ESTIMATION OF GAUSSIAN DIRECTED ACYCLIC GRAPHS USING PARTIAL ORDERING INFORMATION WITH APPLICATIONS TO DREAM3 NETWORKS AND DAIRY CATTLE DATA

BY SYED RAHMAN*, KSHITIJ KHARE†, GEORGE MICHAILIDIS†, CARLOS MARTÍNEZ‡, AND JUAN CARULLA§

Nielsen Company*, University of Florida†, Corporación Colombiana de Investigación Agropecuaria‡, and Universidad Nacional de Colombia§

Estimating a directed acyclic graph (DAG) from observational data represents a canonical learning problem and has generated a lot of interest in recent years. Research has focused mostly on the following two cases: when no information regarding the ordering of the nodes in the DAG is available, and when a domain-specific complete ordering of the nodes is available. In this paper, motivated by a recent application in dairy science, we develop a method for DAG estimation for the middle scenario, where partition based partial ordering of the nodes is known based on domain-specific knowledge. We develop an efficient algorithm that solves the posited problem, coined Partition-DAG. Through extensive simulations using the DREAM3 Yeast networks, we illustrate that Partition-DAG effectively incorporates the partial ordering information to improve both speed and accuracy. We then illustrate the usefulness of Partition-DAG by applying it to recently collected dairy cattle data, and inferring relationships between various variables involved in dairy agroecosystems.

1. Introduction. The problem of estimating a directed acyclic graph (DAG) from high-dimensional observational data has attracted a lot of attention recently in the statistics and machine learning literature, due to its importance in a number of application areas including molecular biology. In the latter area, high throughput techniques have enabled biomedical researchers to profile biomolecular data to better understand causal molecular mechanisms involved in gene regulation and protein signaling (Emmert-Streib et al. [2014]). Further, it provided the impetus for the development of numerous approaches for tackling the problem - see for example the review paper Marbach et al. [2012] and references therein.

This is a challenging learning problem in its general form. It stems from the fact that in order to reconstruct a DAG from data, one has to consider
all possible orderings of nodes and score the resulting network structures based on evidence gleaned from the available data. The computational complexity of obtaining all possible orderings of a set of nodes in a directed graph is of the order $p!$, where $p$ is the number of nodes the graph. In certain applications, one may have access to a complete topological ordering, which renders the problem computationally tractable as discussed below.

An interesting question arises of what advantages, the availability of reliable external information on partial orderings of nodes, brings to solving the problem. This is the key issue addressed in this study, motivated by an application on dairy operations (described later in the introduction and in Section 5).

More formally, suppose $Y_1, ..., Y_n \in \mathbb{R}^p$ are i.i.d. random vectors from a continuous multivariate distribution with mean $\mathbf{0}$ and covariance matrix $\Sigma = \Omega^{-1}$. In sample deprived settings, an effective and popular method for estimating $\Sigma$ imposes sparsity on the entries of $\Sigma$ (covariance graph models), or $\Omega$ (graphical models), or appropriate Cholesky factors of $\Omega$ (directed acyclic graph models or Bayesian networks). The choice of an appropriate model often depends on the application.

In this study, our focus is on learning a DAG from high-dimensional data, assuming sparsity in an appropriate Cholesky factor of $\Omega$. In particular, consider the factorization of the inverse covariance matrix $\Omega = B^tB$, where $B$ can be converted to a lower triangular matrix with positive diagonal entries through a permutation of rows and columns. Such a matrix $B$ can be regarded as the (weighted) adjacency matrix of a directed acyclic graph (DAG), where $B_{ij} \neq 0$ signifies the existence of a directed edge from $i$ to $j$, and $B_{ij} = 0$ signifies the absence of any edge from $i$ to $j$. For ease of exposition, we will identify the weighted adjacency matrix $B$ with its corresponding DAG. Note that the above factorization $\Omega = B^tB$ is not unique, and each permutation of the rows and columns in general leads to a different value for the DAG $B$.

There are three main lines of work that have dealt with DAG estimation in the Gaussian framework - one where the permutation that makes $B$ lower triangular is known, one where it is completely unknown, and one where partial background information is available. We briefly discuss them below.

DAG estimation with known ordering of variables. As previously mentioned, in many applications, a natural ordering, such as time based or location based ordering, of the variables presents itself, and hence a natural choice for the permutation which makes $B$ lower triangular is available. Penalized likelihood methods, which use versions of the $\ell_1$ penalty, and minimize the respective objective functions over the space of lower triangular
matrices, have been developed in Huang et al. [2006], Shojaie and Michailidis [2010], Khare et al. [2017]. Bayesian methods for this setting have been developed in Cao et al. [2017], Altamore et al. [2013], Consonni et al. [2017]. For many of these methods, high dimensional consistency results for the model parameters have also been established.

**DAG estimation with completely unknown ordering of variables.** If the permutation/ordering that makes $B$ lower triangular is unknown, then the problem becomes significantly more challenging, both computationally and theoretically. In this setting, traditional methods assume that the conditional independence relations corresponding to the underlying distribution exactly match the conditional independence relations corresponding to an underlying (unknown) DAG. This assumption is known as faithfulness. Since two DAGs encoding the exact same conditional independence relations are not identifiable from this distribution, the goal then becomes to identify the equivalence class of all graphs which correspond to the exact same conditional independence relations. This equivalence class can be conveniently represented in terms of a partially directed acyclic graph referred to as the CPDAG. Under this regime, several score-based, constraint-based and hybrid algorithms for estimating the underlying CPDAG have been developed and studied in the literature (Spirtes et al. [2001], Geiger and Heckerman [2013], Lam and Bacchus [1994], Heckerman et al. [1995], Chickering [2002], Ellis and Wong [2008], Zhou [2011], Kalisch and Buhlmann [2007], I.Tsamardinos et al. [2006], Gamez et al. [2011, 2012], van de Geer and Buhlmann [2013]). See Aragam and Zhou [2015] for an excellent and detailed review. In recent work, Raskutti and Uhler [2018] relax the faithfulness assumption to the sparsest Markov representation (SMR) assumption, which postulates that (the Markov equivalence class of) the true DAG has the smallest number of edges among all DAGs satisfying the Markov property (with respect to the underlying distribution). They develop the **Sparsest Permutation** (SP) algorithm for inferring (the Markov equivalence class of) the true DAG, and establish consistency under the SMR assumption. They also establish connections of their approach to the $\ell_0$-penalty based method of van de Geer and Buhlmann [2013] in the noiseless Gaussian setting.

Recently, Aragam and Zhou [2015] have developed a penalized likelihood approach called CCDr for sparse estimation of $B$ in a Gaussian setting, which has been shown to be significantly more computationally scalable than previous approaches. In a subtle but important departure from tradition, they do not assume faithfulness. They define two DAGs $B_1$ and $B_2^1$ to be

¡Recall that we (following Aragam and Zhou [2015]) are identifying a DAG with its weighted adjacency matrix, hence any matrix which can be converted to a lower triangular
equivalent if $B^t_1B_1 = B^t_2B_2 = \Omega$, where $\Omega$ is the (unknown) precision matrix of the underlying Gaussian distribution. In other words, they consider the equivalence class

$$\mathcal{E}(\Omega) \overset{\Delta}{=} \{ B : B \text{ is a DAG, } B^tB = \Omega \}.$$ 

The above equivalence is different from Markov equivalence. In fact, it is easy to construct examples of two DAGs in $\mathcal{E}(\Omega)$ which are not Markov equivalent and also do not have the same number of edges (see [Aragam and Zhou, 2015, Example 1]).

Clearly, elements of the equivalence class $\mathcal{E}(\Omega)$ are not identifiable from data. In the high-dimensional setting, however, it makes sense to try and identify the sparsest DAG in $\mathcal{E}(\Omega)$. The ideal approach for this goal would be to minimize an objective function consisting of the Gaussian likelihood and an $\ell_0$ penalty on $B$ (similar to the approach in van de Geer and Buhlmann [2013]). However, with a focus on computational tractability and scalability in moderate and high-dimensional settings, Aragam and Zhou [2015] consider an $\ell_1$ penalty (and other penalties, such as MCP) on $B$, and propose a coordinate-wise minimization algorithm, called the CCDr-$\ell_1$ algorithm, to minimize the resulting objective function. They demonstrate the efficacy and scalability of this approach for high-dimensional settings through various simulations.

**DAG learning with partial domain-specific background information.** In some applications, such as the dairy cattle data studied in Section 5, domain-specific background information regarding the variables is available, which allows for a partition of the variables into sets $V_1, V_2, \cdots, V_k$ such that any possible edge from a vertex in $v_i \in V_i$ to a vertex in $v_j \in V_j$ is directed from $v_i$ to $v_j$ if $i < j$. However, the ordering of the variables in the same subset is not known, and has to be inferred from the data.

Our motivation for considering such a setting comes from a dairy cattle dataset consisting of 25 economic and biological variables associated with dairy cattle operations in a region of Colombia. Grazing dairy cattle operations are characterized by complex interactions between several factors which determine the success of these systems. Most of these operations result in the collection of large amounts of data that are usually analyzed using univariate statistical models for certain variables of interest; therefore, information from relationships between these variables is ignored. In matrix by row and column permutation will be referred to as a DAG.

---

2van de Geer and Buhlmann [2013] define $B_1$ and $B_2$ to be equivalent if $B^t_1B_1 = B^t_2B_2 = \Omega$ and $B_1$ and $B_2$ have the same number of edges. Under faithfulness, this can be shown to be the same as Markov equivalence.
addition, due to the structure of a dairy cattle agroeocystem, it is of great interest to carry out data analysis that permits to implement a systemic approach (Jalvingh [1992], Thornley and France [2007]). Moreover, due to their high relevance when making management decisions and recommendations, knowing not only interaction patterns, but also causal relationships between the components of grazing dairy production systems, is a problem of current interest, and has the potential to have a marked impact on the dairy industry. In any of these systems (such as the data analyzed in Section 5), causal relationships between selected pairs of variables are reliably known, or domain-specific knowledge permits to group variables in such a way that one or more of those contained in group A have a causal effect on one or more of those in group B, but specific relationships are unknown. For example, variables measuring soil and environmental conditions affect the amount and quality of forage which in turn affects milk production. At the lowest end of the hierarchy, we have income and cost variables. See Section 5 for a detailed explanation and discussion of this hierarchy, and the partition of the relevant variables into groups based on this hierarchy. The statistical task then, is to leverage these known relationships and the observed data to estimate the underlying network of relationships between the variables under consideration. The development of statistical methods to achieve this goal will enable implementation of better data-driven decision making in animal production systems, which is extremely needed in Colombian dairy cattle operations.

Such a setting with tiered partitions of variables falls somewhere in the middle of the two extremes of having complete information regarding the ordering, and having no information regarding the ordering. Methods to incorporate background information/knowledge in DAG recovery algorithms such as the PC algorithm have been developed in the literature and are discussed in Section 3. In the current context, where background knowledge is available in the form of edge orientations, an algorithm to integrate this background knowledge into the estimated CPDAG from the PC algorithm is developed in Perkovic et al. [2017]. This approach however, can run into scalability issues in moderate/high-dimensional settings (see Table 5 in Section 4.4), and lead to inconsistencies/contradictions with the edge orientations specified by the background information due to post-hoc incorporation of the background information (see Figure 3 in Section 5).

A fast version of the GES approach (Meek [1997], Chickering [2002]), called FGES, is developed in Ramsey [2015] as part of the TETRAD project. This approach makes several simplifications/adjustments to the original GES approach for computational scalability. An implementation of FGES
with the ability to incorporate tiered partition based ordering is available in the R package \textit{r-causal}. However, we found that the computational speed of the FGES implementation with prior knowledge can suffer in high-dimensions (see Section 4.4). Also, to the best of our knowledge, high-dimensional asymptotic properties, when the number of variables \( p \) increases with the sample size \( n \), have not been established for the FGES approach.

The \( \ell_1 \)-penalty based approaches CCDr (Aragam and Zhou [2015]) and CSCS (Khare et al. [2017]) enjoy a significant computational advantage compared to other approaches in the completely unknown ordering and completely known ordering domains respectively, along with high-dimensional asymptotic performance guarantees. Thus, the goal of this paper is to develop a \textit{computationally scalable approach}, a hybrid of CCDr and CSCS, for DAG estimation which is guaranteed to produce graphs that are \textit{fully consistent with the partition based partial ordering information}, and has asymptotic high-dimensional performance guarantees. Similar to CCDr, we will rely on an appropriate notion of algebraic equivalence between DAGs (Section 2.1) and depart from the traditional faithfulness/Markov equivalence framework. However, unlike CCDr, we will show that using the tiered partition information leads to a reduction in the number of computations and more importantly allows for parallel processing. This can lead to significant improvement in computational speed and statistical performance compared to CCDr (Section 2.3). We will also demonstrate through simulations that the proposed approach can be ten or even hundred times faster than existing approaches which incorporate background information, thereby highlighting its scalability (Section 4.4). Finally, we provide a rigorous underpinning to our approach by establishing high-dimensional support recovery and estimation consistency results (Section 6).

