the weight[3] of $w_e(\hat{P}) = I_e(\hat{P}) - 2\gamma c_e$ based on the empirical distribution $\hat{P}$. Then, consider two edges e and e' such that the weight of e exceeds that of e' with respect to the *true* distribution P, *i.e.*, $w_e(P) > w_{e'}(P)$. We now define the crossover event for two edges e and e' as:

$$C_n(e, e') := \big\{w_e(\hat{P}) \leq w_{e'}(\hat{P})\big\}. \tag{23}$$

As the number of samples $n \to \infty$, the empirical distribution approaches to the true distribution, thus the probability of the crossover event decays to zero, whose decaying rate which we call *crossover rate* is defined as $J_{e,e'} := \lim_{n\to\infty} -\frac{1}{n} \log \mathbb{P}[C_n(e, e')]$. Using this definition of the crossover event, we present Theorem 4.1 that states the decaying rate of the estimation error probability as the number of data samples n grows.

Theorem 4.1 (Decaying rate of **ASYNC-ALGO**). *For any fixed parameter $\gamma \geq 0$,*

$$\lim_{n\to\infty} -\frac{1}{n} \log \mathbb{P}(\mathcal{A}_n(\boldsymbol{x}^{1:n})) = K(\gamma), \tag{24}$$

where

$$K(\gamma) = \min_{e' \notin \hat{E}^\star(\infty)} \min_{e \in \Psi(e'; \hat{E}^\star(\infty))} J_{e,e'}, \tag{25}$$

where $\Psi(e' = (i, j); \hat{E}^\star(\infty)) := \{v_1(= i), v_2, \cdots, v_l(= j)\}$ is the unique path between nodes i and j, such that $(v_k, v_{k+1}) \in \hat{E}^\star(\infty)$ for $1 \leq k \leq l - 1$, and

$$J_{e,e'} = \begin{cases} \inf\limits_{Q \in \mathcal{P}(\mathcal{X}^4)} & \Big\{D_{KL}(Q \parallel P_{e,e'}) : w_e(Q) = w_{e'}(Q)\Big\}, \\ & \text{if } \big\{Q \in \mathcal{P}(\mathcal{X}^4) : w_e(Q) = w_{e'}(Q)\big\} \neq \emptyset, \\ \infty, & \text{otherwise.} \end{cases} \tag{26}$$

[3] We interchangeably use $w_e(\hat{P})$ to denote the assigned weight of an edge e in algorithms, with respect to the empirical distribution $\hat{P}$ from the given samples.

Moreover, we have the following (finite-sample) upper-bound on the error probability: for all $n = 1, 2, \ldots$,

$$\mathbb{P}\Big[\mathcal{A}_n(\boldsymbol{x}^{1:n})\Big] \leq \frac{(d-1)^2(d-2)}{2}\binom{n-1+|\mathcal{X}|^4}{|\mathcal{X}|^4-1}\exp(-n\cdot K(\gamma)). \tag{27}$$

In Theorem 4.1, we observe that the decaying rate of error probability is specified by some topological information of physical/data graphs and the trade-off parameter γ. In particular, the crossover event and its rate $J_{e,e'}$ depend on how difficult it is to differentiate two edge weights under the true data distribution with a consideration of the trade-off parameter γ as well as per-message cost on edges. As interpreted from (26), when $w_e(P) = I_e(P) - 2\gamma c_e$ and $w_{e'}(P) = I_{e'}(P) - 2\gamma c_{e'}$ are close, the confusion between e and e' from samples frequently occurs, leading to high error probability, and we can show the existence of the infimum Q satisfying $w_e(Q) = w_{e'}(Q)$ as by slightly adjusting the true distribution P. Moreover, we remark that the decaying rate $J_{e,e'}$ (and thus $K(\gamma)$) is characterized by a trade-off parameter γ. The error rate becomes smaller (*i.e.*, higher error probability) when γ nearly meets the condition $w_e(P) = w_{e'}(P)$, and the weights becomes deterministic with respect to the samples as γ increases since the portion of the cost in weights grows, resulting to $J_{e,e'} = \infty$ in (26). These interpretations are well-matched to our numerical results in Section 5.

