WAGENINGEN
UNIVERSITY & RESEARCH

Biometris
Quantitative methods brought to life

Performance of causal learning algorithms under many variables

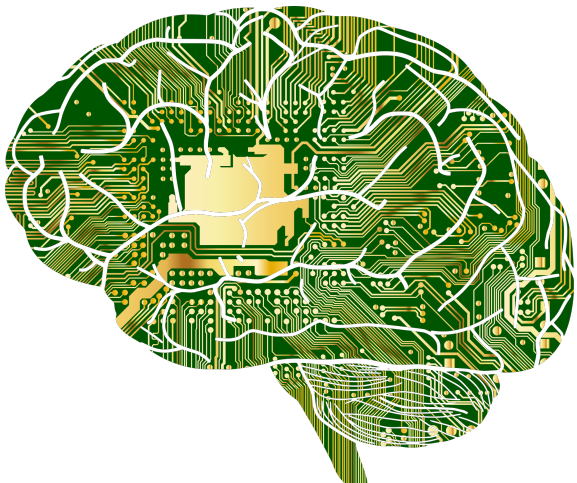
An explorative simulation study on high-dimensional data

Moos van der Veen

Master thesis
Wageningen University & Research
Wageningen, January 2026

Supervisor
Carel Peeters
Mathematical & Statistical Methods group (Biometris)
Wageningen University & Research

Supervisor
Iris van den Boomgaard
Mathematical & Statistical Methods group (Biometris)
Nutrition, Metabolism & Genomics
Wageningen University & Research



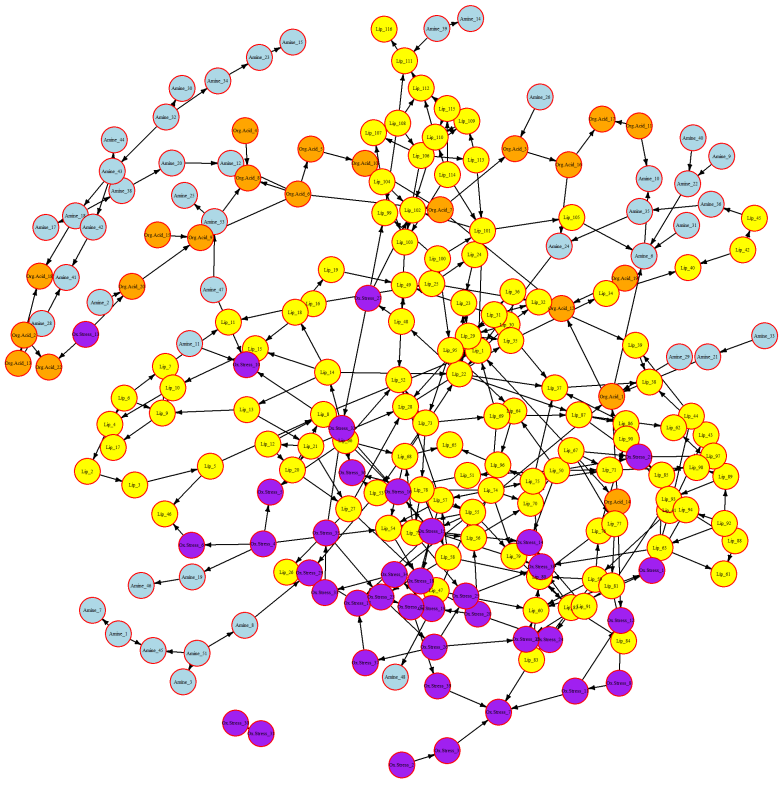
Foreword

This thesis was written as part of my master's thesis project at the Biometris group of Wageningen University & Research. The aim of my thesis was to evaluate the performance of structure learning algorithms in various high-dimensional settings and to provide practical guidance for their application in biomedical research.

I developed the full R-based pipeline for this work, including the simulation of datasets from distinct ground-truth DAGs (generated using bnlearn), benchmarking multiple structure learning algorithms, and constructing and evaluating different types of prior knowledge. I also conducted an illustration study to show how the results can be used as practical guidance and how learned graphs can be made interpretable for hypothesis generation (see graph below).

This thesis used simulated data and confidential metabolomics data from the Amsterdam Dementia Cohort.

I am very grateful for the guidance and expertise of my supervisors, Prof. Dr. Carel Peeters and Iris van den Boomgaard. They were always available for weekly meetings and provided insightful feedback. Their guidance helped me stay on track, especially when I sometimes wanted to explore too many different directions. I am grateful for the things I learned from them and for this thesis topic that challenged me a lot.



Abstract

There is a rapid surge in observational biomedical data with many variables on relatively few samples. This creates a need for transparent methods that can move beyond correlation and help generate causal hypotheses. This thesis benchmarks structure learning algorithms in various settings for such high-dimensional observational biomedical data.

A simulation study was conducted in R using continuous Gaussian data generated from structural equation models on nine ground-truth directed acyclic graphs (DAGs): six DAGs were simulated in sparse and dense settings and three were taken from the bnlearn repository. The datasets were generated with different sample sizes to create low- to high-dimensional settings. Seven structure learning algorithms (GS, Iamb.fdr, PC.stable, HC, Tabu, MMHC and Rsmax2) were evaluated under four different prior knowledge settings: no prior knowledge, a blacklist, a whitelist, and a combination of a blacklist and a whitelist. Performance was evaluated using F1-skeleton, F1-arrow, and structural Hamming distance (SHD).

The algorithms' performance was driven by the interplay of sample size N, graph density/complexity, and the absolute graph size (edges E and variables P proportionally). Notably, the common $P > N$ definition of high-dimensionality was not sufficient to explain performance differences, because high-dimensional DAG learning also depends on graph density (edges relative to variables), which determines how complex the learning task is. Denser graphs require the algorithms to consider more relationships locally at each learning step, making learning harder, especially with limited data. In dense settings, score-based algorithms performed best, whereas in sparse and repository settings hybrid algorithms along with GS and PC.stable performed best. Prior knowledge substantially improved performance, but its effectiveness depends on the type of prior knowledge and setting. A benchmark blacklist yielded the largest improvements in performance, whereas estimated blacklists from graphical models showed inconsistent to small improvements and remained far from the benchmark. Whitelists show consistent improvements across algorithms. Combining both a benchmark blacklist and whitelist improves performance further, but provides no synergistic effects. These findings suggest that algorithm selection should be based on sample size, expected graph density, and type of prior knowledge.

An illustration study with executable R code on Alzheimer's disease metabolomics data demonstrated how the results from this study can guide algorithm and prior knowledge choices in a real-world biomedical situation. The results of this illustration are interpretable for exploratory causal analysis.

Contents

1. Introduction

1.1. **Background.** Technical developments have led to a rapid surge in the volume of observational data. Vast amounts of observational data are collected more easily, with fewer resources and at a faster pace [7]. In particular, observational biomedical data grow fast; high-throughput technologies such as single-cell RNA sequencing, mass cytometry, and other omics approaches can now generate datasets with hundreds and even thousands of variables per sample. The volume of such biomedical data being generated is increasing at an exponential rate, doubling in well under two years. This growth even outpaces Moore's law, which describes the doubling of computing capacity approximately every two years [8].

Seeking to understand the cause behind a particular phenomenon, why something happens, is a key part of biomedical research. Understanding causality already has its roots with thinkers like Aristotle around 350 B.C. [2]. Hume in 1748 defined a cause as: "an object, followed by another, and where all the objects, similar to the first, are followed by objects similar to the second. Or in other words, where, if the first object had not been, the second never had existed." [1]. Yet, the difficulty of distinguishing mere correlation from true causation is widely known, as illustrated by the infamous example of the erroneous link between ice cream sales and shark attacks [3]. The gold standard for establishing causal effects is the randomized controlled trial [4]. However, due to ethical and practical constraints, this is not possible or feasible for many biomedical research questions, and thus observational studies are preferred [5]. Causal inference can be made from observational studies, like in the case of smoking leading to lung cancer. Yet, the use of observational studies for making causal inference is often criticized as holding little to no statistical value, and for its inferential mistakes. For instance, causal inference can be falsely made, as in the case where it was believed that postmenopausal hormone therapy lowered the risk of heart disease. In many cases it is too hard to determine causality, because even with large datasets and sophisticated models, confounding variables and assumptions about the data mean that we cannot definitively know if an observed relationship is causal [4, 5].

With the rapid surge in biomedical observational data, the need for reliable causal inference has become increasingly vital [6]. High-throughput biomedical datasets are commonly of a high-dimensional nature, meaning there are more predictors (P) present, relative to the number of observations (N). This is commonly denoted as $P > N$. High-dimensional data might at first appear as a promising development. For instance, knowing almost all the variables that may lead to a disease could seem like a good way to determine its origin. However, distinguishing the signal from the noise in high-dimensional data becomes complicated. It comes with a problem known as the "curse of dimensionality". As dimensions increase, the distance between data points becomes less informative. As the volume of the dimensional space grows exponentially, distances concentrate around similar values, making it difficult to distinguish data points. This makes it harder to find truly meaningful relations, as it increases computation and creates a higher risk of overfitting [9].

Traditional statistical and machine learning methods are insufficient when it comes to learning causal structure in observational data. This is because observational data may introduce spurious correlations. These are observed relationships between variables that are driven by noise in the data rather than true effects [10]. To address this, structure learning algorithms were developed to uncover the underlying causal structure of the data. The aim of this study is to explore these structure learning algorithms in high-dimensional settings, and to determine their most effective application in the biomedical field according to the type of high-dimensional setting.

1.2. **Structure learning algorithms.** Structure learning algorithms learn causal relationships between variables from observational data. These relationships can then be represented in graphs, for instance a directed acyclic graph (DAG). To illustrate, a graph $G = (V, E)$ consists of a set of vertices (V), representing random variables, and a set of edges (E), representing the causal relationships between the vertices [11]. Graphs can be a visual inference tool to inspect the underlying causal network of a system. DAGs show directional causal relationships between vertices and assume that there are no causal cycles (loops) apparent

in the graph, thus a variable cannot ultimately influence itself through others. DAGs help us to visualize and infer causal relationships between vertices and to simulate how interventions might affect outcomes [12]. An example of a DAG related to lung cancer is illustrated in Figure 1 [46].

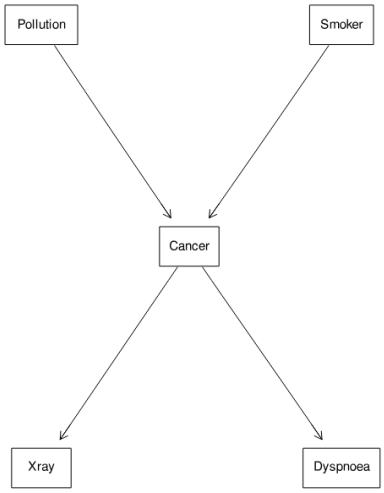


FIGURE 1. This DAG contains five vertices and four edges. We can see that pollution and smoking status have a directed edge to lung cancer, thus showing a causal link. Meanwhile, lung cancer has a causal link towards a positive X-ray result and dyspnoea (shortness of breath) [46].

DAGs are a powerful tool to visualize the underlying causal structure of a network and because they encode conditional independence assumptions. This means that some variables become statistically independent if we condition them on others. To illustrate, in the lung cancer DAG, smoker status and pollution are independent of each other if we condition on lung cancer. In other words, by removing the effect of lung cancer, we find no relationship between pollution and smoking status. Therefore, we can state that these variables are not causally related. Moreover, DAGs provide a framework to simulate interventions. We can simulate how changing one variable might affect others. For instance, in the lung cancer DAG, we might simulate how reducing smoking affects the chance of getting lung cancer. This ability to reason about interventions is especially valuable in biomedical research.

A DAG along with its conditional independencies can be learned from data by a structure learning algorithm. Structure learning algorithms can be distinguished into three main categories [13], namely:

- **Constraint-based algorithms:** Conditional independence tests are first applied to connect the vertices without determining causal direction. This creates an undirected graph (the skeleton) of the DAG. Then, the algorithm uses logical rules to find causal direction for as many edges as possible. This results in a partially directed graph (CPDAG), where some edges show causal direction and some remain undirected because their causal direction could not be learned from the data.
- **Score-based algorithms:** Different possible DAG structures are evaluated using a scoring function, then the structure with the highest score is selected. Common scoring functions are the Bayesian information criterion (BIC) or the Akaike information criterion (AIC).
- **Hybrid algorithms:** Constraint-based and score-based methods are combined.

There are several packages in the statistical programming language R that incorporate these constraint, score-based, and hybrid learning algorithms, such as bnlearn [14]. Even though these algorithms can operate in high-dimensional settings, learning a DAG remains very challenging. This is particularly true when the graphs are dense, meaning there are many edges relative to vertices, which is common in high-dimensional settings [15].

1.3. **Research gap.** Existing research is limited when it comes to structure learning algorithms in high-dimensional settings [16]. Most benchmarking studies are conducted in low- and/or moderate-dimensional settings ($N > P$) [17–20]. These studies primarily use small networks where the number of variables is 20 or lower, whereas high-dimensional settings are usually characterized by a large number of variables and a relatively smaller number of samples.

Some constraint-based, score-based and hybrid algorithms have been tested in high-dimensional settings. More recently, novel methods like non-parametric approaches, a combination of local and global search strategies, and deep learning methods have been researched in these settings too. However, the overall research gap in high-dimensional settings is substantial.

1.3.1. *Research on structure learning algorithms in high-dimensional settings.* The Peter-Clark algorithm (PC-algorithm, a constraint-based algorithm) has been shown to perform successfully in sparse high-dimensional settings [39]. This result established that constraint-based learning is statistically feasible at scale under the assumption of sparsity, meaning that each vertex has relatively few edges connected to it. However, the PC-algorithm is order dependent, meaning that the output varies according to the order in which the variables were given as input. While this is a small issue in low-dimensional settings, it can be problematic in high-dimensional settings. Therefore, modifications to eliminate the order dependence of the PC-algorithm resulted in the PC.stable algorithm. This algorithm achieved stable performance in both low- and high-dimensional settings [40]. However, besides the PC-algorithm, many of the other constraint-based algorithms have not been rigorously tested in various high-dimensional settings. This is especially true with different levels of prior knowledge, which is often available and important in biomedical settings. Prior knowledge refers to the ability to incorporate existing causal information (from, for example, literature, logic, or past evidence) in order to guide and/or constrain the search for a causal structure. This prevents the algorithm from starting from zero.

Additionally, constraint- and score-based methods have also been studied. In low-dimensional settings, they often outperform constraint-based methods in benchmarking studies [17, 41]. However, in high-dimensional settings the results and statements about score-based methods are mixed. The general view is that score-based methods are more accurate than constraint-based methods but they tend to underperform compared to hybrid methods, especially at small sample sizes. Moreover, score-based algorithms are claimed to scale less well to high-dimensional settings [41].

One notable example is the Greedy Equivalence Search (GES), a score-based method which has shown consistency and promising results in sparse high-dimensional settings [42]. In the simulation study designed to test GES, the authors compared Max-Min Hill-Climbing (MMHC, a hybrid approach), the PC-algorithm (constraint-based approach), GES, and Adaptively Restricted GES (ARGES, a hybrid approach of GES that incorporates constraint-based information). Overall, GES, ARGES, and MMHC outperformed the PC-algorithm, with ARGES performing best. However, it is unclear whether this simulation study involved the original PC-algorithm or the PC.stable variant.

Scutari and colleagues [41] conducted one of the few simulation studies that included both low- and high-dimensional settings, evaluating algorithms across all three categories (constraint-based, score-based, and hybrid). They found constraint-based algorithms to be less accurate than Tabu search (a score-based algorithm) across sample sizes. However, they also found most constraint-based algorithms to be more accurate than other score-based algorithms in many simulation settings, and no systematic differences in accuracy between hybrid and constraint-based algorithms. Furthermore, Tabu search was one of the fastest

algorithms as well. Score-based algorithms were also the only algorithms that produced results for data with thousands of variables. They conducted their simulation on both simulated data and real-world data.

These findings were in contrast to existing literature, which often favored hybrid methods [41]. However, they do note that the literature they referenced was using only discrete DAGs or assumed that as a basis, even when not stated explicitly. Importantly, their study did not include the effects of prior causal knowledge on the results, which is widely available and highly relevant in biomedical research.

1.3.2. *New approaches.* Furthermore, several new approaches to structure learning in high-dimensional settings have been proposed. Chakraborty and Shojaie [43] demonstrated that their non-parametric version of the PC.stable algorithm performed nearly as well as the original algorithm for Gaussian distributions, while offering clear advantages in non-Gaussian settings, where linearity cannot be assumed. Xue and colleagues [44] proposed an algorithm that uses both a local and a global causal structure learning. Their algorithm splits the high-dimensional space into smaller, more manageable subsets to learn their local structures, and then merges them into a global DAG. Local approaches were previously impractical with many variables, especially in high-dimensional settings. However, the authors presented their methods as designed for high-dimensional data, yet their simulations were done with $N >> P$ (5000 samples with up to 200 vertices). In addition, they simulated their own DAGs and did not use a benchmark from a repository, nor did they examine the effect of prior knowledge on their algorithm.

Lastly, causal deep learning approaches also show promise in high-dimensional settings. Lagemann and colleagues [45] developed a deep neural architecture combined from convolutional and graph neural networks that can learn from high-dimensional data and prior causal knowledge. However, their approach does not enforce acyclicity, thus the output will most likely not produce a valid DAG. Their focus was more about pairwise accuracy than structural correctness. Moreover, the method relies on prior causal knowledge. If one has none, performance may degrade. Also, this method is a so-called "black box" model, meaning that the interpretation of the learned causal mechanisms is not available. In other words, it is often not known how the algorithm determines whether a causal relationship exists or not. This is especially an issue in biomedical data since transparency is highly valued in this field. Lastly, they only test on very large networks ranging from $1,500$ to $50,000$ variables.

1.3.3. *Broader limitations.* Aside from the aforementioned limitations in high-dimensional settings, the general causal structure learning research (mostly low-dimensional with few variables) faces its own set of limitations. One of these is that some papers generate their own DAGs, whilst others use benchmark DAGs from a repository [26–29]. Also, the DAG generation process across literature varies widely [17, 21–25]. The difference between these two approaches on the performance of structure learning algorithms is not well researched. Similarly, little is known about the difference in performance of structure learning algorithms between dense and sparse DAGs. Lastly, almost all benchmarking studies are performed on discrete networks, whilst a lot of data (especially high-dimensional data) is continuous.

Because current research in high-dimensional settings is limited, it is not known which structure learning algorithms one should choose under different high-dimensional scenarios. The impact of different types of prior knowledge on the structure of the DAGs in high-dimensional settings is also unknown. Restrictive prior knowledge might block causal edges and informative prior knowledge might force causal edges. Furthermore, the knowledge on the difference in performance between ultra high-dimensional data ($P >> N$), moderate high-dimensional data ($P > N$) and almost high-dimensional data ($N \gtrapprox P$) is limited.

This means that there is a large research gap for structure learning algorithms in high-dimensional settings. A benchmark study that explores different high-dimensional scenarios with various types of prior knowledge on a set of diverse structure learning algorithms can provide systematic insights into this research gap. It can offer researchers guidance on effective use of structure learning algorithms in high-dimensional settings.

1.4. **Study aim.** The main aim of this thesis is to evaluate the performance of structure learning algorithms in various high-dimensional settings and to provide practical guidance for their application in biomedical research.

The following are sub-questions to support the study aim:

(1) **Algorithm performance**: How do structure learning algorithms perform across and within high-dimensional settings, ranging from $N \approx P$ to $P > N$?
(2) **Network characteristics**: How does the performance of structure learning algorithms vary between sparse and dense networks?
(3) **Prior knowledge**: How do different types of prior knowledge impact the performance of structure learning algorithms in high-dimensional settings?
(4) **DAG generation**: How does the use of generated DAGs versus repository benchmark DAGs impact the performance of the structure learning algorithms and the external validity of the results?
(5) **Practical guidance**: How does a researcher implement the results of this study in practical research?

1.5. **Overview.** The thesis is structured as follows: **Chapter 2** describes the methods used in the simulation study along with the rationale behind the choices made. A more detailed explanation of DAGs will be given, followed by a description of each step of the simulation. **Chapter 3** presents the results of the simulation study. **Chapter 4** provides an illustrative guide on a real-world high-dimensional Alzheimer's disease dataset. The insights from the simulation study are used to demonstrate how a researcher might apply these results to the dataset, given its characteristics and available prior knowledge. In **Chapter 5** the results will be interpreted in relation to the research questions. Moreover, the limitations of the study will be given along with suggestions for future research in this area. In **Chapter 6** the thesis will be concluded and in **Chapter 7** a declaration of AI usage will be given. Lastly, in the **Appendix**, results, figures, algorithms, and additional information will be provided.

2. Methods

This thesis is designed to evaluate the performance of structure learning algorithms in various high-dimensional settings. A simulation study was conducted in the programming language R (version 4.5.1). The simulation consists of five stages: (1) six distinct ground truth DAGs were simulated along with their structural equation models (SEM), and three were taken from repositories, (2) data was generated from these DAGs, (3) prior knowledge was formed, (4) structure learning algorithms learned the DAGs under different prior knowledge scenarios, (5) the simulation was performed and the learned DAGs were evaluated against the ground truth DAGs with several performance metrics.

2.1. Ground-truth DAGs.

As introduced in Section 1.2, DAGs provide a graphical representation of causal relations between random variables. A DAG is denoted $G = (V, E)$ and consists of a set of vertices (V), representing random variables, and a set of edges (E), representing the causal relationships between the vertices [11]. A missing edge encodes the absence of a causal relationship between variables. A DAG is directed because each edge has a unidirectional orientation, indicating the direction of causal influence. A DAG is acyclic because no vertex can cause itself through other vertices, and therefore loops are not allowed. For any vertex X_i, all vertices that are directly or indirectly caused by X_i are its descendants. The vertices directly caused by X_i are its children. Ancestors are all the vertices directly or indirectly causing X_i, and the vertices that directly cause it are its parents. To give an example, in Figure 1 all the vertices that have an edge directed towards the vertex Cancer are parents (Pollution and Smoker), while Cancer is a child of these parents.

Any DAG can be decomposed into three fundamental structures that describe how causal influence moves between vertices. These three basic structures form the building blocks from which all DAGs are constructed and are used to explain or determine the statistical dependence or independence between vertices, either marginally or conditionally [12]. The three structures are:

- **Chain:** In a chain causal influence flows through the vertices in a sequence, for example $X \to Z \to Y$ or $X \leftarrow Z \leftarrow Y$. In a chain structure, X and Y are marginally dependent, but once we condition on Z, X and Y become conditionally independent.
- **Fork:** In a fork one vertex causes two other vertices separately, for example $X \leftarrow Z \to Y$. In this example Z is a common cause of both X and Y. X and Y are marginally dependent, but once we condition on Z, they become conditionally independent.
- **Collider (also called V-structure):** In a collider structure two vertices both cause another vertex, for example $X \to Z \leftarrow Y$. This structure has the opposite effect of the two other structures above, X and Y are marginally independent, but once we condition on Z they become conditionally dependent. However, this conditional dependence does not represent an actual causal edge; it is a spurious association.

These three structures correspond to the three main sources of bias that can occur when estimating a DAG or inferring causal effects from it:

- **Overcontrol bias:** This type of bias occurs when the causal pathway is broken in a chain structure by conditioning on a vertex within it. For example, conditioning on the mediator Z in the chain structure $X \to Z \to Y$ would block the causal path from X to Y and make them conditionally independent. Therefore, the true causal effect of X on Y conditioned away, leading to an underestimation of the causal effect between X and Y.
- **Confounding bias**: This type of bias occurs when we fail to condition on a common cause. This can happen when a variable that is serving as a common cause (fork structure) has not been measured.
- **Collider bias:** Collider bias occurs when a collider (or one of its descendants) is conditioned on in a collider structure. This creates a spurious association between vertices that are otherwise independent.

These biases highlight the importance of guided conditioning when seeking to understand causal effects. It also shows that causal inference is tricky and why structure learning algorithms often make mistakes. The principle of d-separation was formalized to address the problems of these types of biases. D-separation determines whether two vertices are statistically independent given a set of vertices to condition on. A path p from X to Y is d-separated by conditioning on a set of vertices $\{Z\}$ if:

(1) p contains a chain or a fork such that a middle vertex, M, is in $\{Z\}$; or
(2) p contains a collision such that neither the collider nor any of its descendants is in $\{Z\}$ [48].

If a path p is not d-separated, it is said to be d-connected. D-separation is a method for identifying the conditional independencies in a valid DAG. It is the basis for constraint-based algorithms like the PC-algorithm, since they rely on conditional independence tests on the data to learn the underlying DAG structure. To apply d-separation for structure learning means we assume faithfulness. Faithfulness states that the independencies found in the data are also d-separated in the valid DAG. This assumption is crucial in learning DAGs because it ensures that the conditional independencies observed in the data reflect d-separation from the underlying DAG. This allows algorithms to learn which edges should exist or not exist.

Lastly, the Markov property tells us that a vertex is conditionally independent of its non-descendants given its parents. This means that once you know how the parents influence the vertex, any non-descendant vertices will not provide additional information. This can be encoded in the Markov blanket, which contains the smallest set of vertices that makes a vertex independent of every other vertex in the network. It can be seen as the informational boundary of a vertex. For any vertex X_i, its Markov blanket consists of its parents, children, and co-parents (other vertices that share a child with X_i).

All these DAG properties form the theoretical foundation for structure learning algorithms and explain how they learn DAG structures from data.

2.2. **Structural Equation Model (SEM).** A DAG can be seen as the graphical structure of the causal model, whereas a SEM can be seen as the underlying algebraic system of the DAG [38]. A DAG depicts directed relationships between vertices, but it does not contain weights or distributions of the edges, only the structure. A DAG can be parameterized into a SEM, where each vertex in the DAG is represented by one structural equation as a function of its parents and random error. The edges in a DAG correspond to nonzero entries in the coefficient matrix $\mathbf{B}$ of the SEM, whereas an absence of an edge between vertices corresponds to a zero entry.

Following Pearl [10], for a DAG $G = (V, E)$ with vertices $X_1, \ldots, X_P$, where $P = |V|$ is the number of vertices in the vertex set V. A linear Gaussian SEM for a vertex can be formulated as:

$$X_i = \sum_{j \neq i} \beta_{ij} X_j + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2), \tag{1}$$

where $i = 1, \ldots, P$, and $\text{pa}(X_i)$ denotes the set of all parents of X_i, where one parent in this set denotes X_j. Each parent X_j is with its own coefficient β_{ij} that represents the causal influence of X_j on X_i. In this formula, ϵ_i is the error term. It is normally distributed with mean 0 and residual variance σ_i^2, which is independent. Here, $j \neq i$ means that the summation runs over all vertices X_j except X_i. In matrix form, B contains all coefficients β_{ij} of the SEM, the absence of an edge implies $\beta_{ij} = 0$. The SEM in matrix form can be formulated as:

$$\mathbf{X} = \mathbf{XB} + \mathbf{E}. \tag{2}$$

With a SEM, data can be generated that is consistent with the DAG's structure.

2.2.1. *Simulated DAGs.* Ground truth DAGs were generated using the IC-DAG method from the R package bnlearn [14], which allows sampling random connected DAGs. This method samples a random DAG with uniform probability from the space of all connected DAGs with P vertices, using the IC-MCMC sampler [30]. A connected DAG ensures that all vertices are connected in the same graph. This means that no subgraphs or isolated vertices are present. Algorithm 1 outlines the IC-DAG method.

Algorithm 1: Ide and Cozman's multi-connected DAGs algorithm [31]

Input: Number of vertices (P), number of iterations (T)
Output: Return a connected DAG with P vertices
Initialize a simple ordered tree with P vertices, where all vertices have just one parent, except the first one that does not have any parent
Repeat the next loop T times:
    Generate uniformly a pair of distinct vertices i and j
    If the arc (i, j) exists in the actual graph, delete the arc, provided that the underlying graph remains connected
    else
    Add the arc, provided that the underlying graph remains acyclic
    Otherwise keep the same state
Return the current graph after T iterations

With the IC-DAG method, both dense and sparse DAGs were generated to evaluate how different levels of complexity can affect structure learning performance. Dense graphs contain many edges, which increase the number of conditional dependencies, making it harder to distinguish true edges from spurious associations. This makes retrieving the DAG with structure learning algorithms more difficult. Conversely, sparse graphs contain relatively fewer edges, which makes retrieval of the DAG easier. To capture both dense and sparse settings, ground truth DAGs were generated under different constraint settings. This was done by adjusting the maximum degree parameter, which is a constraint parameter that adjusts the sum of incoming and outgoing edges of a vertex. Sparse DAGs were generated with a maximum degree of 3 and dense DAGs with a maximum degree of 10. Both settings were generated each with three ground truth DAGs. These DAGs have P = 25, 100, 200 vertices, creating in total six ground truth DAGs. To illustrate the difference between sparse and dense DAGs, Figure 2 shows the dense DAG with P = 200 vertices and 972 edges (maximum degree = 10), whereas Figure 3 shows the sparse counterpart with P = 200 vertices but now only with 291 edges (maximum degree = 3). In addition to the generated DAGs, benchmark DAGs from a repository were imported to enable comparisons between generated DAGs and benchmark DAGs.

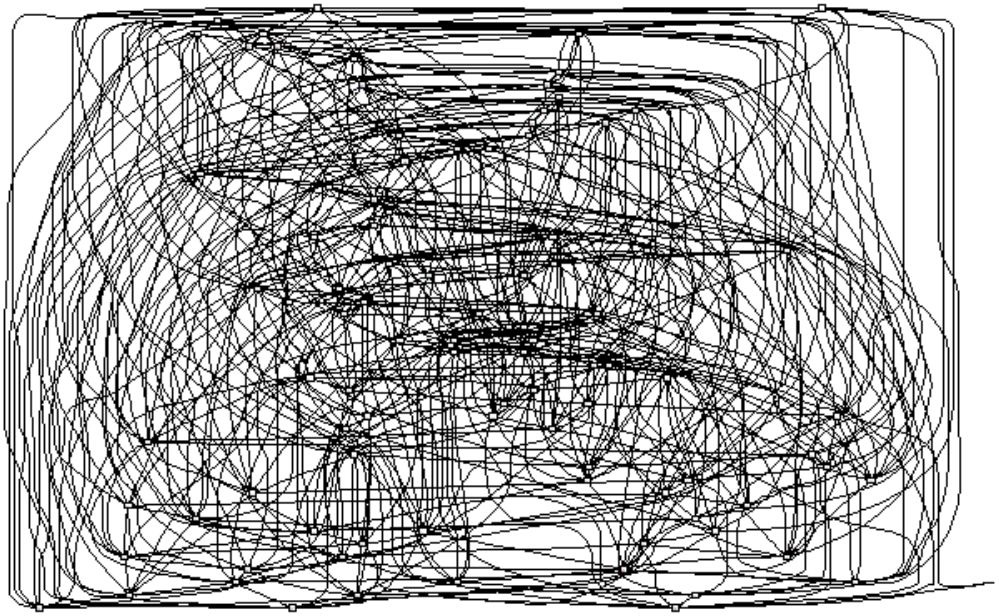


FIGURE 2. Dense DAG 3 with P = 200 vertices and edges = 972.