The remainder of the paper is organized as follows. In Section 2, we develop and describe in detail, our hybrid algorithm called \textit{Partition-DAG}. Section 3 contains a discussion of related work in DAG recovery with background knowledge or interventional data. In Section 4, we perform a detailed experimental study to evaluate and understand the performance of the proposed algorithm. In Sections 4.1, 4.2 and 4.3, we use known DAGs from the DREAM3 competition (Prill et al. [2010], Marbach et al. [2010, 2009]) and perform an extensive simulation study to explore the effectiveness/ability of Partition-DAG to incorporate the ordering information, and how this ability changes with more/less informative partitions. In Section 4.4, we perform a similar simulation study, this time using randomly generated DAGs with more number of variables. In Section 5, we analyze dairy cattle data recently gathered by Universidad Nacional de Colombia using the proposed
Partition-DAG approach. Finally, in Section 6, we establish high dimensional consistency results for the proposed approach.

2. DAG estimation using partition information and the corresponding Partition-DAG algorithm. In order to understand how one can leverage the partial ordering information, it is crucial to understand the workings, similarities, and differences of the Concave penalized Coordinate Descent with reparameterization (CCDr) Aragam and Zhou [2015] and the Convex Sparse Cholesky Selection (CSCS) Khare et al. [2017] algorithms, which are state of the art (in terms on computational scalability and tractability) for the boundary settings with completely unknown and completely known variable ordering respectively.

The CSCS algorithm is derived under the setting where a domain-specific ordering of the variables which makes \( B \) lower triangular is known. Hence, the DAG estimation problem boils down to estimating the sparsity pattern in \( L \), the lower triangular permuted version of \( B \). In other words, \( L \) is a lower triangular matrix with positive diagonal entries such that \( \Omega = L^t L \).

The objective function for CSCS is

\[
Q_{cscs}(L) = \text{trace}(\Omega S) - \log |\Omega| + \lambda \sum_{1 \leq j < i \leq p} |L_{ij}|
\]

where \( S \) is the sample covariance matrix. The first two terms in \( Q_{cscs} \) correspond to the Gaussian log-likelihood, and the third term is an \( \ell_1 \) penalty term which induces sparsity in the lower triangular matrix \( L \).

The CCDr algorithm is derived under the setting where there is no knowledge about the permutation that makes \( B \) lower triangular. Here \( \Omega = B^t B \), where \( B \) varies over the space of \( p \times p \) matrices such that by permuting the rows and columns of \( B \) we can reduce it to a lower triangular matrix with positive diagonal entries. More formally, denoting \( S_p \) to be the group of permutations of \( \{1, 2, \cdots, p\} \), we assume that \( B \in S_p \), where

\[
B_p = \{B : \exists \sigma \in S_p \text{ such that } B_{\sigma(i)\sigma(j)} = 0 \text{ if } i < j\}.
\]

The CCDr-\( \ell_1 \) method uses the following objective function:

\[
Q_{CCDr}(B) = \text{trace}(\Omega S) - \log |\Omega| + \lambda \sum_{1 \leq i < j \leq p} |B_{ij}|
\]

The objective function for CSCS is

\[
Q_{cscs}(L) = \text{trace}(\Omega S) - \log |\Omega| + \lambda \sum_{1 \leq j < i \leq p} |L_{ij}|
\]

where \( S \) is the sample covariance matrix. The first two terms in \( Q_{cscs} \) correspond to the Gaussian log-likelihood, and the third term is an \( \ell_1 \) penalty term which induces sparsity in the lower triangular matrix \( L \).

The CCDr algorithm is derived under the setting where there is no knowledge about the permutation that makes \( B \) lower triangular. Here \( \Omega = B^t B \), where \( B \) varies over the space of \( p \times p \) matrices such that by permuting the rows and columns of \( B \) we can reduce it to a lower triangular matrix with positive diagonal entries. More formally, denoting \( S_p \) to be the group of permutations of \( \{1, 2, \cdots, p\} \), we assume that \( B \in S_p \), where

\[
B_p = \{B : \exists \sigma \in S_p \text{ such that } B_{\sigma(i)\sigma(j)} = 0 \text{ if } i < j\}.
\]

The CCDr-\( \ell_1 \) method uses the following objective function:

\[
Q_{CCDr}(B) = \text{trace}(\Omega S) - \log |\Omega| + \lambda \sum_{1 \leq i < j \leq p} |B_{ij}|
\]
Exactly like CSCS the first two terms correspond to the Gaussian log-likelihood, while the third term tries to impose sparsity on $B$.

While objective functions for CSCS and CCRr-$\ell_1$ look identical, the algorithms to minimize the two objective functions are very different due to the fact that we are minimizing over different spaces. Both objective functions are convex, but CSCS has the added advantage that the range for $Q_{cscs}$, which is the set of lower triangular matrices with positive diagonal entries, is convex as well. This leads to a convex problem (though not strictly convex for $n < p$) and we can establish that the sequence of iterates converges to a global minimum of the objective function. However, the range for $Q_{CCDr}(B)$, which is the set of matrices that can be converted to a lower triangular matrix with positive diagonal entries through a permutation of the rows and columns, is not convex and general results in the literature (at best) only guarantee convergence of the sequence of iterates to a local minimum of the objective function. In addition, while CSCS can be broken down into $p$ parallelizable problems the same cannot be said for CCDr, which leads to a significant computational disadvantage for CCDr. Finally, asymptotic consistency for the general setting (with no restrictions on conditional variances) for CCDr is not available as yet, whereas Theorem 4.1 in Khare et al. [2017] establishes both model selection and estimation consistency for CSCS. See Khare et al. [2017] for a more detailed comparison between these two algorithms.

As stated in the introduction, in many applications, additional data can give information about partitions of the variables where we have prior knowledge about the direction of the edges between partitions, but not within partitions (for example, the dairy cattle data in Section 5, or gene knockout data or more general perturbations data Shojaie et al. [2014a]). We will now discuss how one can create a hybrid algorithm from CSCS and CCDr where we incorporate this information for DAG estimation.

2.1. The case with two partition blocks. For simplicity we will initially work with the case where the variables, $V = \{1, ..., p\}$, are divided into two groups $V_1 = \{1, ..., m\}$ and $V_2 = \{m + 1, ..., p\}$ such that we cannot have an edge from a node in $V_2$ to one in $V_1$, but can have one from a node in $V_1$ to one in $V_2$. Hence, $\forall j \in V_1, \forall k \in V_2$ we have that $B_{kj} = 0$. This implies that $B$ has the block triangular form

\[
B = \begin{pmatrix} B_{11} & 0 \\ B_{21} & B_{22} \end{pmatrix}
\]

(2.1)

The diagonal blocks $B_{11}$ and $B_{22}$ are constrained so that each matrix is a permuted version of lower triangular matrices, i.e., $B_{11} \in B_m$ and $B_{22} \in$
$\mathcal{B}_{p-m}$, the entries of the off-diagonal block $B_{12}$ are all zero. However, there are no constraints on the off-diagonal block $B_{21}$. Hence, we focus on matrices in the space $\tilde{\mathcal{B}}_p$, defined by

$$\tilde{\mathcal{B}}_p = \{ B : B_{11} \in \mathcal{B}_m, B_{22} \in \mathcal{B}_{p-m}, B_{12} = 0 \}.$$ 

Note that, in a similar spirit to CCDr, we will not assume faithfulness, and define $B_1, B_2 \in \tilde{\mathcal{B}}$ to be equivalent if $B_1 B_1^t = B_2 B_2^t = \Omega$. In particular, with high-dimensional settings in mind, our goal is to find the sparsest DAG in the equivalence class

$$\tilde{\mathcal{E}}(\Omega) = \{ B : B \in \tilde{\mathcal{B}}_p, B^t B = \Omega \}.$$ 

For this purpose, we consider a Gaussian log-likelihood based objective function, denoted by $Q_{PDAG}$, given by

$$(2.2) \quad Q_{PDAG}(B) = \text{trace}(B^t B S) - \log |B^t B| + \lambda \sum_{1 \leq i \neq j \leq p} |B_{ij}|.$$ 

Here, our goal is to minimize the above function over the space $\tilde{\mathcal{B}}_p$ (as opposed to CCDr, where the goal is to minimize over the space $\mathcal{B}_p$). Note that since $\mathcal{B}_m$ and $\mathcal{B}_{p-m}$ are not convex sets, $\tilde{\mathcal{B}}_p$ is also not a convex set.

2.1.1. A roadmap for the algorithm. As in CCDr and CSCS, we pursue a coordinate-wise minimization approach. At each iteration, we cycle through minimizing $Q_{PDAG}$ with respect to each non-trivial element of $B$ (fixing the other entries at their current values). The minimizing value is then set as the new value of the corresponding element. We repeat the iterations until the difference between the $B$ values at two successive iterations falls below a user-defined threshold.

Hence, for implementing coordinate-wise minimization, we need to understand how to minimize $Q_{PDAG}$ with respect to an arbitrary element $B_{ij}$ of $B$ given all the other elements. Using straightforward calculations, we get

$$(2.3) \quad Q_{PDAG}(B) = \text{trace}(B^t B S) - \log |B^t B| + \lambda \sum_{1 \leq i \neq j \leq p} |B_{ij}|$$

Given the nature of the constraints on each block of $B \in \tilde{\mathcal{B}}_p$, we consider three different cases.
Case I: (Diagonal entries - $B_{ii}$) It follows from (2.3) that $Q_{PDAG}$ is the sum of quadratic and logarithmic terms in a given diagonal entry $B_{ii}$ (treating other entries as fixed). In particular,

\begin{equation}
Q_{PDAG}(B_{ii}) = S_{ii}B_{ii}^2 + 2B_{ii} \sum_{k=1, k \neq i}^{p} S_{ik}B_{ik} - 2 \log B_{ii} + \text{terms independent of } B_{ii}.
\end{equation}

Simple calculus (see for example, [Khare et al., 2015, Supplemental Section F]) shows that the unique minimizer (with respect to $B_{ii}$) of the above function is given by

\begin{equation}
\hat{B}_{ii} = -\frac{\sum_{k=1, k \neq i}^{p} S_{ik}B_{ik} + \sqrt{(\sum_{k=1, k \neq i}^{p} S_{ik}B_{ik})^2 + 4S_{ii}}}{2S_{ii}}.
\end{equation}

Case II: (Off-diagonal entries in $B_{21}$, the CSCS case) Consider $B_{ij}$, where $m+1 \leq i \leq p$ and $1 \leq j \leq m$. Since $B_{ji} = 0$, it follows from (2.3) that $Q_{PDAG}$ is the sum of quadratic and absolute value terms in $B_{ij}$ (treating other entries as fixed). In particular,

\begin{equation}
Q_{PDAG}(B_{ij}) = S_{jj}B_{ij}^2 + 2B_{ij} \sum_{k=1, k \neq j}^{p} S_{jk}B_{ik} + \lambda |B_{ij}| + \text{terms independent of } B_{ij}.
\end{equation}

It can be shown (see for example, [Mazumder et al., 2011, eq. (4)]) that the unique minimizer (with respect to $B_{ij}$) of the above function is given by

\begin{equation}
\hat{B}_{ij} = S\left(-\frac{\sum_{k=1, k \neq j}^{p} S_{jk}B_{ik}}{2S_{jj}}, \frac{\lambda}{4S_{jj}}\right),
\end{equation}

where $S(x, \lambda) = \text{sign}(x) \max\{|x| - \lambda, 0\}$. This step exactly resembles a typical step of the CSCS algorithm.

