# A Ring-Based Distributed Algorithm for Learning High-Dimensional Bayesian Networks

Jorge D. Laborda[1,2](✉), Pablo Torrijos[1,2], José M. Puerta[1,2], and José A. Gámez[1,2]

[1] Instituto de Investigación en Informática de Albacete (I3A), Universidad de Castilla-La Mancha. Albacete, 02071 Albacete, Spain
{Pablo.Torrijos,Jose.Puerta,Jose.Gamez}@uclm.es

[2] Departamento de Sistemas Informáticos. Universidad de Castilla-La Mancha. Albacete, 02071 Albacete, Spain
JorgeDaniel.Laborda@uclm.es

**Abstract.** Learning Bayesian Networks (BNs) from high-dimensional data is a complex and time-consuming task. Although there are approaches based on horizontal (instances) or vertical (variables) partitioning in the literature, none can guarantee the same theoretical properties as the Greedy Equivalence Search (GES) algorithm, except those based on the GES algorithm itself. In this paper, we propose a directed ring-based distributed method that uses GES as the local learning algorithm, ensuring the same theoretical properties as GES but requiring less CPU time. The method involves partitioning the set of possible edges and constraining each processor in the ring to work only with its received subset. The global learning process is an iterative algorithm that carries out several rounds until a convergence criterion is met. In each round, each processor receives a BN from its predecessor in the ring, fuses it with its own BN model, and uses the result as the starting solution for a local learning process constrained to its set of edges. Subsequently, it sends the model obtained to its successor in the ring. Experiments were carried out on three large domains (400–1000 variables), demonstrating our proposal's effectiveness compared to GES and its fast version (fGES).

**Keywords:** Bayesian network learning · Bayesian network fusion/aggregation · Distributed machine learning

## 1 Introduction

A Bayesian Network (BN) [9,13,18] is a graphical probabilistic model that expresses uncertainty in a problem domain through probability theory. BNs heavily rely on the graphical structure used to produce a symbolic (relevance) analysis [16], which gives them an edge from an interpretability standpoint. The demand for explainable models and the rise of causal models make BNs a cutting-edge technology for knowledge-based problems.

A BN has two parts: A graphical structure that stores the relationships between the domain variables, such as (in)dependences between them, alongside a set of parameters or conditional probability tables that measure the weight of the relationships shown in the graph. Experts in the problem domain can help build both parts of the BN [12]. Unfortunately, this task becomes unsustainable when the scale of the problem grows. Nonetheless, learning BNs with data is a well-researched field, and even though learning the structure of a BN is an NP-hard problem [6], a variety of proposals have been developed to learn BNs from data [3,5,7,22]. Additionally, a number of studies have delved into high-dimensional problems [2,21,25].

The main focus of this paper is to address the problem of structural learning of BNs in high-dimensional domains to reduce its complexity and improve the overall result. To do so, we use a search and score approach within the equivalence class search space [1] while dividing the problem into more minor problems that can be solved simultaneously. Furthermore, our work exploits the advantages of modern hardware by applying parallelism to the majority of the phases of our algorithm.

To achieve these improvements, our research applies, as its core component, the recent proposal for BN fusion [19] alongside an initial partitioning of all of the possible edges of the graph and the GES algorithm [5]. Therefore, in a few words, our algorithm starts by dividing the set of possible edges into different subsets and performing parallel learning of various networks, where each process is restricted to its according subset of edges. Once the batch has finished, the resulting BN is used as input for the following process, creating a circular system where the output of one process is the input of the following process. Our experiments were performed over the three largest BNs in the bnlearn repository [23], showing that our algorithm reduces the time consumed while achieving good representations of these BNs.

The remainder of this paper is organized as follows: Section 2 provides a general introduction to BNs. Next, in Sect. 3, our proposal is explained in detail. In Sect. 4, we describe the methodology used to perform our experiments and present the results obtained. Finally, in Sect. 5, we explain the conclusions we have arrived at throughout our work.