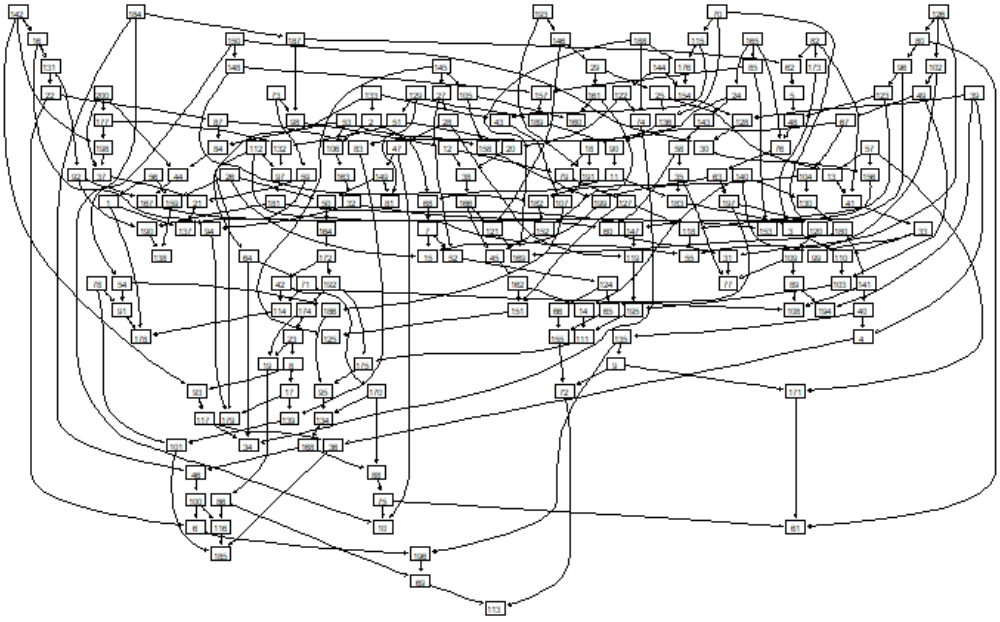


FIGURE 3. Sparse DAG 3 with P = 200 vertices and edges = 291.

2.2.2. *Benchmark DAGs.* Benchmark DAGs from a repository were used. This improves reproducibility and allows a fair comparison with our generated DAGs and with other simulation studies [33]. These

benchmarks are derived from real-world examples. They avoid the inconsistencies that can arise when each study generates its own DAGs using different generation approaches. These benchmarks are stored in open repositories [32–35]. Repositories often overlap and contain mostly the same networks. Moreover, most of the networks in these repositories are discrete. This coincides with the focus of DAG research, where discrete networks are standard and continuous (Gaussian) networks are less explored [36].

High-dimensional data are often continuous in nature and are modeled under the Gaussian assumption. This is especially true in biomedical settings, such as omics datasets (genome, proteome, and metabolome data) [37]. Therefore, continuous Gaussian benchmark DAGs were taken from the bnlearn repository [32] . The selected DAGs range from 46 to 107 vertices and from 70 to 150 arcs. The three selected DAGs are displayed in Figures 4, A5 and A6 (Figures beginning with an "A" refer to Appendix figures). Of these three, the ECOLI70 DAG is displayed here as an illustration in Figure 4.

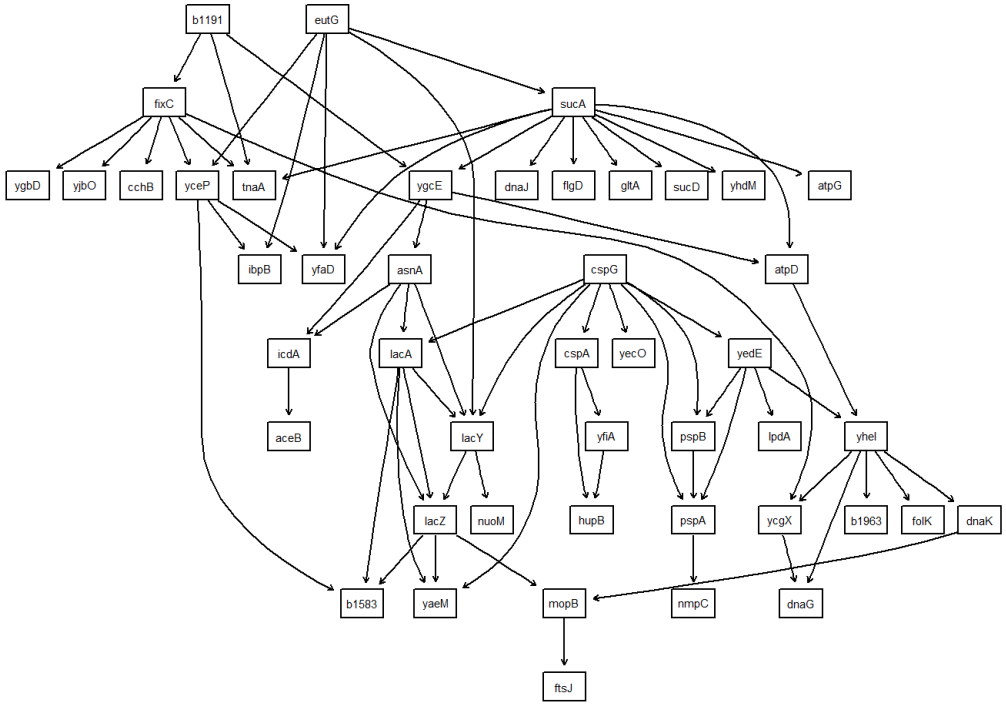


FIGURE 4. ECOLI70 is a real-world gene network of Escherichia coli (E. coli). It contains biologically realistic hubs and a non-random structure, unlike the simulated DAGs [47]. The DAG has P = 46 vertices and edges = 70.

In total, three sparse DAGs, three dense DAGs, and three benchmark DAGs from a repository were included in this study, totaling nine distinct ground-truth DAGs.

2.3. **Data generation.** To be able to generate synthetic datasets from the simulated DAGs, each vertex in the DAG was parameterized as a linear Gaussian SEM. If a vertex has no parents, the SEM only contains an intercept and residual variance. Since the intercept is fixed at 0 and the residual variance at 1, the parentless vertex equals the residual: $X_i = \epsilon_i \sim \mathcal{N}(0, 1)$. If a vertex has parents, a coefficient was randomly sampled from a uniform distribution of $[-1, 1]$ for each parent. This ensures that the strength of each relationship varies across all the edges in the DAG.

For example, if vertex X_3 has as parents X_1 and X_2, its SEM can take the following form:

$$X_3 = -0.9X_1 + 0.4X_2 + \epsilon, \quad \epsilon \sim \mathcal{N}(0,1). \tag{3}$$

This parameterization is done to all vertices in the DAG and results in a fully parametrized SEM. This fitted SEM can be used to simulate continuous datasets, which are consistent with the DAGs' causal structure. For the DAGs obtained from the repository, this parameterization was not necessary because they were already parameterized with a SEM.

A single combination of a DAG with a sample size constitutes one scenario. Datasets are high-dimensional when $P > N$. For the generated DAGs, P = 25, 100, and 200, and for benchmark DAGs, P = 46, 64, and 107. Therefore, sample sizes of N = 20, 50, and 150 were chosen so that most scenarios are high-dimensional. This results in 17 out of the 27 scenarios being high-dimensional. Not all scenarios are high-dimensional $P > N$; some are relatively more extreme $P >> N$, while others are closer to the low-dimensional range $N \approx P$. This allows us to evaluate how structure learning algorithms perform as dimensionality increases.

To obtain reliable performance estimates, multiple datasets were generated for each scenario (see Figure 5). This provides sufficient replications to evaluate differences in accuracy. Therefore, 50 datasets were generated for each scenario.

2.4. **Prior knowledge.** Prior knowledge refers to incorporating existing causal information (from literature, expert knowledge, or logical constraints) to guide structure learning algorithms with the learning of a DAG. This prevents the algorithm from starting without any structural guidance. Prior knowledge has been shown to improve structure learning performance, especially when data are limited [29], which is a hallmark of high-dimensional settings. Combining structure learning algorithms with prior knowledge generally produces more accurate results than relying on raw data alone [30].

Prior knowledge can be incorporated in two ways by structure learning algorithms using the bnlearn package:

(1) **Blacklist:** A blacklist contains pairs of vertices for which a directed edge is not allowed. If a directed edge is in the blacklist (for example, $X \rightarrow Y$), the structure learning algorithm cannot include it in the learned DAG. However, the opposite directed edge ($X \leftarrow Y$) is still allowed.
(2) **Whitelist:** A whitelist contains pairs of vertices for which a directed edge is present in the learned DAG. So, if a directed edge is in the whitelist, the structure learning algorithm must include it in the learned DAG. Constraint-based algorithms allow a whitelisted edge to be in both directions (for instance, both $X \rightarrow Y$ and $X \leftarrow Y$). The edge will be present in the DAG, but the algorithm will make a decision on the direction. Both score-based and hybrid algorithms allow for a whitelisted edge to be in only one direction (for example, either $X \rightarrow Y$ or $X \leftarrow Y$, not both).

2.4.1. *Blacklist.* Prior knowledge about which edges should not be present in the DAG can be obtained from different sources. Domain knowledge (for example, the omics cascade) can restrict edges so that causal direction can only go one way. This can prevent the algorithm from making mistakes. Another source of prior knowledge for the blacklist can be obtained from conditional independencies from other graphs, like for instance Gaussian graphical models (GGM). A great number of packages can learn conditional relationships in high-dimensional settings [49–59]. Under the assumption that the data are generated from a linear Gaussian DAG, and thus the Markov and faithfulness assumptions hold, zero entries in the precision matrix correspond to conditional independencies in the DAG. Therefore, a GGM can identify directed edges that are absent in the DAG, and these edges can be used to construct a blacklist.

For this thesis the R packages Rags2Ridges and Glasso were used to estimate a GGM from the generated data. These packages estimate a regularized precision matrix, from which a partial correlation matrix can be formed. The GGM is then a graphical representation of this partial correlation structure. This GGM can help to reduce the algorithm's search space, therefore potentially improving accuracy. A blacklist guided by a GGM does so by removing edges that are unlikely to be present in a DAG, which means that the

algorithm does not need to score or test those edges. Rags2Ridges is a package that estimates a GGM in high-dimensional settings by employing ridge (ℓ_2) regression and sparsification methods [49]. Glasso estimates a sparse inverse covariance matrix using a lasso (ℓ_1) penalty [52]. From this inverse covariance matrix the conditional independencies can be identified to construct a blacklist. In addition to Rags2Ridges and Glasso, a benchmark GGM was constructed for each scenario directly from the ground truth SEM. This represents the ideal GGM constructed blacklist where the true conditional independencies are known. Using this benchmark allows us to measure the maximum possible improvement that a GGM-constructed blacklist could provide. Thus, in total there were three types of blacklists constructed by the conditional independencies of the GGM, two estimated ones from Rags2Ridges and Glasso, and one ground truth GGM derived from the ground truth SEM.

However, this blacklisting approach shares similarities with constraint-based methods. These methods employ repeated conditional independence tests with a fixed significance threshold, to estimate the undirected skeleton network. After the estimation of the skeleton, logical rules based on d-separation are used to estimate the DAG [60]. Repeating numerous conditional independence tests (with $\alpha = 0.05$) increases the chance of type 1 errors; this is known as the multiple-testing problem. However, Rags2Ridges does not have this problem because it uses a sparsification method: the false discovery rate (FDR). FDR does not control the type 1 error for each individual independence test. Instead, it controls the expected proportion of false positives for all selected edges. This prevents the possible accumulation of type 1 errors that occurs through repeated conditional independence tests. Therefore, the use of a GGM as prior knowledge might still improve the accuracy of constraint-based methods.

Rags2Ridges uses a local FDR-cut to sparsify the estimated partial correlation matrix by keeping only entries that pass a chosen probability threshold for being truly dependent. For example, a local FDR-cut of 0.99 means that the retained partial correlations have at least a 99% probability of being true positives, so that at most about 1% of them are expected to be false positives. The local FDR-cut was set to 0.6 for the main simulation study. Lowering the FDR-cut compared to more default settings (0.8 to 0.99) produces a denser GGM. This means that fewer edges are removed. Therefore, the blacklist becomes smaller but more reliable. This is because the sparsified (blacklisted) partial correlations are more likely to be true conditional independencies. `Lam.min.ratio` in Glasso sets how small the smallest regularization value is relative to the largest one, and therefore controls how dense the final network can become. The `lam.min.ratio` was set to 0.001; this is lower than the default setting of 0.01. This value was chosen because the blacklist will therefore be more likely contain true conditional independencies.

2.4.2. *Whitelist.* A whitelist contains directed edges that must be present in the DAG. Causal relations can be learned from domain knowledge. For example, it is well-known that smoking causes lung cancer. The impact of whitelists in high-dimensional settings is not well studied. Several studies have incorporated whitelists as prior knowledge in low-dimensional settings [29, 61]. In these studies, a subset of true directed edges from the ground truth DAG is placed into the whitelist to evaluate their effects on the structure learning algorithm's performance. These studies have used whitelist sizes ranging from 5% up to 50% of all true directed edges. In this thesis, we examined two whitelists: one containing 5% and another containing 25% of the true directed edges. A whitelist of 50% was not considered because the DAGs in our simulation study are substantially larger (up to 200 vertices) compared to the low-dimensional simulation studies (up to 109 vertices) [29, 61].

2.5. **Structure learning algorithms.** When dealing with health data, it is crucial that algorithms are transparent and interpretable. This is because the output of the algorithms needs to be explainable to the stakeholders, especially in biomedical settings where clinical or scientific decisions can rely on its results. Therefore, we opted for algorithms that are transparent, interpretable, validated, and able to incorporate prior knowledge. Black-box learning algorithms, for instance neural networks or reinforcement learning, were excluded.

Therefore we opted for the following inclusion criteria for the algorithms:

- Must be available in R.
- Must support prior knowledge.
- Must handle continuous Gaussian data.
- Must be able to handle high-dimensional data.
- Must be interpretable (no black-box algorithms).
- Must be established methods that have been evaluated in other simulation studies.

Furthermore, the number of DAGs to be learned by the algorithms (16200 DAGs per algorithm) can make it computationally infeasible to include too many. In addition, we include algorithms from all three classes: constraint-based, score-based, and hybrid. Based on these inclusion criteria, and on the performance and frequency of use of the algorithms in prior simulation studies [17, 18, 20, 62], the following algorithms were included:

Constraint-based algorithms:

(1) **GS:** The Grow Shrink (GS) algorithm estimates the Markov blanket of each vertex using a grow phase where it adds vertices that show conditional dependencies, and a shrink phase where it removes vertices that become conditionally independent given the current blanket. It estimates the edges by connecting vertices that are in each other's Markov blanket, resulting in a skeleton [63]. In bnlearn, orientation rules based on d-separation are applied to estimate directionality of the edges [14]. This results in a partially directed graph (CPDAG), since it cannot always find directions for all edges. See Algorithm A1 in the Appendix for the pseudo-code.

(2) **Iamb.fdr:** This is an FDR modified version of the incremental association Markov boundary (IAMB) algorithm. It estimates Markov blankets with forward and backward conditional independence testing. Each test is evaluated with a FDR instead of a fixed α threshold. Like GS, it estimates the edges by connecting vertices that are in each other's Markov blanket, resulting in a skeleton [64]. After this, orientation rules in bnlearn are applied to learn the CPDAG. See Algorithm A2 for the pseudo-code.

(3) **PC.stable:** The Peter Clark (PC) algorithm starts with a complete undirected graph and removes edges iteratively when it finds a conditional independence, which results in an undirected skeleton. It is the stable variant because during each iteration, it stores the set of all edges that it will remove, and only removes them after the entire round of tests is finished. After the skeleton is found, orientation rules based on d-separation are applied to find directionality, resulting in a CPDAG [40]. See Algorithm A3 for the pseudo-code.

Score-based algorithms:

(4) **HC:** The hill-climbing (HC) algorithm utilizes greedy search to add, remove or reverse edges. After each modification, it computes a score. If the modification improved the score, it is accepted, otherwise it is rejected. The algorithm can use random restarts to avoid local optima. The process repeats until the score does not increase [65]. See Algorithm A4 for the pseudo-code.

(5) **Tabu:** Tabu search (Tabu) is a modification of HC. It uses memory (a tabu list) to avoid returning to visited solutions [65]. See Algorithm A5 for the pseudo-code.

Hybrid algorithms:

(6) **MMHC:** The max-min hill climbing (MMHC) algorithm first learns the skeleton through the constraint-based max-min parent and children (MMPC) algorithm. Then, orientation of the edges in the skeleton is learned using the HC algorithm [67]. See Algorithm A6 for the pseudo-code.

(7) **Rsmax2:** The General 2-Phase Restricted Maximization (Rsmax2) algorithm implements the Sparse Candidate approach. First, it uses a constraint-based approach to limit the set of possible parents for each vertex. Then, orientation of the edges is learned with a score-based algorithm within this restricted parents set. Because the search is performed per vertex rather than the whole search space, complexity is reduced and therefore scales well to high-dimensional settings [68]. See Algorithm A7 for the pseudo-code.

These seven algorithms are all available in the bnlearn package [14], which was used in this simulation study. All algorithms were used using bnlearn's default settings.

2.6. **Simulation procedure.** There were 27 distinct scenarios (9 DAGs $\times$ 3 different sample sizes) to be learned, each with 50 replicates. This resulted in a total of 1350 generated datasets. For every dataset, the seven structure learning algorithms were applied to 12 prior knowledge settings: first, no prior knowledge (just the generated data), second, three blacklist settings (benchmark ground truth partial correlations, Rags2Ridges, Glasso), third, two whitelist variants (5% + 25% of true edges), and lastly, six blacklist + whitelist combinations. This produced $16,200$ learned DAGs per algorithm and $113,400$ learned DAGs in total (see Figure 5 for an overview of the whole simulation procedure).

The simulation was performed in R version 4.5.1 on a Windows 64-bit workstation with Intel Xeon W-2235 CPU (6 cores @ 3.80GHz, 48GB RAM), using parallel computing on 6 workers and fixed random seeds. The results, including information on priors, and the learned DAGs were stored to be analyzed. The R code and analysis outputs are available at `https://thesesmoos.netlify.app/simulation` (see Appendix B).

2.7. **Evaluation metrics.** The learned DAGs were evaluated using structural Hamming distance (SHD), F1-arrow, and F1-skeleton scores. These are widely used metrics in previous studies [13, 19, 20, 61, 62]. Each metric provides a different aspect of performance. This allows for a more complete evaluation. The metrics are based on the concepts of recall (sensitivity), precision and specificity. Precision measures how many of the predicted edges are actually correct out of all the positive predictions, whereas recall measures how many of the true positive edges are recovered by the algorithm out of all the actual positive edges. Specificity measures the proportion of true negatives that are correctly predicted out of all the actual negatives.

- **F1-Arrow:** The F1-score (also known as F1-arrow or arrowhead) is the harmonic mean of precision and recall. It is defined as: $\mathrm{F1} = 2 \times \frac{recall \times precision}{recall + precision}$ [20].
- **F1-Skeleton:** The F1 skeleton score (also known as F1-adjacent) is similar to the F1-arrow score, except that it ignores edge direction. This score is used since it is more fair towards constraint-based algorithms that learn a CPDAG, because any non-oriented edge counts as a mistake in the F1-arrow score [62].
- **SHD:** The structural Hamming distance (SHD) is the number of changes required to turn the learned DAG into the ground truth DAG. So, any edge insertion, deletion or change of direction, counts as one point [13]. It is a widely used metric; however, SHD can be biased towards recall over specificity [20].

2.8. **Overview of the simulation study.** An overview of this simulation study is presented in Figure 5:

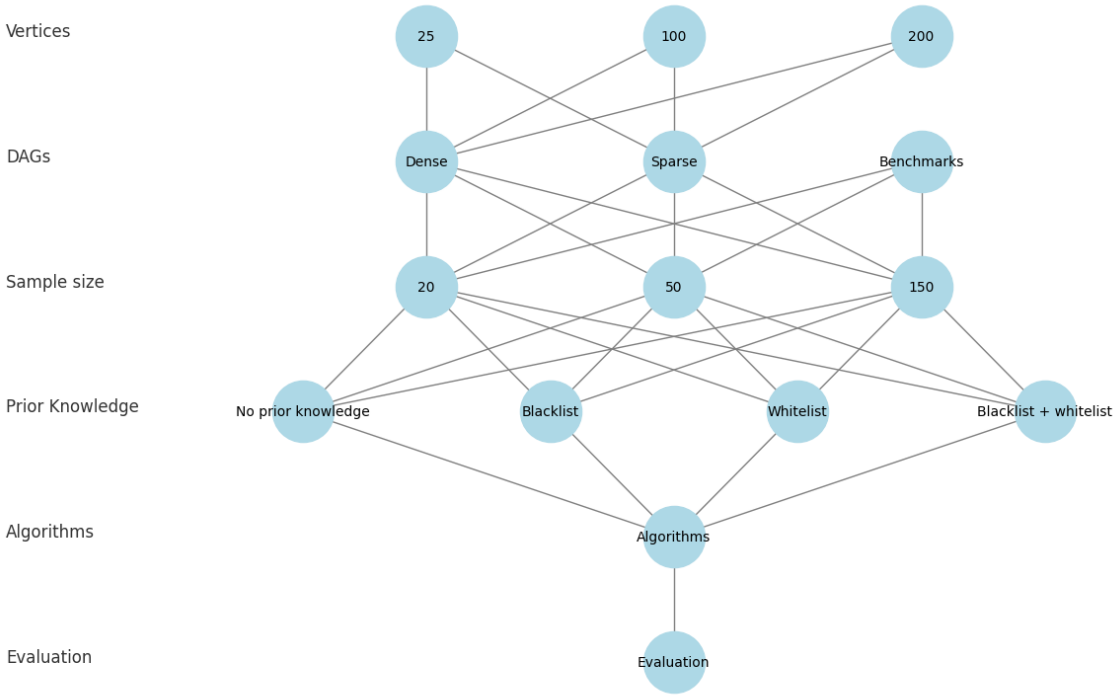


FIGURE 5. Overview of the simulation study. The order of the simulation goes from top to bottom. First, the number of vertices is assigned to the generated dense and sparse DAGs. From all the DAGs, data with the sample sizes 20, 50, and 150 are generated. Then, different prior knowledge types are constructed. Afterwards, the algorithms are constructed and run with all the settings. Lastly, the performance of the algorithms is evaluated. This overview represents all possible combinations in the simulation study.

3. Results

In total there were nine ground truth DAGs; six were generated and three were taken from the bnlearn repository. These are displayed in the following figures:

- **Dense:** DAG 1 (Figure A1), DAG 2 (Figure A2) and DAG 3 (Figure 2).
- **Sparse:** DAG 1 (Figure A3), DAG 2 (Figure A4), and DAG 3 (Figure 3).
- **Bnlearn repository:** MAGIC-IRRI (Magic) (Figure A5), ARTH150 (Arth) (Figure A6), and ECOLI70 (Ecoli) (Figure 4).

The results on the algorithms' performance will be given for the algorithms individually and sometimes about their families: constraint-based (GS, Iamb.fdr, PC.stable), score-based (HC, Tabu), hybrid (MMHC, Rsmax2). To compare algorithm performance, we report the absolute difference between their F1-scores and SHD values, denoted as Δ. We present the results in figures with a rank format, sample size $N = 20, 50, 150$ on the X-axis, and rank one to seven on the Y-axis. Here the highest rank is the best performing algorithm, and thus highest score for a setting. Each algorithm is color coded to assess their rank (see the legend of each results figure) and the score for a particular metric is given for each algorithm in each setting.

3.1. **No prior knowledge.**

3.1.1. *F1-skeleton results. Based on the results in Figure 6.* For the dense DAGs, the score-based algorithms (HC and Tabu) rank 1-2, with a clear performance gap from rank 3. In DAG 1, the difference decreases from $\Delta F1 = 0.22$ ($N = 20$) to 0.17 ($N = 150$), and in DAG 2 it increases from $\Delta F1 = 0.06$ to 0.15 across the sample sizes. In dense DAG 3, Rsmax2, PC.stable, and MMHC rank 1-3 at $N = 20$ and 50, with a minimal difference of $\Delta F1 = 0.01$ between them. At $N = 20$, the difference between rank 1 and 5 is also minimal with $\Delta F1 = 0.01$ between them. For $N = 150$, the score-based algorithms return to ranks 1-2 with a clear performance gap of $\Delta F1 = 0.08$.

For the sparse DAGs, PC.stable and the hybrid algorithms (Rsmax2 and MMHC), occupied ranks 1-3 with a minimal difference of $\Delta F1 = 0.01$ between them across all DAGs and sample sizes. Score-based algorithms rank lower (rank ≥ 5) in these sparse settings when P increases (from sparse DAG1, to sparse DAG3). Within each sparse DAG, the difference for F1-skeleton score between score-based algorithms and rank 1 increases as N grows. For example, in sparse DAG 3 the difference increases from $\Delta F1 = 0.24$ ($N = 20$) to $\Delta F1 = 0.39$ ($N = 150$).

DAGs from a repository share similarities with sparse DAGs. Rsmax2, PC.stable, and MMHC occupy ranks 1-3, with a minimal difference of $\Delta F1 \leq 0.02$ between them across all DAGs and sample sizes. All gaps in performance between ranks are closer for repository DAGs, compared to the performance gaps in dense and sparse DAGs.

Taken together, score-based algorithms perform the best in dense settings, with an exception for DAG 3 at $N = 20, 50$. However, the difference between the top algorithms in dense DAG 3 is negligible. Surprisingly, once settings get more sparse, score-based algorithms perform among the worst algorithms, and PC.stable and hybrid algorithms perform best. Across all settings, Iamb.fdr is the worst performing algorithm.

More broadly, three trends are visible across the F1-skeleton results. First, the type of network (dense, sparse, and repository DAGs) has a substantial impact on overall performance and which algorithm type performs best. Generated sparse DAGs have the largest absolute F1-skeleton scores, followed by repository DAGs and dense DAGs. The algorithm rankings among sparse and repository settings are similar. Second, sample size is a big factor for overall performance; increasing N influences the performance of structure learning algorithms substantially. The performance increase is observed to be stable with increasing N across both $P > N$ and $N \gtrapprox P$, indicating no clear threshold effect around $P \approx N$. This suggests that performance is driven by sample size and graph complexity (E relative to P), rather than simply crossing out of the $P > N$ regime. Third, increasing graph size (P and E proportionally) has a noticeable impact on F1-skeleton performance in the sparse and repository DAGs when N is low, but this effect is weaker at higher N. For example, the sparse DAGs show very similar F1-skeleton scores for rank 1 at $N = 150$

($F1 \approx 0.8$) across them, but at $N = 20$ the performance goes down from $F1 = 0.45$ to $F1 = 0.32$. Repository DAGs show an inconsistent pattern in performance when graph size increases, showing more variability. For example, the Magic network ($P = 64$ and $E = 102$) is smaller than Arth ($P = 107$ and $E = 150$), yet the Arth network has higher F1-skeleton scores. Conversely, dense settings display a more notable decline in performance as graph size increases, especially when N is low. Performance is likely not solely explained by an increasing graph size, or by P or E relative to N alone, but rather by graph complexity (how E scales with P) in combination with N. For example, in the high-dimensional setting, dense DAG 3 with $P = 200$ and $N = 150$ performs similarly to the low-dimensional setting, dense DAG 2, where $P = 100$ and $N = 150$. Although sparse DAG 2 has larger P and E in absolute terms than dense DAG 1, its lower edge density/complexity likely makes the structure easier to learn, resulting in substantially larger F1-scores (except for similar scores at $N = 20$).