Case III: (Off-diagonal entries in $B_{11}$ and $B_{22}$, the CCDr case) Consider $B_{ij}$, where $1 \leq i \neq j \leq m$ or $m+1 \leq i \neq j \leq p$. Since $B_{11} \in B_m$ and $B_{22} \in B_{p-m}$, it follows that at most one of $B_{ij}$ or $B_{ji}$ is non-zero. So as in CCDr Aragam and Zhou [2015], we will jointly minimize $Q_{PDAG}$ as a function of $(B_{ij}, B_{ji})$. This can be done as follows. If adding a non-zero value for $B_{ij}$ violates the DAG constraint, or equivalently the constraint that $B_{11} \in B_m$ and $B_{22} \in B_{p-m}$, then
we set $B_{ij} = 0$, and then minimize $Q_{PDAG}$ as a function of $B_{ji}$ and update the $B_{ji}$ entry as specified in (2.7), (2.9) with the roles of $i$ and $j$ exchanged). If adding a non-zero value for $B_{ji}$ violates the DAG constraint, then we set $B_{ji} = 0$, and then minimize $Q_{PDAG}$ as a function of $B_{ij}$ and update the $B_{ij}$ entry as specified in (2.7), (2.9). However, it is possible that neither $|B_{ij}| > 0$ or $|B_{ji}| > 0$ violates the DAG constraint. In that case, we compute $\hat{B}_{ij}$ and $\hat{B}_{ji}$ using appropriate versions of (2.9), pick the one that makes a larger contribution towards minimizing $Q_{PDAG}$, and set the other one to zero. This step exactly resembles a typical step of the CCDr-$\ell_1$ algorithm.

The resulting coordinatewise minimization algorithm for $Q_{PDAG}$, called Partition-DAG, which repeatedly iterates through all the entries of $B$ based on the three cases discussed above, is provided in Algorithm 1. Case II and Case III, which correspond to typical steps of the CSICS and CCDr algorithm respectively, demonstrate why we regard the Partition-DAG algorithm as a hybrid of these two algorithms.

**Algorithm 1** Partition-DAG algorithm with 2 blocks

1. Input: Set initial estimate $B^o = I_p$, penalty parameter $\lambda > 0$, and tolerance $\epsilon > 0$

2. (Diagonal entries) For every $1 \leq i \leq p$, update $B_{ii}$ by minimizing $Q_{PDAG}(B_{ii})$ as in (2.4) and (2.6)

3. ($B_{21}$ entries) For every $m+1 \leq i \leq p$ and $1 \leq j \leq p$, update $B_{ij}$ as in (2.7) and (2.9)

4. ($B_{11}$ and $B_{22}$ off-diagonal entries) For every $1 \leq i,j \leq m$ and also for every $m+1 \leq i,j \leq p$ (with $i \neq j$), update the pair $(B_{ij}, B_{ji})$ as follows:

   (a) If $i \rightarrow j$ violates DAG, set $B_{ji} = 0$ and update $B_{ij}$ as in (2.7) and (2.9)

   (b) If $j \rightarrow i$ violates DAG, set $B_{ij} = 0$ and update $B_{ji}$ as in (2.7) and (2.9) (reversing the roles of $i$ and $j$)

   (c) If neither violates the DAG, choose the update (from the two choices above) which leads to a smaller value of $Q_{PDAG}$

5. Let $B^o$ denote the current updated version. If $\|B^o - B^o\|_2 \leq \epsilon$, stop. Otherwise set $B^o = B^o$, and go to Step 2

**2.2. The case with multiple partition blocks.** Algorithm 1 can be easily generalized to the case where the variables are partitioned into $R$ blocks, say, $V_1, V_2, \cdots, V_R$, such that any edge from a node $u$ in $V_i$ to a node $v$ in $V_j$ is directed from $u$ to $v$, if $i < j$. However, the ordering within each $V_i$ is not known. In particular, let $V_i = \{m_{i-1} + 1, \cdots, m_i\}$, where $m_0 = 0$ and
Under these constraints, the matrix $B$ has a block lower triangular structure, which can be denoted as follows.

\[
B = \begin{pmatrix}
B_{11} & 0 & \ldots & 0 \\
B_{21} & B_{22} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
B_{R1} & B_{R2} & \ldots & B_{RR}
\end{pmatrix}
\]  

(2.10)

In particular, the parameter $B$ lies in the space $\tilde{B}_{p,R}$ given by

\[
\tilde{B}_{p,R} = \{B : B_{ii} \in B_{m_i-m_{i-1}}, B_{rs} = 0 \text{ if } 1 \leq r < s \leq R\}.
\]

We again use a coordinate-wise minimization approach for minimizing $Q_{PDAG}$ over $\tilde{B}_{p,R}$. Similar to the two partition block case, the coordinate-wise minimizations can be divided into three cases.

- The first case deals with a diagonal entries $B_{ii}$, and the unique minimizer has exactly the same form as in (2.6).
- The second case (the CSCS case) deals with off-diagonal entries $B_{ij}$ which belong to one of the lower triangular blocks $B_{rs}$ with $1 \leq s < r \leq R$, and the unique minimizer has exactly the same form as in (2.7).
- Finally, the third case (the CCDr case) deals with off-diagonal entries $B_{ij}$ which belong to one of the diagonal blocks $B_{ii}$ with $1 \leq r \leq R$, and the unique minimizer has exactly the same form as in (2.9). The algorithm, using the steps described above, is provided as Algorithm 2.

Note that while $Q_{PDAG}$ is jointly convex in $B$, the domain of minimization $\tilde{B}_{p,R}$ is not a convex set. Hence, to the best of our knowledge, existing results in the literature do not imply convergence of the coordinate-wise minimization algorithm (Algorithm 2). Using standard arguments (for example, similar to Theorem 4.1 of Tseng [2001]), the following result can be established.

**Lemma 2.1** Assuming that all the diagonal entries of $S$ are positive, any cluster point of the sequence of iterates produced by Algorithm 2 is a stationary point of $Q_{PDAG}$ in $\tilde{B}_{p,R}$.

2.3. **Computational advantages of Partition-DAG.** Next, we discuss some of the computational and statistical advantages of using the partition based ordering information in the DAG estimation algorithms derived in this paper.
**Algorithm 2** Partition-DAG algorithm with R blocks

1. Input: Set initial estimate $B^0 = I_p$, penalty parameter $\lambda > 0$, and tolerance $\epsilon > 0$

2. (Diagonal entries) For every $1 \leq i \leq p$, update $B_{ii}$ by minimizing $Q_{PDAG}(B_{ii})$ as in (2.4) and (2.6)

3. (B_{rs} entries with $r > s$) For every $(i,j)$ such that $B_{ij}$ belongs to the lower triangular blocks $B_{rs}$ (with $r > s$), update $B_{ij}$ as in (2.7) and (2.9)

4. (B_{rr} off-diagonal entries for $1 \leq r \leq R$) For every $(i,j)$ such that $B_{ij}$ belongs to the diagonal blocks $B_{rr}$ (with $1 \leq r \leq R$) update the pair $(B_{ij}, B_{ji})$ as follows:

   (a) If $i \rightarrow j$ violates DAG, set $B_{ji} = 0$ and update $B_{ij}$ as in (2.7) and (2.9)

   (b) If $j \rightarrow i$ violates DAG, set $B_{ij} = 0$ and update $B_{ji}$ as in (2.7) and (2.9) (reversing the roles of $i$ and $j$)

   (c) If neither violates the DAG, choose the update (from the two choices above) which leads to a smaller value of $Q_{PDAG}$

5. Let $B^n$ denote the current updated version. If $\|B^n - B^0\|_2 \leq \epsilon$, stop. Otherwise set $B^0 = B^n$, and go to Step 2

1. (Parallelizability) Consider the general multiple block case described in Section 2.2. After some manipulations, it can be shown that the objective function $Q_{PDAG}$ can be decomposed as a sum of $R$ functions, where each function exclusively uses entries from a distinct block row of $B$. In particular, it can be shown that

$$Q_{PDAG}(B) = \sum_{r=1}^{R} Q_r(B_{r1}, B_{r2}, \cdots, B_{rr}),$$

where

$$Q_r(B_{r1}, B_{r2}, \cdots, B_{rr}) = \sum_{i=m_r-1+1}^{m_r} \left( \left( \sum_{h=1}^{m_r} S_{hh} B_{ih}^2 + 2 \sum_{k=1}^{m_r-1} \sum_{l=k+1}^{m_r} S_{kl} B_{ik} B_{il} \right) - \log B_{ii} + \lambda \sum_{k=1, k \neq i}^{m_r} |B_{ik}| \right)$$

only depends on the terms in block row $r$. As a result, the minimization of each block row can be implemented in parallel as shown in Algorithm 2. This can lead to huge computational advantages, as illustrated in our experiments.
2. (Number of operations) With the additional partition information, many of the entries in $B$ are automatically set to zero. This reduces the number of operations Partition-DAG performs in comparison to CCDr. In addition, many of the computations for Partition-DAG fall under Case II as discussed in Section 2.1, which is much simpler and faster than computations under Case III, which is what CCDr needs to do for every single coordinate.

3. (Estimation Accuracy) This is very obvious, but leveraging more information can lead to improved estimation accuracy as demonstrated in our simulations studies.

3. Related Work. The problem studied in this paper relates to the problem of DAG recovery from observational data in the presence of additional background knowledge or interventional data. As discussed in the introduction, in a traditional setting with the faithfulness assumption, two Markov equivalent graphs cannot in general be distinguished from observational data, and only a class of Markov equivalent graphs (represented by a CPDAG) can be identified. Methods to reduce the number of DAGs in this class by incorporating background knowledge in the form of edge orientations or partial ordering have been developed in Meek [1995] and Scheines et al. [1998] respectively. The approach by Meek [1995] has been implemented in the `addBgKnowledge` function of the R package `pcalg`, and can be applied to the output of the popular PC algorithm (Spirtes et al. [2001]) to obtain a reduced class of DAGs (see Perkovic et al. [2017]). We examine this approach in the simulations in Section 4.1. Note that while `addBgKnowledge` does a post-hoc incorporation of background knowledge (after a CPDAG is estimated), the `fixedGaps` and `fixedEdges` options in the `pc` function allow the user to a priori guarantee the existence or absence of certain undirected edges in the estimated CPDAG (such information might be harder to come by than absence/presence of directed edges). However, these options do not allow a priori specification of a tiered partition based ordering. A fast version of the scoring based GES algorithm introduced in Meek [1997], Chickering [2002], called FGES, has been implemented as part of the TETRAD project (see Ramsey [2015]). The corresponding algorithm (for both discrete and continuous settings) is implemented in java, and allows a priori specification of a tiered ordering of variables. We provide accuracy and time comparisons with FGES for continuous variables in the simulations in Section 4.4.

Using a structural equations representation, and assuming that all the errors are independent and non-Gaussian, Shimizu et al. [2006] show that it is possible to identify the exact underlying DAG by using a method based on
Independent Component Analysis (ICA). [Shimizu et al., 2011, Section 3.4] develop a more efficient version of this approach, and are able to incorporate prior information about the existence (non-existence) of a directed path between pairs of vertices. Hoyer et al. [2008] develop the PC-LINGAM method, which is a hybrid approach based on ICA and the conditional independence based PC algorithm. This approach is implemented in the LINGAM function of the R package pcalg, and is also examined in the simulations in Section 4.1. In a different context, the ARGES algorithm in Nandy et al. [2018] can incorporate background information that is available in the form of existence/non-existence of undirected edges in the skeleton.

There is another line of work which develops methods for DAG learning based on observational data combined with interventional data (see Hauser and Buhlmann [2012], Shojaie et al. [2014b], Wang et al. [2017], Yang et al. [2018]). Again, the idea is that under certain conditions, information from interventional data can be used to restrict the equivalence class represented by a CPDAG to a more restricted “maximal” class of partial DAGs. Methods for inference (testing) in this setting have also been explored in Peters et al. [2016]. To the best of our understanding, the interventional approach considered in these papers is not directly related to the tiered/partial ordering based setting considered in this paper. Finally, in a recent paper, Li et al. [2020] have focused on high-dimensional inference on the pairwise directed edges in a DAG using constrained likelihood ratio tests. Their method is based on the structural equations interpretation, and involves minimizing the Gaussian likelihood subject to DAG constraints and additional constraints forcing some of the parameters to be zero. However, the error variables in the SEM interpretation are assumed to have equal variances, and given the different goal, there is no $\ell_1$ penalty term in the minimization.

4. Experiments. In this section, we perform extensive simulations to understand/explore the following questions about the Partition-DAG algorithm in realistic settings.

- Can Partition-DAG effectively leverage the partial ordering information to improve performance (as compared to methods such as CCDr which do not incorporate any ordering based information)? How does it perform compared to other DAG estimation algorithms which can incorporate background information?
- As the partitions become finer, does the performance of Partition-DAG improve?
- If the number of sets in the partition is kept the same, but the elements of these sets are changes so that the partition is more informative, then
does the performance of Partition-DAG improve?

- How does the computational speed of Partition-DAG compare to other DAG estimation algorithms which can incorporate background information? How does the computational speed change as the partition becomes finer?

