mathematics

Article

The FEDHC Bayesian Network Learning Algorithm

Michail Tsagris

Department of Economics, University of Crete, Gallos Campus, 74100 Rethymnon, Greece; mtsagris@uoc.gr

Abstract: The paper proposes a new hybrid Bayesian network learning algorithm, termed Forward Early Dropping Hill Climbing (FEDHC), devised to work with either continuous or categorical variables. Further, the paper manifests that the only implementation of MMHC in the statistical software *R* is prohibitively expensive, and a new implementation is offered. Further, specifically for the case of continuous data, a robust to outliers version of FEDHC, which can be adopted by other BN learning algorithms, is proposed. The FEDHC is tested via Monte Carlo simulations that distinctly show that it is computationally efficient, and that it produces Bayesian networks of similar to, or of higher accuracy than MMHC and PCHC. Finally, an application of FEDHC, PCHC and MMHC algorithms to real data, from the field of economics, is demonstrated using the statistical software *R*.

Keywords: causality; Bayesian networks; scalability

MSC: 62H22

check for updates

Citation: Tsagris, M. The FEDHC Bayesian Network Learning Algorithm. *Mathematics* **2022**, *10*, 2604. https://doi.org/10.3390/math10152604

Academic Editors: Snezhana Gocheva-Ilieva, Atanas Ivanov and Hristina Kulina

Received: 23 May 2022
Accepted: 20 July 2022
Published: 26 July 2022

1. Introduction

Learning the causal relationships among variables using non-experimental data is of high importance in many scientific fields, such as economics and econometrics (for a general definition of causality specifically in economics and econometrics, see [1]). When the aim is particularly to recreate the causal mechanism that generated the data, graphical models, such as causal networks and Bayesian Networks (BNs) (also known as Bayes networks, belief networks, decision networks, Bayes(ian) models or probabilistic directed acyclic graphical models) are frequently employed. The advantages of BNs include simultaneous variable selection among all variables, and hence, the detection of conditional associations between variables. On a different route, BNs form the scheme for synthetic population generation [2], and have been used synergetically with agent-based models [3,4].

BNs enjoy applications to numerous fields, but the focus of the current paper is on fields related to economics applications, such as production economics [5], macroeconomics [6] and environmental resource economics [7]. Applications of BNs can also be found in financial econometrics [8], banking and finance [9], credit scoring [10], insurance [11] and customer service [12], to name a few. Despite the plethora of applications of BNs, not many BN algorithms exist, and most importantly, fewer are publicly available in free software environments, such as the statistical software *R*. The Max-Min Hill Climbing (MMHC) [13] is an example of a widely used BN learning algorithm (The relevant paper is one of the classic papers in the Artificial Intelligence field, and has received more than 1870 citations according to scholar.google.com as of July 2022.) that is publicly available, in the *R* package *bnlearn* [14]. PC Hill Climbing (PCHC) [15] is a recently suggested hybrid algorithm that is also publicly available, in the *R* package *pchc* [16].

Ref. [15] showed that when the sample size is in the order of hundreds of thousands, MMHC's implementation in the *R* package *bnlearn* requires more than a day with continuous data, even with 50 variables. On the contrary, PCHC is a computationally efficient and scalable BN learning algorithm [15]. With modern technology and vast data generation, the computational cost is a considerable parameter. Every novel algorithm must be computationally efficient and scalable to large sample sizes. Seen from the green economy point of view, this cost also has an economic and environmental impact; a faster algorithm will produce results in a more timely manner, facilitating faster decision making, consuming less energy and hence reducing its carbon footprint.

Moving along those lines, this paper proposes a new computationally highly efficient algorithm termed Forward with Early Dropping Hill Climbing (FEDHC), which is publicly available in the *R* package *pchc*. FEDHC shares common ideas with PCHC and MMHC. It applies the Forward Backward with Early Dropping (FBED) variable selection algorithm [17] to each variable as a means of skeleton identification, followed by a Hill Climbing (HC) scoring phase. FEDHC can handle millions of observations in just a few minutes, and retains similar or better accuracy levels than PCHC and MMHC. With continuous data, FEDHC performs fewer errors than PCHC, but the converse is true with categorical data. FEDHC further enjoys the same scalability property as PCHC, and its computational cost is proportional to the sample size of the data. Increasing the sample size by a factor increases the execution time by the same factor. Finally, a new, computationally efficient implementation of MMHC is offered that is also publicly available in the *R* package *pchc*.

The choice of the BN learning algorithm is not only computational cost-dependent, but also quality-dependant. Regardless of the algorithm used, the quality of the learned BN can be significantly affected by outliers. Robustness to outliers is an important aspect that, surprisingly enough, has not attracted substantial research attention in the field of BN learning. Ref. [18] were the first to propose a robustified version of the PC algorithm, by replacing the empirical standard deviations with robust scale estimates. Ref. [19] on the other hand, removed the outliers, but their algorithm is only applicable to BNs with a known topology. Robustification of the proposed FEDHC algorithm takes place by adopting techniques from the robust statistics literature. The key concept is to identify and remove the outliers prior to applying FEDHC.

The rest of the paper is structured as follows. Preliminaries regarding BNs that will assist in making the paper comprehensible, and a brief presentation of the PCHC and MMHC algorithms are unveiled in Section 2. The FEDHC is introduced in Section 3, along with its robustified version (the robustified versions are applicable to PCHC and MMHC as well), which will be shown to be remarkable and utterly insensitive to outliers. The theoretical properties and computational details of FEDHC, and the conditional independence tests utilised for continuous and categorical data are delineated in the same section. Section 4 contains Monte Carlo simulation studies comparing FEDHC to PCHC and MMHC in terms of accuracy, computational efficiency and number of tests performed. Section 5 illustrates the FEDHC, PCHC and MMHC on two real cross-sectional datasets using the *R* package *pchc*. The first dataset contains continuous data on the credit history for a sample of applicants for a type of credit card [20]. The second dataset contains categorical data on the household income plus some more demographic variables. Ultimately, Section 6 contains the conclusions drawn from this paper.

2. Preliminaries

Graphical models or probabilistic graphical models express visually the conditional (in)dependencies between random variables (V_i, $i = 1, \dots, D$). Nodes (or vertices) are used to represent the variables V_i and edges between the nodes; for example, $V_i - V_j$ indicates the relationship between the variables V_i and V_j. Directed graphs are graphical models that contain arrows instead of edges, indicating the direction of the relationship; for example, $V_i \rightarrow V_j$. The parents of a node V_i are the nodes whose direction (arrows) points towards V_i. Consequently, node V_i is termed the child of those nodes and the common parents of those nodes are called spouses. Directed acyclic graphs (DAG) are stricter in the sense that they impose no cycles on these directions. For any path between V_i and V_j, $V_i \rightarrow V_k \rightarrow \dots \rightarrow V_j$, no path from V_j to V_i ($V_j \rightarrow \dots \rightarrow V_i$) exists. In other words, the natural sequence or relationship between parents and children or ancestors and descendants is mandated.

2.1. Bayesian Networks

Assume there is a collection $\mathbf{V}$ of D variables whose joint distribution P is known. The BN (BN is a special case of a DAG) [21,22] $B = \langle G, \mathbf{V} \rangle$ arises from linking P to G through the Markov condition (or Markov property), which states that each variable is conditionally independent of its non-descendants, given its parents. By using this condition, the joint distribution P can be factorised as:

$$P(V_1, \ldots, V_D) = \prod_{i=1}^{D} P(V_i | Pa(V_i)), \tag{1}$$

where $Pa(V_i)$ denotes the parents set of V_i in G.

If G entails only conditional independencies in P and all conditional independencies in P are entailed by G, based on the Markov condition, then G, P and G are faithful to each other, and G is a perfect map of P [22].

The BN whose edges can be interpreted causally is called causal BN; an edge $V_i \rightarrow V_j$ exists if V_i is a direct cause of V_j. A necessary assumption made by the algorithms under study is causal sufficiency; there are no latent (hidden, non-observed) variables among the observed variables $\mathbf{V}$.

The triplet (V_i, V_k, V_j), where $V_i \rightarrow V_k \leftarrow V_j$ is called V-structure. If there is no edge between V_i and V_j, the node V_k is called an unshielded collider. In Figure 1, the triplet (V_1, V_3, V_2) is a V-structure as there is no edge between V_1 and V_2, and hence, node V_3 is an unshielded collider. The unshielded collider V_k implies that V_i and V_j are independent conditioning on V_k, provided that the faithfulness property holds true [22]. Conversely, the triplet of nodes (V_i, V_k, V_j) such that $V_k \rightarrow V_i$ and $V_k \rightarrow V_j$ is termed Λ-structure (nodes V_3, V_4 and V_5 in Figure 1 is such an example). The Λ-structure implies that V_i and V_j are conditionally independent, given V_k.

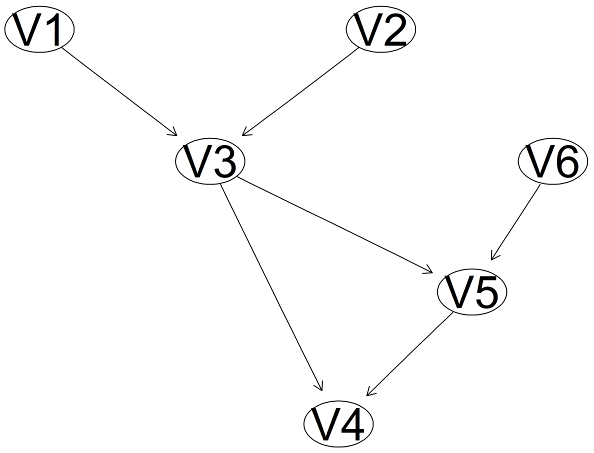


Figure 1. An example of a DAG. Nodes V1 and V2 are the parents of V3, whose children are nodes V4 and V5. V2 is the spouse of V1 (and vice versa, V1 is the spouse of V2).

Two or more BNs are called Markov equivalent, if and only if they have the same skeleton and the same V-structures [23]. The set of all Markov equivalent BNs forms the Markov equivalence class that can be represented by a complete partially DAG, which in addition to directed edges, contains undirected edges. (Undirected edges may be oriented either way in BNs of the Markov equivalence class (in the set of all valid combinations), while directed and missing edges are shared among all equivalent BNs.)

2.2. Classes of BN Learning Algorithms

BN learning algorithms are typically constraint-based, score-based or hybrid. Constraint-based learning algorithms, such as PC (PC stands for Peter and Clark, named after Peter Spirtes and Clark Glymour, the names of the two researchers who invented it) [24] and

FCI [22] employ conditional independence (CI) tests to discover the structure of the network (skeleton), and then orient the edges by repetitively applying orientation rules. On the contrary, score-based methods [25–27] assign a score on the whole network and perform a search in the space of BNs to identify a high-scoring network. Hybrid algorithms, such as MMHC [13] and PCHC [15], combine both aforementioned methods; they first perform CI tests to discover the skeleton of the BN, and then employ a scoring method to direct the edges in the space of BNs. For a series of CI tests for continuous and categorical data, please see Appendix A.

2.3. PCHC and MMHC Algorithms

The PCHC's skeleton identification phase is the same as that of the PC algorithm [15]. The phase commences with all pairwise unconditional associations and removes the edge between ordered pairs that are not statistically significantly associated. Subsequently, CI tests are performed with the cardinality of the conditioning set (denoted by k) increasing by 1 at a time. At every step, the conditioning set consists of subsets of the neighbours, adjacent to each variable V_i ($adj(G, V_i)$). This process is repeated until no edge can be removed.

Ref. [22] suggested three heuristics to select the pairs of variables, and the order is crucial, as it can yield erroneous results. The optimal process, for a given variable V_i, is to first test those variables $\mathbf{V}$ that are least probabilistically dependent on V_i, conditional on those subsets of variables that are most probabilistically dependent on V_i. Note that the pairs are first ordered according to the third heuristic of [22], and so, the order of selection of the pairs is deterministic. Hence, the skeleton identification phase is independent of the order at which the variables are located in the dataset [28].

MMHC's skeleton identification phase performs a variable selection process for each variable (call it target variable, V_i), described as follows. A search for its statistically significantly associated variable V_s is performed via unconditional statistical tests. The associations are stored, and the variable with the highest association (V_j) is chosen; an edge is added between this V_i and V_j, and all non-statistically significant variables are excluded from further consideration. In the second step, all CI tests between the target variable and previously identified variables, conditional on the previously selected variable, are performed $\left(V_i \perp\!\!\!\perp V_m | V_j,\ \ m \neq i, j\right)$, and the non-statistically significant variables are neglected. The previously stored associations are updated; for each variable, the minimum between the old and the new variables is stored. The variable with the highest association (Max-Min heuristic) is next selected. In subsequent steps, while the set of the selected variables increases, the conditioning set does not, as its cardinality is at most equal to k. (This algorithm resembles the classical forward variable selection in statistics with two distinct differences. At each step, non-significant variables are excluded from future searches, instead of conditioning on all selected variables. Secondly, the CI tests for the next variable conditions upon all possible subsets, up to a pre-specified cardinality, of the already selected variables.) Upon completion, a backward phase, in the same spirit as the forward, applies to remove falsely detected variables.

This variable selection process is repeated for all variables. The edges detected remain only if they were identified by all variables. If, for example, V_j was found to be associated with V_i, but V_i was not found to be associated with V_j, then no edge between V_i and V_j will be added.

A slightly modified version of MMHC's skeleton identification phase is implemented in the *R* package *pchc*. The backward phase is not performed, in order to make the algorithm faster. To distinguish between them, *bnlearn*'s implementation will be denoted by MMHC-1, and *pchc*'s implementation will be denoted by MMHC-2, hereafter.

The orientation of the discovered edges takes place in the second, Hill Climbing (HC) scoring, phase of PCHC and MMHC, and is the same phase employed by FEDHC as well.

3. The FEDHC BN Learning Algorithm

Similarly to PCHC and MMHC, the skeleton identification phase of FEDHC relies on a variable selection algorithm. Thus, prior to introducing the FEDHC algorithm, the Forward Backward with Early Dropping (FBED) variable selection algorithm [17] is briefly presented.