## 2 Preliminaries

### 2.1 Bayesian Network

A Bayesian Network (BN) [9,13,18] is a probabilistic graphical model frequently used to model a problem domain with predominant uncertainty. A BN is formally represented as a pair $\mathcal{B} = (G, \mathbf{P})$ where $G$ is a Directed Acyclic Graph (DAG) and $\mathbf{P}$ is a set of conditional probability distributions:

– The DAG is a pair $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{X_1, \dots X_n\}$ is the set of variables of the problem domain, and $\mathbf{E}$ is the set of directed edges between the variables: $\mathbf{E} = \{X \rightarrow Y \mid X \in \mathbf{V}, Y \in \mathbf{V}, X \neq Y\}$ $G$ is capable of representing the conditional (in)dependence relationships between $\mathbf{V}$ using the *d-separation* criterion [18].

– $\mathbf{P}$ is a set of conditional probability distributions that factorizes the joint probability distribution $P(\mathbf{V})$ by using the DAG structure $G$ and Markov's condition:

$$P(\mathbf{V}) = P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | pa_G(X_i)), \tag{1}$$

where $pa_G(X_i)$ is the set of parents of $X_i$ in $G$.

## 2.2 Structural Learning of BNs

Structural learning of BNs is the process of creating their DAG $G$ by using data[1]. This problem is an NP-hard problem [6]; however, many solutions have been developed to learn BNs. We can classify these approaches into two groups: constraint-based and score+search solutions. In addition, some hybrid algorithms have also been developed (e.g., [2,25]). The constraint-based algorithms use hypothesis tests to identify the conditional independences found in the data, while the score+search methods apply a search algorithm to find the best structure for a given score function or metric, which depends entirely on the given data. So, these approaches need a search method to find promising structural candidates and a scoring function to evaluate each candidate. We will only consider discrete variables and focus on the score+search methods.

We can see score+search methods as optimization problems where, given a complete dataset $D$ with $m$ instances over a set of $n$ discrete variables $\mathbf{V}$, the objective is to find the best DAG $G^*$ within the search space of the DAGs of the problem domain $G^n$, by means of a scoring function $f(G : D)$ that measures how well a DAG $G$ fits the given data $D$:

$$G^* = \arg\max_{G \in G^n} f(G : D) \tag{2}$$

Different measurements have been used in the literature. The scoring functions can be divided into Bayesian and information theory-based measures (e.g., [4]). Our work focuses on using the *Bayesian Dirichlet equivalent uniform* (BDeu) score [8], but any other Bayesian score could be used in our proposal. This score is a particular case of BDe where a uniform distribution over all the Dirichlet hyperparameters is assumed.

$$BDeu(G \mid D) = log(P(G)) + \sum_{i=1}^{n} \left[ \sum_{j=1}^{q_i} \left[ log \left( \frac{\Gamma(\frac{\eta}{q_i})}{\Gamma(N_{ij} + \frac{\eta}{q_i})} \right) + \sum_{k=1}^{r_i} log \left( \frac{\Gamma(N_{ijk} + \frac{\eta}{r_i q_i})}{\Gamma(\frac{\eta}{r_i q_i})} \right) \right] \right], \tag{3}$$

[1] In this paper, we only consider the case of complete data, i.e., no missing values in the dataset.

where $r_i$ is the number of states for $X_i$, $q_i$ is the number of state configurations of $Pa_G(X_i)$, $N_{ij}$ and $N_{ijk}$ are the frequencies computed from data for maximum likelihood parameter estimation, $\eta$ is a parameter representing the equivalent sample size and $\Gamma()$ is the *Gamma* function.

A state-of-the-art algorithm for structural learning is the *Greedy Equivalence Search* (GES) [5]. This algorithm performs a greedy approach over the equivalence space, using a scoring metric to search in two stages: *Forward Equivalence Search* (FES) and *Backward Equivalence Search* (BES). The FES stage is in charge of inserting edges into the graph, and when no further insertions improve the overall score, the BES stage begins to delete edges from the graph until there are no further improvements. It is proven that under certain conditions, GES will obtain an optimum BN representation of the problem domain. In our work, we use an alternative approach to GES, as described in [1], where the FES stage is carried out in a totally greedy fashion while maintaining the BES stage intact. This improvement has been proven to be as effective as GES and to retain the same theoretical properties. To use this last algorithm as a control one, we implemented a parallel version of GES where the checking phase of the edges to add or delete is carried out in a distributed manner by using the available threads.