We investigate each of these questions separately in the subsections below. The following algorithms will be used in the subsequent comparisons as appropriate.

- ParDAG: The $\ell_1$ penalty based Partition-DAG algorithm described in Algorithm 2 which leverages partition based ordering information.
- CCDR: The $\ell_1$ penalty based algorithm in Aragam and Zhou [2015] (no ordering information is assumed to be known)
- PC: The testing based PC algorithm from Spirtes et al. [2001] (with no ordering information is assumed to be known).
- LINGAM: The PC-LINGAM algorithm from Hoyer et al. [2008] as implemented in the `LINGAM` function of the R package `pcalg`.
- CSCS: The $\ell_1$ penalty based CSCS algorithm from Khare et al. [2017] which assumes knowledge of the complete ordering of the variables in the dataset.

4.1. Partial ordering info vs. no ordering info: DREAM3 data. The goal of this experiment is to explore if partition based ordering information can help improve accuracy and computational efficiency in realistic settings. With this in mind, we perform a number of simulation studies using gene regulatory networks from the DREAM3 In Silico challenge Prill et al. [2010], Marbach et al. [2010, 2009]. This challenge provides the transcriptional networks for three yeast organisms, which we will denote as Yeast 1, Yeast 2, Yeast 3. These networks mimic activations and regulations that occur in gene regulatory networks. All networks are known and have 50 nodes.

For each DAG, we generated a random $B$ by sampling the off-diagonal non-zero terms from a uniform distribution between 0.3 and 0.7 and assigned them a positive or negative sign with equal probability. The diagonal terms were all set to 1. Then, the “true” $\Omega = B'B$ was computed, and the cor-
responding multivariate Gaussian distribution was used to generate twenty datasets each for sample size \( n \in \{40, 50, 100, 200\} \). The partition provided to ParDAG and PCBGK consists of two sets, one with nodes 1 to 25, and the other with nodes 26 to 50.

For each sample size, each of the algorithms was run for each dataset described above for a range of penalty parameter values (for the PC algorithm and its variants, the significance level for the hypothesis tests was used as a penalty parameter). Note that the DAG estimation problem is a three way classification problem (two classes corresponding to two kinds of directed edges, and one class corresponding to no edge). Hence, the performance was summarized using the corresponding mean AUC-MA (Macro-averaged Area-Under-the-Curve, see for example Tsoumakas et al. [2010]) value over the twenty repetitions. For each method, the appropriate penalty parameter (or significance level in the case of the PC algorithm and its variants) was varied over a range of 30 values to yield a completely sparse network at one end, and a completely dense network (ignoring edge directions) at the other end. The AUC values for the three binary classification problems (corresponding to each class) were then computed and normalized by dividing with the respective false positive rate (FPR) range. The AUC-MA was then computed by taking the average of the three class-wise AUC values. The results for the three different networks (Yeast1, Yeast2, Yeast3) are summarized in Table 1. Since the partition information provided to ParDAG and PCBGK consists of two sets, we refer to the respective methods as ParDAG-2 and PCBGK-2 in the table.

Note that while the Partition-DAG and CCDR algorithms provide a fully-oriented DAG as an output, the PC algorithm and its variants in general provide a partially oriented CPDAG or a maximally oriented PDAG as an output. In a simulation setting, one possibility is to direct all the undirected edges in the correct way, and use the resulting DAG to evaluate the appropriate performance measure. This value, denoted by \( PC_{\text{best}} \), corresponds to the best possible performance by the PC algorithm. Another possibility is to direct all the undirected edges in the wrong way, and use the resulting DAG to evaluate the appropriate performance measure. This value, denoted by \( PC_{\text{worst}} \), corresponds to the worst possible performance by the PC algorithm. Hence, the performance measure for the PC algorithm is reported using the interval \([PC_{\text{worst}}, PC_{\text{best}}]\) instead of a single value. The same holds for PC-PAR, PC-STAB and PCBGK (which provides a maximally oriented PDAG).

As expected, the performance of each method improves with increasing sample size. It is clear that Partition-DAG outperforms the PC algorithm,
the PC algorithm with background knowledge and the CCDr algorithm for all the DAGs by quite a large margin and the performance is more or less consistent regardless of the complexity of the DAG topology. An interesting observation that could be made is that adding background knowledge does not seem to improve the performance of the PC algorithm in these settings. Note that while adding the background knowledge correctly orients some of the non-oriented edges in the CPDAG provided by the PC algorithm, the process can orient further edges by using the orientation rules in Meek [1995]. Since the estimated CPDAG has some incorrectly oriented/identified edges (compared to the underlying true DAG), these further oriented edges can potentially be incorrectly oriented/identified (compared to the underlying true DAG). This explains why the best AUC-MA value for the PC algorithm (obtained by correctly orienting all non-oriented edges in the estimated CPDAG) is marginally better than the corresponding best AUC-MA value with background knowledge (PCBGK).

4.2. Fine partition vs. coarse partition: Yeast 3 network. Note that given a partition, Partition-DAG assumes the ordering of edges between the sets in the partition to be known, and the ordering of edges within each partition set to be unknown. The goal of this subsection is to explore if using a finer partition (and hence more knowledge of the ordering) improves the performance of the Partition-DAG approach. We perform simulations using the Yeast 3 data from the DREAM3 challenge mentioned in Section 4.1. Recall that the underlying network is known, mimics activations and regulations that occur in gene regulatory networks and has \( p = 50 \) nodes. We topologically order the nodes from 1 to 50, so that any edge directs from a smaller node to a bigger node. Again, we construct a “true” \( \Omega \) matrix consistent with the known DAG, and generated hundred multivariate datasets each for sample size \( n \in \{40, 50, 100, 200\} \). We analyze each of the 400 datasets thus generated using several methods: Partition-DAG with a partition consisting of two sets: one with nodes 1 to 36, and the other one with nodes 37 to 50 (referred to as ParDAG-2), Partition-DAG with a partition consisting of three sets: one with the first 24 nodes, one with nodes 25 to 36, and one with nodes 37 to 50 (referred to as ParDAG-3), Partition-DAG with a partition consisting of four sets with one with the first 12 nodes, one with nodes 13 to 24, one with nodes 25 to 36, and one with nodes 37 to 50 (referred to as ParDAG-4), PC algorithm with background knowledge using the above three partitions (PCBGK-2, PCBGK-3, PCBGK-4), and algorithms such as CCDr, original PC and PC-LINGAM which do not use any partition based information. The Macro-averaged Area-Under-the-Curve (AUC-MA) values
Macro Averaged Area-Under-the-Curve (AUC-MA) values for Yeast 1, Yeast 2, and Yeast 3 networks for CCDr, Partition-DAG with partition containing two sets (ParDAG-2), PC algorithm (PC), Stable PC algorithm (PC-STAB), Parallel-PC algorithm (PC-PAR), and PC algorithm with background knowledge using two-set partition (PCBGK-2). PC algorithm variants provide partially oriented DAGs as output, and best/worst AUC-MA values are provided. These are obtained by orienting the non-oriented edges in the output correctly/incorrectly.

<table>
<thead>
<tr>
<th>Method</th>
<th>Sample size</th>
<th>AUC-MA Yeast 1</th>
<th>AUC-MA Yeast 2</th>
<th>AUC-MA Yeast 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCDr</td>
<td>40</td>
<td>0.5254</td>
<td>0.4393</td>
<td>0.4438</td>
</tr>
<tr>
<td>LINGAM</td>
<td>40</td>
<td>0.4393</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>PC</td>
<td>40</td>
<td>[0.4722, 0.5021]</td>
<td>0.4103, 0.4285</td>
<td>0.4171, 0.4323</td>
</tr>
<tr>
<td>PC-PAR</td>
<td>40</td>
<td>[0.4722, 0.5021]</td>
<td>0.4103, 0.4285</td>
<td>0.4171, 0.4323</td>
</tr>
<tr>
<td>PC-STAB</td>
<td>40</td>
<td>[0.4722, 0.5021]</td>
<td>0.4103, 0.4285</td>
<td>0.4171, 0.4323</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>40</td>
<td>[0.4751, 0.5015]</td>
<td>[0.4140, 0.4282]</td>
<td>[0.4196, 0.4315]</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>40</td>
<td>0.6436</td>
<td>0.5963</td>
<td>0.5659</td>
</tr>
<tr>
<td>CCDr</td>
<td>50</td>
<td>0.5213</td>
<td>0.4492</td>
<td>0.4513</td>
</tr>
<tr>
<td>LINGAM</td>
<td>50</td>
<td>0.4690</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>PC</td>
<td>50</td>
<td>[0.4894, 0.5286]</td>
<td>[0.4213, 0.4415]</td>
<td>[0.4255, 0.4392]</td>
</tr>
<tr>
<td>PC-PAR</td>
<td>50</td>
<td>[0.4894, 0.5286]</td>
<td>[0.4213, 0.4415]</td>
<td>[0.4255, 0.4392]</td>
</tr>
<tr>
<td>PC-STAB</td>
<td>50</td>
<td>[0.4894, 0.5286]</td>
<td>[0.4213, 0.4415]</td>
<td>[0.4255, 0.4392]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>50</td>
<td>[0.4931, 0.5288]</td>
<td>[0.4238, 0.4414]</td>
<td>[0.4271, 0.4391]</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>50</td>
<td>0.6543</td>
<td>0.6087</td>
<td>0.5900</td>
</tr>
<tr>
<td>CCDr</td>
<td>100</td>
<td>0.5628</td>
<td>0.4812</td>
<td>0.4746</td>
</tr>
<tr>
<td>LINGAM</td>
<td>100</td>
<td>0.5227</td>
<td>0.4541</td>
<td>0.4589</td>
</tr>
<tr>
<td>PC</td>
<td>100</td>
<td>[0.5475, 0.5770]</td>
<td>[0.4492, 0.4647]</td>
<td>[0.4584, 0.4670]</td>
</tr>
<tr>
<td>PC-PAR</td>
<td>100</td>
<td>[0.5475, 0.5770]</td>
<td>[0.4492, 0.4647]</td>
<td>[0.4584, 0.4670]</td>
</tr>
<tr>
<td>PC-STAB</td>
<td>100</td>
<td>[0.5475, 0.5770]</td>
<td>[0.4492, 0.4647]</td>
<td>[0.4584, 0.4670]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>100</td>
<td>[0.5476, 0.5766]</td>
<td>[0.4484, 0.4635]</td>
<td>[0.4590, 0.4667]</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>100</td>
<td>0.6958</td>
<td>0.6375</td>
<td>0.5994</td>
</tr>
<tr>
<td>CCDr</td>
<td>200</td>
<td>0.5866</td>
<td>0.4922</td>
<td>0.4900</td>
</tr>
<tr>
<td>LINGAM</td>
<td>200</td>
<td>0.5384</td>
<td>0.4523</td>
<td>0.4640</td>
</tr>
<tr>
<td>PC</td>
<td>200</td>
<td>[0.5845, 0.6027]</td>
<td>[0.4806, 0.4936]</td>
<td>[0.4811, 0.4865]</td>
</tr>
<tr>
<td>PC-PAR</td>
<td>200</td>
<td>[0.5845, 0.6027]</td>
<td>[0.4806, 0.4936]</td>
<td>[0.4811, 0.4865]</td>
</tr>
<tr>
<td>PC-STAB</td>
<td>200</td>
<td>[0.5845, 0.6027]</td>
<td>[0.4806, 0.4936]</td>
<td>[0.4811, 0.4865]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>200</td>
<td>[0.5840, 0.6017]</td>
<td>[0.4814, 0.4936]</td>
<td>[0.4807, 0.4864]</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>200</td>
<td>0.6975</td>
<td>0.6539</td>
<td>0.6225</td>
</tr>
</tbody>
</table>