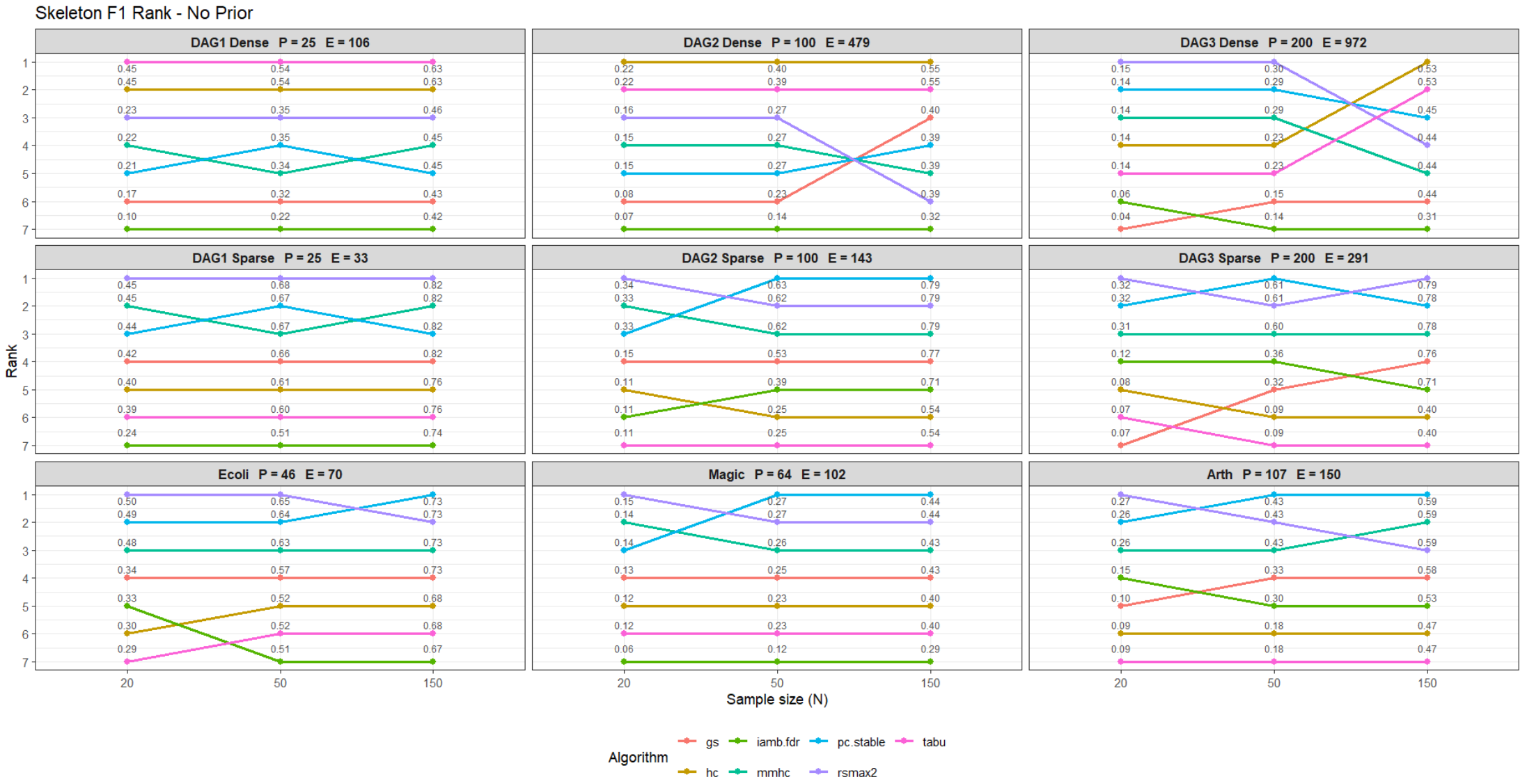


FIGURE 6. Skeleton F1 Rank scores with no prior knowledge.

3.1.2. *F1-arrow results. Based on the results in Figure 7.* The F1-arrow scores are overall between $\Delta F1 = 0.24$ and $\Delta F1 = 0.05$ lower than the F1-skeleton results. The F1-arrow rankings for the dense DAGs are similar to the F1-skeleton ordering. Score-based algorithms typically rank 1-2 in most dense DAGs.

For the sparse DAGs, Rsmax2 and MMHC rank 1-2 for DAG 1. In DAG 2 and DAG 3, GS ranks 1 at $N = 50, 150$ and PC.stable at $N = 20$.

For the DAGs from the repository, there is a similar pattern in algorithm rankings as compared to the sparse DAGs. GS ranks 1 or 2 in the Ecoli and Magic networks, with a performance gap to rank 3 of $\Delta F1 = 0.01$ to $\Delta F1 = 0.05$. PC.stable ranks 1 for the Arth DAG at $N = 20$, and the hybrid algorithms at $N = 50, 150$.

Taken together, the F1-arrow insights and rankings are similar to the F1-skeleton results. However, all F1-arrow scores are $\approx 25\%$ to 50% lower than the F1-skeleton scores. This means that most algorithms make significant mistakes in finding correct edge orientation. One notable difference is the GS algorithm; it ranks higher in the results of the sparse and repository DAGs compared to its ranks in the F1-skeleton results. This suggests that GS is better at learning the correct edge orientation than finding the underlying network structure, for the sparse and repository settings.

More broadly, we observe the same trends as in the F1-skeleton results. First, network type (dense, sparse, or repository DAGs) strongly influences performance and which algorithm (type) performs best. Second, sample size has a substantial effect on performance. Improvements with an increasing N are steady across all DAGs. Third, increasing graph size (P and E proportionally) shows performance declines for small and medium $N = 20, 50$ but little effect with high $N = 150$. Last, defining dimensionality via $P > N$ versus $N > P$ does not reveal a clear threshold effect in performance; instead, performance is better explained by N together with graph complexity/density (E relative to P).

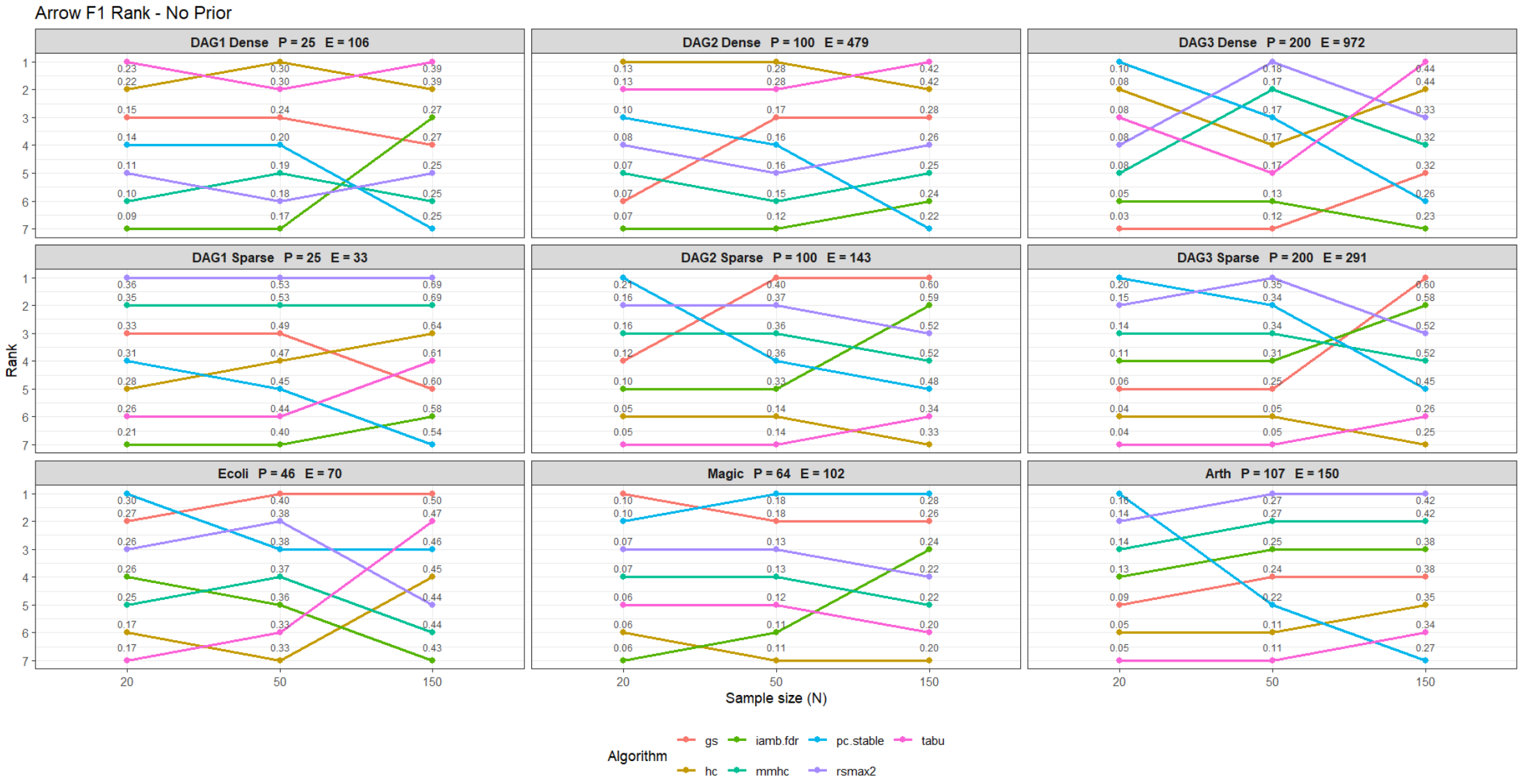


FIGURE 7. F1-arrow rank scores with no prior knowledge.

3.1.3. *SHD results. Based on the results in Figure A7.* SHD ranks differ clearly from the F1-scores for score-based algorithms; score-based algorithms rank 6-7 for all DAGs. They have the largest SHD scores, especially in the more extreme high-dimensional settings. Except for score-based algorithms, the SHD values are comparable across the remaining algorithms, indicating that there is no top performing algorithm.

For the sparse DAGs, Rsmax2 and MMHC rank 1 for DAG 1, while Iamb.fdr and Rsmax2 rank 1 for DAG 2 and 3.

Repository DAGs have similar results compared to the sparse DAGs in terms of ranking. Rsmax2 and MMHC rank 1 for the Ecoli DAG, and Iamb.fdr ranks 1 for both Magic and Arth.

Taken together, SHD displays a different result compared to the F1-scores. Now score-based algorithms are the worst performers across all settings. This clearly indicates that score-based methods produce a lot of false positives. While conservative methods like GS, which performed worst for the F1-scores, now perform among the best algorithms since it does not allow for many edges at all. This shows that SHD emphasizes specificity as it heavily penalizes false positive edges.

3.2. **Blacklist.** The blacklist results are compared to the no prior results, unless stated otherwise.

3.2.1. *F1-skeleton results, benchmark blacklist. Based on the results in Figure 8.* The benchmark blacklist changes the algorithm ranks across all settings. Notably, score-based algorithms now rank 1-2 across all settings. Especially when N is low, the score-based algorithms perform with a clear difference compared to other algorithms.

For the dense DAGs, score-based algorithms show the largest increase in F1-scores in the extreme high-dimensional settings. From DAG 1 to DAG 3, the blacklist improved the score-based F1-score from $\Delta F1 = 0.07$ to $\Delta F1 = 0.33$ (at $N = 20$).

In sparse settings, this increase is larger, from $\Delta F1 = 0.34$ in DAG 1 to $\Delta F1 = 0.68$ ($N = 20$) in DAG 3. For DAG 3 ($N = 20$), this corresponds to a 1071% improvement. Comparing the score-based algorithms to rank 3 for DAG 3 at ($N = 20$), the difference is clear with $\Delta F1 = 0.26$.

In the repository DAGs, there are similar improvements as with sparse settings. Score-based algorithms increase in F1-scores, especially when N is low.

Taken together, the benchmark blacklist improves the score-based algorithms' performance drastically. Score-based algorithms become the best performing algorithms for every setting with a clear performance gap between the other algorithms. Also, the blacklist substantially improves the performance of the score-based algorithms in the most extreme high-dimensional settings, thereby reducing the influence of N on their performance. Therefore, the performance at low N becomes more comparable to that at high N.

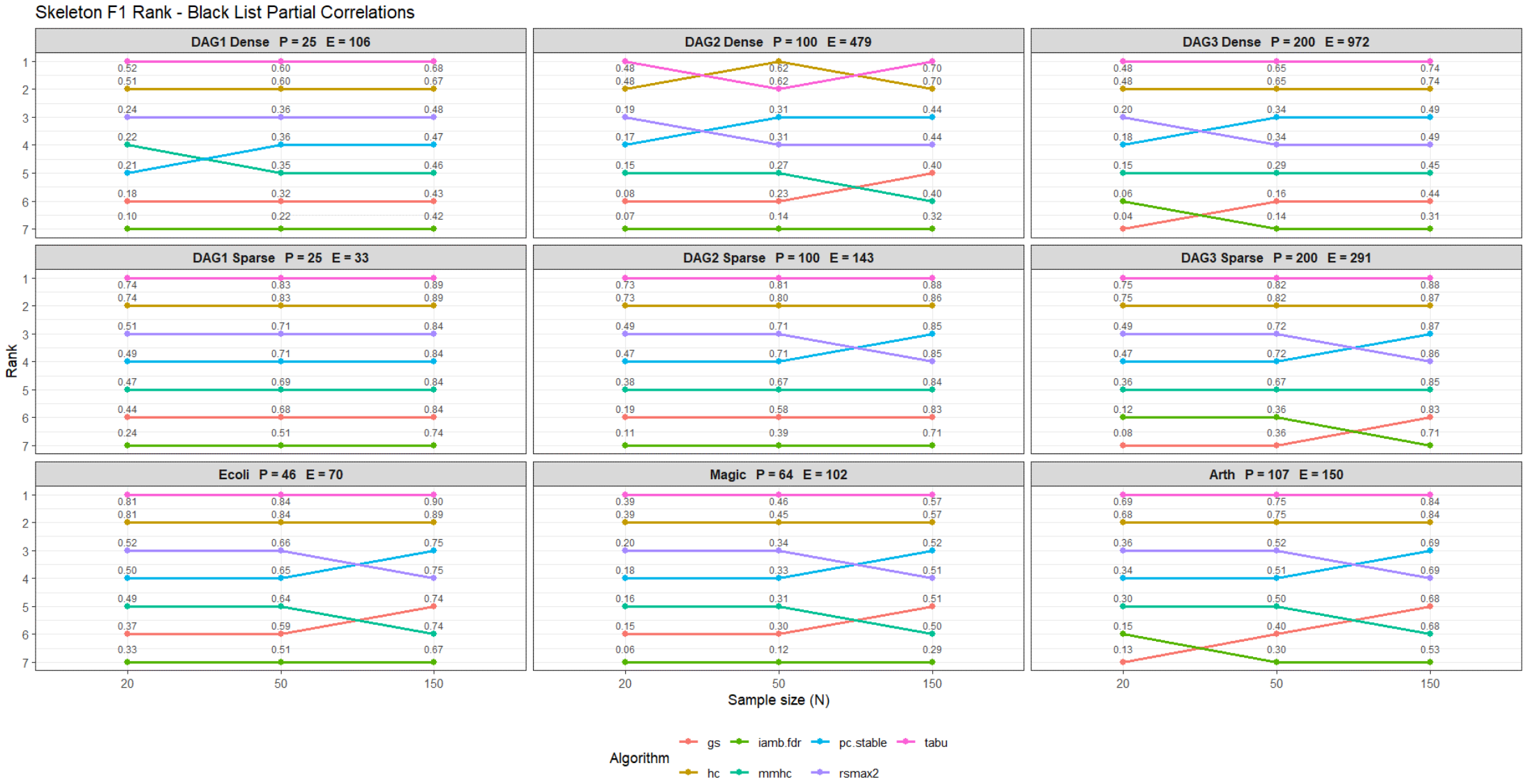


FIGURE 8. F1-skeleton rank scores with benchmark blacklist.

3.2.2. *F1-arrow results, benchmark blacklist. Based on the results in Figure 9.* Compared to the F1-skeleton results, the F1-arrow results show that score-based methods also rank 1-2 across most settings. However, the F1-arrow scores of score-based algorithms decrease more than other algorithms. As a result, in the sparse settings ($N = 50, 150$) of DAG 2, DAG 3 and Magic, PC.stable now ranks 1. The ranks of the other algorithms are similar to the F1-skeleton results.

Taken together, score-based algorithms recover the underlying DAG structure very effectively, indicated by the high performance in F1-skeleton results. However, even though they are still the best performers in F1-arrow results, they do drop faster in performance compared to other algorithms. This suggests that score-based algorithms have more difficulties in finding accurately oriented edges compared to other algorithms, especially PC.stable.

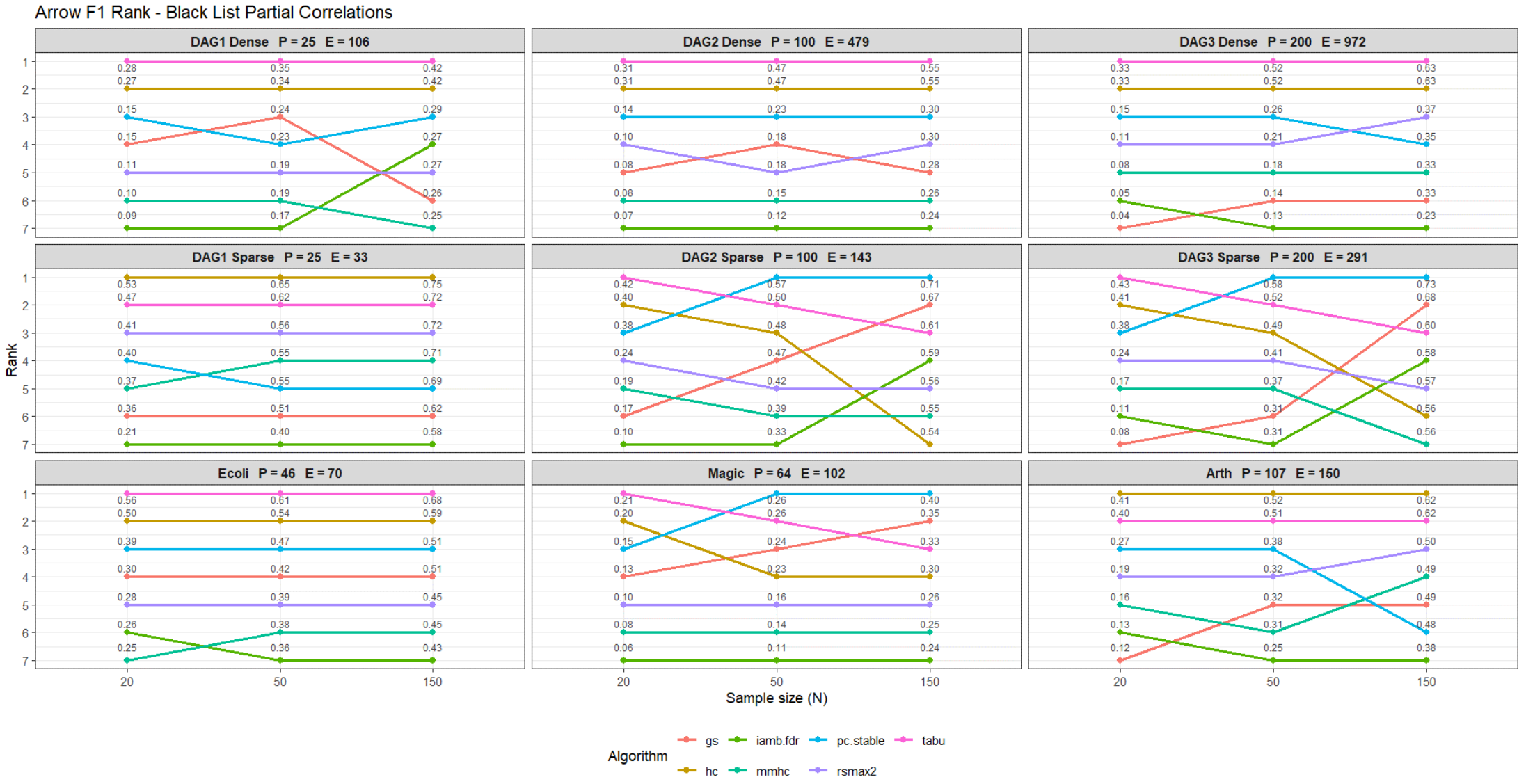


FIGURE 9. F1-arrow rank scores with benchmark blacklist.

3.2.3. *SHD results, benchmark blacklist. Based on the results in Figure A8.* The blacklist changes the SHD results significantly compared to the SHD results with no prior knowledge, where more conservative algorithms like GS ranked high. Score-based algorithms together with PC.stable rank 1 on all but one of the 27 settings. The improvements for score-based algorithms in SHD compared to no prior are large, especially in both sparse and dense DAG 2 and 3. For example, in sparse DAG 3 ($N = 50$), the SHD decreased with $\Delta SHD = 4287$, from 4482 to 195.

Taken together, the performance of score-based algorithms improves substantially with the benchmark blacklist compared to the no prior SHD results. This suggests that the blacklist reduces the likelihood of false positives by restricting the search space consistent with the conditional dependencies of the true underlying DAG. The orientation accuracy of the score-based algorithms is also high, although it drops performance compared to F1-skeleton scores faster than other algorithms. Finally, their SHD values are substantially lower compared to having no prior; this is due to fewer false positives. This highlights the potential of score-based algorithms when the likelihood of false positives is significantly constrained by a blacklist. Furthermore, PC.stable's performance with a blacklist is strong in medium to low-dimensional sparse settings.

3.2.4. *F1-skeleton results, Rags2Ridges blacklist. Based on the results in Figure A9.* The impact of the estimated blacklist on F1-skeleton scores shows variability; it fluctuates both upward and downward across and within settings. However, dense DAG 1 shows a clear drop in performance, with the top ranking algorithms decreasing by $\Delta F1 = -0.17$ to $\Delta F1 = -0.05$ (-27% to -11%). The clearest improvements are for the score-based algorithms and PC.stable, while the remaining algorithms experience either slight decreases or no effect.

Taken together, the estimated blacklist by Rags2Ridges does not improve the performance of the top ranking algorithms notably on the F1-skeleton scores. The results fluctuate considerably between and within settings. However, overall there is a clear improvement for score-based algorithms and PC.stable, especially in sparse high P settings.

3.2.5. *F1-arrow results, Rags2Ridges blacklist. Based on the results in Figure A10.* The estimated blacklist from Rags2Ridges marginally improves the top ranking algorithms' F1-arrow scores, with $\Delta F1 = 0.01$ to $\Delta F1 = 0.08$ for dense DAG 3, sparse DAG 2 and DAG 3, and the repository DAGs Ecoli and Magic. These improvements correspond to relative increases of approximately 5% to 23%. For the remaining DAGs, F1-arrow scores decreased with $\Delta F1 = -0.13$ to $\Delta F1 = -0.01$ (-33% to -3%). The algorithms that benefit the most from the estimated blacklist are score-based and PC.stable. Iamb.fdr, Rsmax2 and MMHC generally perform worse or are indifferent with the estimated blacklist.

Taken together, score-based algorithms and PC.stable improve performance with the estimated blacklist, except for dense DAG 1. The remaining algorithms do not benefit. The overall performance for rank 1 improves with small margins for dense DAG 3, sparse DAG 2, 3 and repository DAGs Ecoli and Magic. The remaining DAGs lose performance for rank 1. This suggests that the estimated blacklist from Rags2Ridges has potential, but that it restricts the search space too strictly or insufficiently in certain scenarios, especially when compared to the benchmark blacklist.

3.2.6. *SHD results, Rags2Ridges blacklist. Based on the results in Figure A11.* The SHD results and rankings remain largely unchanged or show slight improvements compared to the no prior results. Score-based algorithms still rank among the lowest in most scenarios, as in the no prior results. However, they have a substantially lower SHD across all scenarios compared to the no prior results. For dense DAG 2 and DAG 3 at $N = 150$, they rank 1, indicating a similar trend to the benchmark blacklist results where score-based algorithms perform well in high P and high N scenarios.

Taken together, the estimated blacklist has little influence on the SHD, except for score-based algorithms. This suggests that the false positives are greatly reduced for these algorithms.

3.2.7. *Glasso blacklist. Based on the results in Figures A12 to A14.* Across the evaluation metrics (F1-skeleton, F1-arrow, and SHD), both estimated blacklists by Glasso and Rags2Ridges perform similarly. Therefore, the key differences are highlighted below since the trends (or no trends) are described for Rags2Ridges.

For the F1-skeleton results, positive differences can be observed for dense DAG 2 and DAG 3, where Glasso shows marginal improvements compared to no prior results. However, on the repository DAGs Ecoli and Arth, the scores significantly drop at $N = 20, 50$.

For the F1-arrow results, Glasso shows slight improvements for dense DAG 3, sparse DAG 2 and DAG 3. Similar to the F1-skeleton scores, Glasso drops performance on the repository DAGs Ecoli and Arth at low N.

For the SHD results, the estimated blacklist has virtually no impact on SHD values and ranks compared to no prior knowledge, except for score-based algorithms, where SHD performance improves. Compared to the SHD results obtained using Rags2Ridges, the score-based algorithms perform better with Glasso at $N = 20$, similar at $N = 50$, and worse at $N = 150$ across settings.

3.3. **Whitelist.** *Based on the results in Figures A15 to A20.* The whitelist leads to small but consistent improvements across all settings and evaluation metrics (F1-skeleton, F1-arrow and SHD). The increase in F1-skeleton is around 5%, with a slightly higher increase for F1-arrow scores. The increase in performance is most pronounced when the evaluation scores were low, so typically for most high-dimensional situations where N is the lowest. This suggests that the more extreme high-dimensional data might benefit the most from a whitelist, whilst the effect on more low-dimensional data is lower but still relevant, especially in dense settings. The improvements in SHD are generally smaller than those observed for the F1-scores, nevertheless there is still a slight but consistent decrease in SHD. This indicates that the whitelist does not have a substantial impact on the chance of false positives.

The whitelist does not alter the ranking of most algorithms across settings. Only minor rank shifts occur for the score-based methods when using the 25% whitelist. In more extreme high-dimensional settings, and the more sparse a DAG is, score-based algorithms generally do not improve nearly as well as the other algorithms from the whitelist. Whereas the denser the graphs become, the lower the P is, and the more data there is available, score-based algorithms improve the most from the whitelist, and take the top rank with a substantial difference to the other algorithms (except for SHD). This suggests that the whitelist impacts the score-based algorithms differently than a blacklist. The remaining algorithms seem to benefit equally from the whitelist, therefore there are no major differences in the rank order after using the whitelist. One exception is Iamb.fdr for F1-arrow ranks; it performs significantly better in the repository DAGs Ecoli and Magic when the whitelist gets larger, ranking 1 (compared to on average rank 5.33) for each N on these networks with the 25% whitelist.

For the F1-skeleton scores, the performance of the benchmark blacklist is overall greater than the 25% whitelist, except for dense DAG 1 and the Magic network. For the F1-arrow scores, the 25% whitelist performs better for the dense DAGs and repository DAG Magic, and similar for sparse DAGs and repository DAGs Ecoli and Arth. Lastly, for the SHD, the whitelist performs better for the dense DAGs and Magic and the blacklist performs better for sparse DAGs and the Ecoli and Arth network. These results indicate that a whitelist can improve overall edge orientation and a blacklist can improve finding the underlying DAG skeleton and reducing the amount of false positives (for PC.stable and especially score-based algorithms).

3.4. **Whitelist + Blacklist.** *Based on the results in Figures A21 to A38.*

Adding the whitelist with a blacklist does not impact the rank ordering meaningfully, especially with the 5% whitelist. However, the overall performance metrics improve when compared to only a whitelist or blacklist. The estimated blacklist with a whitelist has a similar effect on the rankings as for only having a blacklist. This is in line with the results of only using a whitelist, overall the performance increases, but the ranking remains largely the same.

3.5. **Sensitivity analysis on GGM hyperparameters.** *Based on the results in Figures A39 to A44.* The benchmark blacklist shows good performance for all settings. In contrast, the estimated blacklists were more inconsistent, usually leading to marginal improvements. These improvements remain still far from the benchmark. This raised the question whether hyperparameter tuning could improve the performance of the estimated blacklist. The FDR-cut of 0.6 for Rags2Ridges is likely still set too high for the purpose of the blacklist.

The FDR-cut controls the probability that the retained edges are true edges. However, when forming the blacklist we want to shift that focus to retaining true conditional independencies in the blacklist. This implies that the FDR-cut could be set inversely to be more confident that a missing edge corresponds to a true conditional independence. Therefore, we conducted an exploratory sensitivity analysis to decrease the FDR-cut to 0.01.

For Glasso, the lam.min.ratio was reduced from 0.001 to 0.0001. This can potentially lower the regularization and yield a denser precision matrix estimate. This can increase the likelihood that the edges in the blacklists are true conditional independencies, and reduce the chance of true positives in the blacklist.

The results from the sensitivity analysis show some slight improvements for Rags2Ridges for the score-based algorithms on dense DAGs, but overall the results are similar. For Glasso, the performance stayed the same.

3.6. **Main results.**

- Performance is primarily driven by sample size N, graph density/complexity (E relative to P), and graph size (E and P proportionally) at low and medium N $(20, 50)$, but these effects are weaker at high N (150).
- Dense DAGs are best learned by score-based algorithms (HC, Tabu).
- Sparse and repository DAGs are best learned by PC.stable and hybrid algorithms (Rsmax2, MMHC).
- Score-based algorithms produce many false positives without prior knowledge according to SHD.
- A benchmark blacklist makes score-based algorithms the best performing algorithm in all settings for both F1-skeleton and F1-arrow scores.
- Estimated blacklists (Rags2Ridges, Glasso) give small but inconsistent gains.
- Whitelists give consistent gains for all algorithms across all metrics, corresponding to their size.
- Combining blacklist and whitelist improves scores further but not the algorithm rankings compared to the blacklist results.

4. Illustration study

This section illustrates the results and methods of this thesis. We used metabolomics data of patients with Alzheimer's disease (AD). AD is a progressive neurodegenerative disorder, for which there is no cure. It is responsible for 60-80% of dementia instances. The main risk factors are aging, genetic predispositions, and epigenetic changes [71]. One of these risk factors is the APOE $\varepsilon 4$ allele, a common genetic variant involved in lipid transport in the brain that strongly increases the risk of developing AD [69]. Pathologically, AD is characterized by neural and synaptic loss as a result of the accumulation of amyloid-β (Aβ) plaques and tau neurofibrillary tangles [70]. Studies have shown that dysregulation of energy metabolism in the brain is related to AD progression [70]. APOE $\varepsilon 4$ is also related to changes in lipid metabolism [69]. In this illustration, our focus will be on metabolomics and APOE $\varepsilon 4$. With metabolomics, we refer to the collective of metabolites. Metabolites are small molecules that are a result or an intermediate of metabolism. Some alterations in the metabolism are associated with an increased risk of AD. Furthermore, changes in antioxidant, amino acid, and lipid metabolism have been reported in AD studies [69]. Therefore, learning a DAG can help with hypothesis generation about metabolite dependencies and possible causal pathways under linearity, Gaussianity, no unmeasured confounding, and faithfulness assumptions.

The data used in this illustration are readily available in the package Rags2Ridges [49]. It is a double anonymized subset from the data by De Leeuw and colleagues [69]. The dataset consists of $P = 230$ metabolites on $N = 127$ patients, who were recruited from within the Amsterdam Dementia Cohort [72]. The dataset contains two classes: class 1 ($N = 40$) includes patients without the APOE $\varepsilon 4$ genetic variant, and class 2 ($N = 87$) includes patients with the APOE $\varepsilon 4$ predisposition for AD. The metabolites are 53 amine compounds, 116 lipid compounds, 22 organic acid compounds, and 39 oxidative stress compounds [69].