3.1. The FBED Variable Selection Algorithm

In the classical forward selection algorithm, all available predictor variables are constantly used, and their statistical significance is tested at each step. Assuming that out of 10, 000 predictor variables, only 10 are selected. This implies that almost 10,000 $\times$ 10 regression models must be fitted, and the same amount of statistical tests must be executed. The computational cost is tremendous, rendering this computationally expensive algorithm impractical, and hence, prohibitive. The authors of [17] introduced the FBED algorithm as a speed-up modification of the traditional forward selection algorithm coupled with the backward selection algorithm [29]. FBED relies on the Early Dropping heuristic to speed up the forward selection. The heuristic drops the statistically non-significant predictor variables at each step, thus removing them from further consideration, resulting in a computationally dramatically cheaper algorithm that is presented in Algorithm 1. Especially for the case of continuous data this phase can be completed computationally efficient using only the correlation matrix (see Appendix B for more details).

Algorithm 1 The FBED variable selection algorithm.

```
1: **Input**: A response variable $y$ and a set of $D$ predictor variables $\mathbf{V}$.
2: Let $\mathbf{S} = \emptyset$ denote the set of selected variables.
3: Perform all regression models of $y$ on each $V_i$, $i = 1, \ldots, D$, $y \sim f(V_i)$, where $f$ denotes
   a function of $V_i$; e.g., a linear model $y = a + bV_i + e$, and retain only the statistically
   significant predictor variables $\mathbf{V}_{sig}$.
4: Choose $V_j$ from $\mathbf{V}_{sig}$ that has the highest association, add it in $\mathbf{S}$ and use that to perform
   all regression models of $y$ on the $V_j$ and each $V_\ell$, $y \sim f(V_j, V_\ell)$, where $\ell \in \mathbf{V}$, with $\ell \neq j$
   and again retain only the statistically significant predictor variables, thus reducing
   $|\mathbf{V}_{sig}|$ and increasing $|\mathbf{S}|$.
5: Repeat until no predictor variable is left, i.e., $\mathbf{V}_{sig} = \emptyset$.
6: This process can be repeated $k$ times, using all neglected predictor variables, where $k$ is
   a pre-defined number, until $|\mathbf{S}|$ cannot further increase.
7: Perform a backward selection phase attempting to remove the non statistically significant predictor variables.
8: **Return S**.
```

3.2. Skeleton Identification Phase of the FEDHC Algorithm

The skeleton identification phase of the FEDHC algorithm is the one presented in Algorithm 2, but it must be stressed that the backward phase of FBED is not performed, so as to reduce the computational cost. The FBED algorithm (Algorithm 1) is used for each variable (call it target variable, V_i). This variable selection process is repeated for all variables. The edges detected remain only if they were identified by all variables. If, for example, V_j was found to associated with V_i, but V_i was not found to be associated with V_j, then no edge between V_i and V_j will be added.

Algorithm 2 Skeleton identification phase of the FEDHC algorithm.

```
1: **Input**: Data set on a set of $D$ variables $\mathbf{V}$.
2: Let the adjacency matrix $G$ be full of zeros.
3: **Repeat** for all variables $V_i$, $i = 1, \ldots, D$
4: Perform the FBED algorithm in Algorithm 1, excluding the backward phase, and return $\mathbf{S}_i$.
5: Set $G_{ij} = 1$ for all $j \in \mathbf{S}_i$.
6: **If** $G_{ij} \neq G_{ji}$ set $G_{ij} = G_{ji} = 0$.
7: **Return** $G$.
```

3.3. Hill Climbing Phase of the FEDHC Algorithm

The first phase of FEDHC, MMHC and of PCHC is to discover any possible edges between the nodes, using CI tests. In the second phase, a search for the optimal DAG is performed, where the edges turn to arrows or are deleted towards the maximisation of a score metric. This scoring phase performs a greedy HC search in the space of BNs, commencing with an empty graph [13]. The edge deletion or direction reversal that leads to the largest increase in score, in the space of BNs (This implies that every time an edge removal or arrow direction is implemented, a check for cycles is performed. If cycles are created, the operation is canceled regardless of whether it increases the score), is applied, and the search continues recursively. The fundamental difference from standard greedy search is that the search is constrained to the orientation of the edges discovered by the skeleton identification phase (For more information, see [13]).

Tabu search is an iterative local searching procedure adopted by [13] for this purpose. Its performance is enhanced by using a list where the last 100 structures explored are stored, while searching in the neighborhood of each solution. The search is also capable of escaping from a local optima, in which the normal local search techniques often become stuck. Instead of applying the best local change, the best local change that results in a structure not on the list is performed, in an attempt to escape the local maxima [13]. This change may actually reduce the score. When a number of changes (10–15) occur without an increase in the maximum score ever encountered during a search, the algorithm terminates. The overall best scoring structure is then returned.

The Bayesian Information Criterion (BIC) [30] is a frequent score used for continuous data, while other options include the multivariate normal log-likelihood, the Akaike Information Criterion (AIC) and the Bayesian Gaussian equivalent [31] score. (The term "*equivalent*" refers to their attractive property of giving the same score to equivalent structures (Markov equivalent BNs), i.e., structures that are statistically indistinguishable [13]) The Bayesian Dirichlet equivalent (BDE) [32], the BDe uniform score (BDeu) [27], the multinomial log-likelihood score [33] and the MDL score [34,35] are four scoring options for discrete data.

The combination of the FBED algorithm during the skeleton identification phase with the HC scoring method forms the FEDHC algorithm. Interestingly enough, the skeleton identification phase of FEDHC performs substantially fewer statistical tests than PCHC and MMHC.

3.4. Prior Knowledge

All BN learning algorithms are agnostic of true relationships among the input data. It is customary though, for practitioners and researchers to have prior knowledge of the necessary directions (forbidden or not) of some of the relationships among the variables. For instance, variables such as sex or age cannot be caused by any economic or demographic variables. Economic theory (or theory from any other field) can further assist in improving the quality of the fitted BN by imposing or forbidding directions among some variables. All of the prior information can be inserted into the scoring phase of the aforementioned BN learning algorithms, leading to less errors and more realistic BNs.

3.5. Theoretical Properties of FEDHC

The theoretical properties and guarantees of MMHC and PCHC can be found in [13,15], respectively. As for the FEDHC, while there is no theoretical guarantee of the skeleton identification phase of FEDHC, Ref. [17] showed that running FBED with two repeats recovers the MB of the response variable if the joint distribution of the response and the predictor variables can be faithfully represented by a BN. When used for BN learning though, FBED need not be run more than once for each variable. In this case, FBED, similarly to MMHC, will identify the children and parents of a variable V_i, but not the spouses of the children [17], as this is not necessary during the skeleton identification phase. When FBED is run on the children of the variable V_i, it will again identify the children's parents who are the spouses of the variable V_i. Hence, upon completion, the FEDHC algorithm will have identified the MB of each variable.

Additionally, the early dropping heuristic does not only reduce the computational time, but also reduces the problem of multiple testing, in some sense [17]. When FBED is run only once (as in the current situation), in the worst-case scenario, it is expected to select about $\alpha \cdot D$ variables (where α is the significance level), since all other variables will be dropped in the first (filtering) phase. However, simulation studies have shown that FBED selects fewer false positives than expected and the authors' recommendation is to reduce the number of runs to further limit the number of falsely selected variables, a strategy that FEDHC follows by default.

Similar to MMHC, the FEDHC is a local learning algorithm, and hence during the HC phase, the overall score is decomposed [13], exploiting the Markov property of BNs (1). The local learning has several advantages (see [13]) and the scores (BDe, BIC., etc.) are locally consistent [25].

3.6. Robustification of the FEDHC Algorithm for Continuous Data

It is not uncommon for economic datasets to contain outliers, observations with values that are far from the rest of the data. Income is such an example that contains outliers, but if outliers appear only in that variable, their effect will be minor. The effect of outliers is propagated when they exist in more variables, and in order to mitigate their effect, they must be identified in the multivariate space. If these outliers are not detected or not treated properly, BN learning algorithms will yield erroneous results. FEDHC will employ the Reweighted Minimum Covariance Determinant (RMCD) [36,37] as a means to identify outliers and remove them. (The reason for why one cannot use the robust correlation matrix directly is because the independence property between two variables no longer holds true. The robust correlation between any two variables depends on the other variables, and so adding or removing a variable modifies the correlation structure [38].)

The RMCD estimator is computed in two stages. In the first stage, a subset of observations h ($n/2 \leq h < n$) is selected, such that the covariance matrix has the smallest determinant, and a robust mean vector is also computed. The second stage is a re-weighting scheme that increases the efficiency of the estimator, while preserving its high-breakdown properties. A weight $w_i = 0$ is given to observations whose first-stage robust distance exceeds a threshold value, otherwise the weight of $w_i = 1$ ($i = 1, \ldots, n$) is given. Using the re-weighted robust covariance matrix and mean vector, robust Mahalanobis distances are computed $d^2_{i(RMCD)} = \left(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_{(RMCD)}\right)^T \tilde{\boldsymbol{\Sigma}}^{-1}_{(RMCD)} \left(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_{(RMCD)}\right)$ and proper cut-off values are required to detect the outliers. Those cut-off values are based on the following accurate approximations [39,40]:

$$d^2_{i(RMCD)} \begin{array}{ll} \sim & \frac{(w-1)^2}{w} Be\left(\frac{D}{2}, \frac{w-D-1}{2}\right) \quad \text{if } \; w_i = 1 \\ \sim & \frac{w+1}{w} \frac{(w-1)D}{w-D} F_{D,w-D} \quad \text{if } \; w_i = 0, \end{array}$$

where $w = \sum_{i=1}^n w_i$, and Be and F denote the Beta and F distributions, respectively.

The observations whose Mahalananobis distance $d^2_{i(RMCD)}$ exceeds the 97.5% quantile of either distribution (Be or F) are considered to be outliers, and are hence removed from the dataset. The remainder of the dataset, assumed to be outlier-free, will be used by FEDHC to learn the BN.

The default value for h is $[(n+p+1)/2]$, where $[.]$ denotes the largest integer. This value was proven to have the highest breakdown point [41], but a low efficiency, on the other hand. Changing h yields an estimator with lower robustness properties and increases the computational cost of the RMCD estimator. For these reasons, this is the default value used inside the robustification process of the FEDHC algorithm.

The case of $n < p$ and $n \ll p$ (a very high-dimensional case) in general can be treated in a similar way, by replacing the RMCD estimator with the high-dimensional MCD approach of [42].

4. Monte Carlo Simulation Studies

Extensive experiments were conducted on simulated data to investigate the quality of the estimation of FEDHC compared to PCHC and MMHC-2. MMHC-1 participated in the simulation studies only with categorical data and not with continuous data, because it is prohibitively expensive. The continuous data were generated by synthetic BNs that contained a various number of nodes, $p = (20, 30, 50)$, with an average of three and five neighbours (edges) for each variable. For each case 50 random BNs were generated with Gaussian data and various sample sizes. Categorical data were generated, utilising two famous (in the BN learning community) realistic BNs, and the sample sizes were left varying. The *R* packages *MXM* [43] and *bnlearn* were used for the BN generation, and the *R* packages *pchc* and *bnlearn* were utilised to apply the FEDHC, PCHC, MMHC-2 and the MMHC-1 algorithms, respectively. All simulations were implemented in a desktop computer with an Intel Core i5-9500 CPU at 3.00 GHz with 48 GB RAM and an SSD installed. The *R* codes for the simulation studies are available in the Supplementary Material.

The metrics of quality of the learned BNs were the structural Hamming distance (SHD) [13] of the estimated DAG from the true DAG (This is defined as the number of operations required to make the estimated graph equal to the true graph. Instead of the true DAG, the Markov equivalence graph of the true BN is used; that is, some edges have been un-oriented as their direction cannot be statistically decided. The transformation from the DAG to the Markov equivalence graph was carried out using Chickering's algorithm [44]), the computational cost and the number of tests performed during the skeleton identification phase and the total duration of the algorithm. PCHC and the MMHC-1 algorithms have been implemented in C++; henceforth the comparison of the execution times is not really fair at the programming language level. FEDHC and MMHC-2 have been implemented in R (skeleton identification phase) and C++ (scoring phase).

4.1. Synthetic BNs with Continuous Data

The procedure used to generate the data for X is summarised in the steps below. Let X be a variable in G, and let $Pa(X)$ be the parents of X in G.

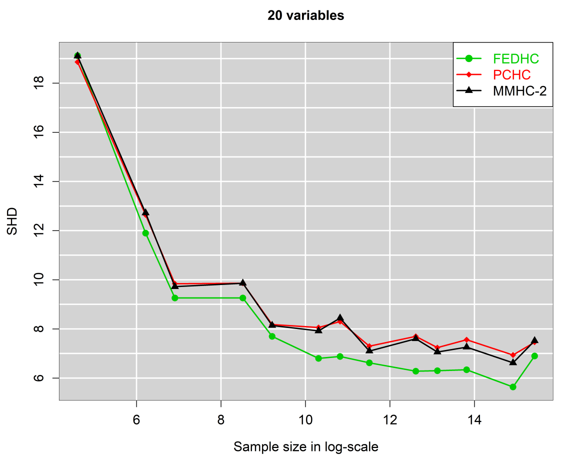
1. Sample the coefficients β of $f(Pa(X))$ uniformly at random from $[-1, -0.1] \cup [0.1, 1]$.
2. In case $Pa(X)$ is empty, X is sampled from the standard normal distribution. Otherwise, $X = f(Pa(X)) = \beta_0 + \sum_i \beta_i Pa_i(X) + \epsilon_X$, a linear function (in general, this can represent any (non-linear) function) depending on X, where ϵ_X is generated from a standard normal distribution.

The average number of connecting edges (neighbours) was set to 3 and 5. The higher the number of edges is, the denser the network is and the harder the inference becomes. The sample sizes considered were $n = (100, 500, 1000, 5000, 1 \times 10^4, 3 \times 10^4, 5 \times 10^4, 1 \times 10^5, 3 \times 10^5, 5 \times 10^5, 1 \times 10^6, 3 \times 10^6, 5 \times 10^6)$.