for each algorithm are provided in Table 2. We see that the performance of Partition-DAG as well as PC algorithm with background knowledge improves with finer partitions (which correspond to more background information). As in Section 4.1, Partition-DAG outperforms the PC algorithm, the PC algorithm with background knowledge and the CCDr algorithm for all the DAGs by quite a large margin for all sample sizes.
<table>
<thead>
<tr>
<th>Method</th>
<th>Sample size</th>
<th>AUC-MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCDr</td>
<td>40</td>
<td>0.4429</td>
</tr>
<tr>
<td>LINGAM</td>
<td>40</td>
<td>NA</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>40</td>
<td>0.5651</td>
</tr>
<tr>
<td>ParDAG-3</td>
<td>40</td>
<td>0.5624</td>
</tr>
<tr>
<td>ParDAG-4</td>
<td>40</td>
<td>0.5937</td>
</tr>
<tr>
<td>CSCS</td>
<td>40</td>
<td>0.8600</td>
</tr>
<tr>
<td>PC</td>
<td>40</td>
<td>[0.4148,0.4393]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>40</td>
<td>[0.4158,0.4393]</td>
</tr>
<tr>
<td>PCBGK-3</td>
<td>40</td>
<td>[0.4167,0.4407]</td>
</tr>
<tr>
<td>PCBGK-4</td>
<td>40</td>
<td>[0.4184,0.4408]</td>
</tr>
<tr>
<td>CCDr</td>
<td>50</td>
<td>0.4400</td>
</tr>
<tr>
<td>LINGAM</td>
<td>50</td>
<td>0.4188</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>50</td>
<td>0.5756</td>
</tr>
<tr>
<td>ParDAG-3</td>
<td>50</td>
<td>0.5898</td>
</tr>
<tr>
<td>ParDAG-4</td>
<td>50</td>
<td>0.6136</td>
</tr>
<tr>
<td>CSCS</td>
<td>50</td>
<td>0.8898</td>
</tr>
<tr>
<td>PC</td>
<td>50</td>
<td>[0.4205,0.4546]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>50</td>
<td>[0.4209,0.4545]</td>
</tr>
<tr>
<td>PCBGK-3</td>
<td>50</td>
<td>[0.4220,0.4540]</td>
</tr>
<tr>
<td>PCBGK-4</td>
<td>50</td>
<td>[0.4242,0.4537]</td>
</tr>
<tr>
<td>CCDr</td>
<td>100</td>
<td>0.4624</td>
</tr>
<tr>
<td>LINGAM</td>
<td>100</td>
<td>0.4715</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>100</td>
<td>0.6059</td>
</tr>
<tr>
<td>ParDAG-3</td>
<td>100</td>
<td>0.6142</td>
</tr>
<tr>
<td>ParDAG-4</td>
<td>100</td>
<td>0.6482</td>
</tr>
<tr>
<td>CSCS</td>
<td>100</td>
<td>0.9454</td>
</tr>
<tr>
<td>PC</td>
<td>100</td>
<td>[0.4519,0.4834]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>100</td>
<td>[0.4513,0.4830]</td>
</tr>
<tr>
<td>PCBGK-3</td>
<td>100</td>
<td>[0.4531,0.4825]</td>
</tr>
<tr>
<td>PCBGK-4</td>
<td>100</td>
<td>[0.4535,0.4822]</td>
</tr>
<tr>
<td>CCDr</td>
<td>200</td>
<td>0.4741</td>
</tr>
<tr>
<td>LINGAM</td>
<td>200</td>
<td>0.4779</td>
</tr>
<tr>
<td>ParDAG-2</td>
<td>200</td>
<td>0.6206</td>
</tr>
<tr>
<td>ParDAG-3</td>
<td>200</td>
<td>0.6314</td>
</tr>
<tr>
<td>ParDAG-4</td>
<td>200</td>
<td>0.6644</td>
</tr>
<tr>
<td>CSCS</td>
<td>200</td>
<td>0.9737</td>
</tr>
<tr>
<td>PC</td>
<td>200</td>
<td>[0.4845,0.5308]</td>
</tr>
<tr>
<td>PCBGK-2</td>
<td>200</td>
<td>[0.4840,0.5035]</td>
</tr>
<tr>
<td>PCBGK-3</td>
<td>200</td>
<td>[0.4833,0.5026]</td>
</tr>
<tr>
<td>PCBGK-4</td>
<td>200</td>
<td>[0.4819,0.5022]</td>
</tr>
</tbody>
</table>

Table 2 Macro Averaged Area-Under-the-Curve (AUC-MA) values for the Yeast 3 network for CCDr, Partition-DAG with partition containing two sets (ParDAG-2), three sets (ParDAG-3) and four sets (ParDAG-4), CSCS, PC algorithm (PC), Stable PC algorithm (PC-STAB), Parallel-PC algorithm (PC-PAR), and PC algorithm with background knowledge using two set (PCBGK-2), three set (PCBGK-3) and four set (PCBGK-4) partitions. PC algorithm variants provide partially oriented DAGs as output, and best/worst AUC-MA values are provided. These are obtained by orienting the non-oriented edges in the output correctly/incorrectly.
4.3. **Informative vs. Non-informative partition: Yeast 1 network.** In this section, we compare the performance of Partition-DAG using two different partitions for the Yeast 1 network. Recall that this network has 50 nodes. We topologically order the nodes from 1 to 50 so that any edge in the true network directs from a smaller node to a larger node. The first partition consists of two sets: \( V_1 = \{4, 9, 11, 16, 24\} \) and its complement. The second partition also consists of two sets: \( V_2 = \{9, 10, 11, 13, 16\} \) and its complement. The partitions are constructed such that in the true graph, any edge between \( V_i \) and \( V_i^c \) directs from \( V_i \) to \( V_i^c \) for \( i = 1, 2 \). The first partition is more “informative” in the sense that in the true network, more edges exist between \( V_1 \) and \( V_1^c \) as compared to edges between \( V_2 \) and \( V_2^c \). Similar to earlier subsections, we generated hundred multivariate datasets (from a multivariate Gaussian consistent with the true network) each for sample size \( n \in \{50, 100, 200\} \), and applied Partition-DAG with the two different partitions discussed above (referred to as ParDAG-INFO and ParDAG-NONINFO) for a range of penalty parameter values. The Macro-averaged Area-Under-the-Curve (AUC-MA) values are provided in Table 3. These results show that the performance of Partition-DAG improves as we go from an non-informative ordering to an informative ordering, but the difference between the two AUC values grows smaller with increasing sample size.

<table>
<thead>
<tr>
<th>Method</th>
<th>Sample size</th>
<th>AUC-MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParDAG-INFO</td>
<td>40</td>
<td>0.6272</td>
</tr>
<tr>
<td>ParDAG-NONINFO</td>
<td>40</td>
<td>0.5864</td>
</tr>
<tr>
<td>ParDAG-INFO</td>
<td>50</td>
<td>0.6321</td>
</tr>
<tr>
<td>ParDAG-NONINFO</td>
<td>50</td>
<td>0.6025</td>
</tr>
<tr>
<td>ParDAG-INFO</td>
<td>100</td>
<td>0.6617</td>
</tr>
<tr>
<td>ParDAG-NONINFO</td>
<td>100</td>
<td>0.6263</td>
</tr>
<tr>
<td>ParDAG-INFO</td>
<td>200</td>
<td>0.6801</td>
</tr>
<tr>
<td>ParDAG-NONINFO</td>
<td>200</td>
<td>0.6401</td>
</tr>
</tbody>
</table>

Table 3

*Macro Averaged Area-Under-the-Curve (AUC-MA) values for the Yeast 1 network for Partition DAG with informative (ParDAG-INFO) vs. non-informative (ParDAG-NONINFO) partitions*

4.4. **Scaling the number of nodes and computational time.** In this section, we consider networks with higher number of nodes (\( p = 100 \) and \( p = 200 \)) to investigate if the parallelizability for Partition-DAG helps improve computational speed as well as statistical accuracy as the partitions grow finer, and also to compare the scalability with existing PC algorithm based approaches. To this end, we generated a “true” \( \mathbf{B} \) matrix of size 100 \( \times \) 100 with a random sparsity pattern of about 95% following DAG restrictions
and conditions similar to those mentioned in previous subsections. We then set $\Omega = \mathbf{B}^T \mathbf{B}$ and generated 20 datasets with multivariate normal distribution with mean 0 and variance $\Sigma = \Omega^{-1}$. The nodes were topologically ordered and partitioned into 4 sets of equal size: $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4$. For each dataset, the following algorithms were used: Partition-DAG with two sets $V_1$ and $V_2 \cup V_3 \cup V_4$ (referred to as ParDAG-2), Partition-DAG with three sets $V_1, V_2$ and $V_3 \cup V_4$ (referred to as ParDAG-3), Partition-DAG with four sets $V_1, V_2, V_3$ and $V_4$ (referred to as ParDAG-4), the PC algorithm, the Parallel PC algorithm (PC-PAR) and PC-LINGAM algorithm (LINGAM).

In this section, we also include computational time and performance accuracy comparisons with the FGES approach in Ramsey [2015]. Note that this approach allows for a priori specification of the tiered ordering information. The FGES algorithm is implemented in the R package r-causal, which provides and R wrapper for the Java based TETRAD library. We use FGES with the four set partition specified above (FGES-4).

Table 4 provides the AUC-MA values, and as expected ParDAG-4 performs the best, followed by ParDAG-3 and then ParDAG-2. Similar to the experiments in Section 4.1 and Section 4.2, the Partition-DAG based approaches perform significantly better than variants of the PC algorithm and also the FGES algorithm.

Table 5 provides the average wall-clock time needed for each algorithm above. During revisions, we implemented the Partition-DAG algorithm in Python (original implementation was purely in R) for additional scalability. The column (ParD4 (Py)) in Table 5 corresponds to average wall-clock times for Partition-DAG obtained using this Python-based implementation. It is clear that the time improves drastically as we consider finer partitions, which is partly due to the fact that we are doing more of the processing in parallel for finer partitions. The results also demonstrate the significant computational scalability of the proposed Partition-DAG (especially the Python based implementation) compared to variants of the PC algorithm.

An interesting finding was that the Parallel-PC algorithm developed in Le et al. [2015] had roughly the same wall-clock time as the PC algorithm. The Parallel-PC algorithm aims to increase the speed of the stable PC algorithm by distributing the testing part of the algorithm to multiple available cores, and then synchronizing the results. This synchronization is needed after every iteration of the PC algorithm, and the cost of synchronization grows with the number of cores used for parallelization. As a result, while the overall computational time for the Parallel-PC algorithm initially decreases with additional cores, it eventually starts increasing with increasing number of cores as the cost of synchronization offsets any benefits given by
parallelization of the testing step. This is further illustrated by the results in Table 5 which were run on a 32 core system. On the other hand, as discussed in Section 2.3, the \( R \) block Partition-DAG algorithm can be more effectively parallelized by distributing the optimization for each of the \( R \) appropriate blocks of the \( B \) matrix with no need for synchronization.

Another interesting observation that comes out from Table 5 is that the computational time for the PC algorithm and its variants increases with increasing sample size. This is because the number of independence tests performed increases with increasing sample size. Note that the same default significance level of 0.01 was used for all the settings in Table 5. On the other hand, the time taken for the Partition-DAG algorithm in Table 5 decreases with increasing sample size. The only way that the sample size affects the objective function \( Q_{PDAG} \) and the Partition-DAG algorithm is through the sample covariance matrix \( S = S_n \). A larger sample size results in \( S_n \) being closer to the true underlying population covariance matrix, and the convergence threshold is achieved in lesser number of iterations (the time taken per iteration is roughly the same across different sample sizes).

Finally, we found that the runtime for the FGES algorithm increases when partition-based prior knowledge was added using forbidden directed edges (as compared to no prior knowledge). As an example, for a setting with the true DAG (\( p = 500 \)) having 10% edge density and a tiered partition-based ordering with 4 sets of 125 vertices each, the FGES algorithm (with default penalty parameter) had a runtime of roughly three hours with prior knowledge and an average runtime of roughly one minute without prior knowledge. This is somewhat counter-intuitive, and could be an issue with how prior knowledge incorporation (in the form of forbidden directed edges) for FGES is implemented in the \r-causal package. The \texttt{Python} based implementation of PDAG with prior knowledge only takes 40 seconds runtime in this setting (averaged over a grid of penalty parameter values). To conclude, the accuracy comparisons in Table 4, the runtime comparisons in Table 5, and the high-dimensional asymptotic guarantees in Section 6 below demonstrate that the proposed Partition-DAG algorithm is a useful addition to the toolbox for DAG recovery with tiered prior knowledge.

5. Analysis of dairy cattle data. Current animal production systems are characterized by the recording of large amounts of data and decision-making based on the information obtained after carrying out statistical analyses. In particular, a typical dairy cattle production system records data on a daily basis, but nowadays, thanks to the advent of the so-called precision agriculture, the number of variables considered has increased dramatically.
and some herds produce records every hour or even every minute (Morota et al. [2018]); consequently, the amount of records is increasing at a very high rate. A major concern in the analysis of data collected in dairy farms, is the lack of a holistic approach that permits inferring causality considering different types of variables (e.g., animal, grassland, economic) and expert knowledge. Most of the current approaches involve separate univariate analyses that ignore the systemic nature of the process taking place in grazing dairy farms.