Apart from GES, we also consider *Fast Greedy Equivalence Search* (fGES) [20] as a competing hypothesis to test our proposal. fGES improves the original GES algorithm by adding parallel calculations.

### 2.3 Bayesian Network Fusion

Bayesian Network Fusion is the process of combining two or more BNs that share the same problem domain. The primary purpose of the fusion is to generate a BN that generalizes all the BNs by representing all the conditional independences codified in all the input BNs. BN fusion is an NP-hard problem; therefore, heuristic algorithms are used to create an approximate solution [17]. To do so, the algorithm relies on a common ordering $\sigma$ of variables, and the final result depends strongly on the ordering $\sigma$ used.

In a recent work [19], a greedy heuristic method (GHO) is proposed to find a good ordering for the fusion process. To achieve a good order, GHO must find an order that minimizes the number of transformations needed. This is accomplished by using the cost of transforming a node into a sink throughout all DAGs, being used as a scoring method to evaluate orders and using it in a heuristic to find a good order.

## 3 Ring-Based Distributed Learning of BNs

Learning BNs for high-dimensional domains is a particularly complex process since it requires a much higher number of statistical calculations, which increases the iterations needed for the learning algorithms to converge. To reduce the computational demand of the learning process, we propose executing several simpler learning processes in parallel that reduce the time spent on the algorithm. We call our proposal *Circular GES* (cGES); it is illustrated in Fig. 1, and the scheme is depicted in Algorithm 1.

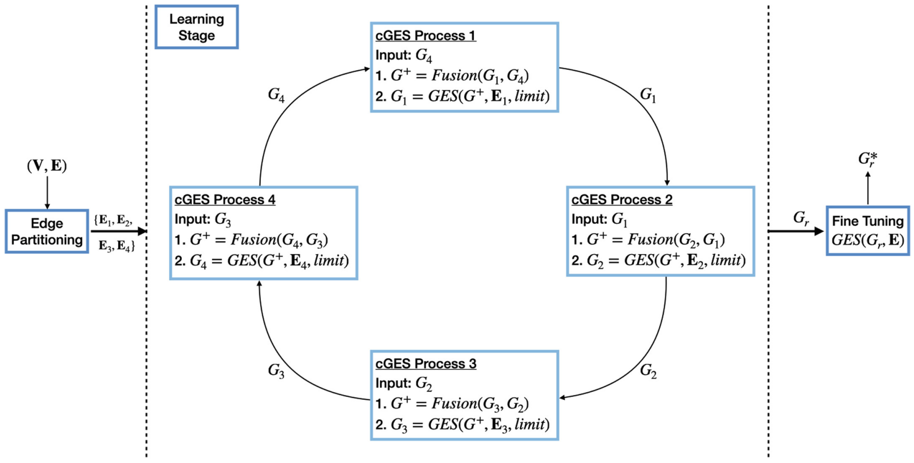
Learning Stage
cGES Process 1
Input: $G_4$
1. $G^+ = Fusion(G_1, G_4)$
2. $G_1 = GES(G^+, \mathbf{E}_1, limit)$
cGES Process 2
Input: $G_1$
1. $G^+ = Fusion(G_2, G_1)$
2. $G_2 = GES(G^+, \mathbf{E}_2, limit)$
cGES Process 3
Input: $G_2$
1. $G^+ = Fusion(G_3, G_2)$
2. $G_3 = GES(G^+, \mathbf{E}_3, limit)$
cGES Process 4
Input: $G_3$
1. $G^+ = Fusion(G_4, G_3)$
2. $G_4 = GES(G^+, \mathbf{E}_4, limit)$
$G_1$
$G_2$
$G_3$
$G_4$
$(\mathbf{V}, \mathbf{E})$
Edge Partitioning
$\{\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4\}$
$G_r$
Fine Tuning
$GES(G_r, \mathbf{E})$
$G_r^*$