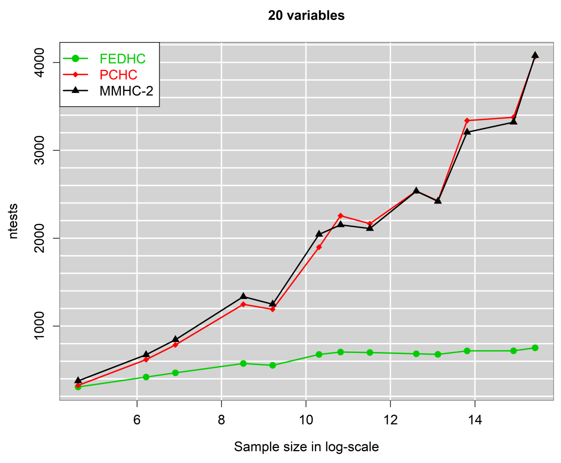
Figures 2–5 present the SHD and the number of CI tests performed of each algorithm for a range of sample sizes (in log-scale). With three neighbours on average per node, the differences in the SHD are noticeably rather small, yet FEDHC achieves lower numbers.

With five neighbours on average though, the differences are more significant and increasing with increasing sample sizes. As for the number of CI tests executed during the skeleton identification phase, FEDHC is the evident winner, as it executes up to six times less tests, regardless of the average neighbours. Moreover, the SHD of the FEDHC is lower than the SHD of its competitors for all cases, and the difference is magnified when the sample size is in the order of millions.

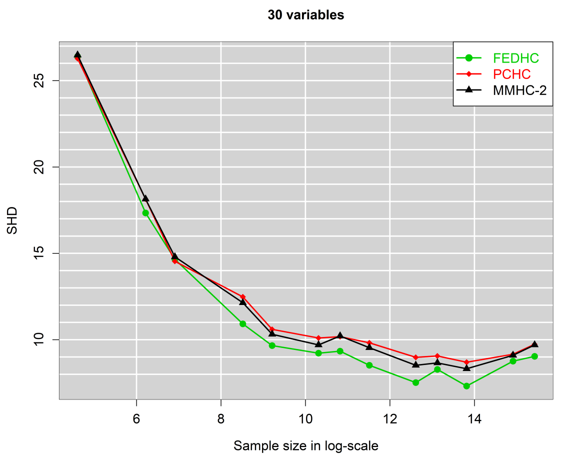
The common observation for all algorithms is that as the dimensionality increases, the SHD requires a greater sample size to achieve low levels. With 20 and 30 variables, the SHD may reach single-digit figures (with three neighbours on average), but with 50 and 100 variables, it never goes below 10. A similar conclusion is drawn when there are five neighbours on average.



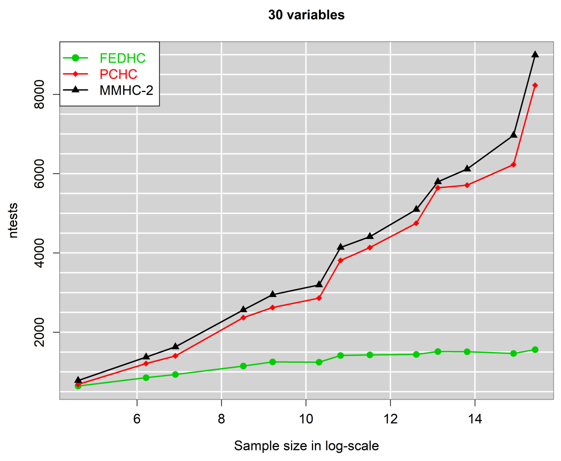
(**a**)



(**b**)

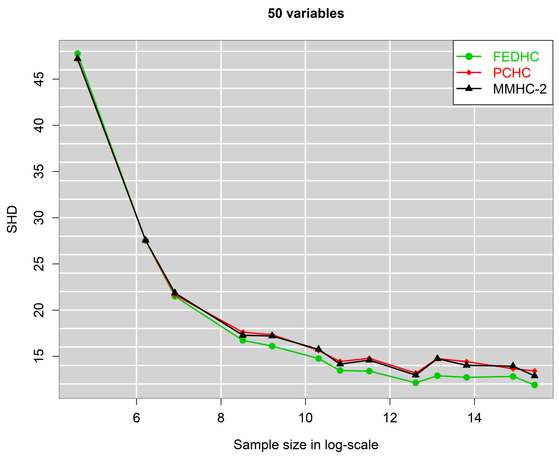


(**c**)

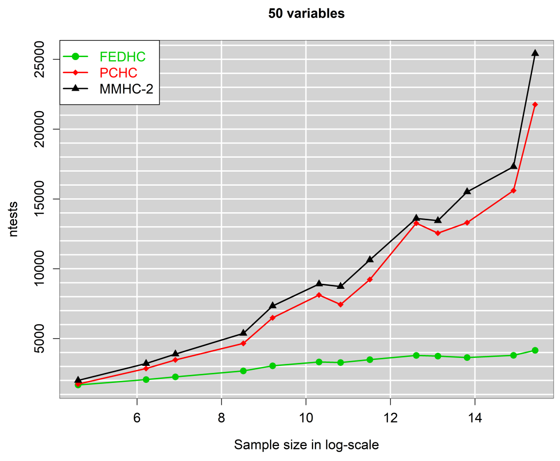


(**d**)

Figure 2. SHD and number of CI tests against log of sample size for 20 and 30 dimensions, with **3 neighbours** on average. (**a**) SHD vs. log of sample size. (**b**) Number of CI tests vs. log of sample size. (**c**) SHD vs. log of sample size. (**d**) Number of CI tests vs. log of sample size.

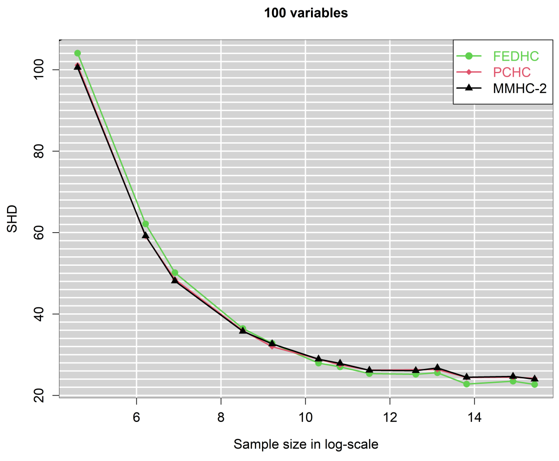


(**a**)

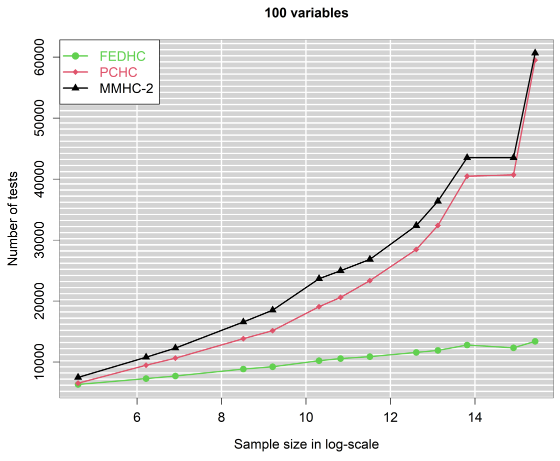


(**b**)

Figure 3. *Cont.*



(**c**)



(**d**)

Figure 3. SHD and number of CI tests against log of sample size for 50 and 100 dimensions, with **3 neighbours** on average. (**a**) SHD vs. log of sample size. (**b**) Number of CI tests vs. log of sample size. (**c**) SHD vs. log of sample size. (**d**) Number of CI tests vs. log of sample size.

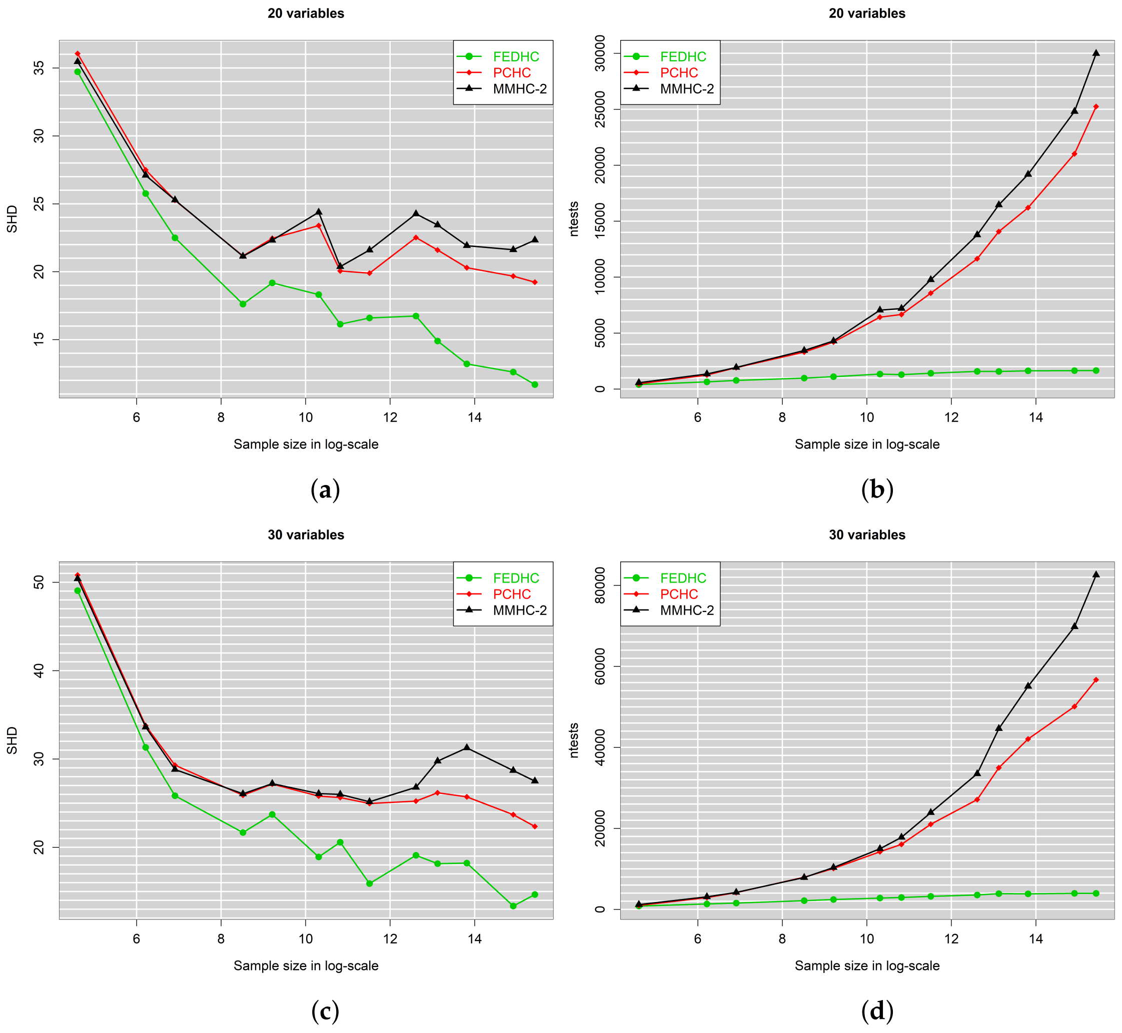


Figure 4. SHD and number of CI tests against log of sample size for 20 and 30 dimensions, with **5 neighbours** on average. (**a**) SHD vs. log of sample size. (**b**) Number of CI tests vs. log of sample size. (**c**) SHD vs. log of sample size. (**d**) Number of CI tests vs. log of sample size.

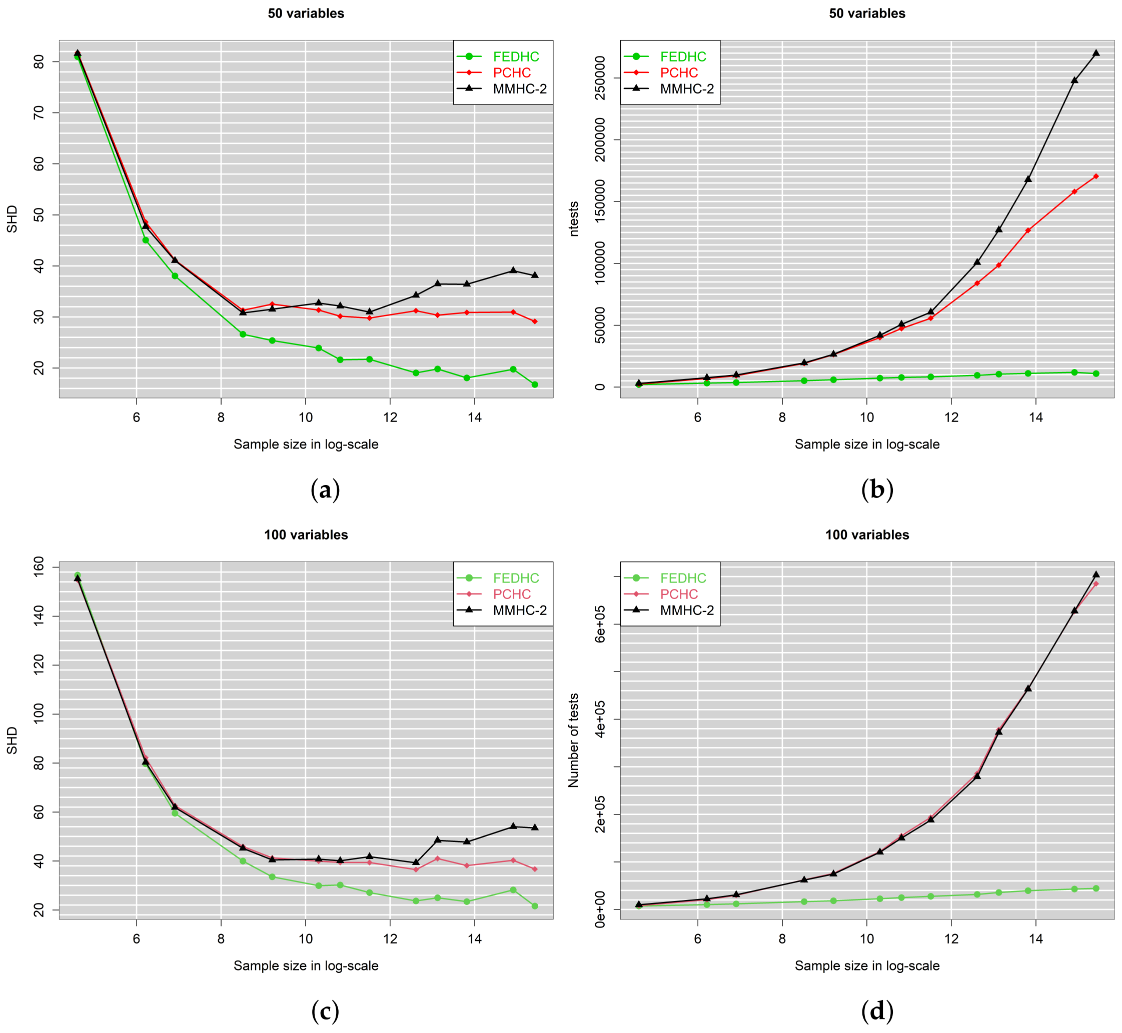
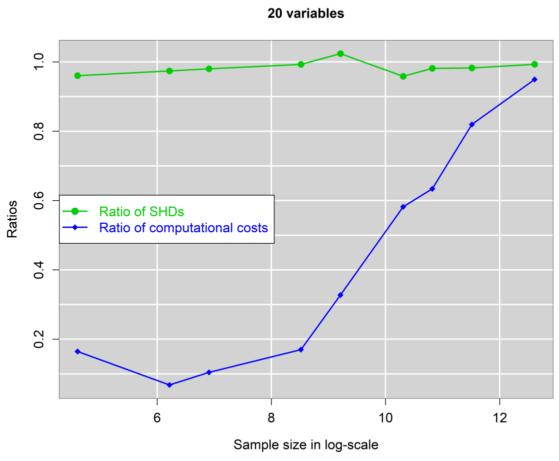


Figure 5. SHD and number of CI tests against log of sample size for various dimensions, with **5 neighbours** on average. (**a**) SHD vs. log of sample size. (**b**) Number of CI tests vs. log of sample size. (**c**) SHD vs. log of sample size. (**d**) Number of CI tests vs. log of sample size.