Proof sketch. The proof of Theorem 4.1 is presented in our technical report [18], and we describe the proof sketch for readers' convenience. Our proof largely follows that of the related work in [28] that analyzes an error exponent of a standard tree structure learning (*i.e.*, known as Chow-Liu algorithm [8]), whose goal is to solely estimate the true data distribution with no consideration of communication cost. Simply, the proof idea follows LDP in the following way. The error event $\mathcal{A}_n(\boldsymbol{x}^{1:n})$ is expressed as a union of small events that **ASYNC-ALGO** estimates only one wrong edge (see the definition of the crossover event in (23)), two wrong edges, and three, etc. Following LDP, the decaying rate of the error probability equals to the decaying rate of the most probable crossover event, which corresponds to the case of only one wrong edge. In more detail, two minimums in (25) specify the most-probably error, whose edge set differs from the optimal data tree structure $\hat{E}^\star(\infty)$ exactly in one edge, , *i.e.*, $\hat{E}^\star(\infty) \setminus \{e\} \cup \{e'\}$, where it contains the non-neighbor node pair e' (as selected in the first minimization) instead of the most probable replacement edge e in the unique path along $\hat{E}^\star(\infty)$ (as in the second minimization). To obtain the minimum crossover rate $J_{e,e'}$, we apply the Sanov's theorem [5], which provides an expression of the probabilistic relationship between $\hat{P}$ and P via their KL divergence. Finally, in addition to the asymptotic decaying rate of the estimation error probability, we also establish its upper bound of the error probability in terms of the number n of data samples, where the first term $(d-1)^2(d-2)/2$ of the bound in (27) implies the number of possible crossover events, and the second term $\binom{n-1+|\mathcal{X}|^4}{|\mathcal{X}|^4-1}$ represents the number of possible empirical distributions $\hat{P}_{e,e'}$.

4.2 Estimation Error of SYNC-ALGO

We conduct a similar analysis here for **SYNC-ALGO** to what we did for **ASYNC-ALGO**, which has more complicated issues for the following reasons: We first denote by $w_e(\hat{P}, T)$ in (17) the assigned weight for edge e to stress its dependence on the corresponding resulting tree structure T and its associated empirical distribution $\hat{P}$. Then, we need to investigate the most probable pattern in the rare event through a certain tree T at some iteration. Simply, the crossover event for two edges e and e' occurs if the order of edge weights from the given finite number of samples becomes reversed to the order of weights from the true data distribution. Among all possible crossover events, we are interested in the crossover event under every tree structure that is obtained on the way of constructing the ideal data structure, denoted by $\hat{E}^{\mathrm{S}}(\infty)$. Let e^t_{true} and T^t_{true} be the selected edge and constructed tree at t-th iteration obtained by running **SYNC-ALGO** w.r.t. the true data distribution P, which would finally find $\hat{E}^{\mathrm{S}}(\infty)$. Then, it is obvious that e^t_{true} has the unique highest edge weight for P, and the crossover event of our interest is defined as:

$$C_n(e^t_{\mathrm{true}}, e'; T^t_{\mathrm{true}}) := \left\{w_{e^t_{\mathrm{true}}}(\hat{P}; T^t_{\mathrm{true}}) \leq w_{e'}(\hat{P}; T^t_{\mathrm{true}})\right\}. \tag{28}$$

We now state Theorem 4.2 that establishes the decaying rate of the estimation error probability as the number of data samples grows.