**Fig. 1.** Graphical description of the proposed approach considering four processes

We can divide the algorithm into three stages:

1. Edge partitioning. Given an input dataset $D$, with $\mathbf{V} = \{X_1, \ldots, X_n\}$, as well as the set of possible edges $\mathbf{E} = \{X \to Y \mid X \in \mathbf{V}, Y \in \mathbf{V}, X \neq Y\}$, this step splits $\mathbf{E}$ into $k$ subsets $\mathbf{E}_1, \ldots, \mathbf{E}_k$, such that $\mathbf{E} = E_1 \cup \cdots \cup E_k$. This is done by using a *score-guided complete-link hierarchical clustering* that partitions $\mathbf{E}$ into $k$ clusters of edges $\mathbf{E}_i$, where each possible edge can only be assigned to one and only one cluster of edges $\mathbf{E}_i$. First, we create $k$ clusters of variables by using the BDeu score (3) [8] difference to measure the similarity or correlation between two variables:

$$s(X_i, X_j) = BDeu(X_i \leftarrow X_j \mid D) - BDeu(X_i \not\leftarrow X_j \mid D). \tag{4}$$

Where, if $s(X_i, X_j)$ (4) is positive, then adding $X_j$ as a parent of $X_i$ has an overall positive effect. The higher the score, the more related are the two variables. $s(X_i, X_j)$ is asymptotically equivalent to the mutual information. It's symmetric but non-negative, and it only measures the similarity of two variables, not the distance between them. We find a similar case in [14].
To apply the complete link approach of the hierarchical clustering, we compute the similarity between clusters $C_r$ and $C_l$ as follows:

$$s(C_r, C_l) = \frac{1}{|C_r| \cdot |C_l|} \sum_{X_i \in C_r} \sum_{X_j \in C_l} s(X_i, X_j) \tag{5}$$

**Algorithm 1:** cGES($D$,$k$)

```
Data: $D$, dataset defined over $\mathbf{V} = \{X_1, \dots, X_n\}$ variables;
$k$, the number of parallel processes;
$l$, the limit of edges that can be added in a single GES process;
Result: $G_r^* = (V, E)$, the resulting DAG learnt over the dataset $D$.
1  $\{\mathbf{E}_1, \dots, \mathbf{E}_k\} \leftarrow$ EdgePartitioning($D, k$)
2  $go \leftarrow True$
3  $G_r \leftarrow \emptyset$
4  for $(i = 1, \dots, k)$ do
5      $G_i \leftarrow \emptyset$
6  while $go$ do
7      /* Learning Stage */
8      for $(i = 1, \dots, k)$ do in parallel
9          $\hat{G} \leftarrow Fusion.edgeUnion(G_i, G_{i-1})$
10         $G_i \leftarrow GES(init = \hat{G}, edges = \mathbf{E}_i, limit = l, D)$
11     /* Convergence Checking */
12     $go \leftarrow False$
13     for $(i = 1, \dots, k)$ do
14         if $(BDeu(G^*, D) - BDeu(G_i, D) \geq 0)$ then
15             $G_r \leftarrow G_i$
16             $go \leftarrow True$
17 /* Fine Tuning */
18 $G_r^* \leftarrow GES(init = G_r, edges = \mathbf{E}, limit = \infty, D)$
19 return $G^*$
```

With the $k$ clusters of variables, we create the same number of clusters of edges. First, we assign all the possible edges among the variables of cluster $C_i$ to the subset $\mathbf{E}_i$. Next, we distribute all the remaining edges of variables belonging to different clusters. We attempt to balance the size of the resulting subsets by assigning the resulting edge with the end variables belonging to two clusters to the subset with the smallest number of edges. Finally, we obtain $k$ disjoint subsets of edges. The execution of this step occurs only once, at the beginning of the algorithm, and the resulting subsets are used to define the search space of each process of the learning stage.