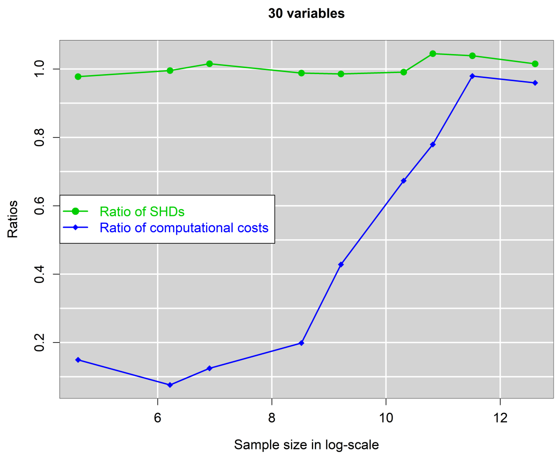
4.2. Robustified FEDHC for Continuous Data

An examination of the robustified FEDHC contains two axes of comparison; the outlier-free and the outliers present cases. At first, the performances of the raw and the robustified FEDHC in the outlier-free case are evaluated.

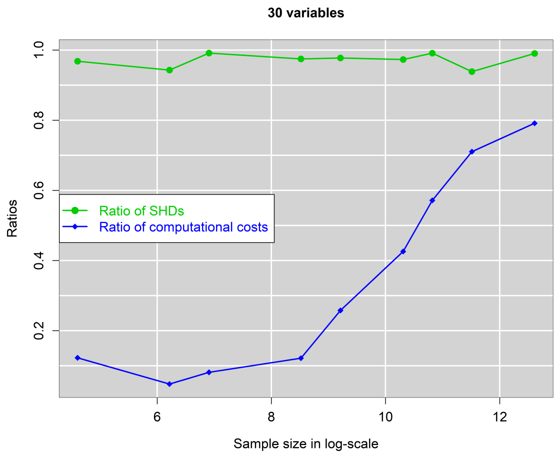
Figures 6 and 7 signify that there is no loss in the accuracy when using the robustified FEDHC over the raw FEDHC. Computationally speaking though, the raw FEDHC is significantly faster than the robustified FEDHC. For small sample sizes, the robustified FEDHC can be 10 times higher, while for large sample sizes, its cost can be double or only 5% higher than that of the raw FEDHC.



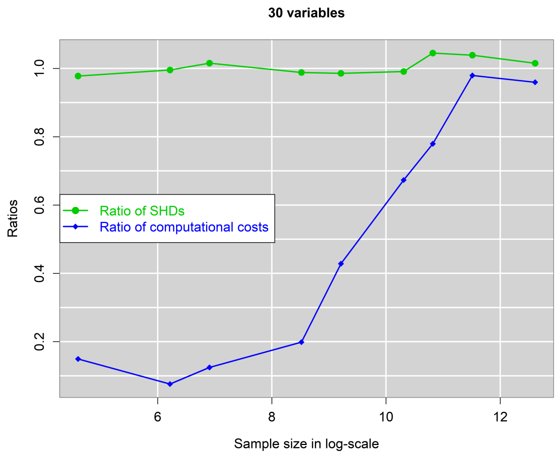
(**a**)



(**b**)

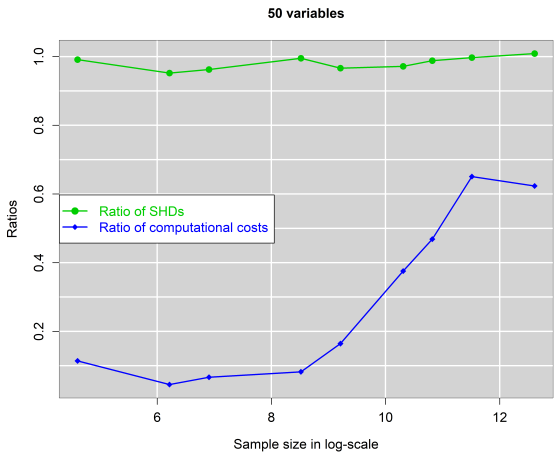


(**c**)

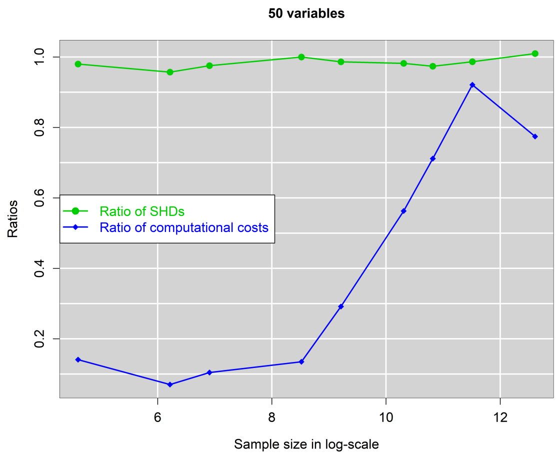


(**d**)

Figure 6. Ratios of SHD and computational cost against log of sample size for various dimensions, with **3 neighbours** and **5 neighbours** on average. The ratios depict the errors and computational cost of the raw FEDHC relative to the robustified FEDHC with NO outliers. (**a**,**c**) 3 neighbours. (**b**,**d**) 5 neighbours.

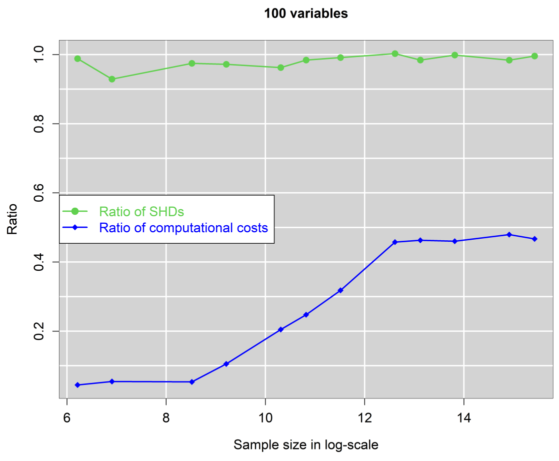


(**a**)

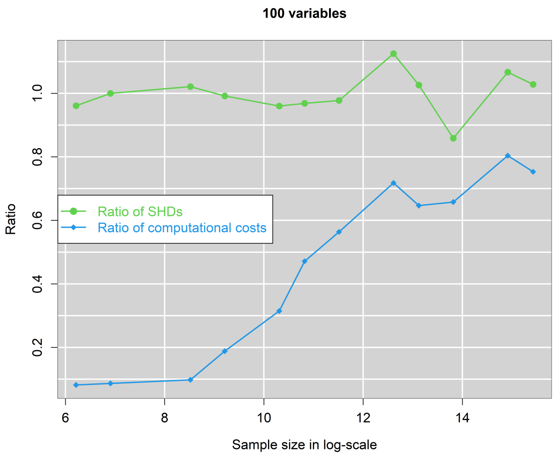


(**b**)

Figure 7. *Cont.*



(**c**)

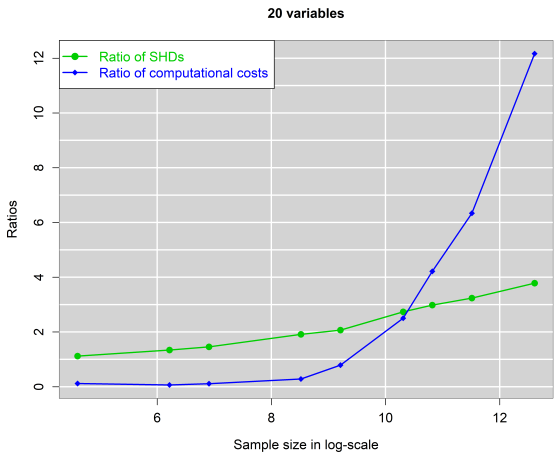


(**d**)

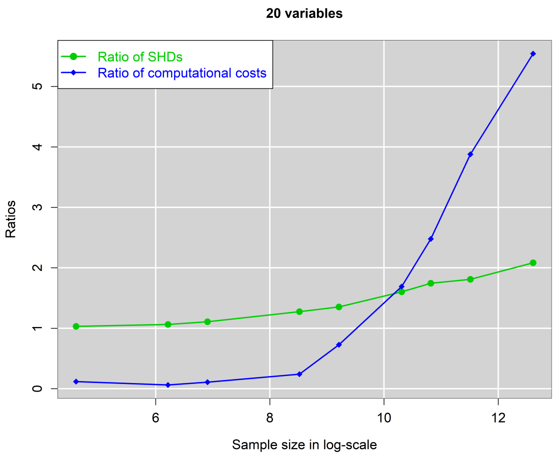
Figure 7. Ratios of SHD and computational cost against log of sample size for various dimensions, with **3 neighbours** and **5 neighbours** on average. The ratios depict the errors and computational cost of the raw FEDHC relative to the robustified FEDHC with NO outliers. (**a**,**c**) 3 neighbours. (**b**,**d**) 5 neighbours.

The performances of the raw FEDHC and of the robustified FEDHC in the presence of extreme outliers are evaluated next. The BN generation scheme is the one described in Section 4.1, with the exception of having added a 5% of outlying observations. The considered sample sizes are smaller, as although FEDHC is computationally efficient, it becomes really slow in the presence of outliers.

The results presented in Figures 8 and 9 evidently show the gain when using the robustified FEDHC over the raw FEDHC. The SHD of the raw FEDHC increases by as little as 100%, and up to 700% with 3 neighbours on average and 50 variables. The duration of the raw FEDHC increases substantially. (Similar patterns were observed for the duration of the skeleton learning phase and for the number of CI tests.) The raw FEDHC becomes up to more than 200 times slower than the robustified version, with hundreds of thousands of observations, and 50 variables. This is attributed to the HC phase of the raw FEDHC, which consumes a tremendous amount of time. The outliers produce noise that escalates the labour of the HC phase.

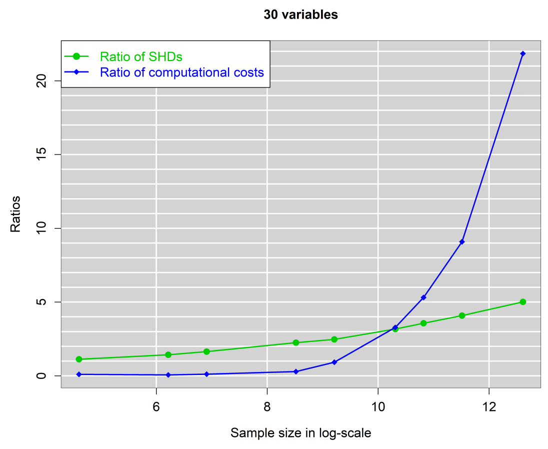


(**a**)

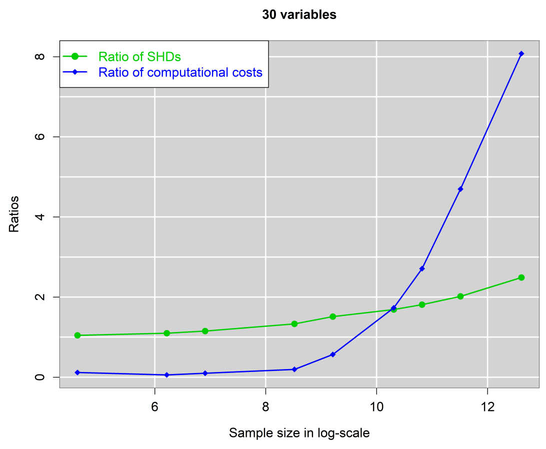


(**b**)

Figure 8. *Cont.*

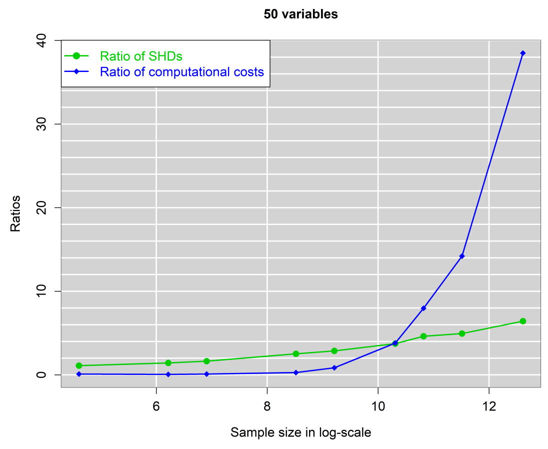


(**c**)