Theorem 4.2 (Decaying rate of **SYNC-ALGO**). *For any fixed parameter $\gamma \geq 0$,*

$$\lim_{n\to\infty} -\frac{1}{n}\log \mathbb{P}(\mathcal{A}_n(\boldsymbol{x}^{1:n})) \geq K(\gamma), \tag{29}$$

where

$$K(\gamma) := \min_{t\in\{1,\cdots,|V|-1\}} \min_{e'\notin T^{t+1}_{\mathrm{true}}} J_{e^t_{\mathrm{true}},e'}(T^t_{\mathrm{true}}), \tag{30}$$

where e^t_{true} and T^t_{true} are the selected edge and constructed tree at t-th iteration by running **SYNC-ALGO** *w.r.t. the true data distribution P,* i.e., *$w_{e^t_{\mathrm{true}}}(P)$ has the maximum edge weight under the tree T^t_{true}, and it is given by: under some tree T, for any e, e',*

$$J_{e,e'}(T) = \begin{cases} \inf\limits_{Q\in\mathcal{P}(\mathcal{X}^4)} & \left\{D_{KL}(Q \parallel P_{e,e'}) : w_e(Q) = w_{e'}(Q)\right\}, \\ & \text{if } \left\{Q \in \mathcal{P}(\mathcal{X}^4) : w_e(Q) = w_{e'}(Q)\right\} \neq \emptyset, \\ \infty, & \text{otherwise.} \end{cases} \tag{31}$$

Moreover, we have the following (finite-sample) upper-bound on the error probability: for all $n = 1, 2, \ldots$,

$$\mathbb{P}\Big[\mathcal{A}_n(\boldsymbol{x}^{1:n})\Big] \leq \frac{(d-1)d(d+1)}{6}\binom{n-1+|\mathcal{X}|^4}{|\mathcal{X}|^4-1}\exp(-n\cdot K(\gamma)). \tag{32}$$

In Theorem 4.2, as seen in (29), the error rate function $K(\gamma)$ in (30) indeed provides a lower-bound of the actual decaying rate of the error event $\mathcal{A}_n(\boldsymbol{x}^{1:n})$, since the crossover event $C_n(e^t_{\mathrm{true}}, e'; T^t_{\mathrm{true}})$ which estimates an edge $e' \notin T^{t+1}_{\mathrm{true}}$ rather than e^t_{true} at any t-th iteration does not guarantee that e' is a wrong edge. Intuitively, the edge weights of **SYNC-ALGO** dynamically change according to a diameter of T^t_{true} as iteration t proceeds, which makes the characterization of the exact error rate of **SYNC-ALGO** be non-trivial.

Proof sketch. Due to space limitation, we present the complete proof in our technical report [18], and we provide a brief proof sketch. The basic idea is similar to the proof of Theorem 4.1. As mentioned there, the crossover event $C_n(e^t_{\mathrm{true}}, e'; T^t_{\mathrm{true}})$ is not a

subset of the error event $\mathcal{A}_n(\boldsymbol{x}^{1:n})$, and as a result, we provide a lower-bound of the decaying error rate in the proof, as established by two minimizations in (31). In particular, the first minimization is taken over all iterations ($1 \le t \le |V|-1$) so that it selects the iteration where the error occurs in the most probable way, and the second minimization specifies the non-neighbor node pair e', which can be estimated instead of e^t_{true}, having the minimum $J_{e^t_{\text{true}},e'}(T^t_{\text{true}})$, In other words, the most probable pattern in the error event of **SYNC-ALGO** is to estimate $T^t_{\text{true}} \setminus \{e^t_{\text{true}}\} \cup \{e'\}$ attained in two minimizations in (30). For the crossover rate $J_{e^t_{\text{true}},e'}(T^t_{\text{true}})$ in (31), when two edges e^t_{true} and e' can be clearly differentiated via their edge weights, since the difference of the cost between two edges dominantly determines the order of the edge weights, *i.e.*, the condition in (31) does not hold, the crossover event does not happen, *i.e.*, $J_{e^t_{\text{true}},e'}(T^t_{\text{true}}) = \infty$. This mostly corresponds to the situation of a large value of the trade-off parameter γ, where the communication cost plays an important role of the error event, which do not depend on the number of samples n. Otherwise, the crossover rate is attained in a similar way to (26). Finally, we establish the upper bound of the error probability in terms of the sample size n, where the first term $(d-1)d(d+1)/6$ of the bound in (32) corresponds to the number of possible crossover events throughout the entire iterations, and the second term implies the number of possible empirical distributions $\hat{P}_{e^t_{\text{true}},e'}$.