Certain pairs of variables have known relationships which are known at different degrees: direction, direction and sign and in certain special cases, direction, sign and magnitude (approximately). However, this domain-specific knowledge does not permit to identify all relationships; consequently, there is a need for a causal inference method that incorporates background knowledge and scales easily. The last feature is extremely relevant due to the mas-
sive implementation of precision agriculture that is expected to take place soon.

Due to these challenges, current research in dairy sciences, especially in developing countries, aims at collecting data at different levels of the production system and looks for statistical methods to analyze them implementing a systemic approach.

5.1. Data background. In a recent research project led by the Universidad Nacional de Colombia and aimed to increase productivity level in high-tropic dairy operations, data on $p = 25$ economic and biological variables associated with dairy cattle operations were collected from $n = 25$ high-tropic dairy farms in the municipality of Guatavita, Department of Cundinamarca, Colombia spanning the period from June, 2016 to August, 2017. A list of the variables, along with associated acronyms is provided in Table 6.

The variables were divided into 9 groups according to causal relationships on the basis of domain-specific knowledge by an expert in dairy science (Dr. Carulla, co-author). Specifically, knowledge on the hierarchical structure of a grazing dairy cattle operation was used (Elgersma et al. [2006]). Colombian dairy operations are based on grasslands; thus, animals are mostly fed fresh forages. This hierarchy follows the natural production flow, it all starts at the soil and environmental conditions (temperature, light, rain), which highly affect the amount and quality of forage (Dillon [2006]). Then, several variables associated to pasture management, cows’ supplementation, and their genetic makeup determine efficiency in the complex process of transforming forage into milk (Bargo et al. [2003]). Since an animal production system is a business, there is always interest in maximizing profit, which is explained by a large number of variables and their interaction, but it is summarized by a simple number: net income. Therefore, at the end of the hierarchy, that is, in the final causal groups, we find economic variables, basically, costs and incomes.

Consequently, pasture growth rate is in the first causal group because this variable can be thought of as an output of the interaction between soil, environment, pasture genetics and pasture management and it highly determines the stocking rate (number of individuals or live weight per unit of area). Along with pasture growth rate, total grazing area defines the number of cows a herd can hold; as a result, the first group comprised of these two variables. Stocking rate is computed as the total number of individuals or total live weight in the herd divided by total grazing area; hence, stocking rate was assigned to the second group. The following groups
contain variables associated to forage allowance and total feed intake, milk yield, and resources used in milk production such as number of workers. The last group comprises of two relevant economic variables: total net income and net income per worker.

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DETAILED MEANING</th>
<th>GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PGR&quot;</td>
<td>Pasture growth rate</td>
<td>Group 1</td>
</tr>
<tr>
<td>&quot;AP&quot;</td>
<td>Total pasture (hectare) area of each herd</td>
<td>Group 1</td>
</tr>
<tr>
<td>&quot;SR&quot;</td>
<td>Stocking rate, i.e., no of individuals per hectare</td>
<td>Group 2</td>
</tr>
<tr>
<td>&quot;OF&quot;</td>
<td>Amount of offered forage (kg per individual)</td>
<td>Group 3</td>
</tr>
<tr>
<td>&quot;AFI&quot;</td>
<td>Average forage intake (kg per individual)</td>
<td>Group 4</td>
</tr>
<tr>
<td>&quot;TFI&quot;</td>
<td>Average total (forage + supp) feed intake (kg per individual)</td>
<td>Group 4</td>
</tr>
<tr>
<td>&quot;AMC&quot;</td>
<td>Average of milking cows in the herd</td>
<td>Group 5</td>
</tr>
<tr>
<td>&quot;AMY&quot;</td>
<td>Average milk yield (lt) per cow per day</td>
<td>Group 5</td>
</tr>
<tr>
<td>&quot;ATS&quot;</td>
<td>Average total solids of milk (%)</td>
<td>Group 5</td>
</tr>
<tr>
<td>&quot;NW&quot;</td>
<td>Number of workers in the herd per month</td>
<td>Group 6</td>
</tr>
<tr>
<td>&quot;SM&quot;</td>
<td>Amount of sold milk (lt) per month</td>
<td>Group 6</td>
</tr>
<tr>
<td>&quot;MH&quot;</td>
<td>Amount of milk per hectare per month</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;TS&quot;</td>
<td>Total solids produced per hectare per day</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;TSC&quot;</td>
<td>Total solids produced per cow per month</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;AMW&quot;</td>
<td>Amount of milk (lt) per worker per month</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;CSC&quot;</td>
<td>Cost (Colombian pesos) of soil correction strategies</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;PMC&quot;</td>
<td>Nutritional and pasture management cost (Colombian pesos)</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;EVM&quot;</td>
<td>Economic value of milk (Colombian pesos per litter)</td>
<td>Group 7</td>
</tr>
<tr>
<td>&quot;MPC&quot;</td>
<td>Milk production cost per litter</td>
<td>Group 8</td>
</tr>
<tr>
<td>&quot;TI&quot;</td>
<td>Total income (Colombian pesos)</td>
<td>Group 8</td>
</tr>
<tr>
<td>&quot;WC&quot;</td>
<td>Cost per worker per month (Colombian pesos)</td>
<td>Group 8</td>
</tr>
<tr>
<td>&quot;CC&quot;</td>
<td>Cost per cow per month (Colombian pesos)</td>
<td>Group 8</td>
</tr>
<tr>
<td>&quot;CH&quot;</td>
<td>Cost per hectare per month (Colombian pesos)</td>
<td>Group 8</td>
</tr>
<tr>
<td>&quot;NI&quot;</td>
<td>Net income (Colombian pesos)</td>
<td>Group 9</td>
</tr>
<tr>
<td>&quot;WI&quot;</td>
<td>Income per worker per month (Colombian pesos)</td>
<td>Group 9</td>
</tr>
</tbody>
</table>

Table 6
Detailed list of dairy agroecosystem variables used in the analysis, along with membership of partition groups. The group with the lowest number is at the top of the causal hierarchy.

The Gaussianity of the variables in the dairy cattle data was examined through univariate QQ plots and normality tests (both Kolmogorov-Smirnov and Cramer von-Mises). The QQ plots and normality tests (details provided in a Supplementary document) are generally consistent with the Gaussianity assumption. The Mardia test for multivariate normality (based on skewness) also returned a favorable result with a $p$-value very close to 1.

5.2. Results and discussion. Given the background information based on a natural hierarchy of causal relationships among various (continuous) variable groups, the Partition-DAG is really well-suited for analyzing this data
set, as it allows us to effectively incorporate this information. The penalty parameter for the Partition-DAG approach was chosen by stability selection (Liu et al. [2010]). In particular, \( N = 100 \) WOR samples \( S_1, S_2, \ldots, S_{100} \) of size \( b = 20 \) were drawn from the 25 data points. For every candidate penalty parameter value \( \lambda \) (chosen from a grid of values), we compute the estimated adjacency matrix \( \hat{\psi}(S_j) \) restricted to the sub-sample \( S_j \) for \( 1 \leq j \leq 100 \). Let \( \hat{\theta}^b \) denote the average of all the 100 \( \hat{\psi}(\cdot) \) matrices, and define

\[
D_b(\lambda) = \sum_{s \neq t} 2 \hat{\theta}_{st}^b (1 - \hat{\theta}_{st}^b) / p^2 - p, \quad \overline{D}_b(\lambda) = \sup_{0 \leq t \leq \lambda} D_b(t).
\]

We choose the penalty parameter value which maximizes \( \overline{D}_b \). The estimated network (DAG) is depicted in Figure 1.

In order to understand/illustrate the difference in the performance of Partition-DAG with changes in the allocation of variables to partitions, we merged some of the 9 groups to obtain a partition with 5 groups, and again estimated a causal network using Partition-DAG. The estimated network with five groups is depicted in Figure 2.

We then used the PC algorithm with level \( \alpha \) for the conditional independence tests chosen by a similar stability selection procedure as the one outlined above. The estimated CPDAG had only three unoriented edges (MH-TS, AMW-CSC, and WI-CSC). The first two unoriented edges are within the same partition group, so background knowledge did not help, but the edge WI-CSC was oriented as CSC -- > WI using background knowledge. The resulting PDAG is provided in Figure 3. More importantly, three of the estimated edges (highlighted in red in Figure 3), namely AFI -- > OF, NI -- > TI, WI -- > EVM, are inconsistent with the partition based hierarchy established earlier in this section (see Table 6). These results highlight the issues with the post-hoc approach used in PCBGK (PC algorithm with background knowledge), where an effort to incorporate edge orientation based background knowledge is made after estimating a CPDAG. On the other hand, the proposed Partition-DAG approach incorporates this knowledge during the estimation process, thereby ensuring results consistent with the provided background information.

We now discuss in particular why the three edges AFI -- > OF, NI -- > TI, WI -- > EVM in the PCBGK estimated network (which contradict the background knowledge) do not make sense. First, the amount of offered grass (OF) controls the intake level (AFI); therefore, the forage intake does not have a causal effect on the amount of offered forage. Second, net income (NI) is the result of subtracting total cost from total income. Therefore, net income is defined by total income (TI) and not the other way. Finally,
Fig 1: Estimated network for dairy cattle variables with the Partition-DAG algorithm using partition with 9 groups. Note that the output of the algorithm is *always a fully-oriented DAG* consistent with the provided partition-based ordering. A list of all the directed edges is provided in the Supplementary material.
Fig 2: Estimated network for dairy cattle variables with the Partition-DAG algorithm using partition with 5 groups. Note that the output of the algorithm is always a fully-oriented DAG consistent with the provided partition-based ordering. A list of all the directed edges is provided in the Supplementary material.
the economic value of milk (EVM) is set by the market and not by income per worker (WI). In this data set, the income per worker increased with the economic value of milk; however, it does not make sense that income per worker has a causal effect on a variable completely determined by the market.

As an additional evaluation, a dairy science expert (Dr. Carulla, co-author) inspected all the estimated edges for each network and first identified estimated edges (ignoring the orientation) which are known or made sense based on current knowledge. This turned out to be roughly two-thirds of the estimated edges for each of the three networks in Figures 1-3. For the 5 groups network, 97% of the identified edges had the correct orientation, while for the 9 groups network, 87% of the identified edges had the correct orientation, and for the PC algorithm, 74% of the identified edges had the correct orientation. The list of relevant edges is provided in the Supplementary material. It is worth mentioning that the expert evaluation of the inferred networks was carried out for the particular conditions of Colombian high altitude tropic dairy cattle operations; it is relevant because certain causal relationships that make sense in this study, may not be valid under different conditions.

The above analysis illustrates the tremendous usefulness and potential of the proposed method in dairy science and in the problem of DAG estimation more generally. In the first place, it is a sound approach to infer a DAG and the associated causal relationships from observational data in animal production systems using the available (partial) information, which is a feature exhibited by most of these agroecosystems. Further, knowledge of causal relationships in dairy and other animal production operations helps in taking management decisions and making recommendations, since it leads to identifying what nodes have the biggest impact on the system outputs; the latter information singles out variables that should be modified and the direction of such modifications. Finally, the analysis of the estimated DAGs sheds light into interesting interactions between components of the production system and thus could be the basis to design follow-up experiments to fully test their validity.

6. Support recovery in a high-dimensional setting. In this section, we establish that the estimator obtained from the Partition-DAG algorithm (Algorithm 1) leads to accurate DAG support recovery and estimation in a high-dimensional setting. High dimensional support recovery results for DAG estimation using a least squares regression and MCP penalty based objective functions are established in Aragam et al. [2016] when no infor-
Fig 3: Estimated network for dairy cattle variables with the PCBGK (PC algorithm with background knowledge) using partition with 9 groups. The resulting partially oriented DAG has two unoriented edges MH-TS and AMV-CSC, both within the same partition group. There are three edges with orientations inconsistent with the partition-based ordering, these are highlighted in red. A list of all the undirected and directed edges is provided in the Supplementary material.
mation on the ordering is available. On the other hand, Khare et al. [2017] provide high-dimensional support recovery results for a Gaussian likelihood and $\ell_1$ penalty based objective function for a completely known ordering. We adapt and generalize the relevant analysis in these two papers to our setting: a Gaussian likelihood and $\ell_1$ penalty based objective function with partial ordering information.

We first discuss what accurate support recovery and estimation means in our setting. Let $\bar{\Sigma} = \bar{\Omega}^{-1}$ denote the true underlying covariance matrix. Then, the equivalence class of DAGs corresponding to $\bar{\Omega}$ is given by $\tilde{\mathcal{E}}(\bar{\Omega}) = \{ \bar{B} : \bar{B} \in \bar{B}_p, \bar{B}'\bar{B} = \bar{\Omega} \}$.