4.1. **R illustration.**

To load the dataset and estimate sparsified partial correlations for constructing the blacklist, we used Rags2Ridges. To learn the DAG and perform bootstrap averaging, we used bnlearn.

```
library(rags2ridges)
library(bnlearn)
```

We separate the data into the two APOE $\varepsilon 4$ classes; `ADclass1` includes patients without the genetic variant and `ADclass2` includes patients with it. Then, we transpose the data to get the samples in the rows and the metabolites in the columns, and standardize each metabolite to have a comparable scale.

```
data("ADdata", package = "rags2ridges")

ADclass1 <- ADmetabolites[, sampleInfo$ApoEClass == "Class 1"]
ADclass1 <- scale(t(ADclass1))

ADclass2 <- ADmetabolites[, sampleInfo$ApoEClass == "Class 2"]
ADclass2 <- scale(t(ADclass2))
```

Next, we found the optimal lambda parameter (via 10-fold cross-validation) to estimate the sparsified partial correlation matrices for both classes. For this estimation, we use the same setup that was used in the illustration of Rags2Ridges on the exact dataset, (see [49]). We sparsify using the localFDR thresholding option with a cut of 0.01, since our simulation results show that this increases both F1-arrow and F1-skeleton results compared to using no prior (see Figures 6, 7, A39 and A40). This yielded a sparsified partial correlation matrix per class, which we can use to construct a blacklist.

```
set.seed(123)
estimate_rags_pcorp <- function(matrix){
  OPT <- optPenalty.kCVauto(matrix,
                            fold = 10,
                            lambdaMin = 1e-07,
                            lambdaMax = 20,
```

```
                                target = default.target(covML(matrix),
                                type = "DUPV"))

  opt_lambda <- OPT$optLambda
  P0 <- sparsify(OPT$optPrec,
                              threshold = "localFDR",
                              FDRcut = 0.01,
                              verbose = FALSE)
return(pruneMatrix(P0$sparseParCor))
}

pcorp_AD1 <- estimate_rags_pcorp(ADclass1)
pcorp_AD2 <- estimate_rags_pcorp(ADclass2)
```

From the sparsified partial correlation matrices, we constructed a blacklist by adding all its estimated conditional independencies. The function below extracts all variable pairs with a zero partial correlation and returns them in bnlearn's blacklist format (from, to).

```
black_list_func <- function(p_cor_matrix) {
  idx <- which(abs(p_cor_matrix) == 0,
                    arr.ind = TRUE)
  cbind(rownames(p_cor_matrix)[idx[,1]], colnames(p_cor_matrix)[idx[,2]])
}

bl_1 <- black_list_func(pcorp_AD1)
bl_2 <- black_list_func(pcorp_AD2)
```

The resulting blacklist was then used as a constraint for learning a DAG. Metabolic networks are considered to be dense/complex networks [49]; therefore, we use the dense setting results as a guideline for selecting an algorithm and an estimated blacklist approach. The dataset contains $P = 230$ variables; this comes close to dense DAG 3 with $P = 200$ (see Figure 2). We chose the structure learning algorithm Tabu since it performs best for both F1-arrow and F1-skeleton results with a blacklist for dense DAG 3, see Figures A39 and A40.

```
class1_data <- as.data.frame(ADclass1)
class2_data <- as.data.frame(ADclass2)

DAG_class1 <- tabu(class1_data, bl = bl_1)
DAG_class2 <- tabu(class2_data, bl = bl_2)
```

Next, we visualized the learned DAGs using the Graphviz plot. This resulted in the following DAGs displayed in Figure 10 and Figure 11.

```
graphviz.plot(DAG_class1)
graphviz.plot(DAG_class2)
```

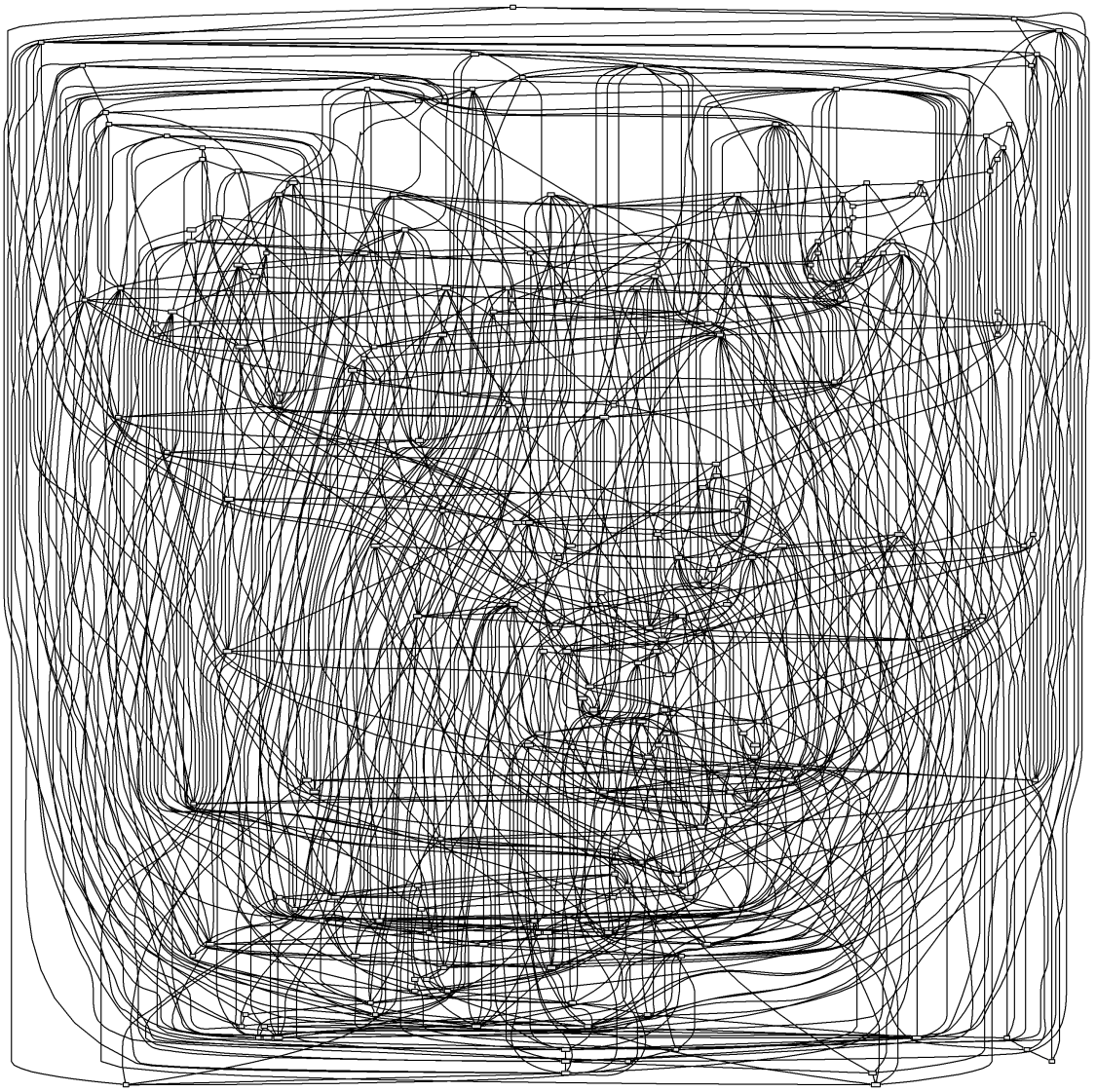


FIGURE 10. DAG class 1.

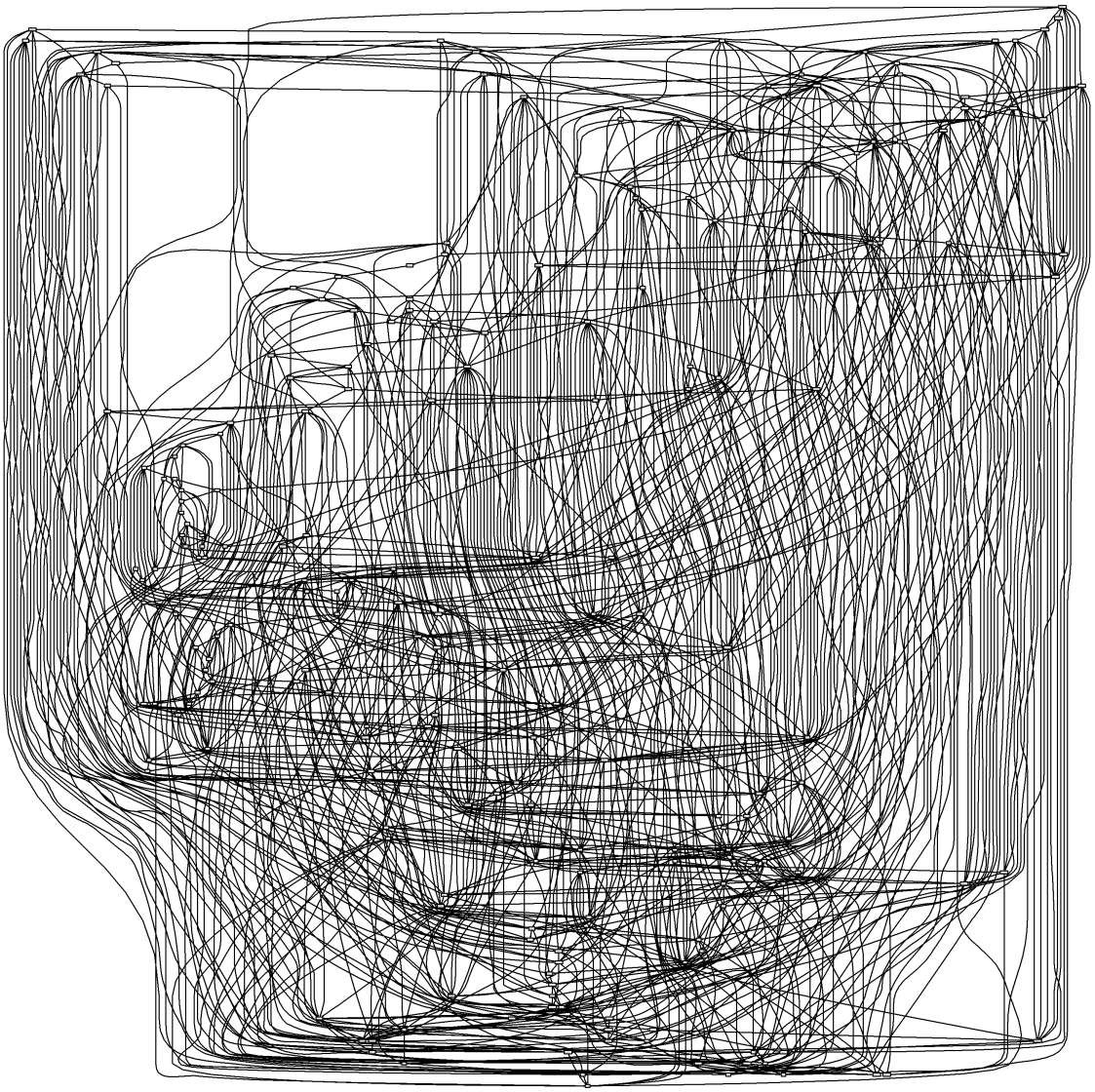


FIGURE 11. DAG class 2.

However, these graphs are difficult to use for inference due to the large number of vertices and edges. Furthermore, we know that these DAGs most likely still contain false positives. To reduce the likelihood of false positives, we apply a bootstrap method from bnlearn. This method bootstraps the data and learns a DAG with Tabu for each bootstrap sample. This approach allowed us to compute the strength of every possible edge, and only allow edges above a certain threshold. This results in an averaged DAG made from the learned bootstrapped DAGs. In our example, we chose 200 bootstraps with an edge threshold of 0.7. We did the same for class 2.

```
boot1 <- boot.strength(
  data = class1_data,
  R = 200,
  algorithm = "tabu",
```

```
  algorithm.args = list(blacklist = bl_1)
)

avg <- averaged.network(boot1, threshold = 0.7)
```

We visualized the averaged network with igraph [73]. The vertices were colored by metabolite class (amines, organic acid, lipids and oxidative stress) to improve interpretability. To improve interpretability further, we removed isolated vertices (vertices with no edges after thresholding in the previous step). A fixed random seed was then used for reproducibility. Next, we used the force-directed layout algorithm that positions connected vertices closer together, making the overall graph easier to read. Although a shared layout (same coordinates for vertices that are present in both graphs) would be ideal for direct visual comparison, producing an interpretable aligned layout for non-identical DAGs after thresholding required substantial additional customization; therefore, we plotted each DAG separately using the same layout algorithm and a fixed seed. This resulted in Figure 12 and provides an interpretable DAG for exploratory hypothesis generation. We did the same for class 2, resulting in Figure 13.

```
library(igraph)

nodes1 <- nodes(avg)

Colors1 <- rep("black", length(nodes1))
names(Colors1) <- nodes1

Colors1[grep("Amine",     nodes1)] <- "lightblue"
Colors1[grep("Org.Acid",  nodes1)] <- "orange"
Colors1[grep("Lip",       nodes1)] <- "yellow"
Colors1[grep("Ox.Stress", nodes1)] <- "purple"

g1 <- as.igraph(avg)

isolates <- which(igraph::degree(g1) == 0)
g1 <- igraph::delete_vertices(g1, isolates)

V(g1)$color       <- Colors1[V(g1)$name]
V(g1)$frame.color <- "red"
V(g1)$label.color <- "black"
V(g1)$label.cex   <- 0.3
V(g1)$size        <- 7
E(g1)$color       <- "black"

set.seed(123)
plot(
  g1,
  layout = layout_with_fr(g1),
  edge.arrow.size = 0.07
)
```

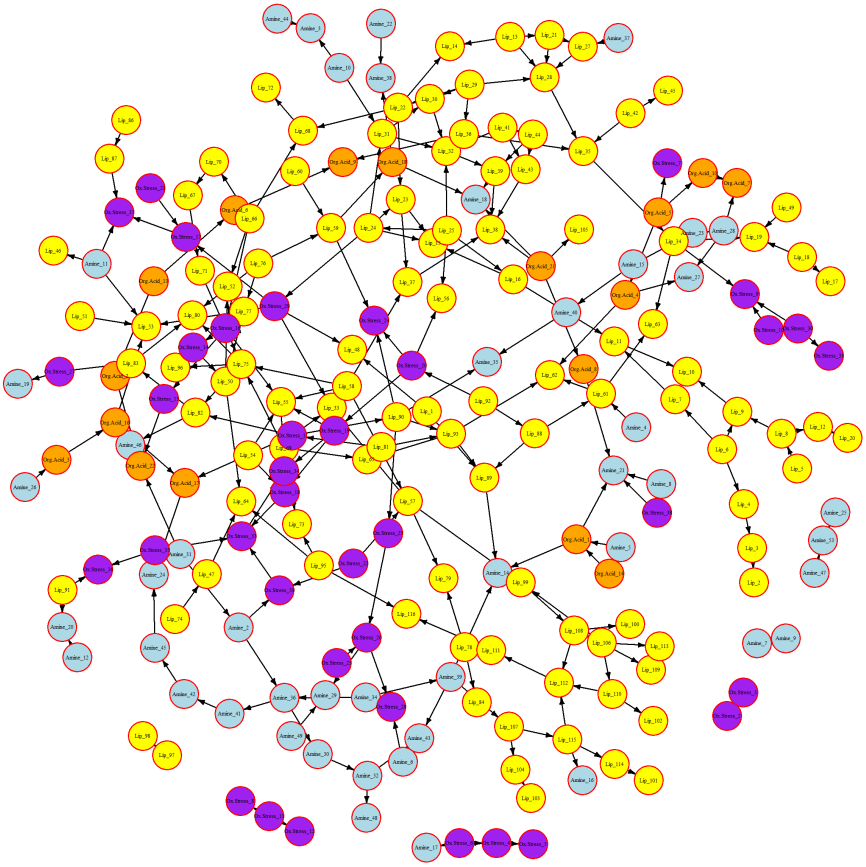


FIGURE 12. DAG class 1, metabolite network for patients without a known predisposition for AD. The vertex colors are: light blue for amines, orange for organic acids, yellow for lipids, and purple for oxidative stress compounds.

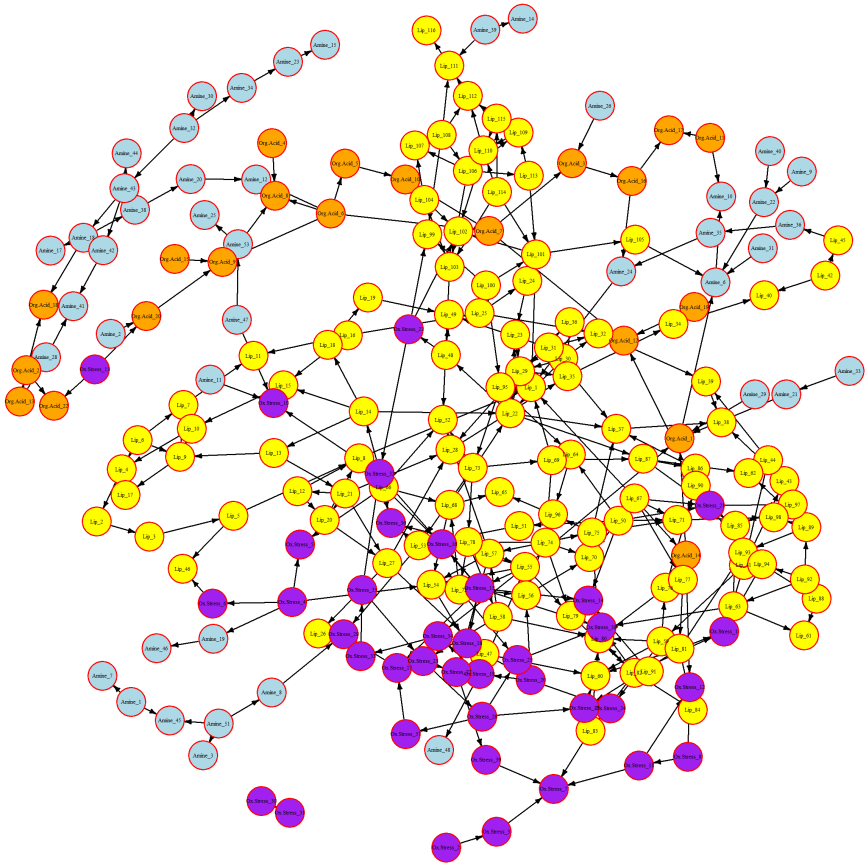


FIGURE 13. DAG class 2, metabolite network for patients with a known predisposition for AD. Note, two edges were undirected because their edge strength was the same from the bootstrap. Orienting these edges would result in cycles, and thus not a valid DAG. Therefore, the undirected edges were removed from this graph.

When compared to APOE $\varepsilon 3$, the APOE $\varepsilon 4$ carrier is described as more poorly lipidated; it tends to carry fewer lipids in the brain and therefore prevents efficient transport of lipids between glia and neurons [74]. Besides lipid alterations, APOE $\varepsilon 4$ has been linked to greater vulnerability of oxidative modification of neuronal proteins [75]. Furthermore, APOE $\varepsilon 4$ status has been associated with differences in amino acid related metabolism [76].

De Leeuw and colleagues [69] found that the metabolite network for AD patients without the APOE $\varepsilon 4$ status is less cohesive compared to the network for AD patients with the APOE $\varepsilon 4$ status. They suggested that this could be due to alternative biochemical dysregulation between the groups. Qualitatively, our DAGs

appear to show a similar pattern; DAG class 2 appears more cohesive (a larger, denser connected core), whereas DAG class 1 appears more dispersed.

Therefore, we can analyze the following exploratory hypothesis based on our two DAGs and the existing research: At the group level, AD patients with the APOE $\varepsilon 4$ status exhibit a metabolite DAG with different directed connectivity patterns within and between lipid-, amine-, and oxidative stress metabolic groups, compared to AD patients without APOE $\varepsilon 4$ status.

First, we analyzed if DAG class 2 is indeed globally less fragmented (more cohesive) than DAG class 1. For this we used the function `components` from the igraph package. The function counts connected components (ignoring edge direction). A component is a connected group of vertices that is disconnected from the rest of the DAG; it is used to assess whether the learned DAG is a more cohesive structure or split into multiple disconnected groups. We report the number of components and the size of the largest component for each DAG.

```
c(components(g1, mode="weak")$no, max(components(g1, mode="weak")$csize));
c(components(g2, mode="weak")$no, max(components(g2, mode="weak")$csize))
```

The number of components in class 1 is seven, with the largest component being 197 vertices, whereas the number of components in class 2 is two, with the largest component being 216 vertices. This suggests that class 2 may be more cohesive (less fragmented) than class 1.

Next, we analyzed the exploratory hypothesis by assessing whether the directed connectivity patterns between the metabolic groups (lipid, amine, and oxidative stress) differ. For this, we made a function that produces a mixing matrix. It works as follows; first it selects all the directed edges in the DAG and assigns each attached vertex to its metabolite group based on its name; second, it keeps only edges where both vertices belong to one of the lipid, amine, or oxidative stress groups; and third, it counts how many edges go from each group (rows) to each other group (columns), including within group edges (diagonal). Furthermore, we also report proportions to account for differences in the total number of edges between the two DAGs. This provides a direct summary aligned with our hypothesis, because it compares directed connectivity patterns within groups and between groups.

```
edge_mix_dir <- function(g, groups = c("Amine","Lip","Ox.Stress")){
  ed <- ends(g, E(g))
  grp <- function(x) sub("_[^_]+$", "", x)
  a <- grp(ed[,1]); b <- grp(ed[,2])
  keep <- a %in% groups & b %in% groups
  table(factor(a[keep], levels=groups), factor(b[keep], levels=groups))
}

m1_dir <- edge_mix_dir(g1)
m2_dir <- edge_mix_dir(g2)

m1_dir
m2_dir
m2_dir - m1_dir
round(m1_dir / sum(m1_dir), 3)
round(m2_dir / sum(m2_dir), 3)
```

The results (see Appendix B) show that the total number of directed edges among lipid, amine, and oxidative stress increased from 226 for class 1 to 305 for class 2 ($\Delta = 79$). In class 2, the directed edges within the lipid group (lipid $\rightarrow$ lipid) increased from 129 to 193 ($\Delta = 64$), and proportionally (for the total edges among these three metabolic groups) from 0.571 to 0.633 ($\Delta = 0.062$). Class 2 also shows more directed edges from lipids to oxidative stress; it increased from 12 to 27, and proportionally 0.053 to 0.089 ($\Delta = 0.036$). This suggests a shift toward more lipid to oxidative stress connectivity, whereas oxidative stress to lipid connectivity decreases from 12 to 7, and proportionally from 0.053 to 0.023 ($\Delta = -0.030$).

Furthermore, the number of directed edges within the oxidative stress group increases from 26 to 38, and proportionally from 0.115 to 0.125. Research suggests APOE $\varepsilon 4$ is associated with altered lipid biology and greater vulnerability to oxidative damage, so differences in lipid-oxidative stress connectivity between and within groups are plausible [74, 75]. Amines only show minor changes in connectivity. Proportionally, in class 2, the amine edges make up a smaller share of the total edges among these groups. Using a DAG, rather than an undirected network, allows us to quantify differences in directed connectivity patterns between metabolic groups. This is useful for exploratory hypothesis generation about potential pathway directionality. Our results reflect this with the asymmetric lipid–oxidative stress connectivity: lipid to oxidative stress connectivity increases and oxidative stress to lipid connectivity decreases.

Overall, these exploratory results suggest that APOE $\varepsilon 4$ status is associated with different directed connectivity patterns within and between the lipid and oxidative-stress groups, with less pronounced differences for amines, compared to no APOE $\varepsilon 4$ status.

5. Discussion

The main aim of this study was to evaluate the performance of structure learning algorithms in various high-dimensional settings and to provide practical guidance for their application in biomedical research. To this end, a simulation study and an illustration study on AD data were conducted. The simulation study revealed how structure learning algorithms perform across a range of low- to high-dimensional settings. This was achieved by testing the effects of DAG sparsity, different forms of prior knowledge, and the use of benchmark versus generated DAGs. The illustration study showed how to implement the results of this study in practice on a real-world biomedical dataset, and how to infer insights from the findings.

5.1. **Research questions.** Because the main motivation of this study is about causal inference from high-dimensional observational biomedical data, the accuracy of edge orientation is the primary goal. Therefore, although SHD and F1-skeleton results provide important information about structural accuracy, the F1-arrow score is most aligned with evaluating the algorithms based on causal inference. For this reason, F1-arrow scores are used as the primary evaluation metric to answer the research questions. SHD and F1-skeleton scores are interpreted as secondary but still informative metrics about learning the underlying structure and the types of errors an algorithm can make.

5.1.1. *How do structure learning algorithms perform across and within high-dimensional settings, ranging from $N \approx P$ to $P > N$?* Across the range of low- to high-dimensional settings, the results indicate that the performance of structure learning algorithms is driven by the interplay of absolute sample size N in combination with graph complexity (how E scales with P) and graph size (E and P proportionally), rather than by the $P > N$ regime alone. Across our evaluated sample sizes ($N = 20, 50, 150$), increasing N improves performance across the evaluation metrics for all algorithms and settings. Defining settings as high-, mid-, or low-dimensional via $P > N$ versus $N \gtrapprox P$ does not reveal a clear threshold effect or explain differences in performance. Instead, the improvements with an increasing N are steady and do not show an abrupt change around $N \approx P$. In addition, graph complexity impacts the performance of structure learning algorithms. The more edges present relative to P, the more difficult it is to learn a DAG, likely because a dense graph increases the size of the local conditioning/regression problems faced during each learning step of conditional independence testing (constraint-based and hybrid algorithms) and parent-set selection (score-based algorithms). This could be interpreted as a form of local dimensionality, because the effective dimension of conditional independence tests (conditioning-set size) and local regressions increases with higher density/complexity at the vertex level. This might also explain why the effect of increasing graph size is moderate when graph complexity remains comparable, because structure learning algorithms are largely operating at a local level. Still, the absolute graph size impacts the performance, but more conditionally. Its impact is more substantial when N is small, whereas when N is large, the impact of a larger graph size is lower (for F1-scores). For SHD, the number of edges and sample size are substantial factors that influence performance.

Within dense settings, score-based algorithms are the strongest performers in terms of F1-scores. Hybrid algorithms, GS, and PC.stable perform best in the sparse and repository settings. Notably, although there are general trends regarding which algorithm (type) performs best in each setting, variability remains both within and across settings. Therefore, it is important to select the correct algorithm based on the results from this simulation study. This is because no algorithm performs best across all settings, and each shows its own setting and evaluation-metric-specific strengths and weaknesses.

5.1.2. *How does the performance of structure learning algorithms vary between sparse and dense networks?* The type of network (dense, sparse, and repository DAGs) has a substantial impact on overall performance and which algorithm type performs best. Score-based algorithms perform best for dense settings according to F1-scores. For the sparse and repository settings, hybrid algorithms, GS and PC.stable, perform best according to their F1-scores. Generated sparse DAGs have the highest overall evaluation scores, suggesting that these are the easiest to learn for the algorithms; these are followed by repository DAGs and lastly dense

DAGs. The algorithm rankings among sparse and repository settings are similar. This indicates that the repository DAGs have a similar level of sparsity, and that sparsity has a substantial influence on algorithm performance. Therefore, it is highly relevant to know (or have an indication) of the sparsity of the network that a structure learning algorithm has to learn.

SHD further clarifies the effects of density by penalizing false positives the most. Score-based algorithms rank worst under SHD across all settings with no prior knowledge, and especially in dense settings. This shows that score-based algorithms contain many false positives. Conversely, conservative methods such as Iamb.fdr perform well under SHD. However, these conservative methods perform well under SHD because their learned DAGs are small, but this does not mean that they are accurate, as their F1-scores indicate. Differences in algorithm rankings reflect the differences between the F1-scores and SHD, with F1-scores balancing precision and recall, while SHD puts a stronger penalty on false positives. As a result, these evaluation metrics reflect different objectives, with F1-scores recovering as many true edges as possible and SHD limiting false positives.

5.1.3. *How do different types of prior knowledge impact the performance of structure learning algorithms in high-dimensional settings?* Prior knowledge can change the algorithms' performance substantially, but the magnitude of performance increase is highly dependent on the type of prior knowledge used and the setting. The benchmark blacklist produces the best performance increase, especially for the score-based algorithms. Score-based algorithms perform the best for almost all settings with the benchmark blacklist. One limitation of the score-based algorithms without a prior is their SHD, which, compared to other algorithms is extremely high especially in dense settings. This indicates that score-based algorithms produce a lot of false positives, which is amplified when there are a lot of edges present. A blacklist reduces the likelihood of false positives and therefore improves the performance of score-based algorithms substantially. The largest increase in performance is observed in the SHD dense settings, where the likelihood of false positives is highlighted the most. Besides score-based algorithms, PC.stable also increases in performance substantially, often taking the second or, in some cases, the first rank. The other algorithms do not improve nearly as well from the blacklist, and Iamb.fdr shows no increase in performance.

In contrast, the estimated blacklists from Rags2Ridges and Glasso do not achieve the same performance increase as the benchmark blacklist for F1-scores. Their impact is more inconsistent across settings, fluctuating between moderate gains and small losses in performance. Improvements are again more apparent for the score-based algorithms in dense settings, and PC.stable in the sparse settings. The repository settings fluctuate the most, showing small improvements and declines in performance across and within settings. Rags2Ridges and Glasso produce similar results overall. The performance gap to the benchmark blacklist indicates that the estimated blacklists are limited by the accuracy of the underlying GGM estimates. The sensitivity analysis supports this since adjusting the hyperparameters to reduce errors yields only minor changes. This suggests that the gap to the benchmark blacklist is primarily driven by estimation error. However, SHD shows a substantial improvement for score-based algorithms with the estimated blacklists, most likely due to a reduction in false positive edges. These SHD improvements for score-based algorithms differ between the methods, depending on the sample size. Notably, Rags2Ridges improves SHD with a higher sample size ($N = 150$), whereas Glasso improves SHD at a lower sample size ($N = 20$). Importantly, this difference is only observed for SHD and only for score-based methods; it is not apparent for other evaluation metrics (F1-skeleton, F1-arrow) or for other structure learning algorithms. This indicates that SHD is particularly sensitive to the removal of extra edges in dense estimated DAGs. As score-based algorithms tend to include many false positives, removing a lot of edges can strongly improve SHD, even if this effect is not reflected in other evaluation metrics. This suggests that the apparent performance differences between Rags2Ridges and Glasso are not due to generally better estimated blacklists, but rather due to how strongly SHD rewards the removal of extra edges (more conservative DAGs). At low sample sizes Glasso removes many edges, while at higher sample sizes Rags2Ridges can more selectively remove false positives, leading to improved SHD.