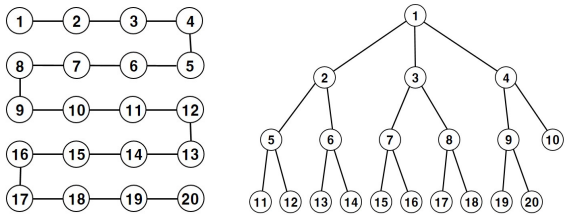
5 NUMERICAL RESULTS

In this section, we provide a set of numerical experiments to validate our analytical results of **ASYNC-ALGO** and **SYNC-ALGO** under various numbers of data samples, communication costs, and trade-off parameters.

5.1 Setup

Physical graph. We use a physical network $G = (V, E_P)$ consisting of 20 nodes forming a line topology, where node i can directly communicate only with nodes $i-1$ and $i+1$, see Figure 2(a). We assign some constant cost of single message-passing for each edge $e = (i, i+1)$: $c_{i,i+1} = \kappa \times 1.1^i$, except for $c_{1,2} = 4\kappa$, $c_{3,4} = 2\kappa$, $c_{6,7} = 0.1\kappa$, where we appropriately choose κ to adjust the scale of total communication cost of two learning algorithms in the same range, for clear comparison with the same values of γ. In the message-passing between non-neighboring (w.r.t. the physical graph) node pairs (i, j), we simply assume that it expenses the sum of the costs when it is passed along the unique shortest multi-hop path $\Psi((i,j); G)$ on G, *i.e.*, $c_{i,j} = \sum_{e' \in \Psi((i,j);G)} c_{e'}$. For example, $c_{1,4} = c_{1,2} + c_{2,3} + c_{3,4}$. We use this line topology for an exemplar physical graph to clearly observe the difference between **ASYNC-ALGO** and **SYNC-ALGO**, where it leads to a significantly huge amount of communication cost for **SYNC-MAP**, due to large diameter value $\texttt{diam}(G) = 19$.

Data graph. As an underlying statistical dependencies among 20 nodes in the data graph, we consider a 3-regular tree $T = (V, E_D)$, except for boundary nodes, where the node 1 is a root node and every node has a degree of 3 or less, as depicted in Figure 2(b). Each random variable X_i associated to a node i is set to follow a Bernoulli distribution. For a root node 1, it has $P(X_1 = 0) = 0.7$ and $P(X_1 = 1) = 0.3$, and for other neighboring node pairs i and j, we



(a) Physical graph of 20 nodes forming a line topology.

(b) Data graph of 20 nodes forming a 3-regular tree (except for the leaves).

Figure 2: Physical and data graphs used for our simulations.

set the conditional distribution between X_i and X_j by

$$P(X_i = 0|X_j = 0) = 0.7, \quad \text{and} \quad P(X_i = 0|X_j = 1) = 0.3 \qquad (33)$$

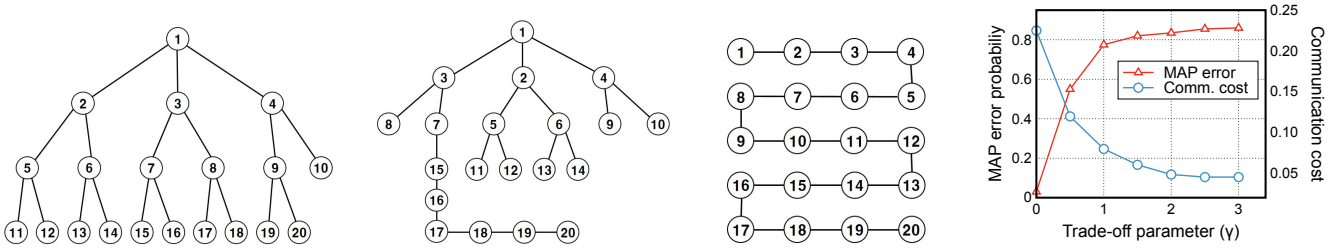
whenever $i < j$. With this setting of per-node distribution, it turns out that neighboring node pairs have high correlations, and thus have distinct values of the mutual information.