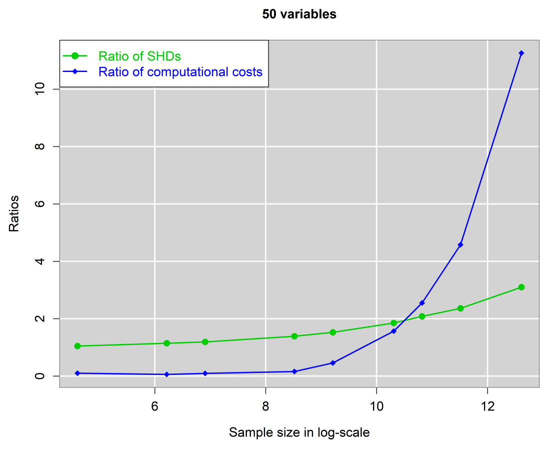


(**d**)

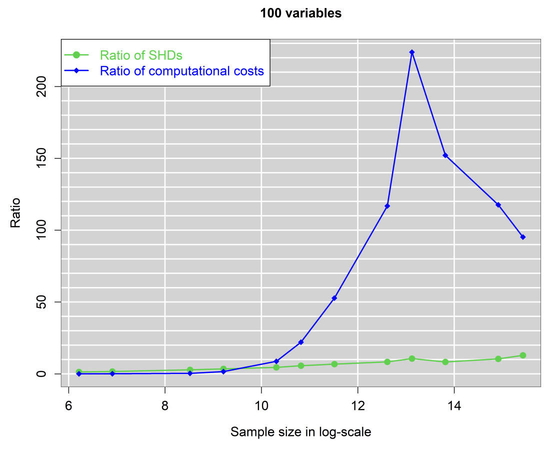
Figure 8. Ratios of SHD and computational cost against log of sample size for 20 and 30 dimensions with **3 neighbours** and **5 neighbours** on average. The ratios depict the errors and computational cost of the raw FEDHC relatively to the robustified FEDHC with 5% outliers. (**a**,**c**) 3 neighbours. (**b**,**d**) 5 neighbours.



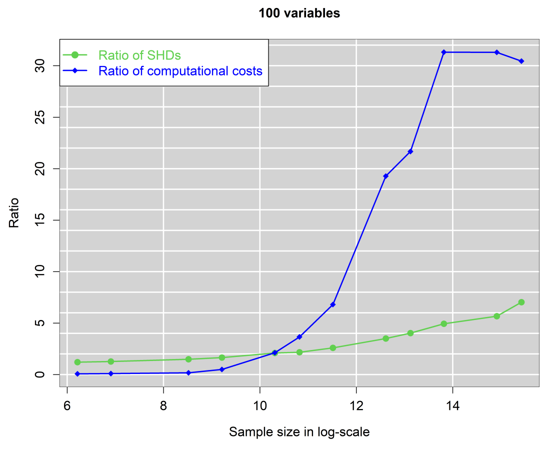
(**a**)



(**b**)



(**c**)



(**d**)

Figure 9. Ratios of SHD and computational cost against log of sample size for 50 and 100 dimensions with **3 neighbours** and **5 neighbours** on average. The ratios depict the errors and computational cost of the raw FEDHC relatively to the robustified FEDHC with 5% outliers. (**a**,**c**) 3 neighbours. (**b**,**d**) 5 neighbours.

4.3. Realistic BNs with Categorical Data

The $f(Pa(X))$ function utilised in the continuous data case relies on the β coefficients. The larger the magnitude of their values, the stronger the association between the edges becomes, and hence, it becomes easier to identify them. For BNs with categorical data, one could apply the same generation technique and then discretise the simulated data. To avoid biased or optimistic estimates favoring one or the other method, two real BNs with categorical data were utilised to simulate data. These are (a) the *Insurance* BN, used for evaluating car insurance risks [45], which consists of 27 variable (nodes) and 52 (directed) edges and (b) the *Alarm* BN, designed to provide an alarm message system for patient monitoring, and consists of 37 variables and 45 (directed) edges. The *R* package *bnlearn* contains a few thousand categorical instantiations from these BNs, but for the purpose of the simulation studies more instantiations were generated using the same package. The sample sizes considered were $n = (1 \times 10^4, 2 \times 10^4, 5 \times 10^4, 1 \times 10^5, 2 \times 10^5, 5 \times 10^5, 1 \times 10^6, 2 \times 10^6, 5 \times 10^6)$.

Figure 10 shows the SHD and the number of CI tests executed by each algorithm against the sample size. The MMHC-1 evidently has the poorest performance in both axes of comparison. Our implementation (MMHC-2) performs substantially better, but the overall winner is the PCHC. FEDHC, on the other hand, performs better than MMHC-1, yet is the second-best option.

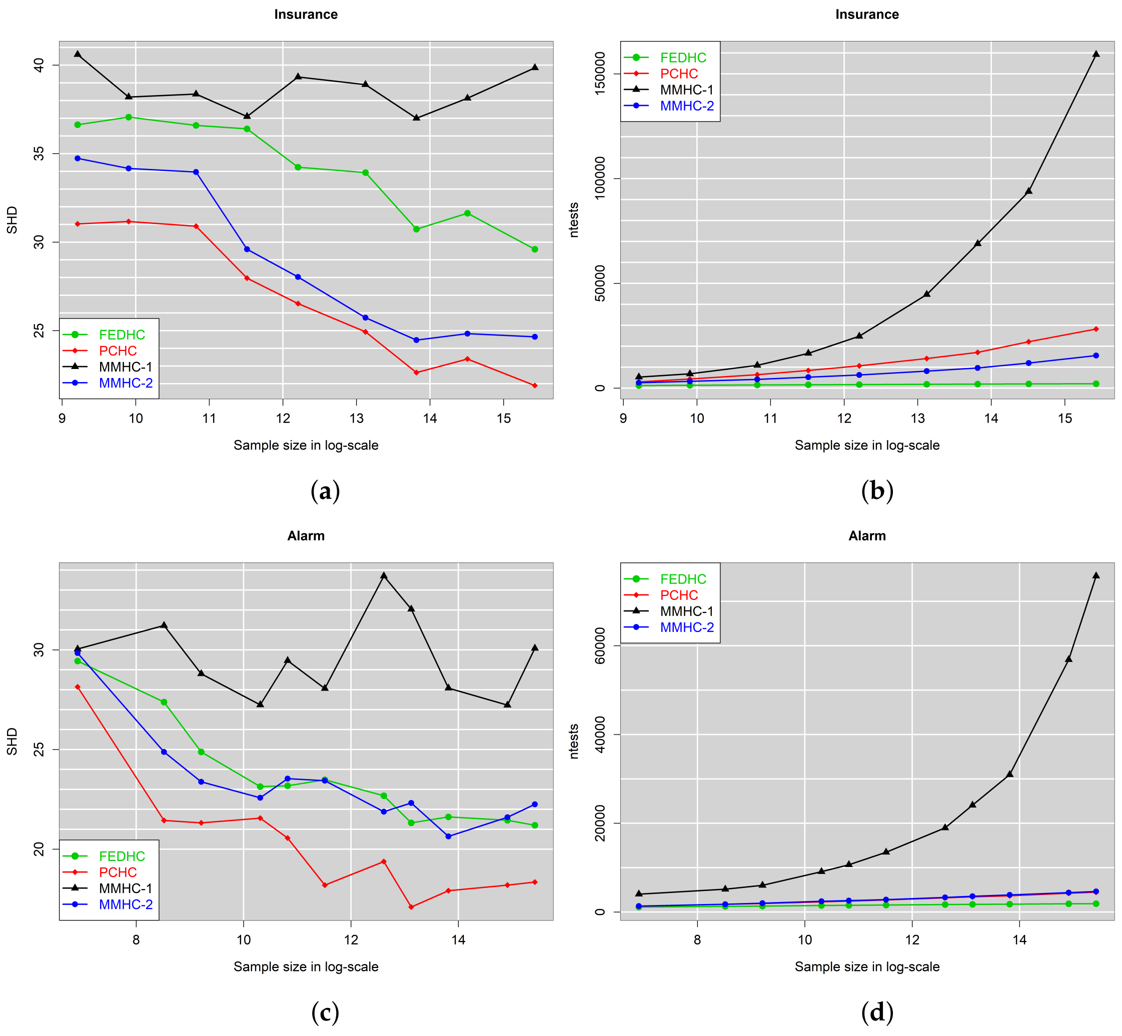
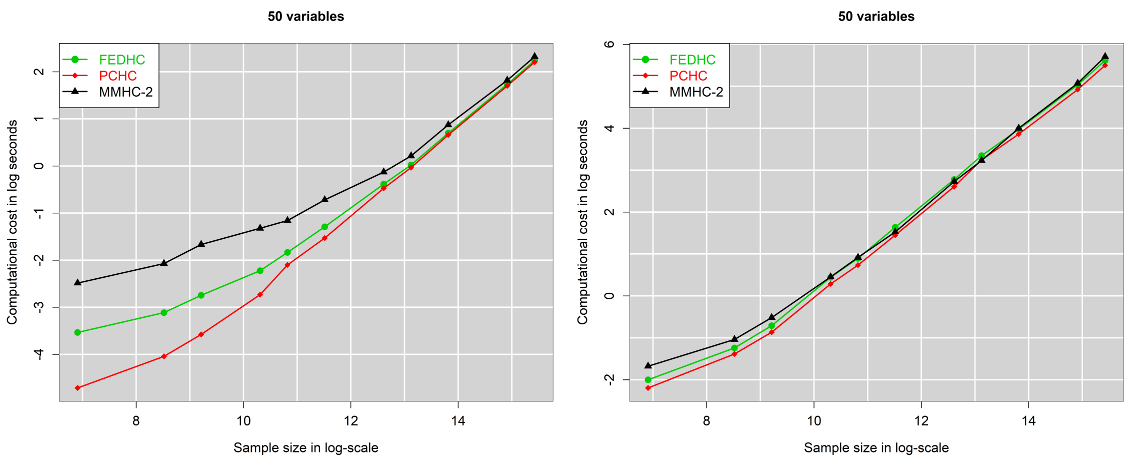


Figure 10. SHD and number of CI tests against log of sample size with **categorical** data. (**a**,**c**) SHD vs. log of sample size. (**b**,**d**) Number of CI tests vs. log of sample size.

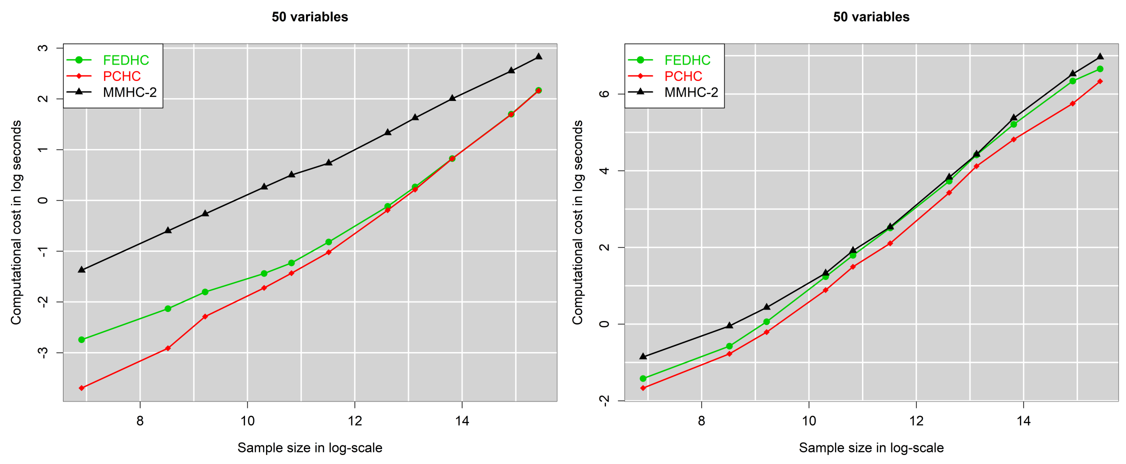
4.4. Scalability of FEDHC

The computational cost of each algorithm was also measured, appearing in Figure 11 as a function of the sample size. The empirical slopes of all lines in Figure 11 are nearly equal to 1, indicating that the scalability of FEDHC, PCHC, and MMHC-2 is linear in the sample size. Hence, the computational cost of all algorithms increases linearly with respect to the sample size. For any percentage-wise increase in the sample size, the time increases by the same percentage. Surprisingly enough, the computational cost of MMHC-1 was not evaluated for the categorical data case, because similarly to the continuous data case, it was too high to evaluate.

It is surprising though that the computational cost of FEDHC is similar to that of PCHC and MMHC-2. In fact the skeleton identification phase requires about the same amount of time, and it requires only 8 seconds with 5 million observations. The scoring phase is the most expensive phase of the algorithms, absorbing 73–99% of the total computation time. Regarding FEDHC and MMHC-2, since the initial phase of both has been implemented in *R*, this can be attributed to the fact that the calculations of the partial correlation in FEDHC are heavier than those in MMHC-2, because the conditioning set in the former can grow larger than the conditioning set in MMHC-2, which is always bounded by a pre-specified value *k*. Thus, MMHC-2 performs more but computationally lighter calculations than FEDHC.

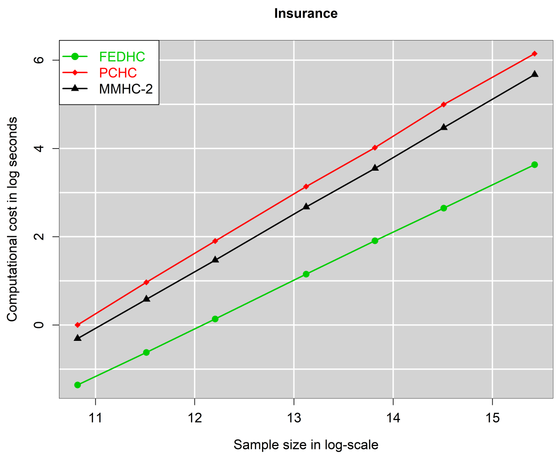


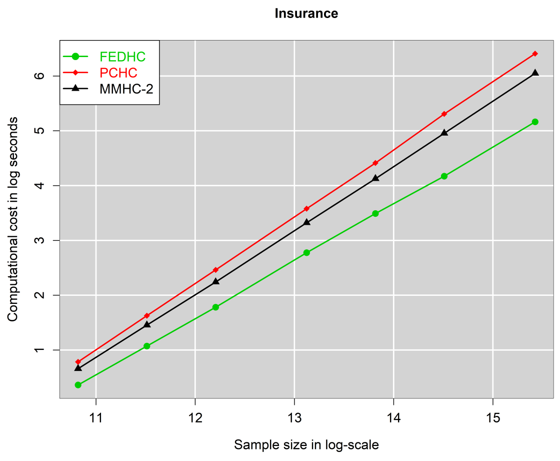
Continuous data with 3 neighbours on average.