Whitelists show a consistent improvement in performance for all algorithms. These performance gains scale with the proportion of correct edges provided by the whitelist. Moreover, a whitelist has stronger effects when data is limited. The improvements in F1-skeleton scores are marginal, especially at high N. This suggests that a whitelist does not improve the ability of the algorithms to recover the remaining true edges. However, for F1-arrow scores in dense settings, we observe that providing a whitelist marginally improves the algorithms' ability to correctly orient additional edges. Furthermore, whitelists yield more improvements in F1-scores compared to SHD, where the change is modest. This suggests that a whitelist does not reduce the ability of the algorithms to produce fewer false positives. A 25% whitelist shows a similar increase in performance as the benchmark blacklist for F1-scores. Overall, the use of a whitelist functions primarily as an additive improvement, rather than improving the algorithms' learning ability.

Combining both a benchmark blacklist and whitelist produces similar rankings compared to only the blacklist, with absolute scores improving slightly. This again indicates that a whitelist is an additive component, and if the performance of an algorithm is already high, the whitelist will only provide a small marginal extra gain. Thus, there appears to be no synergy in learning ability.

5.1.4. *How does the use of generated DAGs versus repository benchmark DAGs impact the performance of the structure learning algorithms and the external validity of the results?* The repository DAGs have similar algorithm orderings to the sparse DAGs. This can be explained by their comparable density, as both have similar ratios between P and E. This further suggests that density is a key determinant in algorithm ordering and, therefore, a deciding factor in the choice of algorithm for learning a DAG. Furthermore, there is a substantial performance gap between the generated DAGs and the repository DAGs. The repository DAGs have considerably lower F1-scores and slightly worse SHD compared to the sparse DAGs. This suggests that the repository DAGs are harder to learn, likely due to differences in structure and parametric complexity. Importantly, repository DAGs may capture structural differences that are difficult to reproduce with random generated DAGs. This suggests that benchmark results based solely on generated DAGs may be overly optimistic when translated to applied biomedical settings, given the consistent performance gap observed. Consequently, using repository DAGs more prominently would likely yield more realistic expectations of algorithm performance in practice and comparability between studies.

5.1.5. *How does a researcher implement the results of this study in practical research?* This research question is addressed in the illustration study, as shown in Chapter 4. The results of this study emphasized the importance of selecting the best performing algorithm based on the density of the data, sample size, and the availability of prior knowledge. We demonstrated in the illustration how these results can be used practically for causal structure learning on a real-world high-dimensional biomedical dataset.

Based on the results, for prior knowledge we chose an estimated blacklist with Rags2Ridges with local FDR-cut set to 0.01 (see Figures 6, 7, A39 and A40). We chose this since the results show an increase in both F1-scores compared to no prior knowledge and a substantially lower SHD; this lessens the likelihood of false positives in the DAG (while not being too conservative since F1-scores also improved). When examining the results, the algorithm choice was Tabu search since this was the best performing algorithm according to F1-scores in these dense settings and because causal inference was the primary objective. Furthermore, by applying bootstrapping and setting a high threshold for retaining an edge, we improved the likelihood that an edge was correct and reduced the likelihood that the resulting DAG contained false positives. This reduced the structure of the DAG to a much smaller DAG, which was more suitable for causal inference as well.

The resulting DAGs were two different metabolite networks for patients with and without a predisposition to AD. These were then displayed in a clear graphical representation so that causal inference can be made. The resulting networks show clear structural differences between the two groups. This suggests that the learned DAGs capture distinct dependency structures across the metabolic pathways.

5.2. **Strengths and limitations.** The controlled simulation design allowed the effects of sample size, number of vertices, DAG structure (sparse, dense, and repository), algorithm type, and prior knowledge to

be evaluated in relation to each other. Therefore, this study provides a broad comparison and evaluation of structure learning algorithm performance in high-dimensional settings. The focus on causality in high-dimensional data is aligned with the study gap and challenges that are encountered in biomedical research. The illustration study demonstrates how the findings can be used to guide analysis in a real-world biomedical setting.

However, the generalization of the results is restricted to the study settings. All simulations are based on continuous data and assumptions of linearity, Gaussianity, and faithfulness. Moreover, the study does not incorporate latent confounding. This choice was made to ensure comparability between the generated DAGs and the repository DAGs, which are fully observed and do not model latent confounding. Additionally, modeling latent confounding would increase methodological complexity; given the number of factors already considered, it was therefore not included. Lastly, some structure learning algorithms were left out because it was computationally infeasible to include too many.

5.3. **Future directions.** Future work could further advance the understanding of the identified research gap in high-dimensional settings. For example, settings with latent confounding could be modeled by adding unobserved variables or by introducing hidden common causes. This would allow evaluation of structure learning methods under even more realistic biomedical scenarios. Furthermore, larger networks with hundreds to thousands of vertices could be explored as the availability of biomedical data continues to increase. Additionally, future benchmarking studies should include repository benchmark DAGs more prominently, as they may better reflect real-world scenario's. This would improve external validity and reduce the risk of overly optimistic conclusions based solely on generated DAGs. Moreover, additional structure learning algorithms that were not included in this simulation study could be evaluated. Another important extension could be the exploration of nonlinear, non-Gaussian, and mixed data in high-dimensional settings.

Prior knowledge could be further investigated in these settings and in DAG research more broadly as many aspects are currently unknown. For example, there are other ways to restrict the search space by constructing a blacklist, such as the omics cascade, which allows causal direction to flow only one way. Lastly, the benchmark blacklist shows great promise in improving score-based algorithms. Improvements in the accuracy of constructing an estimated blacklist from a GGM could be highly beneficial for enhancing the performance of score-based algorithms.

6. Conclusion

The main aim of this study was to evaluate the performance of structure learning algorithms in various high-dimensional settings and to provide practical guidance for their application in biomedical research. A simulation study was conducted, complemented by an illustration study with Alzheimer's disease metabolomics data to support this aim.

Overall, the algorithms' performance was driven by the interplay of sample size N, network density/-complexity (edges relative to vertices), and the graph size (E and P proportionally). In contrast, defining settings as high- or low-dimensional solely via $P > N$ versus $N > P$ did not reveal a clear threshold effect or explain differences in performance. This is because $P > N$ does not account for E. The algorithms are limited by local complexity because a dense graph increases the size of conditioning/regression problems faced during conditional independence testing (constraint-based and hybrid algorithms) and parent-set selection (score-based algorithms). Thus, high-dimensionality in DAG structure learning could be better interpreted through the interplay of sample size, graph size, and graph complexity (how E scales with P) than by $P > N$ alone.

In dense settings, score-based algorithms performed best according to their F1-scores, whereas in sparse and repository settings hybrid algorithms along with GS and PC.stable performed best. The findings suggest that no single algorithm is optimal and that algorithm selection should be based on sample size and expected network density.

Prior knowledge substantially improved performance, but its effectiveness depends on the type of prior knowledge and setting. A benchmark blacklist derived from the structural equation model of the DAG yielded the largest improvements in performance, especially for score-based algorithms. This was due to the substantial reduction of false positives, indicated by the large SHD performance increase. Estimated blacklists derived from Rags2Ridges and Glasso resulted in inconsistent improvements in F1-scores. Gains are mainly for score-based algorithms in dense settings, and PC.stable in the sparse settings, while repository DAGs' performance fluctuates the most. Although Rags2Ridges slightly outperforms Glasso, the performance gap to the benchmark blacklist indicates that the estimated blacklists are constrained by the accuracy of the underlying GGM estimates. Whitelists show a consistent and equal improvement in performance for all algorithms. Combining both a benchmark blacklist and whitelist improves performance further, however it produces similar rankings compared to only the blacklist.

Repository DAGs have similar characteristics to sparse settings, so the same algorithms perform well in both settings. However, there is a substantial performance gap in F1-scores between the generated DAGs and the repository DAGs.

Finally, the illustration study demonstrated how the findings from this study can be used to guide analysis of a real-world high-dimensional biomedical dataset. By combining an estimated blacklist, the Tabu search algorithm (based on the expectation of a dense network), and bootstrap-based filtering to retain likely edges, the DAGs became interpretable for exploratory causal analysis.

7. Declaration of AI usage

During the preparation of this work, the author used ChatGPT (version 5) in order to improve grammar, spelling, fluency, and to assist with R code. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the report.

References

[1] D. Hume, "An Enquiry concerning Human Understanding," in *Oxford University Press eBooks*, 1748, pp. 134–198. doi: 10.1093/oseo/instance.00032980.

[2] N. Stein, *Causality and causal explanation in Aristotle*. Oxford University Press, 2023.

[3] T. L. Kvamme, T. Ros, and M. Overgaard, "Can neurofeedback provide evidence of direct brain-behavior causality?," *NeuroImage*, vol. 258, p. 119400, Jun. 2022, doi: 10.1016/j.neuroimage.2022.119400.

[4] M. A. Hernán, J. Hsu, and B. Healy, "A second chance to get causal inference right: a classification of data science tasks," *CHANCE*, vol. 32, no. 1, pp. 42–49, Jan. 2019, doi: 10.1080/09332480.2019.1579578.

[5] N. Black, "Why we need observational studies to evaluate the effectiveness of health care," *BMJ*, vol. 312, no. 7040, pp. 1215–1218, May 1996, doi: 10.1136/bmj.312.7040.1215.

[6] S. Bhattacharyya, A. E. Hassanien, D. Gupta, A. Khanna, and I. Pan, *International Conference on Innovative Computing and Communications*. Springer, 2018, pp. 261–269. doi: 10.1007/978-981-13-2354-6.

[7] N. Vervoort, K. Goossens, M. Baeten, and Q. Chen, "Recent advances in analytical techniques for high throughput experimentation," *Analytical Science Advances*, vol. 2, no. 3–4, pp. 109–127, Jan. 2021, doi: 10.1002/ansa.202000155.

[8] K. R. Moon *et al.*, "Visualizing structure and transitions in high-dimensional biological data," *Nature Biotechnology*, vol. 37, no. 12, pp. 1482–1492, Dec. 2019, doi: 10.1038/s41587-019-0336-3.

[9] C. Giraud, Introduction to High-Dimensional Statistics, 2nd ed. New York, United States of America, 2021. doi: 10.1201/9781003158745.

[10] J. Pearl, *Causality*. 2009. doi: 10.1017/cbo9780511803161.

[11] N. Hartsfield and G. Ringel, Pearls in graph Theory: A Comprehensive Introduction. Courier Corporation, 2003.

[12] F. Dablander, "An Introduction to Causal Inference," PsyArXiv, DOI: 10.31234/osf.io/b3fkw, 2020. [Online]. Available: https://doi.org/10.31234/osf.io/b3fkw

[13] N. K. Kitson, A. C. Constantinou, Z. Guo, Y. Liu, and K. Chobtham, "A survey of Bayesian Network structure learning," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8721–8814, Jan. 2023, doi: 10.1007/s10462-022-10351-w.

[14] M. Scutari, "Learning Bayesian Networks with thebnlearnRPackage," *Journal of Statistical Software*, vol. 35, no. 3, Jan. 2010, doi: 10.18637/jss.v035.i03.

[15] Z. Fang, S. Zhu, J. Zhang, Y. Liu, Z. Chen, and Y. He, "On Low-Rank directed acyclic graphs and causal structure learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4924–4937, May 2023, doi: 10.1109/tnnls.2023.3273353.

[16] N. B. Aragam, "Structure learning of Linear Bayesian Networks in High-Dimensions," 2015. https://escholarship.org/uc/item/9gs5787w

[17] L. Farnia, M. Alibegovic, and E. Cruickshank, "On causal structural learning algorithms: Oracles' simulations and considerations," *Knowledge-Based Systems*, vol. 276, p. 110694, Jun. 2023, doi: 10.1016/j.knosys.2023.110694.

[18] L. Zhang, L. O. Rodrigues, N. R. Narain, and V. R. Akmaev, "BAICIS: A novel Bayesian Network Structural Learning Algorithm and its comprehensive performance Evaluation against Open-Source software," *Journal of Computational Biology*, vol. 27, no. 5, pp. 698–708, Sep. 2019, doi: 10.1089/cmb.2019.0210.

[19] M. Scutari, C. E. Graafland, and J. M. Gutiérrez, "Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms," *International Journal of Approximate Reasoning*, vol. 115, pp. 235–253, Oct. 2019, doi: 10.1016/j.ijar.2019.10.003.

[20] A. C. Constantinou, Y. Liu, K. Chobtham, Z. Guo, and N. K. Kitson, "Large-scale empirical validation of Bayesian Network structure learning algorithms with noisy data," *International Journal of Approximate Reasoning*, vol. 131, pp. 151–188, Jan. 2021, doi: 10.1016/j.ijar.2021.01.001.

[21] P. Rütimann and P. Bühlmann, "high-dimensional sparse covariance estimation via directed acyclic graphs," *Electronic Journal of Statistics*, vol. 3, no. none, Jan. 2009, doi: 10.1214/09-ejs534.

[22] M. J. Ha and W. Sun, "Estimation of high-dimensional directed acyclic graphs with surrogate intervention," *Biostatistics*, vol. 21, no. 4, pp. 659–675, Nov. 2018, doi: 10.1093/biostatistics/kxy080.

[23] M. H. Maathuis, M. Kalisch, and P. Bühlmann, "Estimating high-dimensional intervention effects from observational data," *The Annals of Statistics*, vol. 37, no. 6A, Aug. 2009, doi: 10.1214/09-aos685.

[24] Chris. J. Oates, J. Q. Smith, and S. Mukherjee, "Estimating causal structure using conditional DAG models," *arXiv (Cornell University)*, Jan. 2014, doi: 10.48550/arxiv.1411.2755.

[25] M. Kalisch and P. Bühlmann, "Estimating high-dimensional directed acyclic graphs with the PC-algorithm," *arXiv (Cornell University)*, Jan. 2005, doi: 10.48550/arxiv.math/0510436.

[26] M. Gasse, A. Aussem, and H. Elghazel, "A hybrid algorithm for Bayesian network structure learning with application to multi-label learning," *Expert Systems With Applications*, vol. 41, no. 15, pp. 6755–6772, May 2014, doi: 10.1016/j.eswa.2014.04.032.

[27] C. Sharma and P. Van Beek, "Scalable Bayesian Network Structure Learning with Splines," *PMLR*, Sep. 19, 2022. https://proceedings.mlr.press/v186/sharma22a.html

[28] W. Liu, M. A. Brookhart, S. Schneeweiss, X. Mi, and S. Setoguchi, "Implications of M bias in epidemiologic studies: a simulation study," *American Journal of Epidemiology*, vol. 176, no. 10, pp. 938–948, Oct. 2012, doi: 10.1093/aje/kws165.

[29] A. C. Constantinou, Z. Guo, and N. K. Kitson, "The impact of prior knowledge on causal structure learning," *Knowledge and Information Systems*, vol. 65, no. 8, pp. 3385–3434, Apr. 2023, doi: 10.1007/s10115-023-01858-x.

[30] J. S. Ide and F. G. Cozman, "Random generation of Bayesian networks," in *Lecture notes in computer science*, 2002, pp. 366–376. doi: 10.1007/3-540-36127-8_35.

[31] N. Cruz-Ramírez *et al.*, "How good is crude MDL for solving the Bias-Variance dilemma? An empirical investigation based on Bayesian networks," *PLoS ONE*, vol. 9, no. 3, p. e92866, Mar. 2014, doi: 10.1371/journal.pone.0092866.

[32] "Bnlearn - Bayesian Network Repository." https://www.bnlearn.com/bnrepository/

[33] "Bayesian Network Repository." https://www.cs.huji.ac.il/w̃galel/Repository/

[34] M. Leonelli, "A Repository of Bayesian Networks from the Academic Literature [R package bnRep version 0.0.5]," Jul. 23, 2025. https://cran.r-project.org/web/packages/bnRep/index.html

[35] A. C. Constantinou *et al.*, "The Bayesys data and Bayesian network repository," 2020. [Online]. Available: http://constantinou.info/downloads/bayesys/bayesys_repository.pdf

[36] C. E. Graafland and J. M. Gutiérrez, "Learning complex dependency structure of gene regulatory networks from high-dimensional microarray data with Gaussian Bayesian networks," *Scientific Reports*, vol. 12, no. 1, Nov. 2022, doi: 10.1038/s41598-022-21957-z.

[37] J. Rahnenführer *et al.*, "Statistical analysis of high-dimensional biomedical data: a gentle introduction to analytical goals, common approaches and challenges," *BMC Medicine*, vol. 21, no. 1, May 2023, doi: 10.1186/s12916-023-02858-y.

[38] Z. J. Kunicki, M. L. Smith, and E. J. Murray, "A primer on structural equation model diagrams and directed acyclic graphs: when and how to use each in psychological and epidemiological research," *Advances in Methods and Practices in Psychological Science*, vol. 6, no. 2, p. 251524592311560, Apr. 2023, doi: 10.1177/25152459231156085.

[39] M. Kalisch and P. Bühlmann, "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm," Mar. 2007. [Online]. Available: https://www.jmlr.org/papers/volume8/kalisch07a/kalisch07a.pdf

[40] D. Colombo, M. H. Maathuis, Seminar for Statistics, and ETH Zurich, "Order-Independent Constraint-Based causal structure learning," Nov. 2014. [Online]. Available: https://jmlr.org/papers/volume15/colombo14a/colombo14a.pdf?utm_

[41] M. Scutari, C. E. Graafland, and J. M. Gutiérrez, "Who learns Better Bayesian Network Structures: Accuracy and speed of structure learning algorithms," *arXiv.org*, May 30, 2018. https://arxiv.org/abs/1805.11908

[42] P. Nandy, A. Hauser, and M. H. Maathuis, "High-dimensional consistency in score-based and hybrid structure learning," *arXiv.org*, Jul. 09, 2015. https://arxiv.org/abs/1507.02608

[43] S. Chakraborty and A. Shojaie, "Nonparametric causal structure learning in high dimensions," *Entropy*, vol. 24, no. 3, p. 351, Feb. 2022, doi: 10.3390/e24030351.

[44] "Causal Structure learning of High-Dimensional data based on local and global collaboration," *IEEE Conference Publication — IEEE Xplore*, Jul. 27, 2024. https://ieeexplore.ieee.org/document/10702301

[45] K. Lagemann, C. Lagemann, B. Taschler, and S. Mukherjee, "Deep learning of causal structures in high dimensions under data limitations," *Nature Machine Intelligence*, vol. 5, no. 11, pp. 1306–1316, Oct. 2023, doi: 10.1038/s42256-023-00744-z.

[46] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence*. 2010. doi: 10.1201/b10391.

[47] J. Schäfer and K. Strimmer, "A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, no. 1, p. Article32, Jan. 2005, doi: 10.2202/1544-6115.1175.

[48] C. F. W. Peeters, *Directed Acyclic Graphs and Causality: Causal Inference I, Lecture 3*, lecture slides, Wageningen University & Research, Biometris, n.d.

[49] C. F. W. Peeters, A. E. Bilgrau, and W. N. Van Wieringen, "rags2ridges: A One-Stop- ℓ_2 -Shop for Graphical Modeling of High-Dimensional Precision Matrices," *Journal of Statistical Software*, vol. 102, no. 4, Jan. 2022, doi: 10.18637/jss.v102.i04.

[50] S. Hermes, J. Van Heerwaarden, and P. Behrouzi, "Copula Graphical models for heterogeneous mixed data," *Journal of Computational and Graphical Statistics*, vol. 33, no. 3, pp. 991–1005, Dec. 2023, doi: 10.1080/10618600.2023.2289545.

[51] L. Augugliaro, G. Sottile, E. C. Wit, and V. Vinciotti, "cglasso: An R Package for Conditional Graphical Lasso Inference with Censored and Missing Values," *Journal of Statistical Software*, vol. 105, no. 1, Jan. 2023, doi: 10.18637/jss.v105.i01.

[52] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, Dec. 2007, doi: 10.1093/biostatistics/kxm045.

[53] T. Zhao, H. Liu, K. Roeder, J. Lafferty, and L. Wasserman, "The huge Package for High-dimensional Undirected Graph Estimation in R," Apr. 01, 2012. https://pmc.ncbi.nlm.nih.gov/articles/PMC4729207/

[54] T. Cai, W. Liu, and X. Luo, "A constraintL1Minimization approach to sparse precision matrix estimation," *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 594–607, Mar. 2011, doi: 10.1198/jasa.2011.tm10155.

[55] R. Opgen-Rhein and K. Strimmer, "From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data," *BMC Systems Biology*, vol. 1, no. 1, Aug. 2007, doi: 10.1186/1752-0509-1-37.

[56] M. J. Ha and W. Sun, "Partial correlation matrix estimation using ridge penalty followed by thresholding and re-estimation," *Biometrics*, vol. 70, no. 3, pp. 762–770, May 2014, doi: 10.1111/biom.12186.

[57] A. E. Seffernick *et al.*, "Bootstrap Evaluation of Association Matrices (BEAM) for Integrating Multiple Omics Profiles with Multiple Outcomes," *bioRxiv (Cold Spring Harbor Laboratory)*, Aug. 2024, doi: 10.1101/2024.07.31.605805.

[58] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, vol. 76, no. 2, pp. 373–397, Aug. 2013, doi: 10.1111/rssb.12033.

[59] C. A. Class, M. J. Ha, V. Baladandayuthapani, and K.-A. Do, "iDINGO—integrative differential network analysis in genomics with Shiny application," *Bioinformatics*, vol. 34, no. 7, pp. 1243–1245, Nov. 2017, doi: 10.1093/bioinformatics/btx750.

[60] R. Nagarajan, M. Scutari, and S. Lèbre, *Bayesian Networks in R*. 2013. doi: 10.1007/978-1-4614-6446-4.

[61] U. Hasan and M. O. Gani, "KCRL: A Prior Knowledge Based Causal Discovery Framework with Reinforcement Learning," *Proceedings of Machine Learning Research*, vol. 182, pp. 1–24, 2022, [Online]. Available: https://proceedings.mlr.press/v182/hasan22a/hasan22a.pdf

[62] S. Liu, Y.-S. Ong, and K. Tang, "Scalable structure learning of Bayesian networks by learning algorithm ensembles," Aug. 2021. [Online]. Available: https://arxiv.org/pdf/2506.22848

[63] D. Margaritis, "Learning Bayesian Network Model Structure from Data," PhD dissertation, Carnegie Mellon University, 2003. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA461103.pdf

[64] J. M. Peña and IFM, Linköping University, "Learning Gaussian Graphical Models of Gene Networks with False Discovery Rate Control," 2007. [Online]. Available: https://www.ida.liu.se/~jospe50/mnfdrevobio2008b.pdf

[65] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. New York, NY, USA: Prentice Hall, 2009.

[66] N. K. Kitson, A. C. Constantinou, Z. Guo, Y. Liu, and K. Chobtham, "A survey of Bayesian Network structure learning," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8721–8814, Jan. 2023, doi: 10.1007/s10462-022-10351-w.

[67] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Machine Learning*, vol. 65, no. 1, pp. 31–78, Mar. 2006, doi: 10.1007/s10994-006-6889-7.

[68] N. Friedman, I. Nachman, and D. Pe'er, "Learning Bayesian Network Structure from Massive Datasets: The Sparse Candidate Algorithm" 1999. https://www.semanticscholar.org/paper/Learning-Bayesian-Network-Structure-from-Massive-Friedman-Nachman/863a5e02d003dfc3bffc2484ae3ff665ba8a21a9

[69] F. A. De Leeuw *et al.*, "Blood-based metabolic signatures in Alzheimer's disease," *Alzheimer S & Dementia Diagnosis Assessment & Disease Monitoring*, vol. 8, no. 1, pp. 196–207, Jan. 2017, doi: 10.1016/j.dadm.2017.07.006.

[70] Y. Yuan, G. Zhao, and Y. Zhao, "Dysregulation of energy metabolism in Alzheimer's disease," *Journal of Neurology*, vol. 272, no. 1, p. 2, Dec. 2024, doi: 10.1007/s00415-024-12800-8.

[71] B. Penke, M. Szűcs, and F. Bogár, "New pathways identify novel drug targets for the prevention and treatment of Alzheimer's disease," *International Journal of Molecular Sciences*, vol. 24, no. 6, p. 5383, Mar. 2023, doi: 10.3390/ijms24065383.

[72] W. M. Van Der Flier *et al.*, "Optimizing patient care and research: The Amsterdam Dementia Cohort," *Journal of Alzheimer's Disease*, vol. 41, no. 1, pp. 313–327, Jun. 2014, doi: 10.3233/jad-132306.

[73] G. Csárdi *et al.*, "igraph: Network Analysis and Visualization." Mar. 01, 2006. doi: 10.32614/cran.package.igraph.

[74] L. G. Yang, Z. M. March, R. A. Stephenson, and P. S. Narayan, "Apolipoprotein E in lipid metabolism and neurodegenerative disease," *Trends in Endocrinology and Metabolism*, vol. 34, no. 8, pp. 430–445, Jun. 2023, doi: 10.1016/j.tem.2023.05.002.

[75] D. A. Butterfield and M. P. Mattson, "Apolipoprotein E and oxidative stress in brain with relevance to Alzheimer's disease," *Neurobiology of Disease*, vol. 138, p. 104795, Feb. 2020, doi: 10.1016/j.nbd.2020.104795.

[76] M. Arnold *et al.*, "Sex and APOE ε4 genotype modify the Alzheimer's disease serum metabolome," *Nature Communications*, vol. 11, no. 1, p. 1148, Mar. 2020, doi: 10.1038/s41467-020-14959-w.

Appendix A. Ground truth DAGs

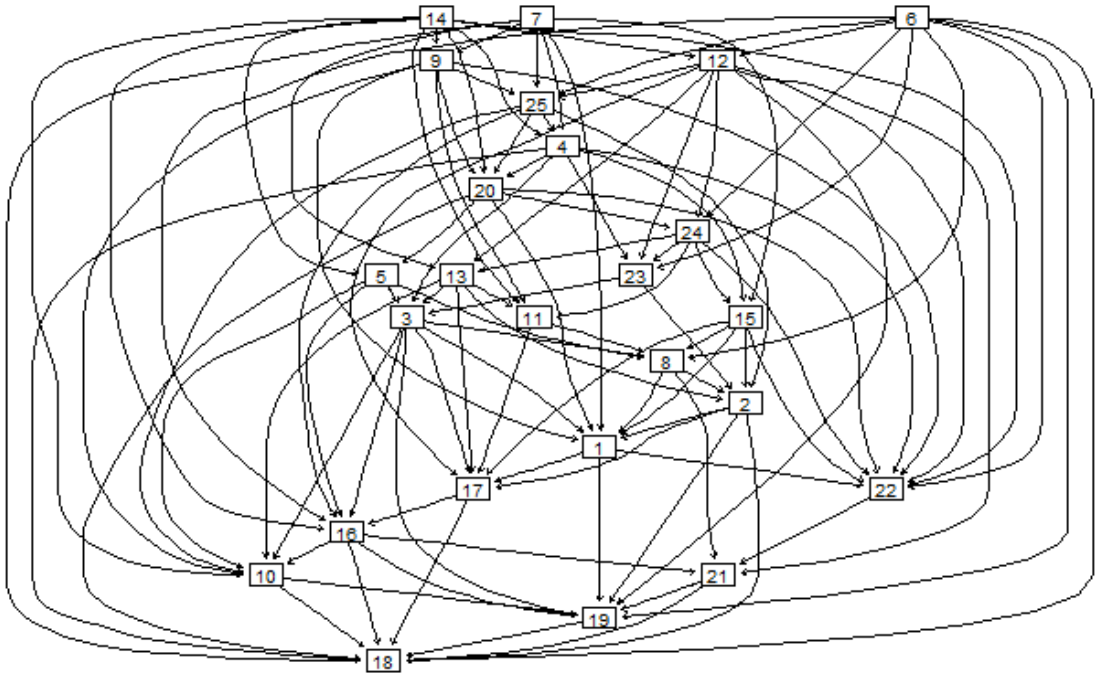


Figure A1. Dense DAG 1 with $P = 25$ vertices and edges = 106.

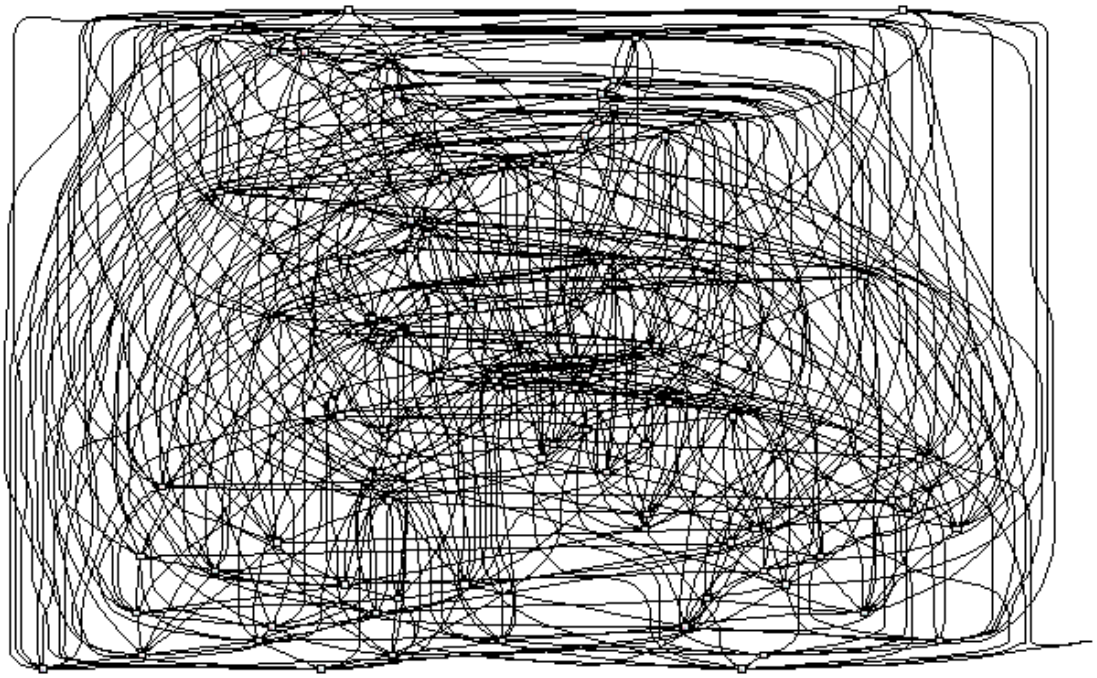


FIGURE A2. Dense DAG 2 with P = 100 vertices and edges = 479.