Under this choice of physical and data graphs, we obtain numerical examples to show the performance of **ASYNC-ALGO** and **SYNC-ALGO** for various values of trade-off parameter γ, ranging from 0 to 4, and a fixed $\beta = 1$ in our results. For a fixed $n \in \mathbb{N}$, we first generate n i.i.d. samples $\boldsymbol{x}^{1:n}$ from $P(\boldsymbol{x})$ in (33). Then, we compute the empirical distribution $\hat{P}(\boldsymbol{x}^{1:n})$ and the empirical mutual information of all possible node pairs $\{I_e(\hat{P})\}_{e \in V \times V}$. Then, we learn the cost-efficient data tree by running **ASYNC-ALGO** or **SYNC-ALGO**, and estimate how well the proposed algorithms recover the ideal data graph by investigating the estimation error probability as n grows.

5.2 Results

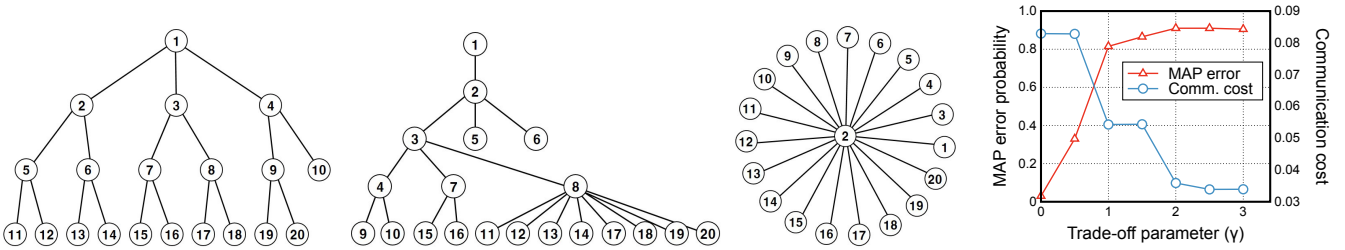
(i) Estimated trees with varying γ. Figures 3 and 4 show that the estimated data trees by **ASYNC-ALGO** and **SYNC-ALGO** for various γ. We recall that the value of γ parameterizes the amount of priority for communication cost compared to the inference quality, see (3), where smaller γ leads to higher priority to the inference quality. In both algorithms, we observe that they with $\gamma = 0$ estimate the exact data graph in Figure 2(b), since the goal is to achieve the highest inference accuracy. However, as γ grows, each of two algorithms estimates a different structure for data tree, since **ASYNC-MAP** and **SYNC-MAP** have different forms of communication costs. In particular, in **ASYNC-ALGO**, as γ grows, we observe that the algorithm produces the estimated data tree with more resemblance to the physical graph, and finally it estimates the data tree that is the same as the physical graph with $\gamma = 4$, see Figures 3(b) and 3(c). We note that for a large value of γ, the goal of **ASYNC-ALGO** is to find a MWST of minimum total cost, which accords with the physical graph of line-topology. However, the communication cost of **SYCN-ALGO** increases in proportion to the *diameter* of the estimated tree, thus it estimates a tree that is of a *star-like* topology, *i.e.*, a tree with a small diameter as seen in Figures 4(b) and 4(c), to significantly reduce the cost, as γ grows.

(ii) Quantifying trade-off between inference accuracy and cost. We now quantify how the trade-off between inference accuracy of the MAP estimator in (4) behaves and the total communication cost is captured for different values of γ. To support the trade-off



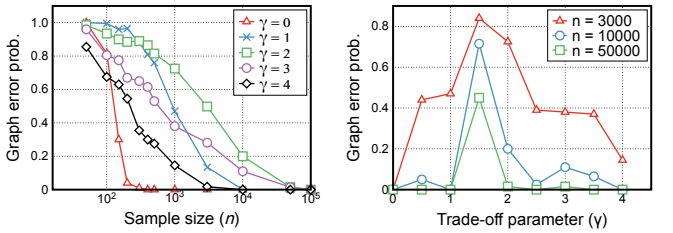
(a) Estimated tree structure of $\gamma = 0$ (b) Estimated tree structure of $\gamma = 0.5$ (c) Estimated tree structure of $\gamma = 4$ (d) Trade-off between MAP accuracy and cost

Figure 3: An instance of estimated tree structure by ASYNC-ALGO with distinct trade-off parameter $\gamma = 0, 0.5, 4$, and the trade-off between MAP accuracy and communication cost.