Continuous data with 5 neighbours on average.

Figure 11. *Cont.*





Categorical data.

Figure 11. Scalability of the algorithms with respect to the sample size for some selected cases. The results for the other cases convey the same message and are thus not presented. The left column refers to the skeleton identification phase, whereas the right column refers to both phases.

5. Illustration of the Algorithms on Real Economics Data Using the *R* Package *pchc*

The *R* package *pchc* is used with two examples with the datasets used in [15] to illustrate the performance of FEDHC against its competitors, PCHC, MMHC-1 and MMHC-2. More details about the package can be found in Appendix C. The advantages of BNs have already been discussed in [15], and hence the examples focus on the comparison of FEDHC to PCHC, MMHC-1 and MMHC-2.

5.1. Expenditure Data

The first example concerns a dataset with continuous variables containing information on the monthly credit card expenditure of individuals. This is the **Expenditure** dataset [20] and is publicly accessible via the *R* package *AER* [46]. The dataset contains information about 1319 observations (10% of the original data set) on the following 12 variables. Whether the application for a credit card was accepted or not (**Card**), the number of major derogatory reports, (**Reports**), the age in years plus twelfths of a year (**Age**), the yearly income in \$10,000s (**Income**), the ratio of monthly credit card expenditure to yearly income (**Share**), the average monthly credit card expenditure (**Expenditure**), whether the person owns their home or they rent (**Owner**), whether the person is self employed or not (**Selfemp**), the number of dependents + 1 (**Dependents**), the number of months living at current address (**Months**), the number of major credit cards held (**Majorcards**) and the number of active credit accounts (**Active**).

The *R* package *AER* contains the data and must be loaded and processed for the algorithms to run.

```
> library(AER)  ## CreditCard are available
> library(bnlearn)  ## To run MMHC-1
> data(CreditCard)  ## load the data
> x <- CreditCard
> colnames(x) <- c( ‘‘Card’’, ‘‘Reports’’, ‘‘Age’’, ‘‘Income’’, ‘‘Share’’, ‘‘Expenditure’’,
+             ‘‘Owner’’, ‘‘Selfemp’’, ‘‘Dependents’’, ‘‘Months’’,  ‘‘Majorcards’’, ‘‘Active’’)
## Prepare the data
> for (i in 1:12)  x[, i] <- as.numeric(x[, i])
> x <- as.matrix(x)
> x[, c(1, 7, 8)] <- x[, c(1, 7, 8)] - 1
## Run all 4 algorithms
> a1 <- bnlearn::mmhc( as.data.frame(x), restrict.args =
+                       list(alpha = 0.05, test = ‘‘zf’’) )
> a2 <- pchc::mmhc(x, alpha = 0.05)
> a3 <- pchc::pchc(x, alpha = 0.05)
> a4 <- pchc::fedhc(x, alpha = 0.05)
```

In order to plot the fitted BNs of each algorithm, the following commands were used.

```
> pchc::bnplot(a1)
> pchc::bnplot(a2$dag)
> pchc::bnplot(a3$dag)
> pchc::bnplot(a4\$dag)
```

This example shows the practical usefulness of the BNs. Evidently, this small scale experiment shows that companies can customise their products according to the key factors that determine the consumers' behaviours. Instead of selecting one variable only, a researcher/practitioner can identify the relationships among all of the variables by estimating the causal mechanistic system that generated the data. The BN can further reveal information about the variables that are statistically significantly related.

According to FEDHC (Figure 12a), the age of the individual affects their income, the number of months they have been living at their current address, whether they own their home or not, and the ratio of their monthly credit card expenditure to their yearly income. The only variables associated with the number of major derogatory reports (Reports) are whether the consumer's application for a credit card was accepted or not (Card) and the number of active credit accounts (Active). In fact, these two variables are parents of Reports, as the arrows are directed towards it. A third advantage of BNs is that they provide a solution to the variable selection problem. The parents of the variable Majorcards (number of major credit cards held) are Card (whether the application for credit card was accepted or not) and Income (yearly income in $10,000), its only child is Active (number of active credit accounts) and its only spouse (parent of Active) is Owner (whether the consumer owns their home). The collection of those parents, children and spouses form the Majorcards' MB. That is, any other variable does not increase the information on the number of major credit cards held by the consumer. For any given variable, one can straightforwardly obtain (and visualise) its MB, which can be used for the construction of the appropriate linear regression model.

Figure 12 contains the BNs using both implementations of MMHC, the PCHC and the FEDHC algorithms fitted to the expenditure data, with the variables sorted in a topological order [44], a tree-like structure. The BIC of the BN learned by MMHC-1 and MMHC-2 are equal to $-32,171.75$ and $-32,171.22$, and for the PCHC and FEDHC, they are both equal to $-32,171.75$. This is an indication that all four algorithms produced BNs of nearly the same quality. On a closer examination of the graphs, one can detect some differences between the algorithms. For instance, **Age** is directly related to **Active**, according to PCHC and MMHC-2, but not according to FEDHC and MMHC-1. Further, all algorithms have identified the **Owner** as the parent of **Income**, and not vice-versa. This is related to the prior knowledge discussed in Section 3.4, and will be examined in the next categorical example dataset.

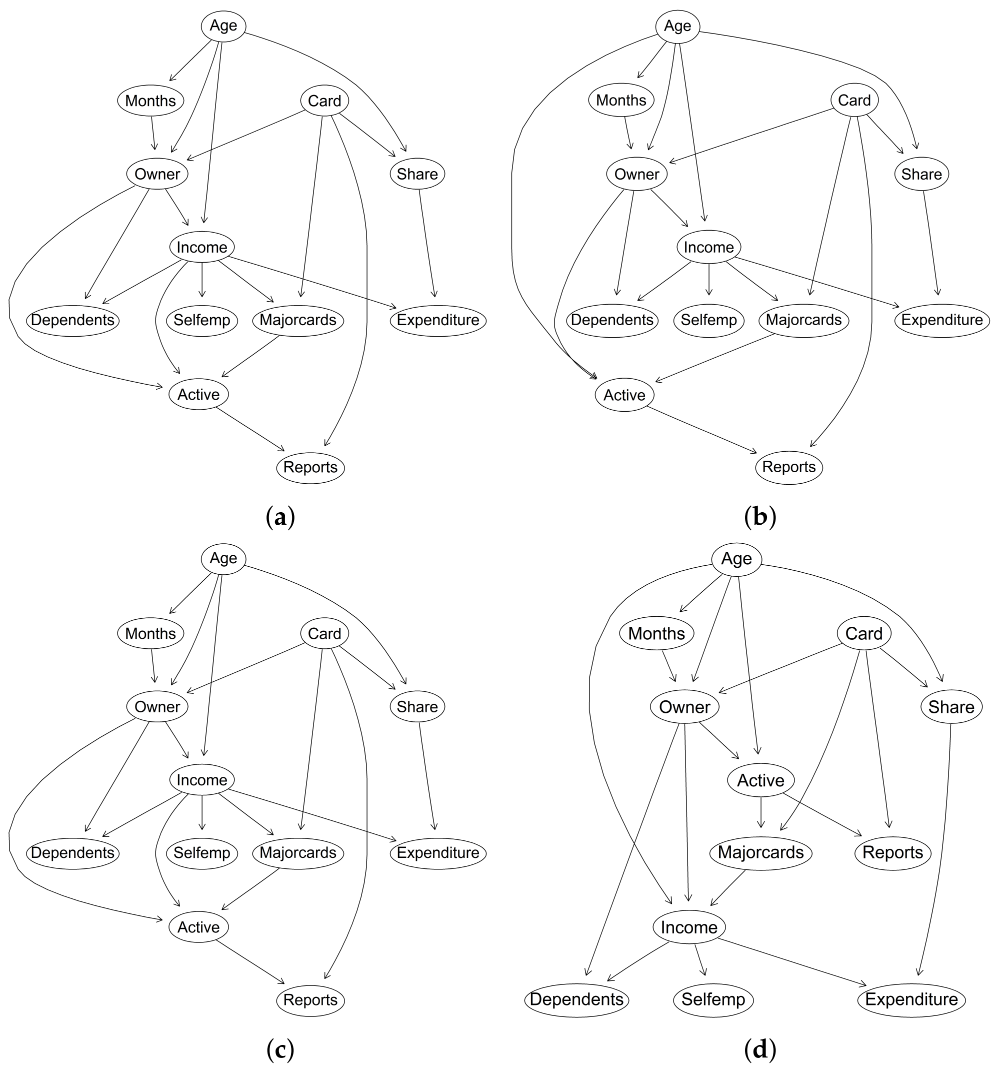


Figure 12. Expenditure data. Estimated BNs using (**a**) FEDHC, (**b**) PCHC, (**c**) MMHC-1 and (**d**) MMHC-2.

5.2. Income Data

The second example dataset contains categorical variables and originates from an example in the book "The Elements of Statistical Learning" [47], and is publicly available from the *R* package *arules* [48]. It consists of 6876 instances (obtained from the original dataset with 9409 instances, by removing observations with a missing annual income) and a mixture of 13 categorical and continuous demographic variables. The continuous variables were discretised, as suggested by the authors of the package. The continuous variables (age, education, income, years in bay area, number in household and number of children) were discretised based on their median values. **Income**: "$0–$40,000" or "$40,000+", **Sex**: "male" or "female", **Marriage**: "married", "cohabitation", "divorced", "widowed" or "single", **Age**: "14–34" or "35+", **Education**: "college graduate" or "no college graduate", **Occupation**, "professional/managerial", "sales", "laborer", "clerical/service", "homemaker", "student", "military", "retired" or "unemployed", **Bay** (number of years in bay area): "1–9" or "10+", **No of people** (number of of people living in the house): "1" or "2+", **Children**: "0" or "1+", **Rent**: "own", "rent" or "live with parents/family", **Type**: "house", "condominium", "apartment", "mobile home" or "other", **Ethnicity**: "American Indian", "Asian", "black", "east Indian", "hispanic", "white", "pacific islander" or "other" and **Language** (language spoken at home): "english", "spanish" or "other".

The dataset is at first accessed via the *R* package *arules* and is prepossessed, as suggested in *arules*.

```
> library(arules)
> data(IncomeESL)
## remove incomplete cases
> IncomeESL <- IncomeESL[complete.cases(IncomeESL), ]
## preparing the data set
> IncomeESL[[‘‘income‘‘]] <- factor((as.numeric(IncomeESL[[‘‘income’’]]) > 6) + 1,
+ levels = 1 : 2 , labels = c(‘‘0-40,000’’, ‘‘40,000+’’))
> IncomeESL[[‘‘age‘‘]] <- factor((as.numeric(IncomeESL[[‘‘age’’]]) > 3) + 1,
+ levels = 1 : 2 , labels = c(‘‘14-34’’, ‘‘35+’’))
> IncomeESL[[‘‘education‘‘]] <- factor((as.numeric(IncomeESL[[‘‘education’’]]) > 4) +
+ 1, levels = 1 : 2 , labels = c(‘‘no college graduate‘‘, ‘‘college graduate’’))
> IncomeESL[[‘‘years in bay area’’]] <- factor(
+ (as.numeric(IncomeESL[[‘‘years in bay area’’]]) > 4) + 1,
+ levels = 1 : 2 , labels = c(‘‘1-9’’, ‘‘10+’’))
> IncomeESL[[‘‘number in household’’]] <- factor(
+ (as.numeric(IncomeESL[[‘‘number in household’’]]) > 3) + 1,
+ levels = 1 : 2 , labels = c(‘‘1’’, ‘‘2+’’))
> IncomeESL[[‘‘number of children’’]] <- factor(
+ (as.numeric(IncomeESL[[‘‘number of children’’]]) > 1) + 0,
+ levels = 0 : 1 , labels = c(‘‘0’’, ‘‘1+’’))
```

Some more steps are required prior to running the BN algorithms.

```
> x <- IncomeESL
> x <- x[, -8]
> colnames(x) <- c( "Income", "Sex", "Marriage", "Age", "Education", "Occupation",
+     "Bay", "No of people", "Children", "Rent", "Type", "Ethnicity", "Language" )
> nam <- colnames(x)
```

The importance of prior knowledge incorporation discussed in Section 3.4 becomes evident in this example. Prior knowledge can be added in the **blacklist** argument denoting the forbidden directions (arrows).

```
> black <- matrix(nrow = 26, ncol = 2)
> black <- as.data.frame(black)
> for (i in 1:13)  black[i, ] <- c(nam[i], nam[2])
> for (i in 1:13)  black[13 + i, ] <- c(nam[i], nam[4])
> black <- black[-c(2, 17), ]
> black <- rbind( black, c(nam[9], nam[3]) )
> black <- rbind( black, c(nam[3], nam[6]) )
> black <- rbind( black, c(nam[9], nam[6]) )
> black <- rbind( black, c(nam[6], nam[5]) )
> black <- rbind( black, c(nam[3], nam[1]) )
> black <- rbind( black, c(nam[1], nam[5]) )
> black <- rbind( black, c(nam[10], nam[1]) )
> black <- rbind( black, c(nam[10], nam[5]) )
> black <- rbind( black, c(nam[10], nam[6]) )
> black <- rbind( black, c(nam[13], nam[12]) )
> colnames(black) <- c(‘‘from’’, ‘‘to’’)
```

Finally, the four BN algorithms are applied to the Income data.

```
> b1 <- bnlearn::mmhc( x, blacklist = black, restrict.args =
+                       list(alpha = 0.05, test = ‘‘mi’’) )
> b2 <- pchc::mmhc(x, method = ‘‘cat’’, alpha = 0.05, blacklist = black,
+ score = ‘‘bic’’)
```

```
> b3 <- pchc::pchc(x, method = ‘‘cat’’, alpha = 0.05, blacklist = black,
+ score = ‘‘bic’’)
> b4 <- pchc::fedhc(x, method = ‘‘cat’’, alpha = 0.05, blacklist = black,
+ score = ‘‘bic’’)
```

Figure 13 presents the fitted BNs of the MMHC-1, MMHC-2, PCHC and FEDHC algorithms. There are some distinct differences between the algorithms. For instance, PCHC is the only algorithm that has not identified **Education** as the parent of **Bay**. Additionally, the BN learned by MMHC-2 is the densest one (contains more arrows), whereas PCHC learned the BN with the fewest arrows.