2. Learning stage. In this stage, $k$ processes learn the structure of a BN. Each process $i$ receives the BN learned by its predecessor $(i - 1)$ process and its $\mathbf{E}_i$ edge cluster as input. In every iteration, all the processes are executed in parallel, where each process is limited to their assigned $\mathbf{E}_i$ edge cluster. Each process works as follows: First, the process starts by carrying out a BN fusion

[19] between the predecessor's BN and the BN the process has learned so far. If it is the first iteration, the fusion step is skipped since no BNs have been learned yet, and we use an empty graph as starting point. Next, with the result of the fusion as a starting point, a GES algorithm is launched where the edges considered for addition and deletion are restrained to the edges of its $\mathbf{E}_i$ cluster. Furthermore, an additional option is to limit the number of edges that can be added in each iteration, resulting in a shorter number of iterations and avoiding introducing complex structures that would later be pruned during the merging process. After a preliminary examination, this limitation was set to $(10/k)\sqrt{n}$, ensuring that the limitation is tailored to the size of the problem, as well as to the number of subsets $\mathbf{E}_1, \ldots, \mathbf{E}_k$ used.
Once each process learns a BN, it is used as input for the next process, creating a ring topology structure. All the processes are independent and are executed in parallel. Each inner calculation needed by GES is also performed in parallel. As noted in the above section, we use the parallel version of GES, and all the processes store the scores computed in a concurrent safe data structure to avoid unnecessary calculations. Finally, when an iteration has finished, the convergence is checked by comparing whether any of the resulting BNs has improved its BDeu score over the best BN constructed so far. When no BN has outperformed the up to now best BN, the learning stage finishes.

3. Fine tuning. Once the learning stage has finished, the parallel version of the GES algorithm is executed using the resulting BN as a starting point. This time, the GES algorithm uses all the edges of $\mathbf{E}$ without adding any limitation. As we expect to start from a solution close to the optimal, this stage will only carry out a few iterations. Since we apply a complete run of GES (FES+BES) over the resulting graph, all the theoretical properties of GES will be maintained as they are independent of the starting network considered.

It is important to notice that, by using this ring topology, the fusion step only takes two networks as input, thus avoiding obtaining very complex (dense) structures and so reducing overfitting. Furthermore, throughout each iteration, the BNs generated by each process will be of greater quality, generalizing better with each iteration since more information is shared. By limiting the number of edges added, the complexity of each BN is smaller, and the fusions make smaller changes, creating more consistent BNs in each process. A general overview of the learning stage can be seen in Fig. 1.

## 4 Experimental Evaluation

This section describes the experimental evaluation of our proposal against competing hypotheses. The domains and algorithms considered, the experimental methodology, and the results obtained are described below.

### 4.1 Algorithms

In this study, we examined the following algorithms:

– An enhanced version of the GES algorithm [5] introduced in [1] (see Sect. 2.2). Notably, the implementation in this study incorporates parallelism to expedite the computational processes. In each iteration, to find the best edge to be added or deleted, the computation of the scores is implemented in parallel by distributing them among the available threads.
– The fGES algorithm, introduced in [20].
– The proposed cGES algorithm (see Sect. 3). We evaluate this algorithm with 2, 4, and 8 edge clusters, as well as limiting and non-limiting configurations for the number of edges inserted in each iteration.

### 4.2 Methodology

Our methodology for evaluating Bayesian network learning algorithms involved the following steps:

First, we selected three real-world BNs from the Bayesian Network Repository in bnlearn [23] and sampled 11 datasets of 5000 instances for each BN. The largest BNs with discrete variables, namely `link`, `pigs`, and `munin`, were chosen for analysis. For each BN, Table 1 provides information about the number of *nodes*, *edges*, *parameters* in the conditional probability tables, the *maximum* number of *parents* per variable, average *BDeu* value of the *empty* network, and the structural Hamming distance between the *empty* network and the moralized graph of the original BN (*SMHD*) [10].

**Table 1.** Bayesian networks used in the experiments.

| Network | Features | | | | | |
|---|---|---|---|---|---|---|
| | Nodes | Edges | Parameters | Max parents | Empty BDeu | Empty SMHD |
| Link | 724 | 1125 | 14211 | 3 | −410.4589 | 1739 |
| Pigs | 441 | 592 | 5618 | 2 | −459.7571 | 806 |
| Munin | 1041 | 1397 | 80592 | 3 | −345.3291 | 1843 |

We considered several evaluation scores to assess the algorithms' efficiency and accuracy. These included the CPU time required by each algorithm for learning the BN model from data, the BDeu score [8] measuring the goodness of fit of the learned BN with respect to the data normalized by the number of instances as in [24], and the Structural Moral Hamming Distance (SMHD) between the learned and original BN, measuring the actual resemblance between the set of probabilistic independences of the moralized graph of the two models (see, e.g., [11]).