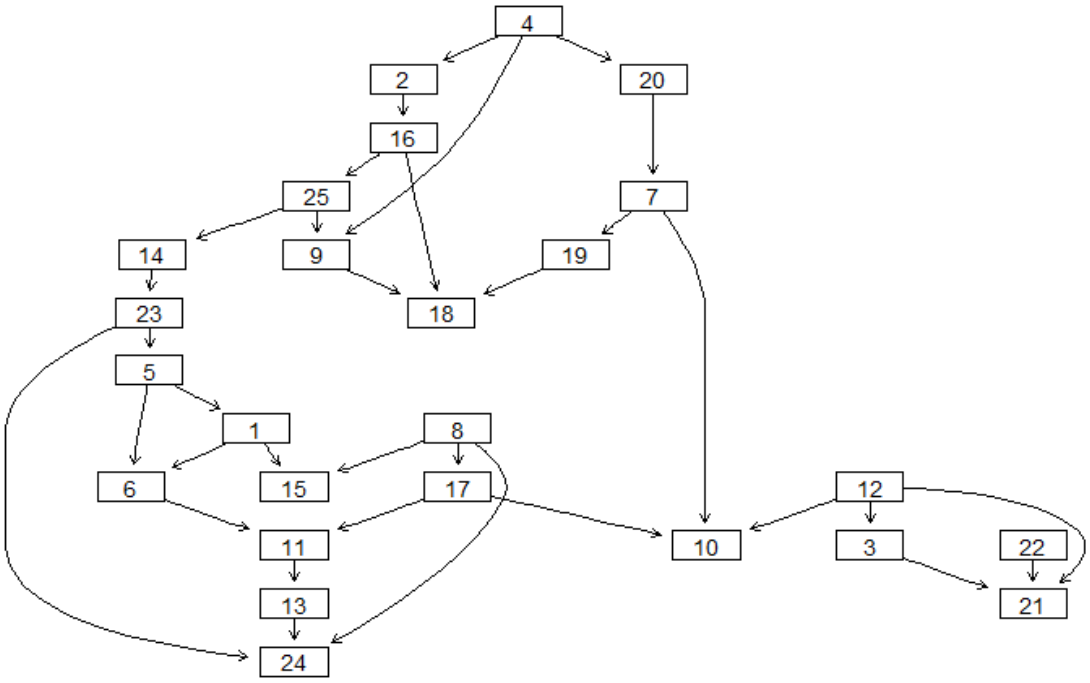


FIGURE A3. Sparse DAG 1 with P = 25 vertices and edges = 33.

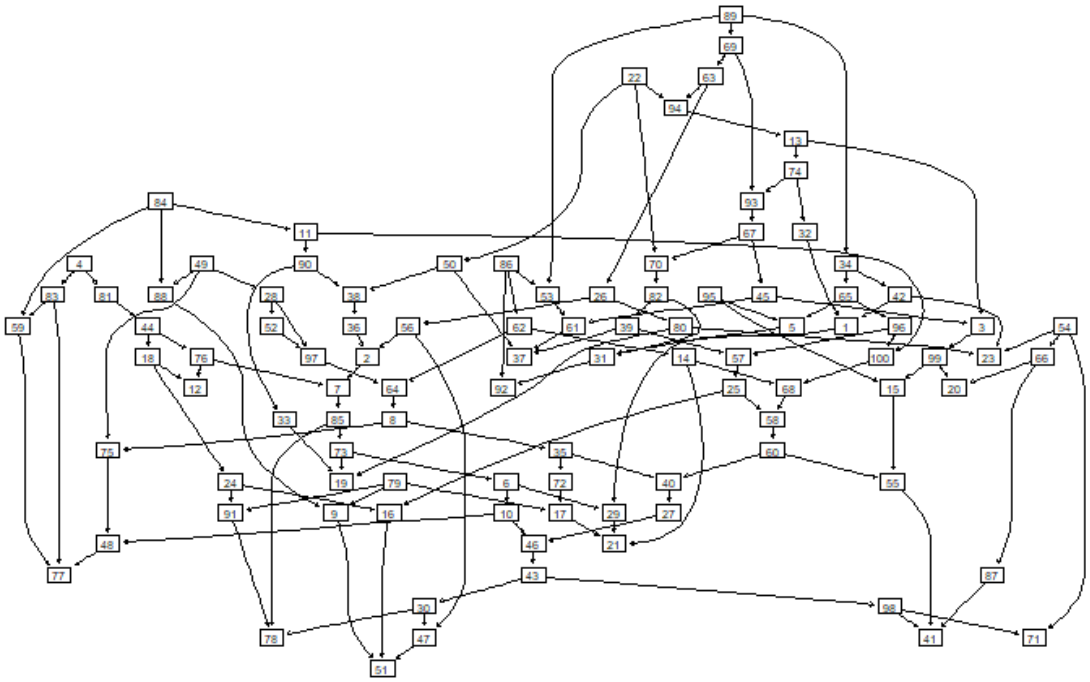


FIGURE A4. Sparse DAG 2 with P = 100 vertices and edges = 143.

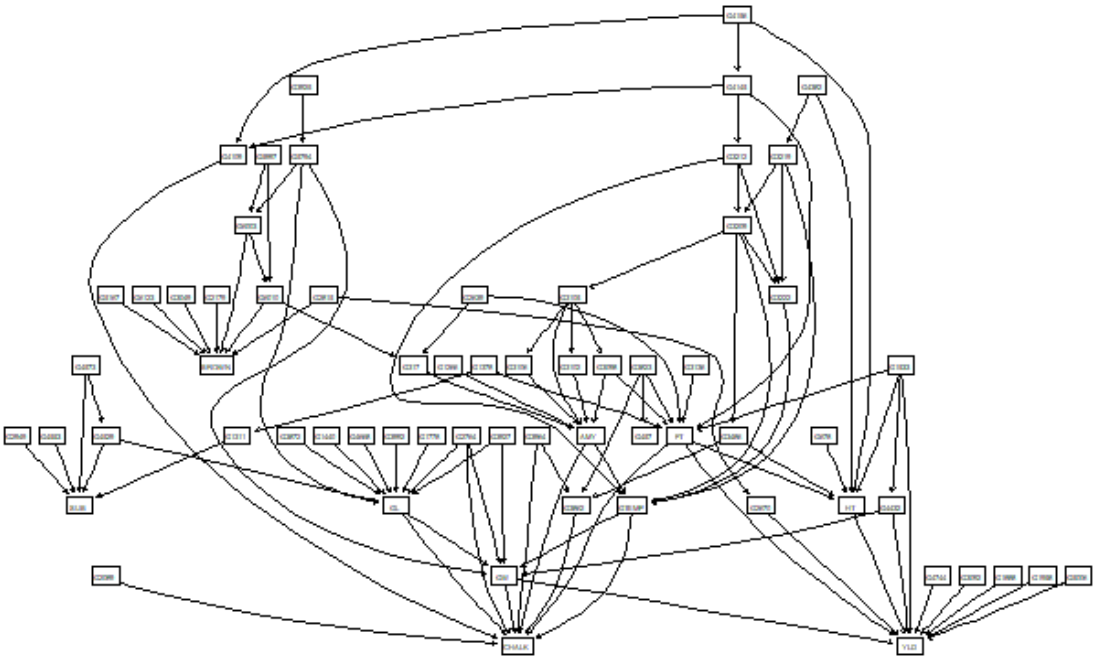


FIGURE A5. MAGIC-IRRI DAG with P = 64 vertices and edges = 102.

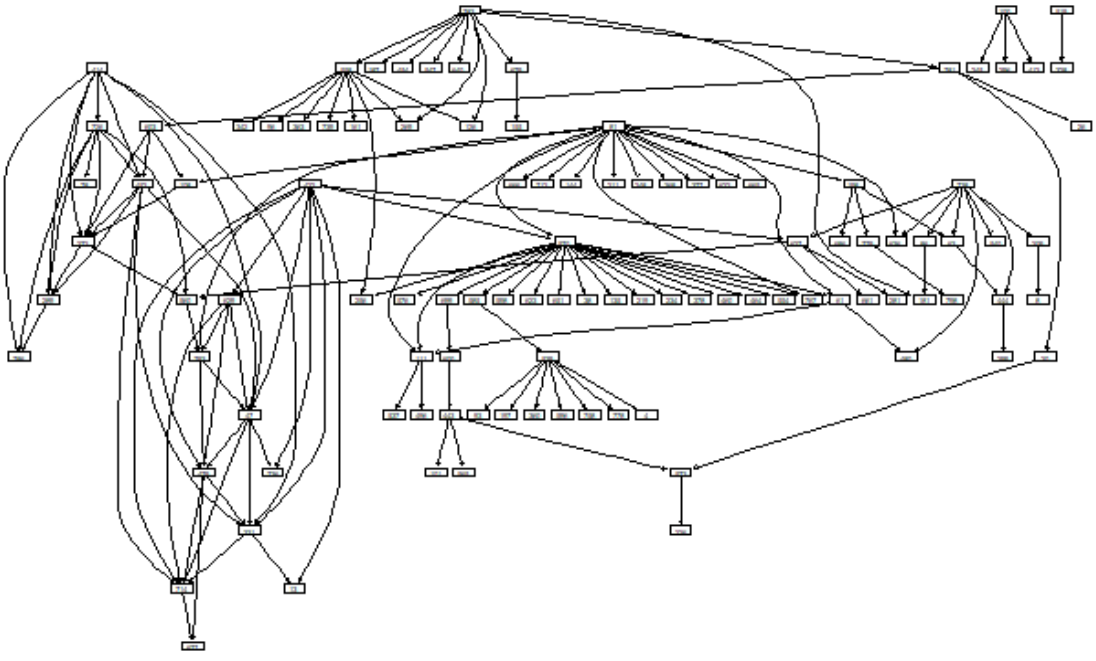


FIGURE A6. ARTH150 DAG with P = 107 vertices and edges = 150.

Appendix B. Results

An HTML document with the R code and output of the analysis can be found at `https://thesesmoos.netlify.app/simulation`.

B.0.1. ***Illustration results***.

Class 1: Mixing matrix (counts).

	Amine	Lip	Ox.Stress
Amine	28	6	3
Lip	6	129	12
Ox.Stress	4	12	26

Class 2: Mixing matrix (counts).

	Amine	Lip	Ox.Stress
Amine	31	2	3
Lip	2	193	27
Ox.Stress	2	7	38

Class 1: Mixing matrix (proportion).

	Amine	Lip	Ox.Stress
Amine	0.124	0.027	0.013
Lip	0.027	0.571	0.053
Ox.Stress	0.018	0.053	0.115

Class 2: Mixing matrix (proportion).

	Amine	Lip	Ox.Stress
Amine	0.102	0.007	0.010
Lip	0.007	0.633	0.089
Ox.Stress	0.007	0.023	0.125

B.0.2. ***Simulation results***.

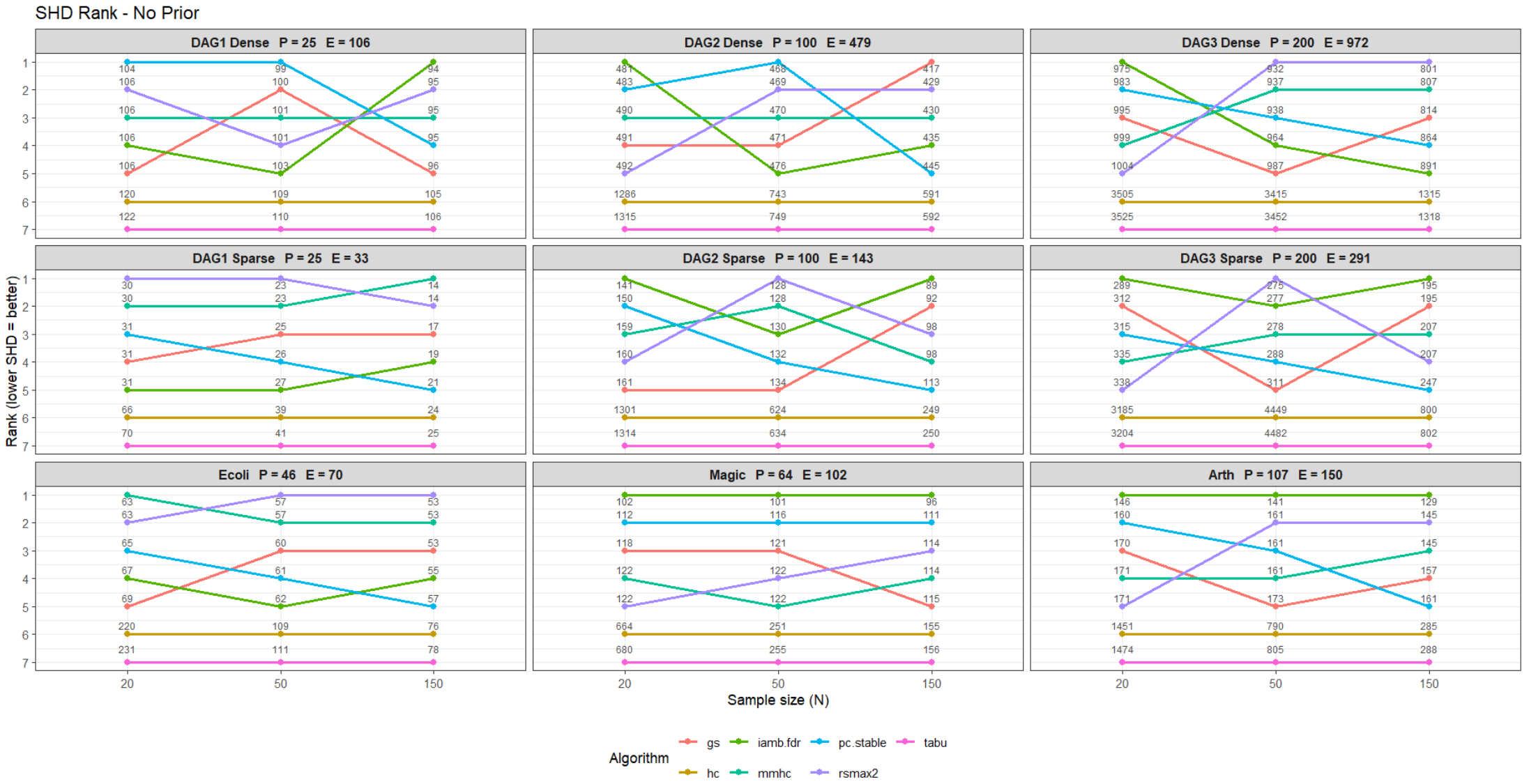


FIGURE A7. SHD rank scores with no prior knowledge.

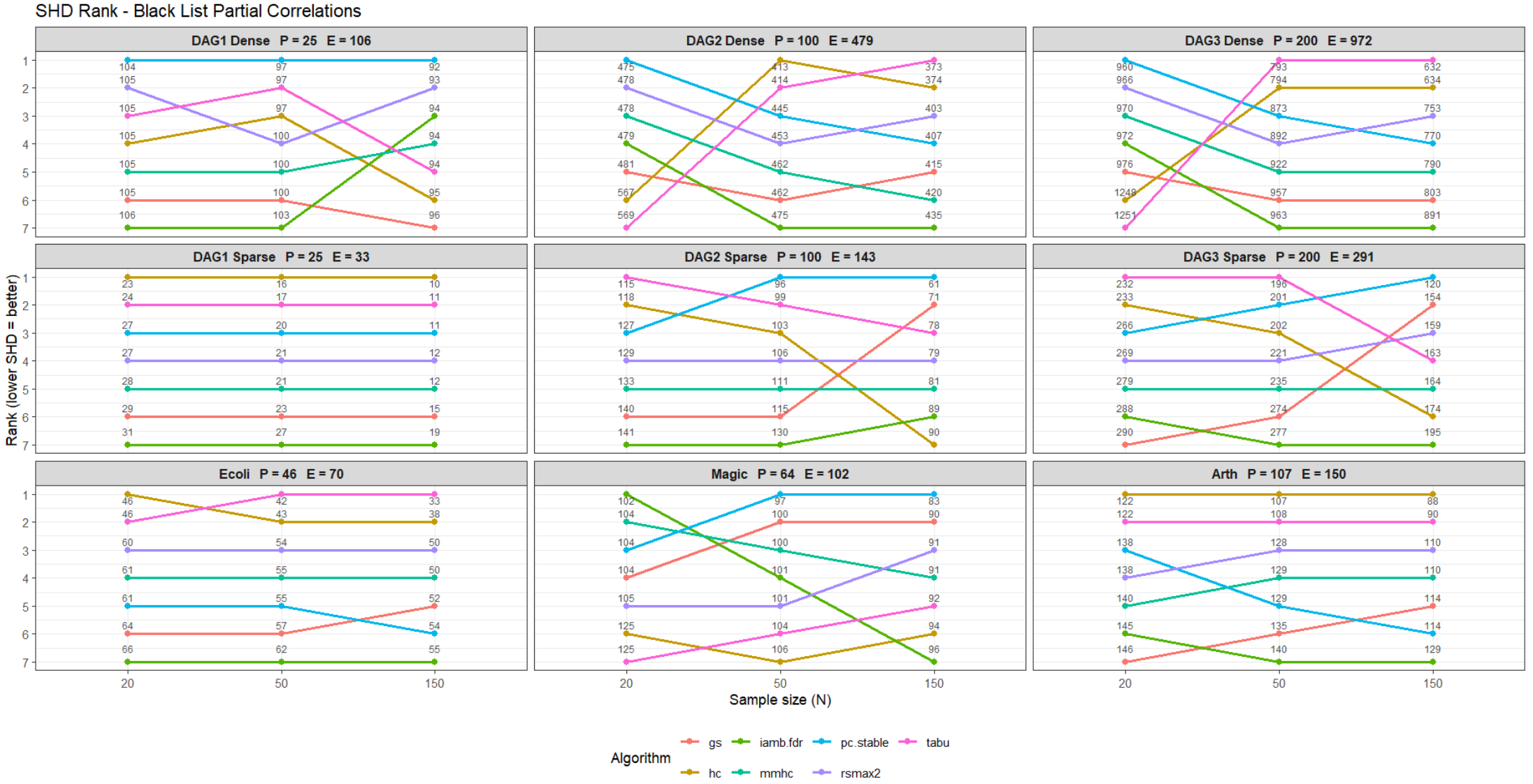


FIGURE A8. SHD rank scores with benchmark blacklist.

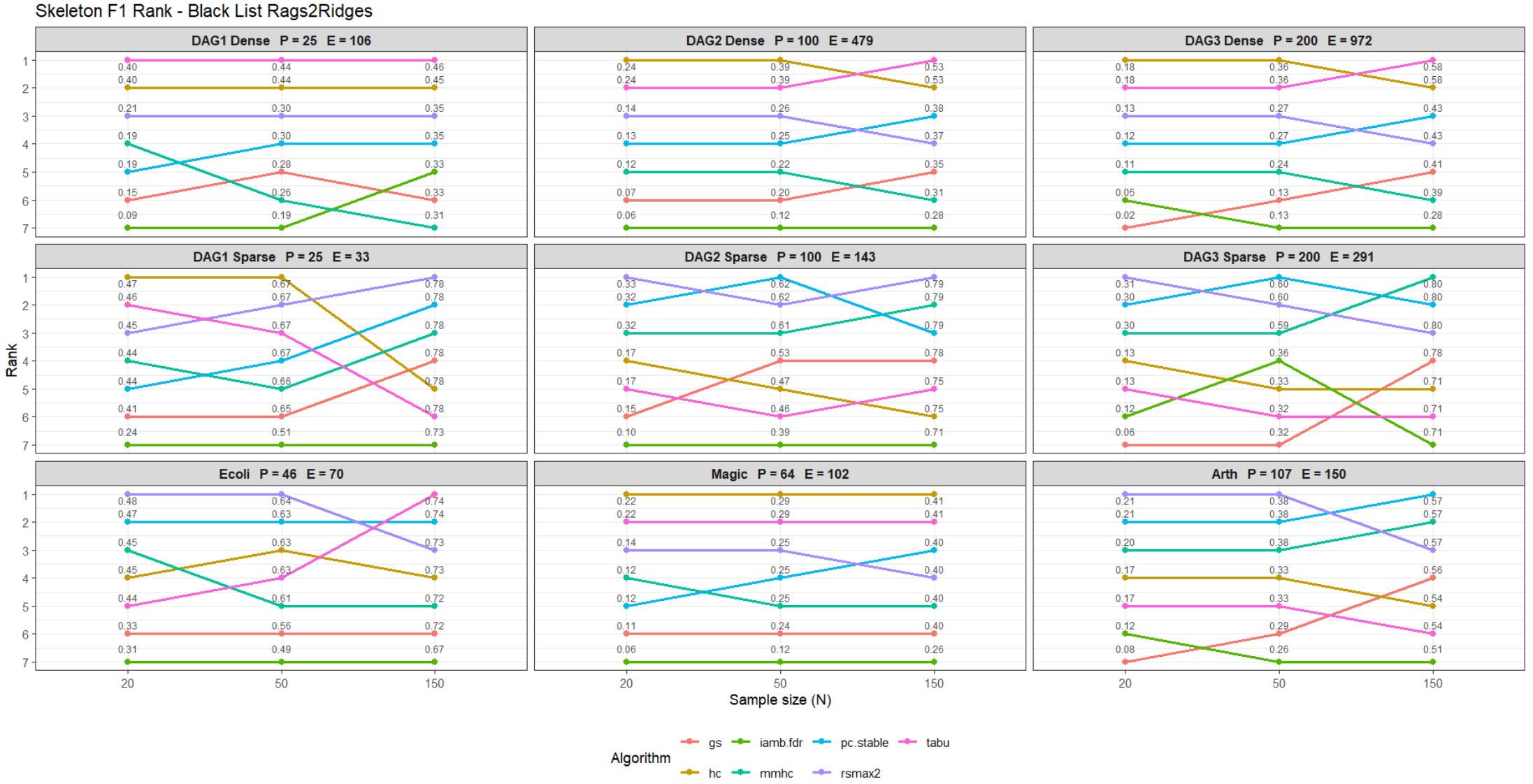


FIGURE A9. F1-skeleton rank scores with Rags2Ridges blacklist.

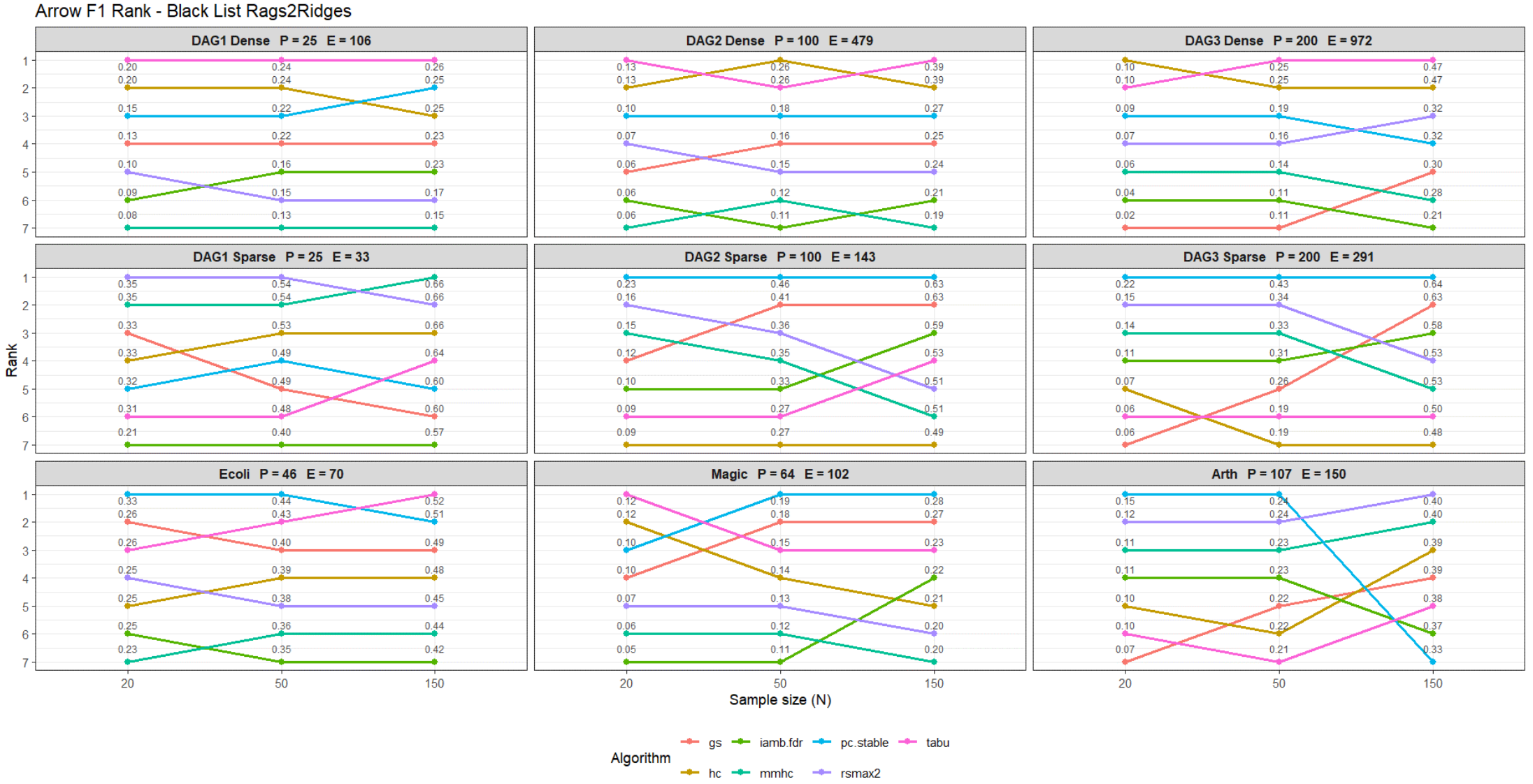


FIGURE A10. F1-arrow rank scores with Rags2Ridges blacklist.

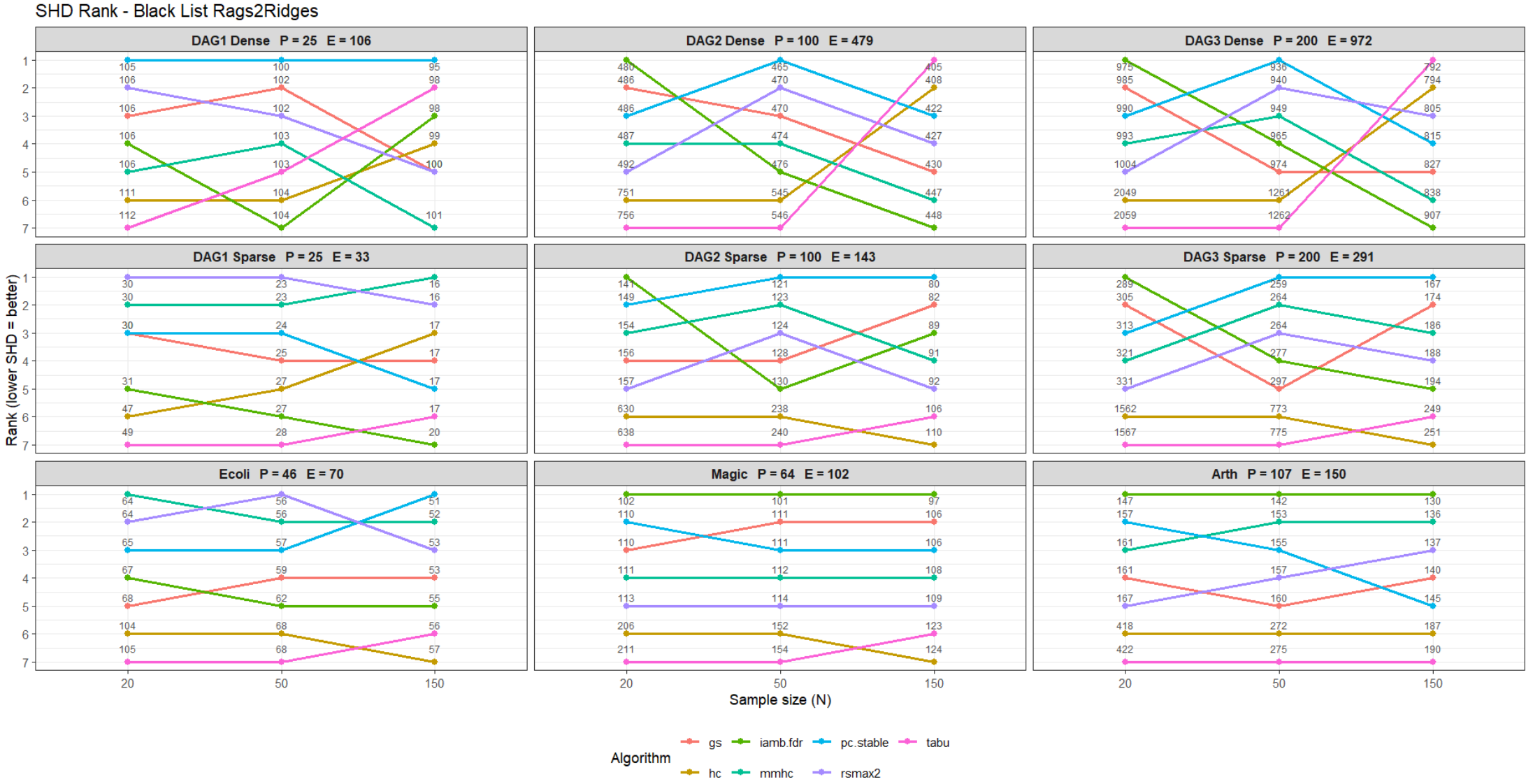


FIGURE A11. SHD rank scores with Rags2Ridges blacklist.