This example further demonstrates the necessity of prior knowledge. BN learning algorithms fit a model to the data, ignoring the underlying truthfulness and ignoring the relevant economic theory. Economic theory can be used as prior knowledge to help mitigate the errors and lead to more truthful BNs. The exclusion of the blacklist argument (forbidden directions) would yield some irrational directions; for instance, that occupation or age might affect sex, or that marriage affects age, simply because these directions could increase the score. Finally, BNs are related to synthetic population generation, where the data are usually categorical. This task requires the specification of the joint distribution of the data, and BNs accomplish this [2]. Based on the Markov condition (1), the joint distribution can be written down explicitly, allowing for synthetic population generation in a sequential order. One commences by generating values for education and sex. Using these two variables, values for occupation are generated. These values, along with income and age, can be used to generate for the marital status, and so on.

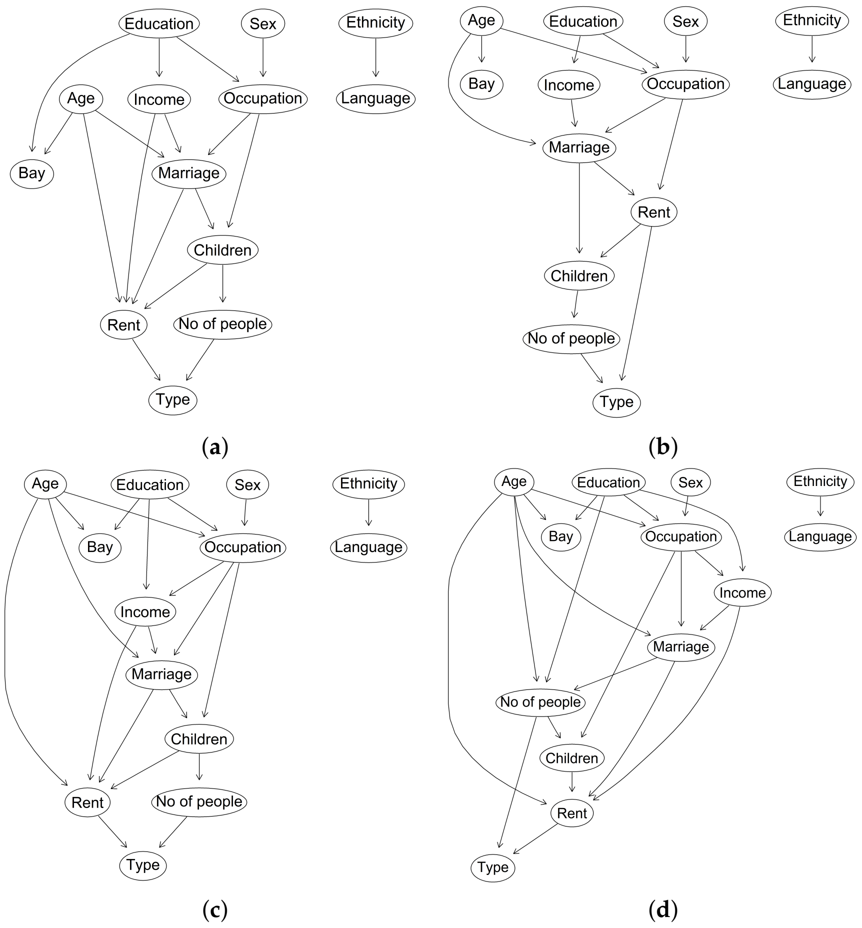


Figure 13. Income data. Estimated BNs using (**a**) FEDHC, (**b**) PCHC, (**c**) MMHC-1 and (**d**) MMHC-2.

6. Conclusions

This paper proposed to combine the first phase of the FBED variable selection algorithm with the HC scoring phase, leading to a new hybrid algorithm, termed FEDHC. Additionally, a new implementation of the MMHC algorithm was provided. Finally, the paper presented robustified (against outliers) versions of FEDHC, PCHC and MMHC. The robustified version of FEDHC was shown to be nearly 40 times faster than the raw version, and yielded BNs of higher quality, when outliers were present. Simulation studies manifested that in terms of computational efficiency, FEDHC is comparable to PHCHC, and along with MMHC-2, FEDHC was able to fit BNs to continuous data, with sample sizes in the order of hundreds of thousands in a few seconds and in the order of millions in a few minutes. It must be highlighted though that the skeleton identification phase of FEDHC and MMHC-1 have been implemented in *R* and not in *C++*. Additionally, FEDHC always executed significantly fewer CI tests than its competitors. Ultimately, in terms of accuracy, FEDHC outperformed is competitors with continuous data, and it was more accurate than or on par with MMHC-1 and MMHC-2 with categorical data, but less accurate than PCHC.

The rationale of MMHC and PCHC is to perform variable selection to each node, and then to apply a HC to the resulting network. In the same spirit, Ref. [49] used LASSO for variable selection with the scopus of constructing an un-directed network. The HC phase could be incorporated in the graphical LASSO to learn the underlying BN. Broadly speaking, the combination of a network learning phase with a scoring phase can yield hybrid algorithms. Other modern hybrid methods for BN learning include [50] on hybrid structure learning and sampling. They combine constraint-based pruning with MCMC inference schemes (also to improve the overall search space) and find a combination that is relatively efficient, with relatively good performance. The constraint-based part is interchangeable, and could connect well with MMHC, PCHC or FEDHC.

FEDHC is not the first algorithm that has outperformed MMHC. Recent algorithms include PCHC [15], the SP algorithm for Gaussian DAGs [51] and the NOTEARS [52]. The algorithms of [53,54] were also shown to outperform MMHC in the presence of latent confounders, not examined here. The advantage of the latter two is that they employ non-parametric tests such as the kernel CI test, thus allowing for non-linear relationships. BNs that detect non-linear relationships among the variables, such as the algorithms proposed by [53,54] is what this paper did not cover. Further, our comparative analysis was only with MMHC [13] and PCHC [15], due to their close relationship with FEDHC.

Future research includes a comparison of all algorithms in terms of more directions. For instance, (a) the effect of the Pearson and Spearman CI tests and the effect of X^2 and G^2 CI tests, (b) the effect of the outliers, (c) the effect of the scoring methods (Tabu search and HC), (d) the effect of the average neighbours (network density), and (e) the effect of the number of variables on the quality of the BN learned by either algorithm. These directions can be used to numerically evaluate the asymptotic properties of the BN learning algorithms with tens of millions of observations. Another interesting direction is the incorporation of fast non-linear CI tests, such as the distance correlation [55–58]. The distance correlation could be utilised during the skeleton identification of the FEDHC, mainly because it performs fewer CI tests than its competitors.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math10152604/s1, *R* codes for simulation studies.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Conditional Independence Tests

The type of CI tests executed during the skeleton identification phase depends upon the nature of the data, and they are used to test the following. Let X and Y be two random variables, and $\mathbf{Z}$ be a (possibly empty) set of random variables. Statistically speaking, X and Y are conditionally independent, given $\mathbf{Z}$ ($X \perp\!\!\!\perp Y|\mathbf{Z}$), if $P(X, Y|\mathbf{Z}) = P(X|\mathbf{Z}) \cdot P(Y|\mathbf{Z})$, and this holds for all values of X, Y and $\mathbf{Z}$. Equivalently, the conditional independence of X and Y, given $\mathbf{Z}$ implies $P(X|Y, \mathbf{Z}) = P(X|\mathbf{Z})$ and $P(Y|X, \mathbf{Z}) = P(Y|\mathbf{Z})$.

Appendix A.1. Pearson Correlation for Continuous Data

A frequently employed CI test for two continuous variables X and Y conditional on a set of variables $\mathbf{Z}$ is the partial correlation test [59] that assumes linear relationships among the variables. The test statistic for the partial Pearson correlation is given by:

$$T_p = \frac{1}{2}\left|\log\frac{1 + r_{X,Y|\mathbf{Z}}}{1 - r_{X,Y|\mathbf{Z}}}\right|\sqrt{n - |\mathbf{Z}| - 3}, \tag{A1}$$

where n is the sample size, $|\mathbf{Z}|$ denotes the number of conditioning variables and $r_{X,Y|\mathbf{z}}$ is the partial Pearson correlation (The partial correlation is efficiently computed using the correlation matrix of X, Y and $\mathbf{Z}$ [59]) of X and Y conditioning on $\mathbf{Z}$ (In the R package *Rfast*, its implementation of the PC algorithm compares T_p (A1) against a t distribution with $n - |\mathbf{Z}| - 3$ degrees of freedom, whereas the MMHC algorithm in the R package *bnlearn* compares T against the standard normal distribution. The differences are evident in small sample sizes, but become negligible when the sample sizes are in the order of a few tens). When $\mathbf{Z}$ is empty ($|\mathbf{Z}| = 0$), the partial correlation drops to the usual Pearson correlation coefficient.

Appendix A.2. Spearman Correlation for Continuous Data

The non-parametric alternative that is assumed to be more robust to outliers is the Spearman correlation coefficient. The Spearman correlation is equivalent to the Pearson correlation applied to the ranks of the variables. Its test statistic, however, is given by $T_s = T_p \times 1.029563$ [60,61].

Appendix A.3. G^2 Test of Independence for Categorical Data

The G^2 test of independence of two categorical variables X and Y, conditional on a set of variables $\mathbf{Z}$, is defined as [62]:

$$G^2 = 2\sum_l \sum_{i,j} O_{ij|l} \log \frac{O_{ij|l}}{E_{ij|l}}, \tag{A2}$$

where O_{ij} are the observed frequencies of the i-th and j-th values of X and Y, respectively, for the l-th value of $\mathbf{Z}$. The E_{ij} are their corresponding expected frequencies computed by $E_{ij} = \frac{O_{i+|l}O_{+j|l}}{O_{++|l}}$, where $O_{i+|l} = \sum_{j=1}^{n} O_{ij|l}$, $O_{+j|l} = \sum_{i=1}^{n} O_{ij|l}$ and $O_{++|l} = n_l$. Under the conditional independence assumption, the G^2 test statistic follows the χ^2 distribution with $(|X| - 1)(|X| - 1)(|\mathbf{Z}| - 1)$ degrees of freedom, where $|\mathbf{Z}|$ refers to the cardinality of $\mathbf{Z}$ and the total number of values of $\mathbf{Z}$.

Appendix A.3.1. X^2 Test of Independence for Categorical Data

Alternatively, one could use the Pearson X^2 test statistic $X^2 = \sum_l \sum_{i,j} \frac{\left(O_{ij|l} - E_{ij|l}\right)^2}{E_{ij|l}^2}$ that has the same properties as the G^2 test statistic (A2). The drawback of X^2 is that it cannot be computed when $E_{ij|l} = 0$. On the contrary, G^2 is computed in such cases, since $\lim_{x \to 0} x \log x = 0$. For either aforementioned test, when $|\mathbf{Z}|$ is the empty set, both tests examine the unconditional association between variables X and Y. (For a practical comparison between the two tests based on extensive simulation studies, see [63].)

Appendix A.3.2. Permutation-Based p-Values

The aforementioned test statistics produce asymptotic p-values. In the case of small sample sizes, computationally intensive methods such as permutations might be preferable. With continuous variables for instance, when testing for unconditional independence, the idea is to distort the pairs multiple times, and each time, calculate the relevant test statistic. For the conditional independence of X and Y conditional on $\mathbf{Z}$, the partial correlation is computed from the residuals of two linear regression models, $X \sim \mathbf{Z}$ and $Y \sim \mathbf{Z}$. In this instance, the pairs of the residual vectors are distorted multiple times. With categorical variables, this approach is more complicated, and care must be taken so as to retain the row and column totals of the resulting contingency tables. For either case, the p-value is then computed as the proportion of times that the permuted test statistics exceed the observed test statistic that is computed using the original data. Permutation-based techniques have shown to improve the quality of BNs [64] in small sample sized cases. On the contrary, the FEDHC algorithm aims at making inferences on datasets with large sample sizes, for which asymptotic statistical tests are valid and reliable enough to produce the correct decisions.

Appendix B. Computational Details of FEDHC

With continuous data, the correlation matrix is computed once and utilised throughout the skeleton identification phase. FEDHC returns the correlation matrix and the matrix of the p-values of all pairwise associations that are useful in a second run of the algorithm with a different significance level. This is a significant advantage when BNs have to fit to large scale datasets and the correlation matrix can be given as an input to FEDHC to further reduce FEDHC's computational cost.

The partial correlation coefficient is given by:

$$r_{X,Y|\mathbf{Z}} = \left\{ \begin{array}{ll} \frac{R_{X,Y} - R_{X,z} R_{Y,z}}{\sqrt{\left(1 - R^2_{X,Z}\right)^T \left(1 - R^2_{Y,z}\right)}} & \text{if } |\mathbf{Z}| = 1 \\ -\frac{\mathbf{A}_{1,2}}{\sqrt{\mathbf{A}_{1,1}\mathbf{A}_{2,2}}} & \text{if } |\mathbf{Z}| > 1 \end{array} \right\},$$

where $R_{X,Y}$ is the correlation between the variables X and Y; $R_{X,Z}$ and $R_{Y,Z}$ denote the correlations between X & Z and Y & Z. $\mathbf{A} = R^{-1}_{X,Y,\mathbf{Z}}$, with $\mathbf{A}$ denoting the sub-correlation matrix of variables $X, Y, \mathbf{Z}$ and $A_{i,j}$ symbolises the element in the i-row and j-th column of matrix A.