We provide an alternative characterization of $\tilde{\mathcal{E}}(\bar{\Omega})$ which will be useful in the subsequent analysis. Let $S_1$ denote the group of permutations of $\{1, 2, \ldots, m\}$, and $S_2$ denote the group of permutations of $\{m + 1, m + 2, \ldots, p\}$. Fix $\pi_1 \in S_1$ and $\pi_2 \in S_2$. Then, $\pi = (\pi_1, \pi_2)$ corresponds to a permutation of $\{1, 2, \ldots, p\}$. Let $P_{\pi_1, \pi_2}$ denote the corresponding permutation matrix, i.e., for any matrix $A$, we have $\left( P_{\pi_1, \pi_2} A P_{\pi_1, \pi_2}' \right)_{ij} = A_{\pi_1(i), \pi_2(j)}$ for every $1 \leq i, j \leq p$. Let $\bar{L}(\pi_1, \pi_2)$ be the Cholesky factor of $P_{\pi_1, \pi_2} \bar{\Omega}(\pi_1, \pi_2) P_{\pi_1, \pi_2}'$, i.e., $\bar{L}(\pi_1, \pi_2)$ is a lower triangular matrix with positive diagonal entries such that $P_{\pi_1, \pi_2} \bar{\Omega}(\pi_1, \pi_2) P_{\pi_1, \pi_2}' = \bar{L}(\pi_1, \pi_2) \bar{L}(\pi_1, \pi_2)'$.

Let $\tilde{\mathcal{E}}(\Omega) = \{ \bar{B}(\pi_1, \pi_2) : \pi_1 \in S_1, \pi_2 \in S_2 \}$.

In other words, $\tilde{\mathcal{E}}(\Omega)$ can be obtained by considering all possible permutations of $\bar{\Omega}$, and choosing appropriately permuted versions of the corresponding Cholesky factors.

Let $\hat{\bar{B}}$ denote the estimator obtained by minimizing $Q_{PDAG}(B)$ in (2.2), i.e.,

$$\hat{\bar{B}} \in \arg\min_{B \in \bar{B}_p} Q_{PDAG}(B).$$

Let $\hat{\pi}_1 \in S_1$ and $\hat{\pi}_2 \in S_2$ denote (random) permutations such that permuting the rows and columns according to $\hat{\pi} = (\hat{\pi}_1, \hat{\pi}_2)$ makes $\hat{\bar{B}}$ lower triangular.

We will first show that the sparsity pattern in $\hat{\bar{B}}$ is identical to the sparsity pattern in $\bar{B}(\hat{\pi})$ with high probability (Theorem 6.1). We will then show that $\bar{B}(\hat{\pi})$ is well-behaved in terms of sparsity by showing that the $\ell_1$-norm of the
vector of its off-diagonal entries is of the same order of the smallest such $\ell_1$-norm among all DAGs in $\tilde{E}(\Omega)$ (Theorem 6.2). Together these results show that the sparsity pattern in the Partition-DAG estimator $\hat{B}$ is close to the sparsity pattern of a population DAG which has the smallest $\ell_1$ norm of its vectorized off-diagonal entries among all candidate DAGs in $\tilde{E}(\bar{\Omega})$. In the challenging non-identifiable and high-dimensional Gaussian DAG setting, this is perhaps close to the best that one can hope for.

For any matrix $A$, let $\text{supp}(A)$ denote the collection of all pairs $(i, j)$ such that $A_{ij} \neq 0$. The first key step in our analysis will be to express the event of interest $\{\text{supp}(\hat{B}) = \text{supp}(\bar{B}(\hat{\pi}))\}$ based on the global minimizer $\hat{B}$ in terms of restricted minimizers $\{\hat{B}(\pi_1, \pi_2)\}_{\pi_1 \in S_1, \pi_2 \in S_2}$, which are defined as follows. For every $\pi_1 \in S_1$ and $\pi_2 \in S_2$, let

$$\tilde{B}_p(\pi_1, \pi_2) = \{B \in \tilde{B}_p : P_{\pi_1, \pi_2}BP_{\pi_1, \pi_2}^T \text{ is lower triangular}\}$$

and

$$\hat{B}(\pi_1, \pi_2) \in \arg\min_{B \in \tilde{B}_p(\pi_1, \pi_2)} Q_{PDAG}(B).$$

The restricted minimization problem of $Q_{PDAG}$ over $\tilde{B}_p(\pi_1, \pi_2)$ is a simpler problem since we know the ordering $\pi$ which makes $B$ lower triangular. Let $\hat{\beta}_i(\pi_1, \pi_2)$ denote the $i^{th}$ row of $\hat{B}(\pi_1, \pi_2)$, and $N_i(\pi_1, \pi_2)$ denote all vertices which are smaller than $i$ in the ordering specified by $(\pi_1, \pi_2)$, i.e.,

$$N_i(\pi_1, \pi_2) = \{j : \pi_1^{-1}(j) < \pi_1^{-1}(i)\}$$

for $1 \leq i \leq m$, and

$$N_i(\pi_1, \pi_2) = \{1, 2, \cdots, m\} \cup \{j : \pi_2^{-1}(j) < \pi_2^{-1}(i)\}$$

for $m + 1 \leq i \leq p$. Let

$$N_i(\pi_1, \pi_2)^c = \{1, 2, \cdots, p\} \setminus \{N_i(\pi_1, \pi_2), i\}.$$

Using

$$Q_{PDAG}(B) = \sum_{i=1}^{p} \left\{ -\log B_{ii} + B_{i}^tSB_i + \lambda \sum_{j \neq i} |B_{ij}| \right\},$$

it follows that

(6.1)

$$\hat{\beta}_i(\pi_1, \pi_2) \in \arg\min_{x \in \mathbb{R}^p, x_i \geq 0, x_{N_i(\pi_1, \pi_2)^c} = 0} \left\{ -\log x_i + x^tSx + \lambda \sum_{j \neq i} |x_j| \right\}$$

for $1 \leq i \leq p$, and

$$\hat{\beta}_i(\pi_1, \pi_2) \in \arg\min_{x \in \mathbb{R}^p, x_i \geq 0, x_{N_i(\pi_1, \pi_2)^c} = 0} \left\{ -\log x_i + x^tSx + \lambda \sum_{j \neq i} |x_j| \right\}$$

for $m + 1 \leq i \leq p$.
for every $1 \leq i \leq p$. Let $\bar{\beta}_i(\pi_1, \pi_2)$ denote the $i^{th}$ row of $\bar{B}(\pi_1, \pi_2)$. Since $\hat{B} = \bar{B}(\hat{\pi})$, it follows that

$$\{\text{supp}(\hat{B}) \neq \text{supp}(\bar{B}(\hat{\pi}))\} \subseteq \bigcup_{\pi_1 \in S_1, \pi_2 \in S_2} \{\text{supp}(\bar{B}(\pi_1, \pi_2)) \neq \text{supp}(\bar{B}(\pi_1, \pi_2))\}$$

(6.2) \hspace{1cm} \subseteq \bigcup_{i=1}^p \bigcup_{\pi_1 \in S_1, \pi_2 \in S_2} \{\text{supp}(\hat{\beta}_i(\pi_1, \pi_2)) \neq \text{supp}(\bar{\beta}_i(\pi_1, \pi_2))\}.$$ 

It is implicitly understood that $\{\text{supp}(\hat{B}) \neq \text{supp}(\bar{B}(\hat{\pi}))\}$ represents the event that the support of $\hat{B}$ does not equal the support of $\bar{B}(\hat{\pi})$ for at least one global minimizer $\hat{B}$ of $Q_{PDAG}$. A similar interpretation holds for other events in (6.2). Using results in Khare et al. [2017], we can provide a uniform exponential bound on the probability of the individual support recovery failure events $\{\text{supp}(\hat{\beta}_i(\pi_1, \pi_2)) \neq \text{supp}(\bar{\beta}_i(\pi_1, \pi_2))\}$. However, the union over $S_1 \times S_2$ has $m! \times (p - m)!$ terms which dominates the exponential bound.

As in Aragam et al. [2016], we demonstrate that the above union is contained in a union of a much smaller number of key support recovery failure events. Let

$$s = \max_{1 \leq i \leq p, \pi_1 \in S_1, \pi_2 \in S_2} |\text{supp}(\bar{\beta}_i(\pi_1, \pi_2))|$$

denote the maximum number of non-zero entries in a row for any matrix in $\tilde{E}(\tilde{\Omega})$, and

$$\tilde{N}_i = \{N : N = \text{supp}(\bar{\beta}_i(\pi_1, \pi_2)) \text{ for some } 1 \leq i \leq p, \pi_1 \in S_1, \pi_2 \in S_2\}$$

denote the collection of supports of all the rows of matrices in $\tilde{E}(\tilde{\Omega})$. It follows that

$$|\tilde{N}_i| \leq \sum_{j=0}^s \binom{p}{j} \leq (p + 1)^s.$$ 

Fix $1 \leq i \leq p, \pi_1 \in S_1, \pi_2 \in S_2$ arbitrarily. Let $N \in \tilde{N}_i$ be such that $N = \text{supp}(\bar{\beta}_i(\pi_1, \pi_2))$. Let $X \sim N_i(0, \Sigma)$ and

$$\mathcal{M}(N) = \{\tilde{N} : N \subseteq \tilde{N}, X \perp \perp X_{\tilde{N}\setminus N} \mid X_N\}.$$ 

It follows by the Gaussianity of $X$ that if $\tilde{N}_1, \tilde{N}_2 \in \mathcal{M}(N)$, then $\tilde{N}_1 \cup \tilde{N}_2 \in \mathcal{M}(N)$. Hence $\mathcal{M}(N)$ has a maximal element. Let $M(N)$ denote the maximal element of $\mathcal{M}(N)$. We define

$$\beta_i(N) = \arg\min_{x \in \mathbb{R}^p, x_i \geq 0, x_M(N)^c = 0} \left\{ -\log x_i + x^T \Sigma x \right\}$$

(6.3)
and

\begin{equation}
\hat{\beta}_i(N) \in \arg\min_{x \in \mathbb{R}, x_i \geq 0, x \in \mathcal{M}(N) \in } \left\{-\log x_i + x^T S x + \lambda \sum_{j \neq i} |x_j| \right\}.
\end{equation}

For any \(k \times k\) positive definite matrix

\[
A = \begin{bmatrix}
a_{11} & A_{12} \\
A_{12}^T & A_{11}
\end{bmatrix}
\]

it follows by straightforward calculations that

\[
\frac{1}{\sqrt{a_{11} - A_{12}^T A_{12}^{-1} A_{12}}} (1, -A_{12}^{-1} A_{12}) = \arg\min_{x \in \mathbb{R}, x_i \geq 0} \left\{-\log x_1 + x^T A x \right\}.
\]

Using this fact, along with the definition of \(\mathcal{M}(N)\), we get that

\[
\text{supp}(\bar{\beta}_i(N)) = \text{supp}(\tilde{\beta}_i(\pi_1, \pi_2)) = N.
\]

It follows that if \(\text{supp}(\bar{\beta}_i(N)) = \text{supp}(\tilde{\beta}_i(N))\) for every minimizer in (6.4), then \(\bar{\beta}_i(N)\) is a candidate for the minimization problem in (6.1). Since \(N_i(\pi_1, \pi_2) \subseteq \mathcal{M}(N)\), it follows that \(\bar{\beta}_i(\pi_1, \pi_2)\) is a candidate for the minimization problem in (6.4). Given that \(\bar{\beta}_i(N)\) and \(\tilde{\beta}_i(\pi_1, \pi_2)\) are global minimizers for the respective minimization problems in (6.4) and (6.1), we get

\[
\left\{\text{supp}(\bar{\beta}_i(N)) = \text{supp}(\tilde{\beta}_i(N))\right\} \subseteq \left\{\text{supp}(\bar{\beta}_i(\pi_1, \pi_2)) = \text{supp}(\tilde{\beta}_i(\pi_1, \pi_2))\right\}.
\]

Using (6.2), it follows that

\begin{equation}
\{\text{supp}(\hat{\beta}) \neq \text{supp}(\tilde{\beta}^{\hat{x}})\} \subseteq \cup_{i=1}^p \cup_{N \in \mathcal{N}_i} \left\{\bar{\beta}_i(N) = \tilde{\beta}_i(N)\right\}.
\end{equation}

Hence, the union over \(m! \times (p-m)!\) support recovery failure events in (6.2) has been reduced to a union of at most \((p+1)^{s+1}\) support recovery failure events. A uniform exponential bound for these events can now be obtained by using results from Khare et al. [2017]. We state the following assumptions which are standard for consistency results using \(\ell_1\)-penalty based minimizations, but are adapted to the non-identifiable DAG setting.