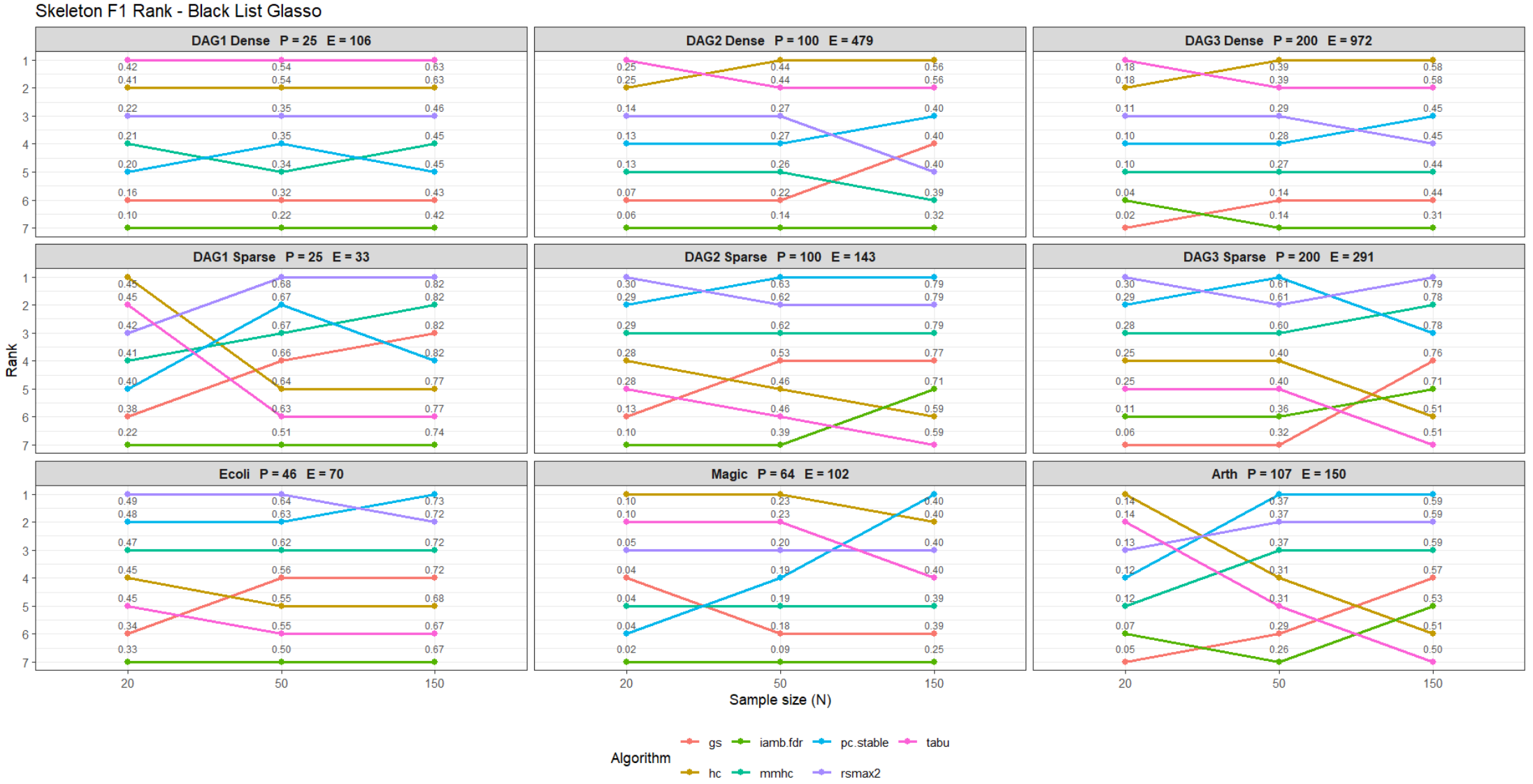


FIGURE A12. F1-skeleton rank scores with Glasso blacklist.

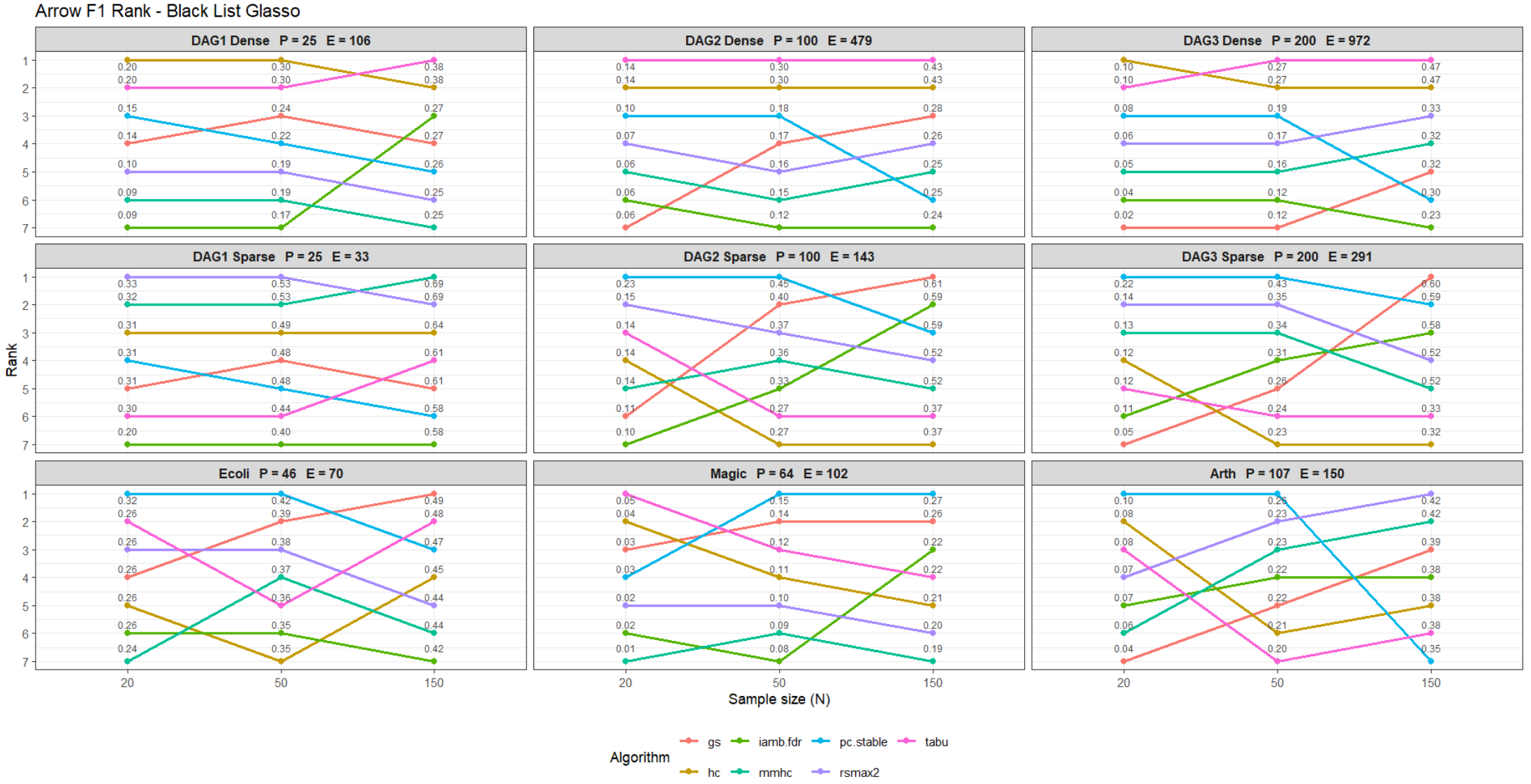


FIGURE A13. F1-arrow rank scores with Glasso blacklist.

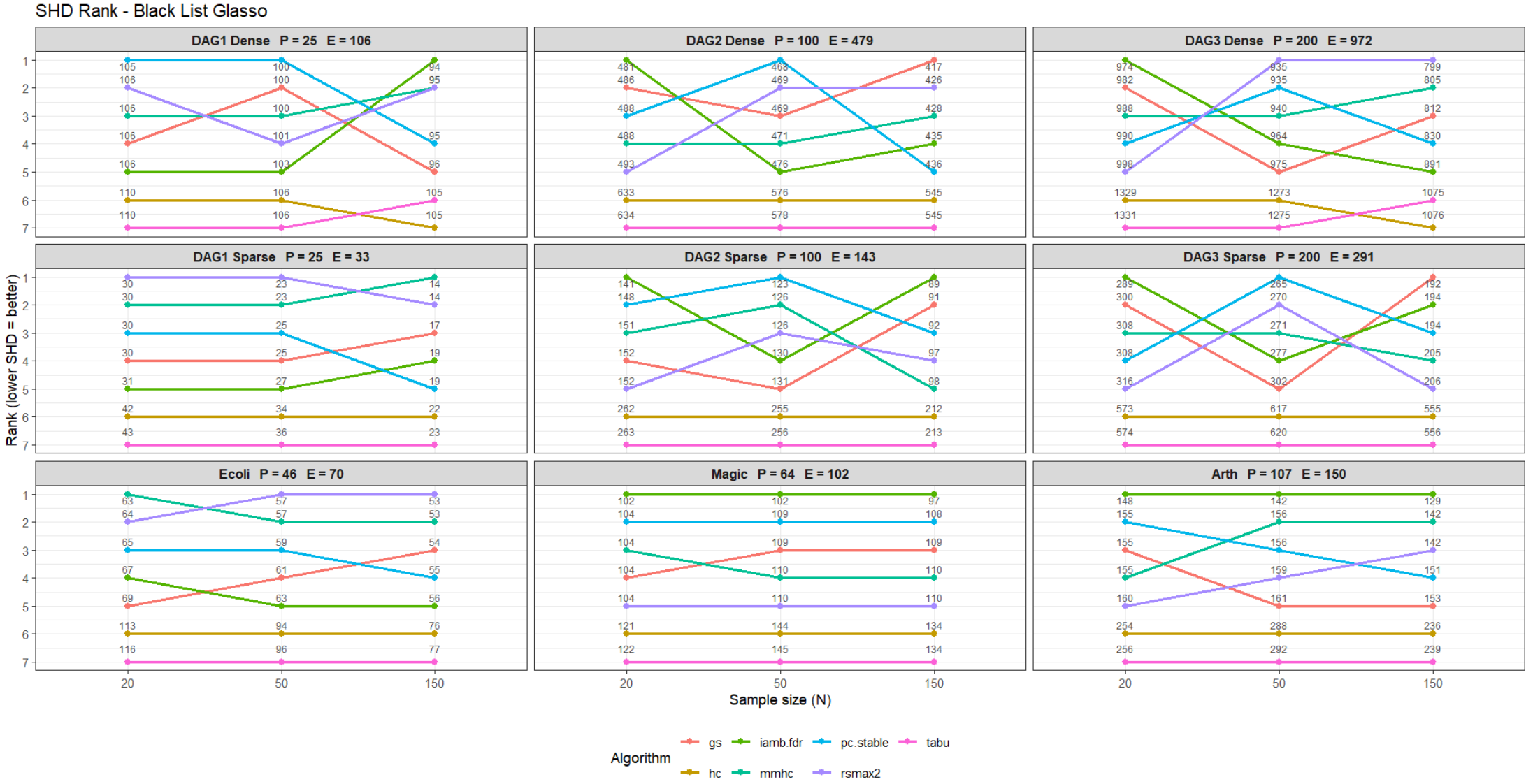


FIGURE A14. SHD rank scores with Glasso blacklist.

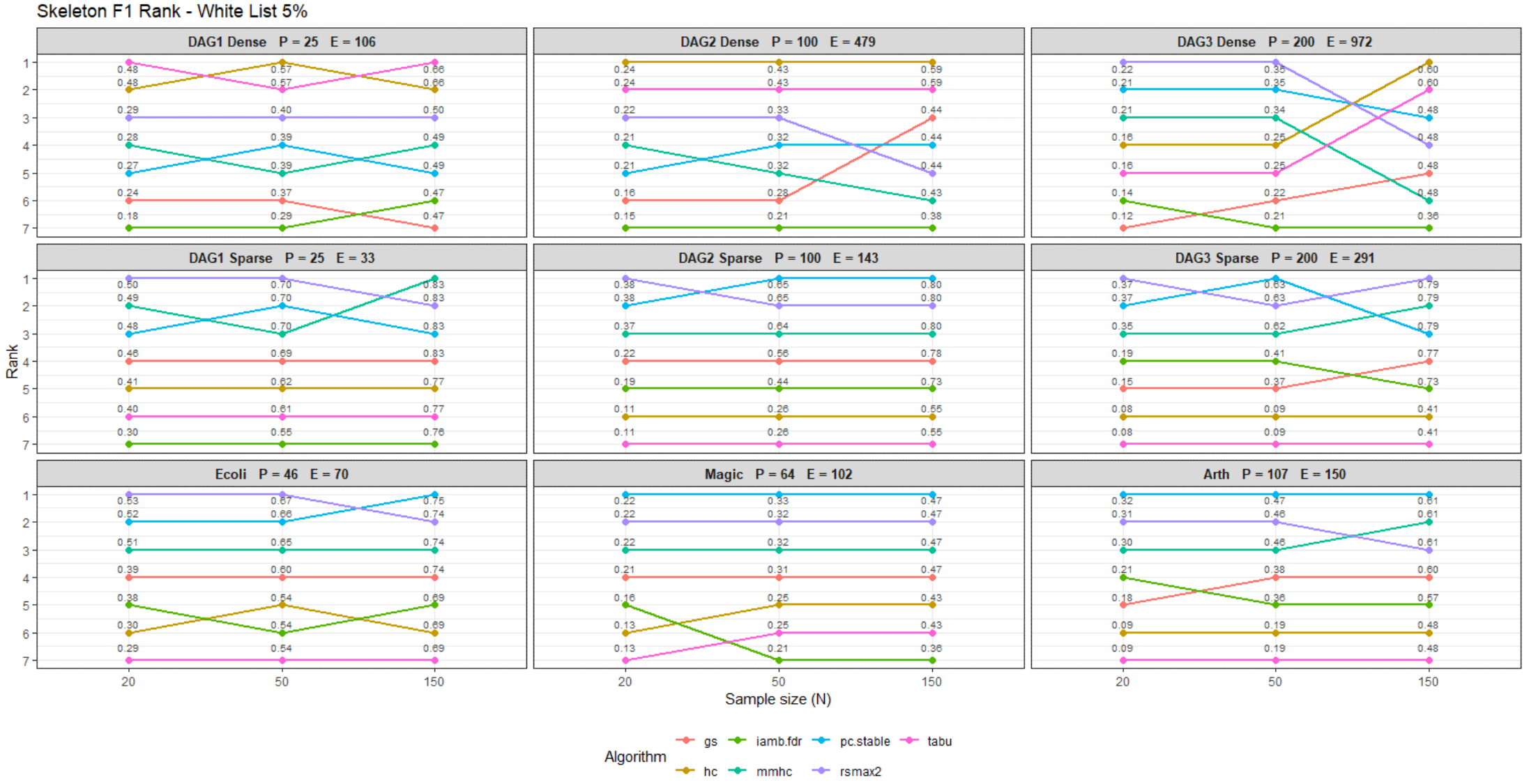


FIGURE A15. F1-skeleton rank scores with whitelist 5%.

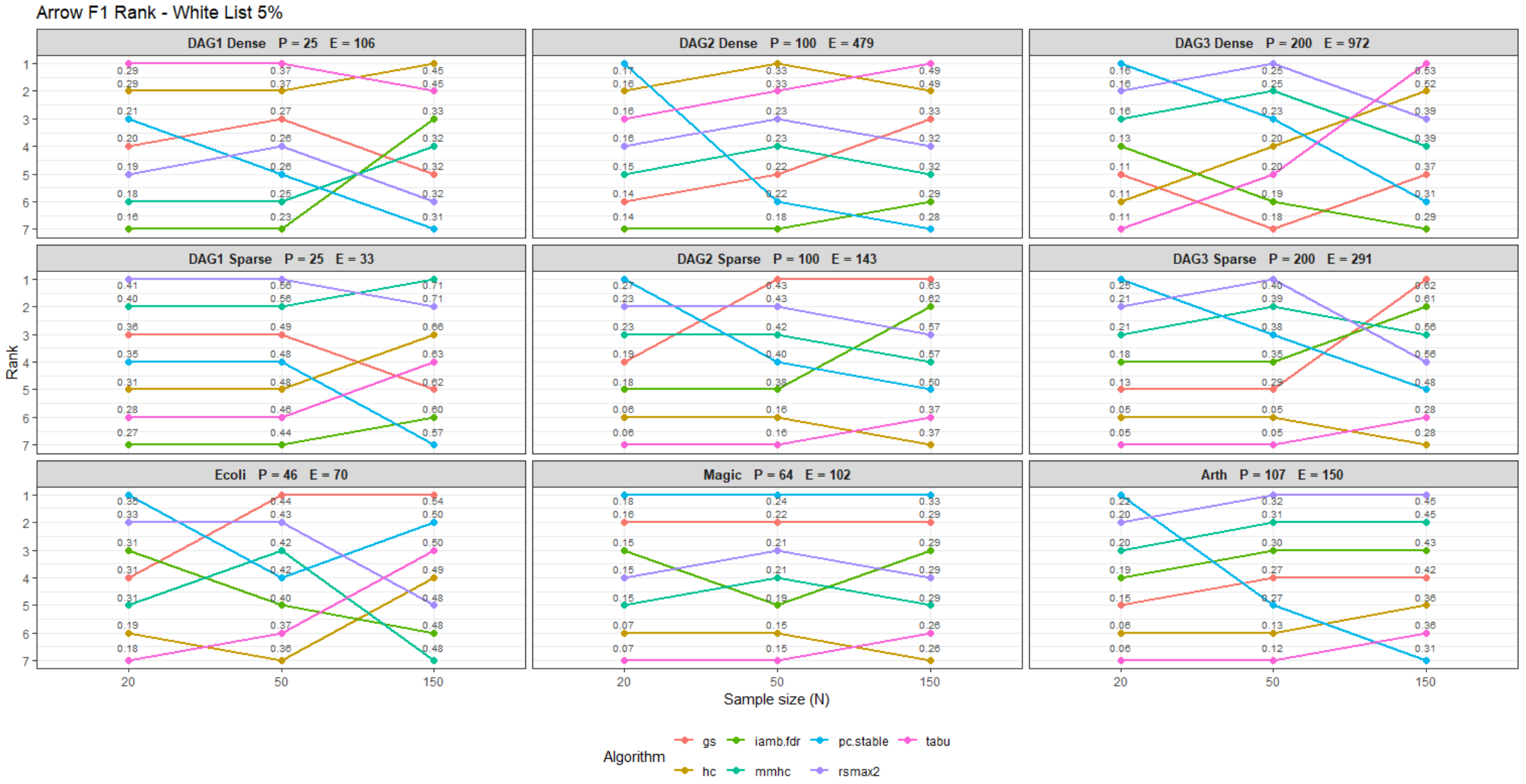


FIGURE A16. F1-arrow rank scores with whitelist 5%.

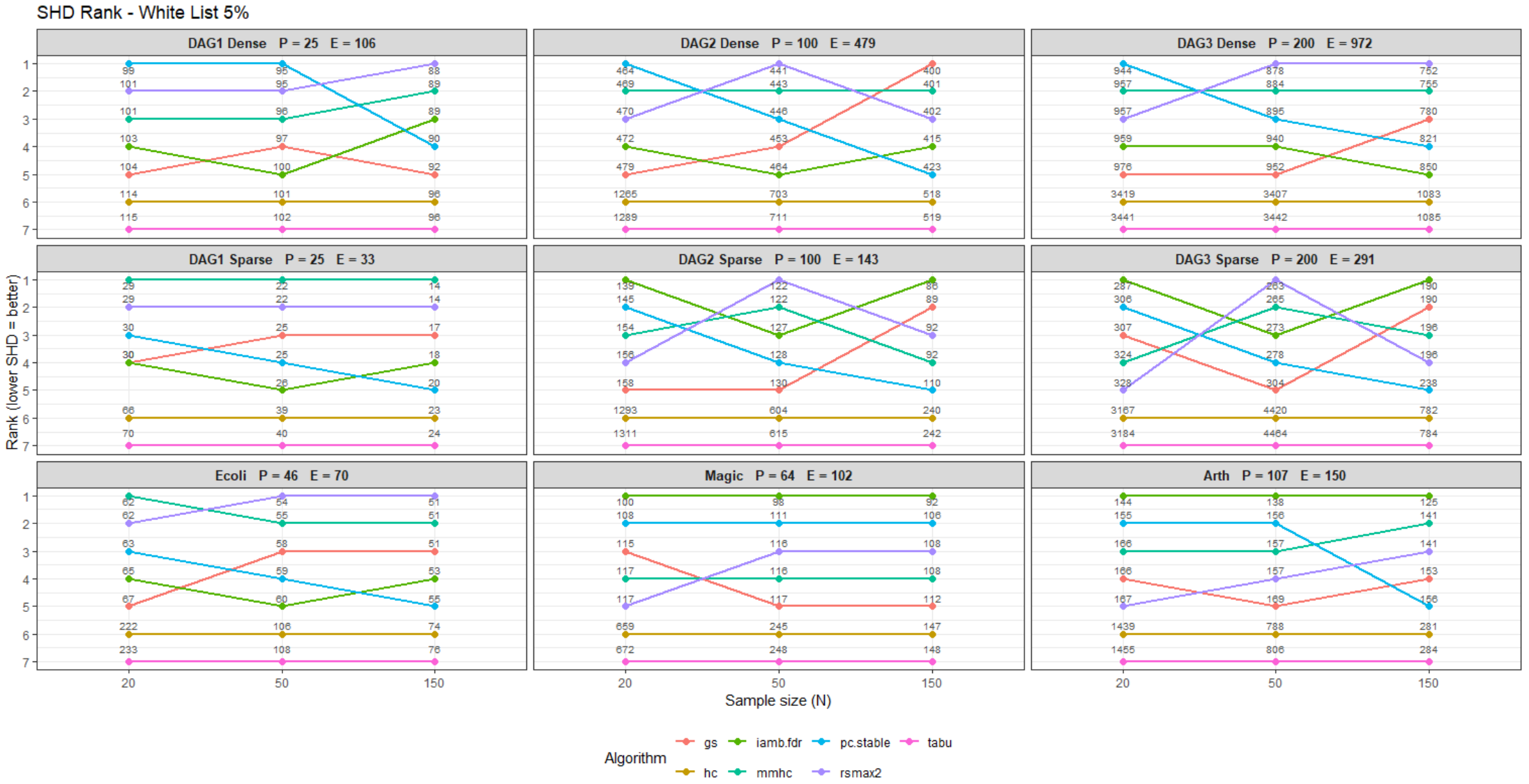


FIGURE A17. SHD rank scores with whitelist 5%.

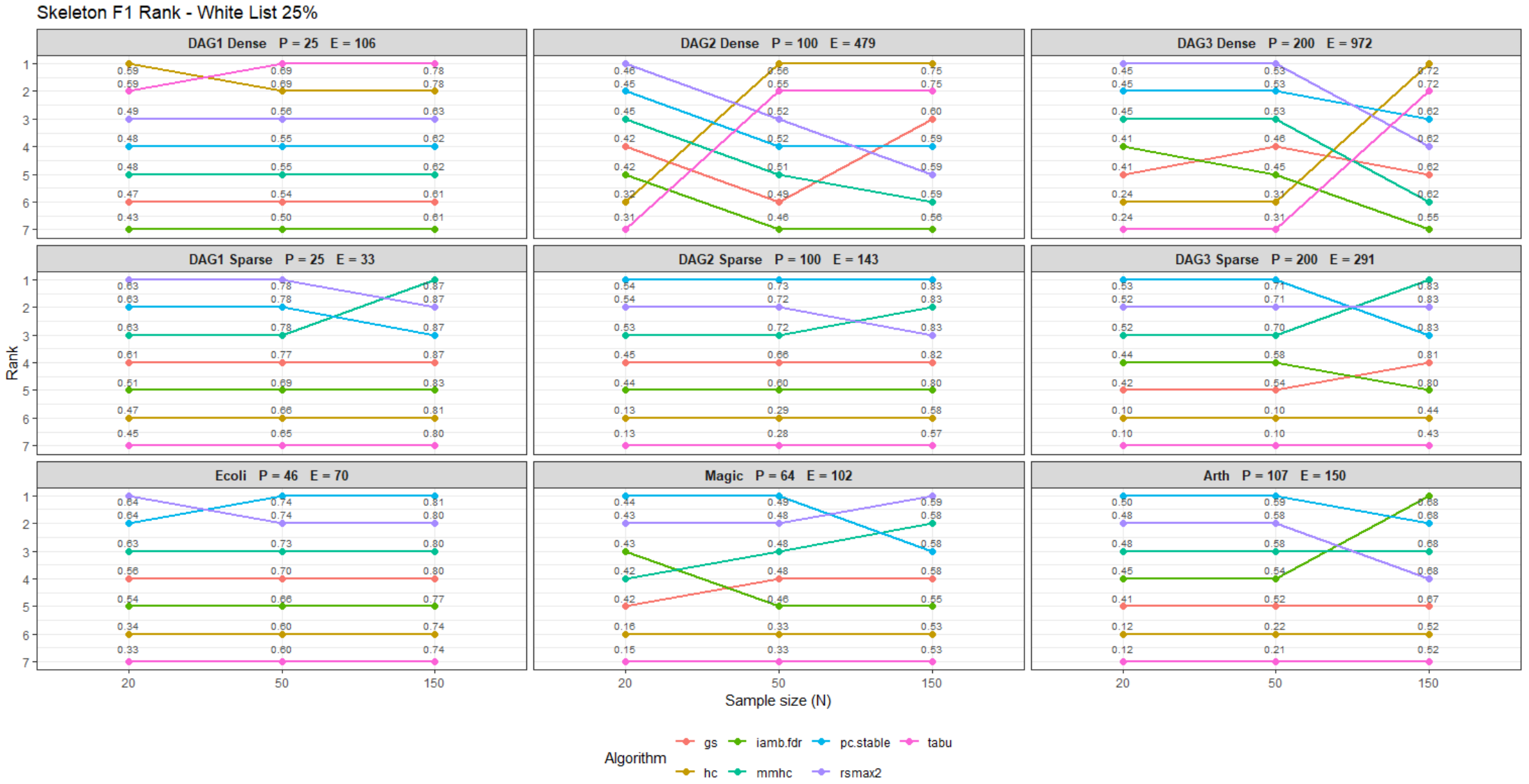


FIGURE A18. F1-skeleton rank scores with whitelist 25%.

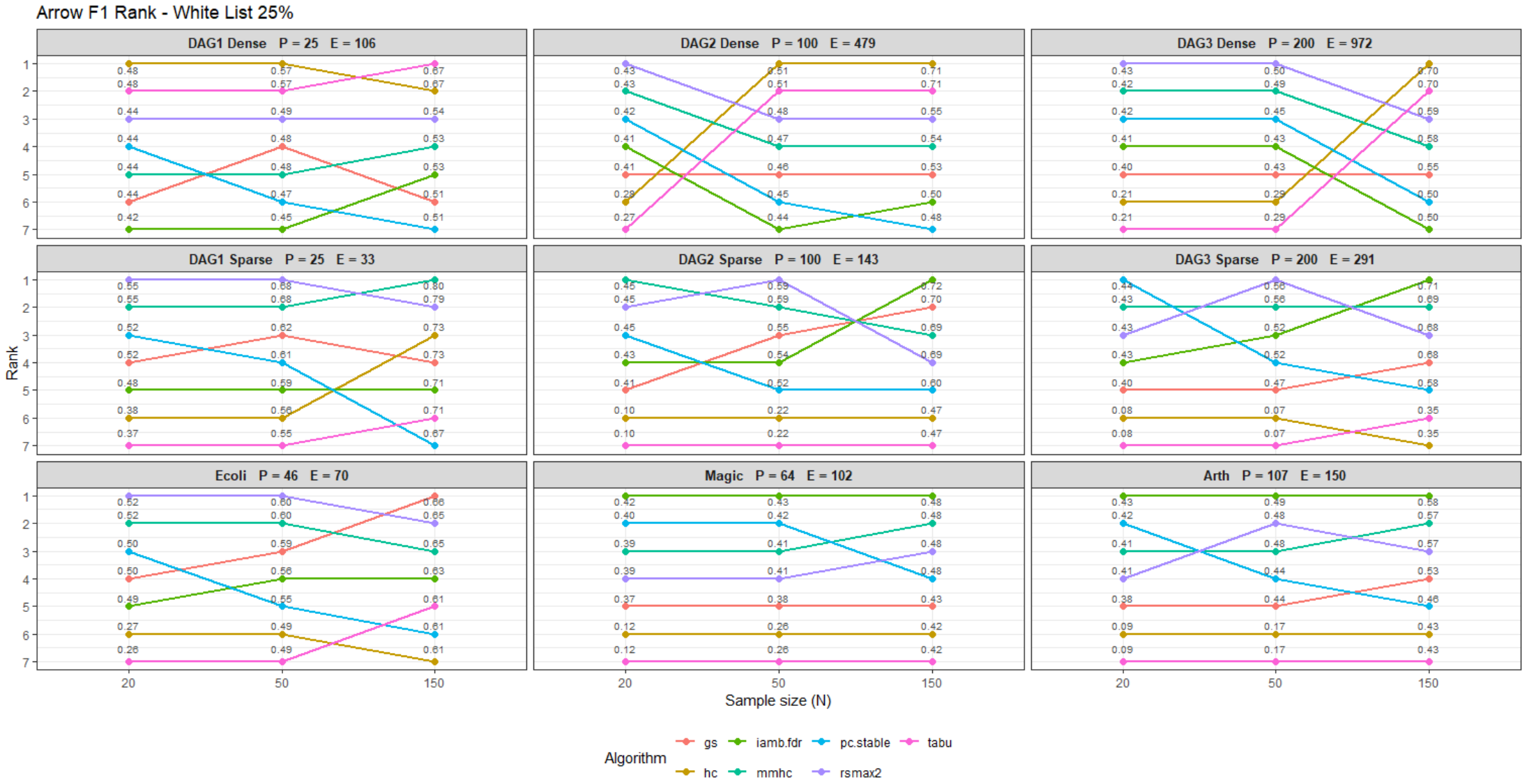


FIGURE A19. F1-arrow rank scores with whitelist 25%.