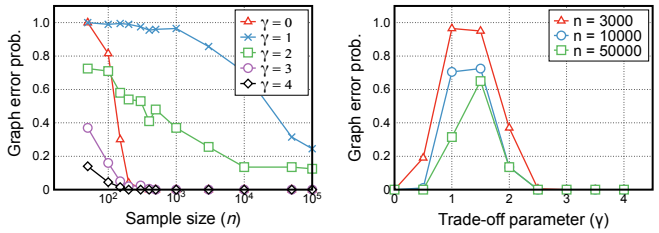
(a) Estimated tree structure of $\gamma = 0$ (b) Estimated tree structure of $\gamma = 1$ (c) Estimated tree structure of $\gamma = 4$ (d) Trade-off between MAP accuracy and cost

Figure 4: An instance of estimated tree structure by SYNC-ALGO with distinct trade-off parameter $\gamma = 0, 1, 4$, and the trade-off between MAP accuracy and communication cost.



(a) Error probability w.r.t. the sample size n (b) Error probability with varying trade-off parameter γ

Figure 5: Error probability of ASYNC-ALGO.



(a) Error probability w.r.t. the sample size n (b) Error probability with varying trade-off parameter γ

Figure 6: Error probability of SYNC-ALGO.

parameterized by γ in the optimization problem in (3), we vary γ from 0 to 4 and plot the accuracy of MAP estimator and the total cost on the learnt data dependency graph as the red and blue lines, respectively, in Figures 3(d) and 4(d). In particular, we run the **ASYNC-ALGO** and **SYNC-ALGO** with $n = 200$ samples, respectively, and run the max-product algorithm on the learnt data tree to obtain the MAP estimator. We repeatedly run for 200 times, and measure the error probability that the MAP estimator on the learnt data tree differs from the MAP estimator on the true data graph, as a metric of inference accuracy. The average (over the 200 results) of the communication cost on the learnt data tree is measured by the form of (6) and (7) for each algorithm. In Figure 4(d), we observe that the MAP estimation error and cost with $\gamma = 0.5$ is 0.33 and 0.083, respectively, while those with $\gamma = 2.5$ is 0.91 and 0.034, respectively. The impact of γ on the trade-off for two algorithms seems similar, as seen in Figures 3(d) and 4(d).

(iii) Impact of data sample size on graph estimation accuracy. Finally, we demonstrate the theoretical findings in Theorems 4.1 and 4.2 on the decaying rate of the error probability w.r.t. the number of samples n for various values of γ. In both **ASYNC-ALGO** and **SYNC-ALGO**, for a fixed γ, we run both algorithms for 200 times each, and measure their error probabilities. In Figures 5(a) and 6(a), we observe that the error probability $\mathbb{P}(\mathcal{A}_n)$ for every γ decays exponentially as the sample size n increases, as established in (27) and (32). It is interesting to see that a different choice of γ leads to a different decaying rate, which can be understood by our analytical findings of the crossover rate in (26) and (31), simply given by:

$$J_{e,e'}(T) = \inf_{Q \in \mathcal{P}(\mathcal{X}^4)} \Big\{ D_{\mathrm{KL}}(Q \,\|\, P_{e,e'}) : w_e(Q) = w_{e'}(Q) \Big\},$$

where the edge weights for **ASYNC-ALGO** and **SYNC-ALGO** are assigned in different forms, yet depending on the value of γ, as

seen in (9) and (17). Some choice of γ makes a difference of the corresponding edge weights highly small, so that it becomes easier to estimate wrong edges with an insufficient number of samples. In our simulation, **ASYNC-ALGO** with $\gamma = 2$ shows higher error probability of 0.2 with $n = 10^4$ samples, while that with $\gamma = 0$ achieves almost 0 error probability with less than 3000 samples, see Figure 5(a). This impact of γ on the error probability is presented in Figures 5(b) and 6(b) for both algorithms, where for large γ, we observe the error probability decays at a higher rate in general, since the priority to the inference accuracy is insignificant, leading to less chance of experiencing the crossover event.