- (A1 - Bounded eigenvalues) The eigenvalues of \(\bar{\Omega}\) are bounded below by \(\theta_{\min} > 0\), and bounded above by \(\theta_{\max} < \infty\).
• (A2 - Uniform Incoherence condition) There exists $\delta < 1$ such that for every $\pi_1 \in S_1, \pi_2 \in S_2, 1 \leq i \leq p$, and $j \notin \text{supp}(\bar{\beta}_i(\pi_1, \pi_2))$, we have

$$
\left| \Sigma_{j,N} \left( \Sigma_{NN} + \frac{2}{B_{ii}} \Delta \right)^{-1} \text{sign}(\bar{B}_{iN}) \right| \leq \delta.
$$

Here $N = \text{supp}(\bar{\beta}_i(\pi_1, \pi_2))$ and $\Delta$ is a matrix with the $i^{th}$ diagonal entry equal to 1 and all other entries equal to 0. This assumption is a version of the standard mutual incoherence condition needed for consistency of $\ell_1$-penalized estimators.

Let $\tau$ denote the minimal signal size, i.e.,

$$
\tau = \min_{\pi_1 \in S_1, \pi_2 \in S_2} \{ \max_{1 \leq i,j \leq p} |S_{ij} - \bar{\Sigma}_{ij}| : (\bar{B}(\pi_1, \pi_2))_{ij} \neq 0 \}.
$$

Under Assumption A1, using similar arguments to the proof of Lemma E.1 in Khare et al. [2017] it can be shown that

$$
P \left( \max_{1 \leq i,j \leq p} \left| S_{ij} - \bar{\Sigma}_{ij} \right| \geq K \sqrt{\frac{s \log p}{n}} \right) \leq \frac{4}{(p + 1)^{s+2}}
$$

for an appropriate constant $K$. Here $P$ denotes the probability measure corresponding to the true data generating model. Now, following closely the arguments as in Lemma E.2 - Lemma E.8 in Khare et al. [2017] the only things different are that $\max_{1 \leq i,j \leq p} |S_{ij} - \bar{\Sigma}_{ij}|$ should be bounded by $K \sqrt{\frac{s \log p}{n}}$ and not $K \sqrt{\frac{\log p}{n}}$, and the bound on the derivatives in the statement of Lemma E.4 should be of the order $\sqrt{\frac{s \log p}{n}}$ and not $K \sqrt{\frac{\log p}{n}}$, with the corresponding probability lower bound equal to $1 - \frac{4}{(p + 1)^{s+2}}$. It follows from (6.5) that

$$
\supp(\hat{\beta}(\pi)) = \supp(\bar{\beta}(\pi))
$$

The only things different are that $\max_{1 \leq i,j \leq p} |S_{ij} - \bar{\Sigma}_{ij}|$ should be bounded by $K \sqrt{\frac{s \log p}{n}}$ and not $K \sqrt{\frac{\log p}{n}}$, and the bound on the derivatives in the statement of Lemma E.4 should be of the order $\sqrt{\frac{s \log p}{n}}$ and not $K \sqrt{\frac{\log p}{n}}$, with the corresponding probability lower bound equal to $1 - \frac{4}{(p + 1)^{s+2}}$. It follows from (6.5) that

$$
\supp(\hat{\beta}(\pi)) = \supp(\bar{\beta}(\pi))
$$
and
\[ \left\| \hat{\beta}_i(\hat{\pi}) - \beta_i(\hat{\pi}) \right\| \leq C \sqrt{\left| \text{supp}(\beta_i(\hat{\pi})) \right|} \lambda \]
for every \( 1 \leq i \leq p \) with probability at least \( 1 - \frac{64}{p+1} \). The above non-asymptotic bound can be easily converted into the following asymptotic support recovery consistency result.

**Theorem 6.1 (Support recovery and estimation consistency)** Suppose (A1) and (A2) are satisfied. If \( s \sqrt{\frac{\log p}{n}} = o(\min(1, \lambda)) \), \( s \lambda = o(\min(1, \tau)) \), then any minimizer \( \hat{B} \) of \( Q_{PDAG} \) and the corresponding permutation \( \hat{\pi} \) which makes \( \hat{B} \) lower-triangular satisfy \( \text{supp}(\hat{B}) = \text{supp}(\bar{B}(\hat{\pi})) \) and
\[ \left\| \hat{\beta}_i(\hat{\pi}) - \beta_i(\hat{\pi}) \right\| \leq C \sqrt{\left| \text{supp}(\beta_i(\hat{\pi})) \right|} \lambda \]
for every \( 1 \leq i \leq p \) with probability tending to 1 as \( n \to \infty \). The above result establishes that the estimated DAG \( \hat{B} \) has the same support and is vanishingly close in magnitude to \( \bar{B}(\hat{\pi}) \) - a member of the class of non-identifiable population DAGs \( \tilde{E}(\bar{\Omega}) \). We now show that \( \bar{B}(\hat{\pi}) \) is close to optimal in the following sense. Note that our goal is to identify the sparsest DAG in \( \tilde{E}(\bar{\Omega}) \), which ideally corresponds to using the \( \ell_0 \)-penalty in our objective function. However, as is common, we use the \( \ell_1 \)-penalty as a proxy for the \( \ell_0 \)-penalty for computational scalability. Let \( \hat{\pi} = (\hat{\pi}_1, \hat{\pi}_2) \) denote the permutation which minimizes the off-diagonal \( \ell_1 \)-norm over DAGs in \( \tilde{E}(\bar{\Omega}) \), i.e.,
\[ (\hat{\pi}_1, \hat{\pi}_2) = \arg\min_{\pi_1 \in S_1, \pi_2 \in S_2} \left\| B(\pi_1, \pi_2) \right\|_{1, \text{off}}, \]
where \( \left\| B(\pi_1, \pi_2) \right\|_{1, \text{off}} = \sum_{i \neq j} \left| (B(\pi_1, \pi_2))_{ij} \right| \). We will show that \( \left\| B(\hat{\pi}) \right\|_{1, \text{off}} \) is very close to \( \left\| B(\hat{\pi}) \right\|_{1, \text{off}} \) with high probability.

Note that if \( \bar{\Omega} \) is diagonal, then all the DAGs in \( \tilde{E}(\bar{\Omega}) \) are trivially optimal in off-diagonal \( \ell_1 \)-norm. Hence, we assume that \( \bar{B}(\hat{\pi}) \) does not correspond to a diagonal matrix. Since \( \hat{B} \) is a global minimizer of \( Q_{PDAG} \), it follows that
\[ Q_{PDAG}(\hat{B}) \leq Q_{PDAG}(\bar{B}(\hat{\pi})). \]
Since \( \bar{B}(\hat{\pi})^t \bar{B}(\hat{\pi}) = \bar{B}(\hat{\pi})^t \bar{B}(\hat{\pi}) = \bar{\Omega} \), it follows that
\[ \left\{ \text{trace}(\hat{B}^t \bar{B}S) - \log \left| \hat{B}^t \hat{B} \right| - \text{trace}(\bar{B}(\hat{\pi})^t \bar{B}(\hat{\pi})) + \log \left| \bar{B}(\hat{\pi})^t \bar{B}(\hat{\pi}) \right| \right\} + \lambda \left\| \hat{B} \right\|_{1, \text{off}} \]
\[ \leq \lambda \left\| \bar{B}(\hat{\pi}) \right\|_{1, \text{off}} \]
(6.6)
Let $\|B\|_{0, off}$ denote the number of off-diagonal entries in $B$. Using Assumptions (A1) and (A2) along with Theorem 6.1, we get

\[
\text{trace}(\hat{B}'\hat{B}S) - \log \|\hat{B}'\hat{B}\| - \text{trace}(\hat{B}(\hat{\pi})'\tilde{B}(\hat{\pi})) + \log \|\tilde{B}(\hat{\pi})'\tilde{B}(\hat{\pi})\| = \sum_{i=1}^{p} \left\{ \hat{\beta}_i(\hat{\pi})'S\hat{\beta}_i - \hat{\beta}_i(\hat{\pi})'S\hat{\beta}_i - 2\log \hat{B}_{ii} + 2\log \tilde{B}(\hat{\pi})_{ii} \right\}
\]

\[
\geq -\bar{C} \sum_{i=1}^{p} \|\hat{\beta}_i(\hat{\pi}) - \hat{\beta}_i(\hat{\pi})\|^2
\]

\[
\geq -C\bar{C}\lambda^2 \sum_{i=1}^{p} |\text{supp}(\hat{\beta}_i(\hat{\pi}))|
\]

\[
\geq -C\bar{C} \left( p + \|\tilde{B}(\hat{\pi})\|_{0, off} \right) \lambda^2
\]

\[
= -C\bar{C} \left( p + \|\tilde{B}(\hat{\pi})\|_{0, off} \right) \lambda^2
\]

and

\[
\|\tilde{B}\|_{1, off} - \|\tilde{B}(\hat{\pi})\|_{1, off} \geq -\sum_{i=1}^{p} \|\hat{\beta}_i(\hat{\pi}) - \hat{\beta}_i(\hat{\pi})\|_1
\]

\[
\geq -\sum_{i=1}^{p} \sqrt{|\text{supp}(\hat{\beta}_i(\hat{\pi}))|} \|\hat{\beta}_i(\hat{\pi}) - \hat{\beta}_i(\hat{\pi})\|
\]

\[
\geq -C\lambda \sum_{i=1}^{p} |\text{supp}(\hat{\beta}_i(\hat{\pi}))|
\]

\[
\geq - C \left( p + \|\tilde{B}(\hat{\pi})\|_{0, off} \right) \lambda
\]

with probability at least $1 - \frac{64}{p+1}$, for an appropriate constant $\bar{C}$. It follows by (6.6) that

\[
\left( 1 - \frac{C(1 + \bar{C})\lambda (p + \|\tilde{B}(\hat{\pi})\|_{0, off})}{\|\tilde{B}(\hat{\pi})\|_{1, off}} \right) \|\tilde{B}(\hat{\pi})\|_{1, off} \leq \|\tilde{B}(\hat{\pi})\|_{1, off}.
\]

with probability at least $1 - \frac{64}{p+1}$. Since $\|\tilde{B}(\hat{\pi})\|_{1, off} \geq \tau \|\tilde{B}(\hat{\pi})\|_{0, off}$ and $\|\tilde{B}(\hat{\pi})\|_{0, off} \geq \|\tilde{B}(\hat{\pi})\|_{0, off}$, we get

\[
\left( 1 - C(1 + \bar{C})\frac{\lambda}{\tau} \left( \frac{p}{\|\tilde{B}(\hat{\pi})\|_{0, off}} + 1 \right) \right) \|\tilde{B}(\hat{\pi})\|_{1, off} \leq \|\tilde{B}(\hat{\pi})\|_{1, off}
\]
with probability at least $1 - \frac{64}{p+x}$. The above non-asymptotic bound can now be converted to the following result.

**Theorem 6.2 (Optimality of $\hat{\pi}$)** Suppose the conditions in Theorem 6.1 hold, and $\lambda p = o\left(\tau \|\bar{B}(\bar{\pi})\|_{0,off}\right)$. Then

$$(1 - b_n) \|B(\bar{\pi})\|_{1,off} \leq \|\bar{B}(\bar{\pi})\|_{1,off} \leq \|\bar{B}(\bar{\pi})\|_{1,off}$$

with probability tending to 1 as $n \to \infty$, with $b_n = o(1)$.

Since $s\lambda = o(\tau)$ in the assumptions needed for Theorem 6.1, a sufficient condition for the condition $\lambda p = o\left(\tau \|B(\bar{\pi})\|_{0,off}\right)$ to hold is that $p = o\left(s \|\bar{B}(\bar{\pi})\|_{0,off}\right)$. This will be satisfied for example when each row of $\bar{B}(\bar{\pi})$ has at least one non-zero entry, such as the setting where the permuted lower triangular version of $\bar{B}(\bar{\pi})$ has a banded structure.

Note that the theoretical results above are proved for a global minimizer $\hat{B}$ of $Q_{PDAG}$. The output of the Partition-DAG algorithm in general is only guaranteed to be a stationary point of $Q_{PDAG}$ (see Lemma 2.1). This is a common issue with the analysis of non-convex optimization problems. A practical approach to address this is to use multiple starting points (based on time/computational budget) and choose the output corresponding to the minimum $Q_{PDAG}$ value as the estimator.

**Code:** The functions and code used for the experiments and data analysis in Sections 4 and 5 is available at https://github.com/shr264/PDAG

**References.**