Our methodology tested the configuration of each algorithm on the 11 samples for each of the three BNs. The results reported are the average of these runs for each evaluation score. This approach allowed us to systematically evaluate the performance of the BN learning algorithms across multiple datasets and provide comprehensive insights into their efficiency and accuracy.

## 4.3 Reproducibility

To ensure consistent conditions, we implemented all the algorithms from scratch, using Java (OpenJDK 8) and the Tetrad 7.1.2-2[2] causal reasoning library. The experiments were conducted on machines equipped with Intel Xeon E5-2650 8-Core Processors and 64 GB of RAM per execution running the CentOS 7 operating system.

To facilitate reproducibility, we have made the datasets, code, and execution scripts available on GitHub[3]. Specifically, we utilized the version 1.0 release for the experiments presented in this article. Additionally, we have provided a common repository on OpenML[4] containing the 11 datasets sampled for each BN referencing their original papers.

## 4.4 Results

Table 2 present the corresponding results for the BDeu score (2a), Structural Moral Hamming Distance (SMHD) (2b), and execution time (2c) of each algorithm configuration discussed in Sect. 4.1. The notation cGES-l refers to the variant of cGES that imposes limitations on the number of added edges per iteration, while the numbers 2, 4, and 8 indicate the number of processes in the ring. The algorithm exhibiting the best performance for each Bayesian network is highlighted in bold to emphasize the superior results.

These results lead us to the following conclusions:

– Of the algorithms evaluated, fGES stands out as the least effective option, producing subpar results or exhibiting significantly longer execution times when obtaining a good result. In terms of the quality of the BN generated, fGES yields unsatisfactory outcomes, as evidenced by low BDeu scores and high SMHD values in the `pigs` and `link` networks. Furthermore, when aiming to construct a reasonable network, fGES requires substantially longer execution times compared to both GES and all cGES variants. This is evident in the case of the `munin` network.
– Upon comparing the versions of cGES, namely cGES-l and cGES, which respectively impose limits on the number of edges that can be added by each FES run to $(10/numClusters)\sqrt{nodes}$ and have no such restriction, it becomes evident that cGES-l outshines cGES in terms of performance. In most cases, cGES-l demonstrates superior performance in generating high-quality BNs compared to cGES. Additionally, it consistently achieves an

[2] https://github.com/cmu-phil/tetrad/releases/tag/v7.1.2-2.
[3] https://github.com/JLaborda/cges.
[4] https://www.openml.org/search?type=data&uploader_id=%3D_33148&tags.tag=bnlearn.

**Table 2.** Results (BDeu, SHMD and CPU Time)

| Network | Algorithm | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | fGES | GES | cGES 2 | cGES 4 | cGES 8 | cGES-l 2 | cGES-l 4 | cGES-l 8 |
| Pigs | −345.1826 | **−334.9989** | −335.6668 | −335.8876 | −335.5411 | −335.1105 | −335.1276 | −335.1865 |
| Link | −286.1877 | −228.3056 | −228.3288 | −227.1207 | **−226.4319** | −227.6806 | −227.9895 | −227.2155 |
| Munin | **−186.6973** | −187.0736 | −187.1536 | −186.7651 | −187.8554 | −186.9388 | −187.2936 | −187.4198 |

(a) BDeu score

| **Network** | **Algorithm** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | fGES | GES | cGES 2 | cGES 4 | cGES 8 | cGES-l 2 | cGES-l 4 | cGES-l 8 |
| Pigs | 309.00 | **0.00** | 31.00 | 36.91 | 21.00 | 4.36 | 4.18 | 5.18 |
| Link | 1370.45 | 1032.36 | 1042.18 | 953.18 | 940.64 | **937.91** | 952.64 | 941.55 |
| Munin | 1489.64 | **1468.45** | 1531.18 | 1521.38 | 1668.89 | 1503.25 | 1558.30 | 1623.22 |