The CI tests executed during the initial phase compute the logarithm of the p-value, instead of the p-value itself, to avoid numerical overflows observed with a large test statistic that produces a p-value that is equal to 0. Additionally, the computational cost of FEDHC's first phase can be further reduced via parallel programming.

It is also possible to store the p-values of each CI test for future reference. When a different significance level must be used, this will further decrease the associated computational cost of the skeleton identification phase in a second run. However, as will be exposed in Section 4.4, the cost of this phase is very small (a few seconds), even for millions of observations. The largest portion of this phase's computational cost is attributed to the calculation of the correlation matrix, which can be passed into subsequent runs of the algorithm.

Finally, Ref. [15] disregarded the potential of applying the PC-orientation rules [22,24] prior to the scoring phase as a means of improving the performance of FEDHC and MMHC, and this is not pursued any further.

Appendix C. The *R* Package *pchc*

The package *pchc* was first launched in *R* in July 2020 and initially contained the PCHC algorithm. It now includes the FEDHC and MMHC-2 algorithms, functions for testing (un)conditional independence with continuous and categorical data, data generation, BN visualisation and utility functions. It imports the *R* packages *bnlearn* and *Rfast*, and the built-in package *stats*. *pchc* is distributed as part of the CRAN R package repository and is compatible with MacOS-X, Windows, Solaris and Linux operating systems. Once the package is installed and loaded,

```
> install.packages (‘‘pchc’’)
> library(pchc)
```

it is ready to use without internet connection. The signature of the function **fedhc**, along with a short explanation of its arguments, is displayed below.

```
> fedhc(x, method = ‘‘pearson", alpha = 0.05, robust = FALSE, ini.stat = NULL,
+  R = NULL, restart = 10, score = ‘‘bic-g", blacklist = NULL, whitelist = NULL)
```

- x: A numerical matrix with the variables. If you have a data.frame (i.e. categorical data) turn them into a matrix using. Note, that for the categorical case data, the numbers must start from 0. No missing data are allowed.
- method: If you have continuous data, this must be “pearson” (default value) or ””cat” if you have categorical data . With categorical data, one has to make sure that the minimum value of each variable is zero. The function *g2test* from the package *Rfast* and the relevant functions work that way.
- alpha: The significance level for assessing the *p*-values. The default value is 0.05.
- robust: If outliers are be removed prior to applying the FEDHC algorithm, this must be set to TRUE.
- ini.stat: If the initial test statistics (univariate associations) are available, they can passed to this argument.
- R: If the correlation matrix is available, pass it here.
- restart: An integer, the number of random restarts. The default value is 10.
- score: A character string, the label of the network score to be used in the algorithm. If none is specified, the default score is the Bayesian Information Criterion for continuous data sets. The available score for continuous variables are: “bic-g” (default value), “loglik-g”, “aic-g”, “bic-g” or ”bge”. The available score of categorical variables are: ”bde”, “loglik” or “bic”.
- blacklist: A data frame with two columns (optionally labeled “from” and “to”), containing a set of forbidden directions.
- whitelist: A data frame with two columns (optionally labeled “from” and “to”), containing a set of must-add directions.

The output of the **fedhc** function is a list including:

- ini: A list including the output of the *fedhc.skel* function.
- dag: A “bn” class output, a list including the outcome of the Hill-Climbing phase. See the package *bnlearn* for more details.
- scoring: The highest score value observed during the scoring phase.
- runtime: The duration of the algorithm.

References

1. Hoover, K.D. *Causality in Economics and Econometrics*; Palgrave: Macmillan UK, 2017; pp. 1–13.
2. Sun, L.; Erath, A. A Bayesian network approach for population synthesis. *Transp. Res. Part Emerg. Technol.* **2015**, *61*, 49–62. [CrossRef]
3. Kocabas, V.; Dragicevic, S. Agent-based model validation using Bayesian networks and vector spatial data. *Environ. Plan. Plan. Des.* **2009**, *36*, 787–801. [CrossRef]
4. Kocabas, V.; Dragicevic, S. Bayesian networks and agent-based modeling approach for urban land-use and population density change: A BNAS model. *J. Geogr. Syst.* **2013**, *15*, 403–426. [CrossRef]

5. Hosseini, S.; Barker, K. A Bayesian network model for resilience-based supplier selection. *Int. J. Prod. Econ.* **2016**, *180*, 68–87. [CrossRef]
6. Spiegler, R. Bayesian networks and boundedly rational expectations. *Q. J. Econ.* **2016**, *131*, 1243–1290. [CrossRef]
7. Xue, J.; Gui, D.; Lei, J.; Sun, H.; Zeng, F.; Feng, X. A hybrid Bayesian network approach for trade-offs between environmental flows and agricultural water using dynamic discretization. *Adv. Water Resour.* **2017**, *110*, 445–458. [CrossRef]
8. Mele, A. A structural model of dense network formation. *Econometrica* **2017**, *85*, 825–850. [CrossRef]
9. Chong, C; Kluppelberg, C. Contagion in financial systems: A Bayesian network approach. *SIAM J. Financ. Math.* **2018**, *9*, 28–53. [CrossRef]
10. Leong, C.K. Credit risk scoring with Bayesian network models. *Comput. Econ.* **2016**, *47*, 423–446. [CrossRef]
11. Sheehan, B.; Murphy, F.; Ryan, C.; Mullins, M.; Liu, H.Y. Semi-autonomous vehicle motor insurance: A Bayesian Network risk transfer approach. *Transp. Res. Part Emerg. Technol.* **2017**, *82*, 124–137. [CrossRef]
12. Cugnata, F.; Kenett, R.; Salini, S. Bayesian network applications to customer surveys and InfoQ. *Procedia Econ. Financ.* **2014**, *17*, 3–9. [CrossRef]
13. Tsamardinos, I.; Brown, L.E.; Aliferis, C.F. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.* **2006**, *65*, 31–78. [CrossRef]
14. Scutari, M. Learning Bayesian Networks with the bnlearn R Package. *J. Stat. Softw.* **2010**, *35*, 1–22. [CrossRef]
15. Tsagris, M. A new scalable Bayesian network learning algorithm with application to economics. *Comput. Econ.* **2021**, *57*, 341–367. [CrossRef]
16. Tsagris, M. pchc: Bayesian Network Learning with the PCHC and Related Algorithms. R package version 0.5. 2021. Available online: https://cran.r-project.org/web/packages/pchc/index.html (accessed on 22 May 2022).
17. Borboudakis, G; Tsamardinos, I. Forward-Backward selection with Early Dropping. *J. Mach. Learn. Res.* **2019**, *20*, 276–314.
18. Kalisch, M.; Bühlmann, P. Robustification of the PC-algorithm for directed acyclic graphs. *J. Comput. Graph. Stat.* **2008**, *17*, 773–789. [CrossRef]
19. Cheng, Y.; Diakonikolas, I.; Kane, D.; Stewart, A. Robust learning of fixed-structure Bayesian networks. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 10283–10295.
20. Greene, W.H. *Econometric Analysis*; Pearson Education India: Noida, India, 2003.
21. Pearl, J. *Probabilistic Reasoning in iNtelligent Systems: Networks of Plausible Reasoning*; Morgan Kaufmann Publishers: Los Altos, CA, USA, 1988.
22. Spirtes, P.; Glymour, C.N.; Scheines, R. *Causation, Prediction, and Search*; MIT Press: Cambridge, MA, USA, 2000.
23. Verma, T.; Pearl, J. Equivalence and synthesis of causal models. In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, USA, 27–29 July 1991; pp. 220–227.
24. Spirtes, P.; Glymour, C. An algorithm for fast recovery of sparse causal graphs. *Soc. Sci. Comput. Rev.* **1991**, *9*, 62–72. [CrossRef]
25. Chickering, D.M. Optimal structure identification with greedy search. *J. Mach. Learn. Res.* **2002,** *3*, 507–554.
26. Cooper, F.G.; Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.* **1992**, *9*, 309–347. [CrossRef]
27. Heckerman, D.; Geiger, D.; Chickering, D.M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **1995**, *20*, 197–243. [CrossRef]
28. Tsagris, M. Bayesian network learning with the PC algorithm: An improved and correct variation. *Appl. Artif. Intell.* **2019**, *33*, 101–123. [CrossRef]
29. Draper; R, N.; Smith, H. *Applied Regression Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 1998.
30. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* **1978**, *6*, 461–464. [CrossRef]
31. Geiger, D; Heckerman, D. Learning Gaussian networks. In Proceedings of the 10th International Conference on Uncertainty in Artificial Intelligence, Seattle, WA, USA, 29–31 July 1994; pp. 235–243.
32. Buntine, W. Theory refinement on Bayesian networks. In *Uncertainty Proceedings*; Elsevier: Amsterdam, The Netherlands, 1991; pp. 52–60.
33. Bouckaert, R.R. *Bayesian Belief Networks: From Construction to Inference*. Ph.D. Thesis, University of Utrecht, Utrecht, The Netherlands, 1995.
34. Lam, W.; Bacchus, F. Learning Bayesian belief networks: An approach based on the MDL principle. *Comput. Intell.* **1994**, *10*, 269–293. [CrossRef]
35. Suzuki, J. A construction of Bayesian networks from databases based on an MDL principle. In *Uncertainty in Artificial Intelligence*; Morgan Kaufmann: Burlington, MA, USA, 1993; pp. 266–273.
36. Rousseeuw, P.J. Multivariate estimation with high breakdown point. *Math. Stat. Appl.* **1985**, *8*, 283–297.
37. Rousseeuw, P.J.; Driessen, K.V. A fast algorithm for the minimum covariance determinant estimator. *Technometrics* **1999**, *41*, 212–223. [CrossRef]
38. Raymaekers, J.; Rousseeuw, P.J. Fast robust correlation for high-dimensional data. *Technometrics* **2021**, *63*, 184–198. [CrossRef]
39. Cerchiello, P; Giudici, P. Big data analysis for financial risk management. *J. Big Data* **2016**, *3*, 18. [CrossRef]
40. Cerioli, A. Multivariate outlier detection with high-breakdown estimators. *J. Am. Stat. Assoc.* **2010**, *105*, 147–156. [CrossRef]
41. Hubert, M.; Debruyne, M. Minimum covariance determinant. *Comput. Stat.* **2010**, *2*, 36–43. [CrossRef]
42. Ro, K.; Zou, C.; Wang, Z.; Yin, G. Outlier detection for high-dimensional data. *Biometrika* **2015**, *102*, 589–599. [CrossRef]

43. Tsagris, M.; Tsamardinos, I. Feature selection with the R package MXM. *F1000Research* **2019**, *7*, 1505. [CrossRef] [PubMed]
44. Chickering, D.M. A transformational characterization of equivalent Bayesian network structures. In Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–20 August 1995; pp. 87–98.
45. Beinlich, I.A.; Suermondt, H.J.; Chavez, R.M.; Cooper, G.F. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *AIME 89*; Springer: Berlin/Heidelberg, Germany, 1989; pp. 247–256.
46. Kleiber, C.; Zeileis, A. *Applied Econometrics with R*; Springer: New York, NY, USA, 2008.
47. Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Springer: New York, NY, USA, 2001.
48. Hahsler, M.S.; Chelluboina; Hornik, K.; Buchta, C. The arules R-Package Ecosystem: Analyzing Interesting Patterns from Large Transaction Datasets. *J. Mach. Learn. Res.* **2011**, *12*, 1977–1981.
49. Meinshausen, N.; Bühlmann, P. High-dimensional graphs and variable selection with the Lasso. *Ann. Stat.* **2006**, *34*, 1436–1462. [CrossRef]
50. Kuipers, J.; Suter, P.; Moffa, G. Efficient sampling and structure learning of Bayesian networks. *arXiv* **2020**, arXiv:1803.07859.
51. Raskutti, G.; Uhler; C Learning directed acyclic graph models based on sparsest permutations. *Stat* **2018**, *7*, e183. [CrossRef]
52. Zheng, X.; Aragam, B.; Ravikumar, P.K.; Xing, E.P. Dags with no tears: Continuous optimization for structure learning. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 9492–9503.
53. Zhang; Peters, K.; Janzing, J.; D; Schölkopf, B. Kernel-based conditional independence test and application in causal discovery. In Proceedings of the 27th conference on Uncertainty in Artificial Intelligence, Quebec City, QC, Canada, 270030 July 2012; pp. 804–813.
54. Chalupka, K.; Perona, P.; Eberhardt, F. Fast conditional independence test for vector variables with large sample sizes. *arXiv* **2018**, arXiv:1804.02747.
55. Huo, X.; Székely, G.J. Fast computing for distance covariance. *Technometrics* **2016**, *58*, 435–447. [CrossRef]
56. Shen, C.; Panda, S.; Vogelstein, J.T. The chi-square test of distance correlation. *J. Comput. Graph. Stat.* **2022**, *31*, 254–262. [CrossRef]
57. Székely, G.J.; Rizzo, M.L. Partial distance correlation with methods for dissimilarities. *Ann. Stat.* **2014**, *42*, 2382–2412. [CrossRef]
58. Székely, G.J.; Rizzo, M.L.; Bakirov, N.K. Measuring and testing dependence by correlation of distances. *Ann. Stat.* **2007**, *35*, 2769–2794. [CrossRef]
59. Baba, K.; Shibata, R.; Sibuya, M. Partial correlation and conditional correlation as measures of conditional independence. *Aust. N. Z. J. Stat.* **2004**, *46*, 657–664. [CrossRef]
60. Fieller, E.C.; Hartley, H.O.; Pearson, E.S. Tests for rank correlation coefficients. I. *Biometrika* **1957**, *44*, 470–481. [CrossRef]
61. Fieller, C.E.; Pearson, E.S. Tests for rank correlation coefficients: II. *Biometrika* **1961**, *48*, 29–40.
62. Agresti, A. *Categorical Data Analysis*, 2nd ed.; Wiley Series in Probability and Statistics: Hoboken, NJ, USA, 2002.
63. Alenazi, A. A Monte Carlo comparison of categorical tests of independence. *arXiv* **2020**, arXiv:2004.00973.
64. Tsamardinos, I.; Borboudakis, G. Permutation testing improves Bayesian network learning. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Barcelona, Spain, 20–24 September 2010; pp. 322–337.