6 CONCLUSION

In many multi-agent networked systems, a variety of applications involve distributed in-network statistical inference tasks, such as MAP (maximum a posteriori), exploiting a given knowledge of statistical dependencies among agents. When agents are spatially-separated, running an inference algorithm leads to a non-negligible amount of communication cost due to inevitable message-passing, coming from the difference between data dependency and physical connectivity. In this paper, we consider a structure learning problem which recovers the statistical dependency from a set of data samples, which also considers the communication cost incurred by the applied distributed inference algorithms to the learnt data graph. To this end, we first formulate an optimization problem formalizing the trade-off between inference accuracy and cost, whose solution chooses a tunable point in-between them. As an inference task, we studied the distributed MAP and their two implementations **ASYNC-MAP** and **SYNC-MAP** that have different cost generation structures. In **ASYNC-MAP**, we developed a polynomial time, optimal algorithm, inspired by the problem of finding a maximum weight spanning tree, while we proved that the optimal learning in **SYNC-MAP** is NP-hard, thus proposed a greedy heuristic. For both algorithms, we then established how the error probability that the learnt data graph differs from the ideal one decays as the number of samples grows, using the large deviation principle.

ACKNOWLEDGMENTS

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2016-0-00160, Versatile Network System Architecture for Multi-dimensional Diversity). The work of H. Jang was supported by the Brain Korea 21 Program for Leading Universities and Students under Grant BK21 PLUS.