(b) Structural Moral Hamming Distance (SHMD)

| **Network** | **Algorithm** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | fGES | GES | cGES 2 | cGES 4 | cGES 8 | cGES-l 2 | cGES-l 4 | cGES-l 8 |
| Pigs | **20.26** | 175.43 | 122.47 | 108.08 | 121.80 | 76.59 | 58.06 | 73.84 |
| Link | **41.12** | 746.54 | 694.08 | 463.92 | 447.62 | 383.04 | 276.72 | 286.56 |
| Munin | 12331.31 | 2000.00 | 1883.78 | 1330.62 | 1454.72 | 1433.19 | 895.76 | **791.36** |

(c) CPU Time (seconds)

impressive speed-up, with execution times reduced by approximately a half compared to cGES. These findings highlight the effectiveness of the edge limitation strategy employed in cGES-l and its significant impact on the learning process's quality and efficiency.

– When comparing the algorithms based on the number of ring processes (processes or edge subsets), it is challenging to establish a consistent pattern regarding the quality of the BNs generated. While there is a general trend of cGES performing slightly better with more partitions and cGES-l with fewer, this pattern may vary depending on the BN. However, regarding execution time, it is evident that using 4 or 8 clusters improves the efficiency compared to using 2 clusters. In particular, as the size of the BN increases, using 8 clusters tends to yield better execution times.
– Lastly, comparing the fastest variant of cGES in two out of three BNs (cGES-l 4) with GES yields noticeable speed improvements. `pigs`, `link`, and `munin` BNs experience speed-ups of 3.02, 2.70, and 2.23, respectively. These values are significant considering that both algorithms run parallel utilizing 8 CPU threads. Notably, the reduced speed-up execution time does not come at the cost of lower-quality BNs. In fact, GES performs better on `pigs` and `munin` BNs, while cGES-l 4 excels with the `link` BN. However, these differences in performance are not as pronounced as those observed with the BN generated by fGES on the `pigs` and `link` networks.

## 5 Conclusions

Our study introduces cGES, an algorithm for structural learning of Bayesian Networks in high-dimensional domains. It employs a divide-and-conquer approach, parallelism, and fusion techniques to reduce complexity and improve learning efficiency. Our experimentation demonstrates that cGES generates high-quality BNs in significantly less time than traditional methods. While it may not always produce the absolute best solution, cGES strikes a favourable balance between BN quality and generation time. Another important point to be considered is that cGES exhibits the same theoretical properties as GES, as an unrestricted GES is run by taking the network identified by the ring-distributed learning process as its starting point.

As future works, the algorithm's modular structure opens up possibilities for applications such as federated learning [15], ensuring privacy and precision.

**Acknowledgements.** This work has been funded by the Government of Castilla-La Mancha and "ERDF A way of making Europe" under project SBPLY/21/180225/000062. It is also partially funded by MCIN/AEI/10.13039/501100011033 and "ESF Investing your future" through PID2019–106758GB–C33, TED2021-131291B-I00 and FPU21/01074 projects. Furthermore, this work has been supported by the University of Castilla-La Mancha and "ERDF A Way of Making Europe" under project 2023-GRIN-34437. Finally, this work has also been funded by the predoctoral contract with code 2019-PREDUCLM-10188, granted by the University of Castilla-La Mancha.

## References

1. Alonso-Barba, J.I., delaOssa, L., Gámez, J.A., Puerta, J.M.: Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes. Int. J. Approximate Reasoning **54**(4), 429–451 (2013). https://doi.org/10.1016/j.ijar.2012.09.004
2. Arias, J., Gámez, J.A., Puerta, J.M.: Structural learning of Bayesian networks via constrained hill climbing algorithms: adjusting trade-off between efficiency and accuracy. Int. J. Intell. Syst. **30**(3), 292–325 (2015). https://doi.org/10.1002/int.21701
3. de Campos, C.P., Ji, Q.: Efficient structure learning of Bayesian networks using constraints. J. Mach. Learn. Res. **12**, 663–689 (2011). http://jmlr.org/papers/v12/decampos11a.html
4. de Campos, L.M.: A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. J. Mach. Learn. Res. **7**, 2149–2187 (2006). http://jmlr.org/papers/v7/decampos06a.html
5. Chickering, D.M.: Optimal structure identification with greedy search. J. Mach. Learn. Res. **3**(Nov), 507–554 (2002). http://www.jmlr.org/papers/v3/chickering02b.html
6. Chickering, D.M., Heckerman, D., Meek, C.: Large-sample learning of bayesian networks is np-hard. J. Mach. Learn. Res. **5**, 1287–1330 (2004). https://www.jmlr.org/papers/v5/chickering04a.html

7. Gámez, J.A., Mateo, J.L., Puerta, J.M.: Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. Data Mining Knowl. Discov. **22**(1), 106–148 (2011). https://doi.org/10.1007/s10618-010-0178-6
8. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: the combination of knowledge and statistical data. Mach. Learn. **20**, 197–243 (1995). https://doi.org/10.1007/BF00994016
9. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer, New York (2007). https://doi.org/10.1007/978-0-387-68282-2
10. de Jongh, M., Druzdzel, M.J.: A comparison of structural distance measures for causal Bayesian network models. In: Klopotek, M., Przepiorkowski, A., Wierzchon, S.T., Trojanowski, K. (eds.) Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series, pp. 443–456. Academic Publishing House EXIT (2009). https://doi.org/10.1007/978-3-030-34152-7
11. Kim, G.-H., Kim, S.-H.: Marginal information for structure learning. Stat. Comput. **30**(2), 331–349 (2019). https://doi.org/10.1007/s11222-019-09877-x
12. Kjaerulff, U.B., Madsen, A.L.: Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis, 2nd edn. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-5104-4
13. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, Cambridge (2009)
14. Krier, C., François, D., Rossi, F., Verleysen, M.: Feature clustering and mutual information for the selection of variables in spectral data, pp. 157–162 (2007)
15. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Signal Process. Mag. **37**(3), 50–60 (2020). https://doi.org/10.1109/MSP.2020.2975749
16. Lin, Y., Druzdzel, M.J.: Computational advantages of relevance reasoning in Bayesian belief networks. In: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, pp. 342–350. UAI 1997, Morgan Kaufmann Publishers Inc. (1997)
17. Peña, J.: Finding consensus Bayesian network structures. J. Artif. Intell. Res. (JAIR) **42**, 661–687 (2011). https://doi.org/10.1613/jair.3427
18. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
19. Puerta, J.M., Aledo, J.A., Gámez, J.A., Laborda, J.D.: Efficient and accurate structural fusion of Bayesian networks. Inf. Fusion **66**, 155–169 (2021). https://doi.org/10.1016/j.inffus.2020.09.003
20. Ramsey, J., Glymour, M., Sanchez-Romero, R., Glymour, C.: A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. Int. J. Data Sci. Anal. **3**(2), 121–129 (2016). https://doi.org/10.1007/s41060-016-0032-z
21. Scanagatta, M., Campos, C.P.D., Corani, G., Zaffalon, M.: Learning Bayesian networks with thousands of variables. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 2, pp. 1864–1872. NIPS 2015, MIT Press (2015)
22. Scanagatta, M., Salmerón, A., Stella, F.: A survey on Bayesian network structure learning from data. Progress Artif. Intell. **8**(4), 425–439 (2019). https://doi.org/10.1007/s13748-019-00194-y

23. Scutari, M.: Learning Bayesian networks with the bnlearn R package. J. Stat. Softw. **35**(3), 1–22 (2010). https://doi.org/10.18637/jss.v035.i03
24. Teyssier, M., Koller, D.: Ordering-based search: a simple and effective algorithm for learning Bayesian networks, pp. 584–590. UAI 2005, AUAI Press, Arlington, Virginia, USA (2005)
25. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Mach. Learn. **65**(1), 31–78 (2006). https://doi.org/10.1007/s10994-006-6889-7