REFERENCES

[1] Pieter Abbeel, Daphne Koller, and Andrew Y Ng. 2006. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research* 7, Aug (2006), 1743–1788.
[2] Daron Acemoglu and Asuman Ozdaglar. 2011. Opinion dynamics and learning social networks. *Dynamic Games and Applications* 1, 1 (2011), 3–49.
[3] Anima Anandkumar, Furong Huang, Daniel J Hsu, and Sham M Kakade. 2012. Learning mixtures of tree graphical models. In *Proc. of Neural Information Processing Systems*.
[4] Guy Bresler. 2015. Efficiently learning Ising models on arbitrary graphs. In *Proc. of Symposium on Theory of Computing*. ACM.
[5] James A Bucklew. 1990. *Large deviation techniques in decision, simulation, and estimation*. Wiley New York.
[6] Jean-Francois Chamberland and Venugopal V Veeravalli. 2007. Wireless sensors in distributed detection applications. *IEEE Signal Processing Magazine* 24, 3 (2007), 16–25.
[7] Lei Chen, Mujdat Cetin, and Alan S Willsky. 2005. Distributed data association for multi-target tracking in sensor networks. (2005).
[8] C Chow and C Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 14, 3 (1968), 462–467.
[9] Sanjoy Dasgupta. 1999. Learning polytrees. In *Proc. of Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc.
[10] Justin Dauwels. 2007. On variational message passing on factor graphs. In *Proc. of International Symposium on Information Theory*. IEEE.
[11] Mohammadreza Doostmohammadian and Usman A Khan. 2014. Graph-theoretic distributed inference in social networks. *IEEE Journal of Selected Topics in Signal Processing* 8, 4 (2014), 613–623.
[12] Gal Elidan, Ian McGraw, and Daphne Koller. 2012. Residual belief propagation: Informed scheduling for asynchronous message passing. *ArXiv preprint arXiv:1206.6837* (2012).
[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9, 3 (2008), 432–441.
[14] Xinzhe Fu, Zhongzhao Hu, Zhiying Xu, Luoyi Fu, and Xinbing Wang. 2017. De-anonymization of Social Networks with Communities: When Quantifications Meet Algorithms. *ArXiv preprint arXiv:1703.09028* (2017).
[15] Amir Globerson and Tommi S Jaakkola. 2008. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Proc. of Neural Information Processing Systems*.
[16] Alexander T Ihler, John W Fisher, Randolph L Moses, and Alan S Willsky. 2005. Nonparametric belief propagation for self-localization of sensor networks. *IEEE Journal on Selected Areas in Communications* 23, 4 (2005), 809–819.
[17] Alexander T Ihler, W Fisher John III, and Alan S Willsky. 2005. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research* 6, May (2005), 905–936.
[18] Hyeryung Jang, HyungSeok Song, and Yung Yi. 2018. Learning Data Dependency with Communication Cost. *ArXiv preprint arXiv:1804.10942* (2018).
[19] Usman A Khan and Jose MF Moura. 2008. Distributing the Kalman filter for large-scale systems. *IEEE Transactions on Singal Processing* 56, 10 (2008), 4919–4935.
[20] O Patrick Kreidl and Alan S Willsky. 2006. Inference with minimal communication: A decision-theoretic variational approach. In *Proc. of Neural Information Processing Systems*.
[21] Marina Meila and Michael I Jordan. 2000. Learning with mixtures of trees. *Journal of Machine Learning Research* 1, Oct (2000), 1–48.
[22] Cetin Mujdat, Lei Chen, John W Fisher, Alexander T Ihler, Randolph L Moses, Martin J Wainwright, and Alan S Willsky. 2006. Distributed fusion in sensor networks. *IEEE Singal Processing Magazine* 23, 4 (2006), 42–55.
[23] Mark Paskin, Carlos Guestrin, and Jim McFadden. 2005. A robust architecture for distributed inference in sensor networks. In *Proc. of Information Processing in Sensor Networks*. IEEE.
[24] Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.
[25] Pradeep Ravikumar, Martin J Wainwright, John D Lafferty, et al. 2010. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *The Annals of Statistics* 38, 3 (2010), 1287–1319.
[26] A Rényi and G Szekeres. 1967. On the hieght of trees. *Journal of Australian Mathematical Society* 7, 4 (1967), 497–507.
[27] Jeremy Schiff, Dominic Antonelli, Alexandros G Dimakis, David Chu, and Martin J Wainwright. 2007. Robust message-passing for statistical inference in sensor networks. In *Proc. of Information Processing in Sensor Networks*. IEEE.
[28] Vincent YF Tan, Animashree Anandkumar, Lang Tong, and Alan S Willsky. 2011. A large-deviation analysis of the maximum-likelihood learning of Markov tree structures. *IEEE Transactions on Information Theory* 57, 3 (2011), 1714–1735.
[29] Venugopal V Veeravalli and Pramod K Varshney. 2012. Distributed inference in wireless sensor networks. *Philosophical Transactions of the Royal Sociaty A* 34, 3 (2012), 100–117.
[30] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. 2003. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory* 49, 5 (2003), 1120–1146.
[31] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. 2005. MAP estimation via greement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory* 51, 11 (2005), 3697–3717.
[32] Yair Weiss and William T Freeman. 2001. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory* 47, 2 (2001), 736–744.
[33] Jianru Xue, Nanning Zheng, Jason Geng, and Xiaopin Zhong. 2008. Tracking multiple visual targets via particle-based belief propagation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38, 1 (2008), 196–209.
[34] Wei Zhao and Yao Liang. 2015. Energy-efficient and robust in-network inference in wireless sensor networks. *IEEE Transactions on Cybernetics* 45, 10 (2015), 2105–2118.
[35] Chongyu Zhou, Chen-Khong Tham, and Mehul Montani. 2016. Optimizing Graphical Model Structure for Distributed Inference in Wireless Sensor Networks. In *Proc. of Sensing, Communication, and Networking*. IEEE.
[36] Or Zuk, Shiri Margel, and Eytan Domany. 2012. On the number of samples needed to learn the correct structure of a Bayesian network. *Arxiv preprint arXiv:1206.6862* (2012).