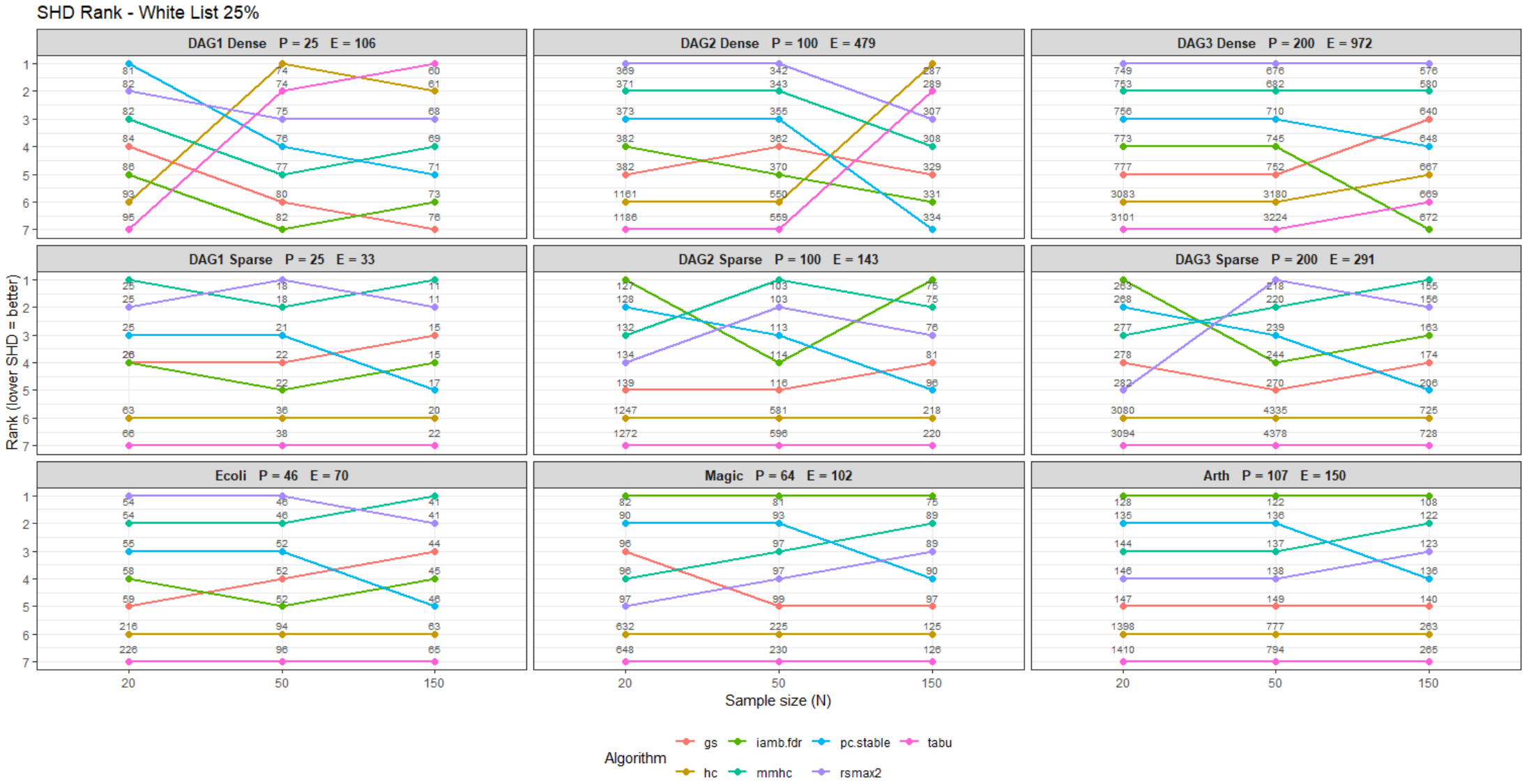


FIGURE A20. SHD rank scores with whitelist 25%.

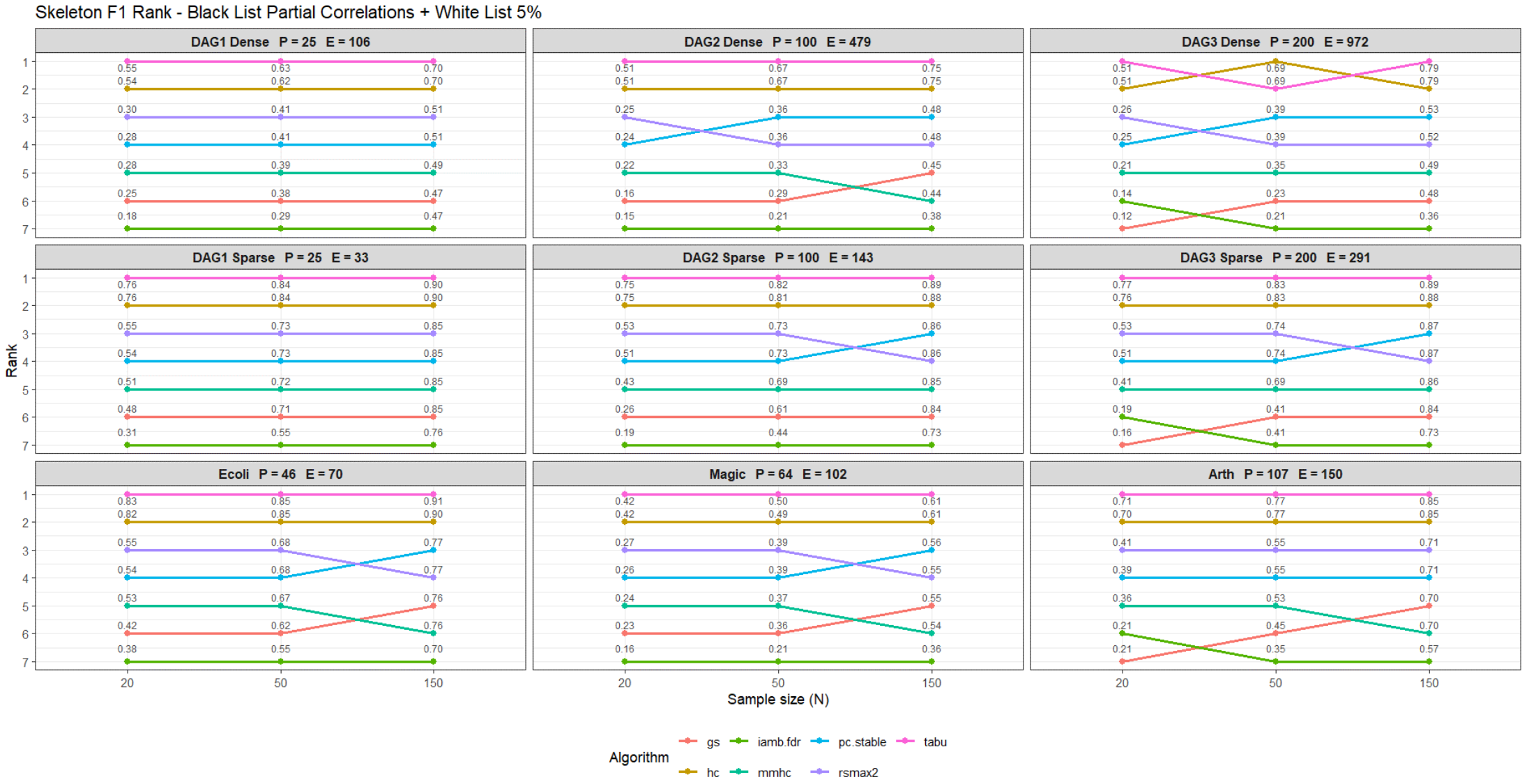


FIGURE A21. F1-skeleton rank scores with benchmark blacklist + whitelist 5%.

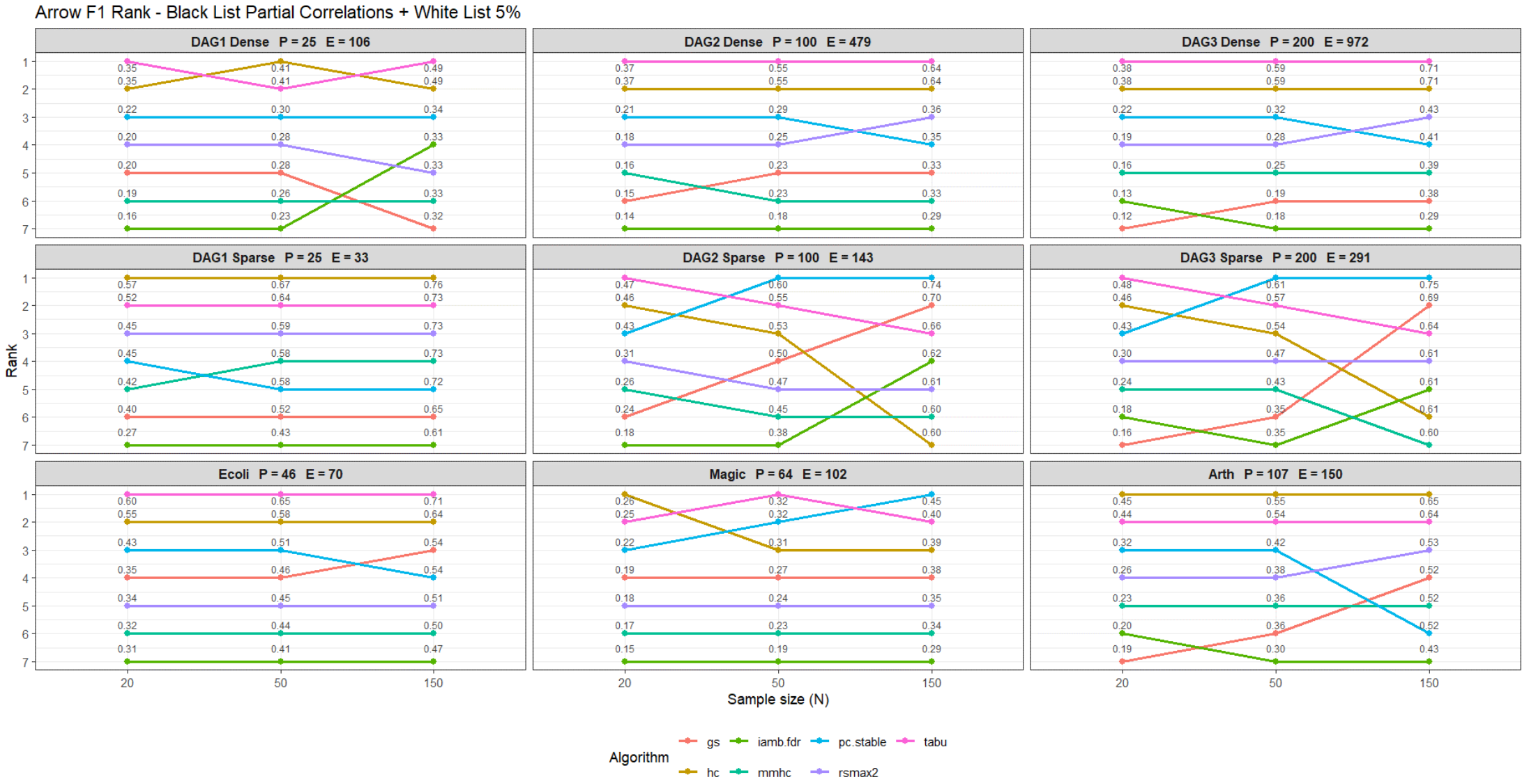


FIGURE A22. F1-arrow rank scores with benchmark blacklist + whitelist 5%.

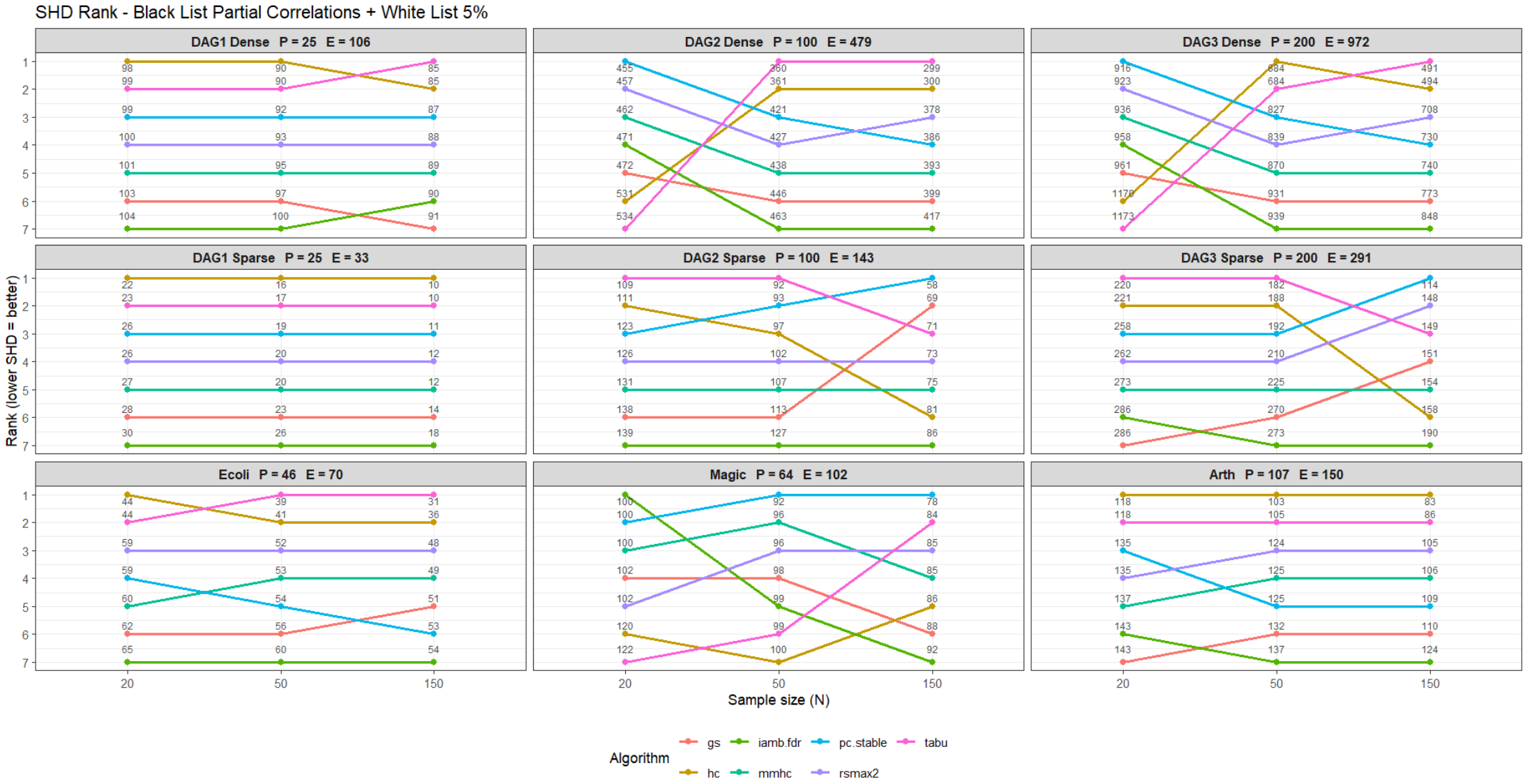


FIGURE A23. SHD rank scores with benchmark blacklist + whitelist 5%.

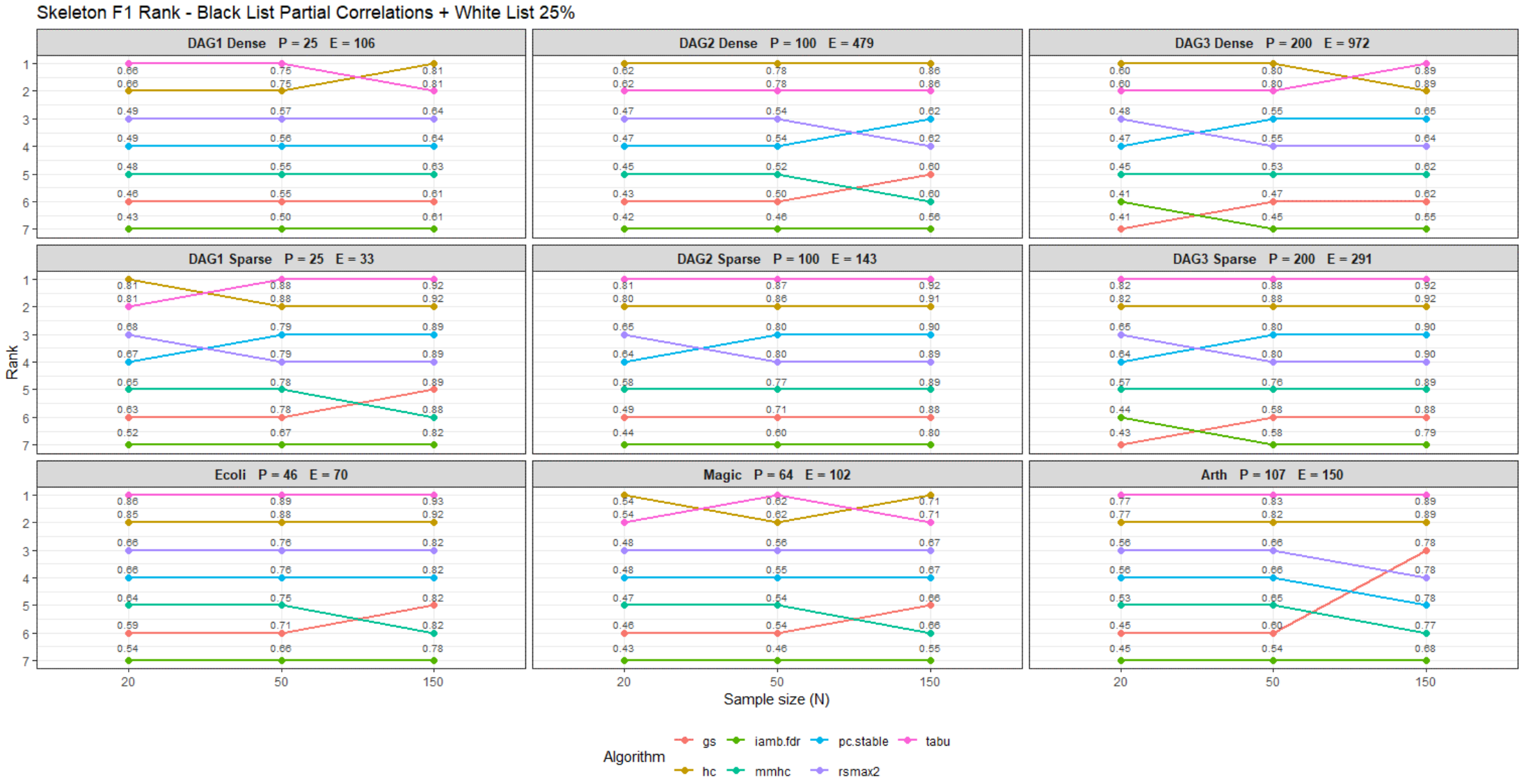


FIGURE A24. F1-skeleton rank scores with benchmark blacklist + whitelist 25%.

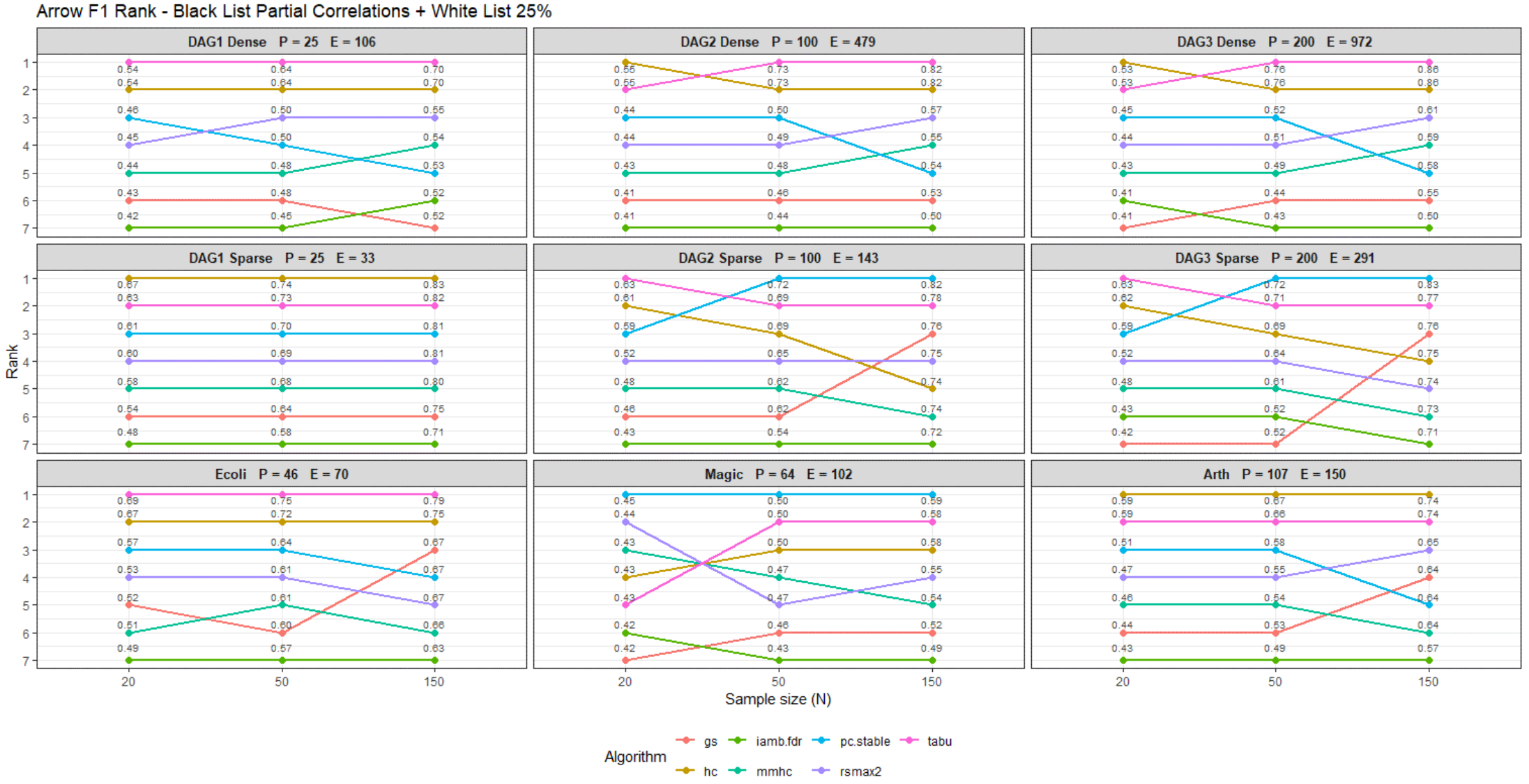


FIGURE A25. F1-arrow rank scores with benchmark blacklist + whitelist 25%.

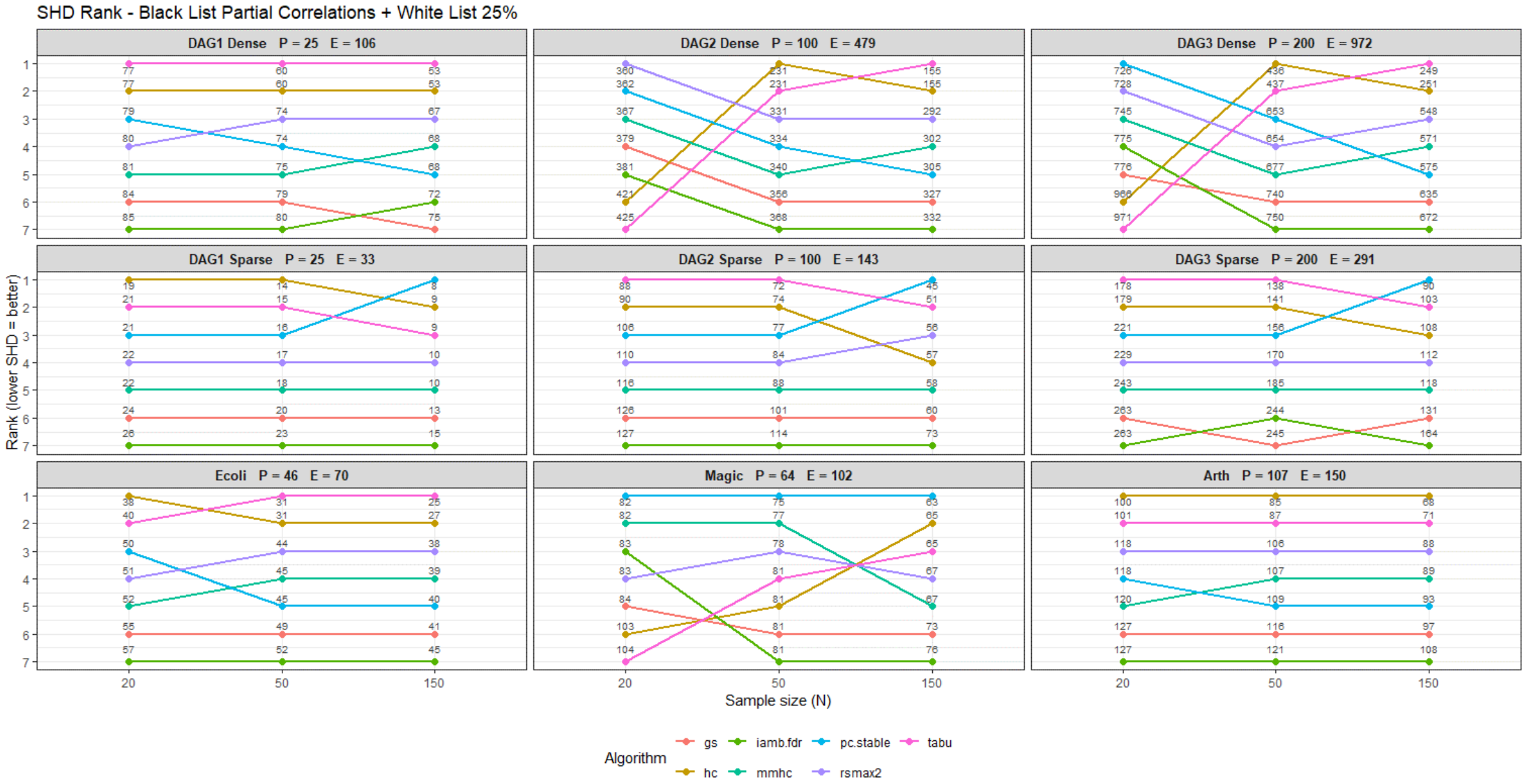


FIGURE A26. SHD rank scores with benchmark blacklist + whitelist 25%.

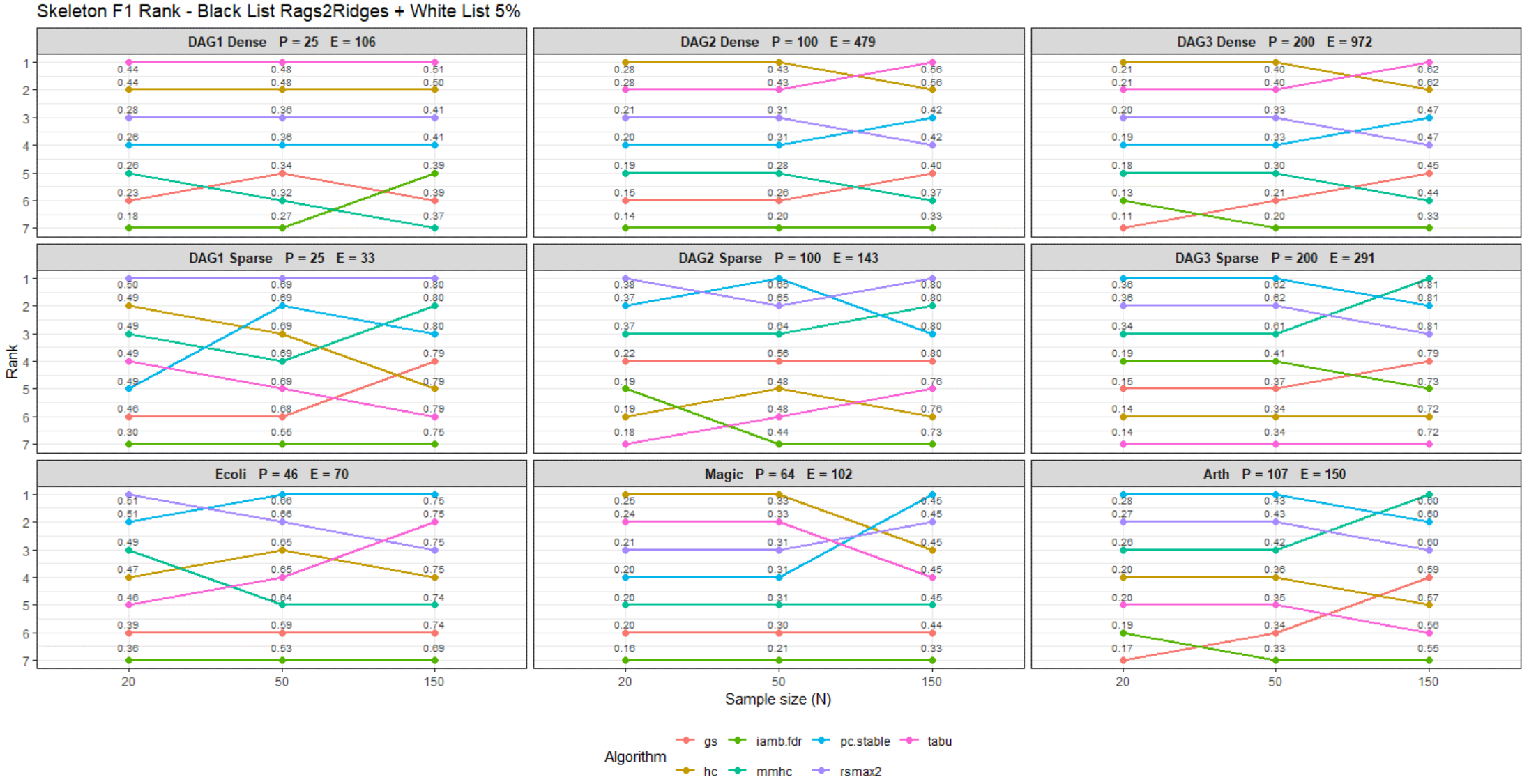


FIGURE A27. F1-skeleton rank scores with Rags2Ridges blacklist + whitelist 5%.

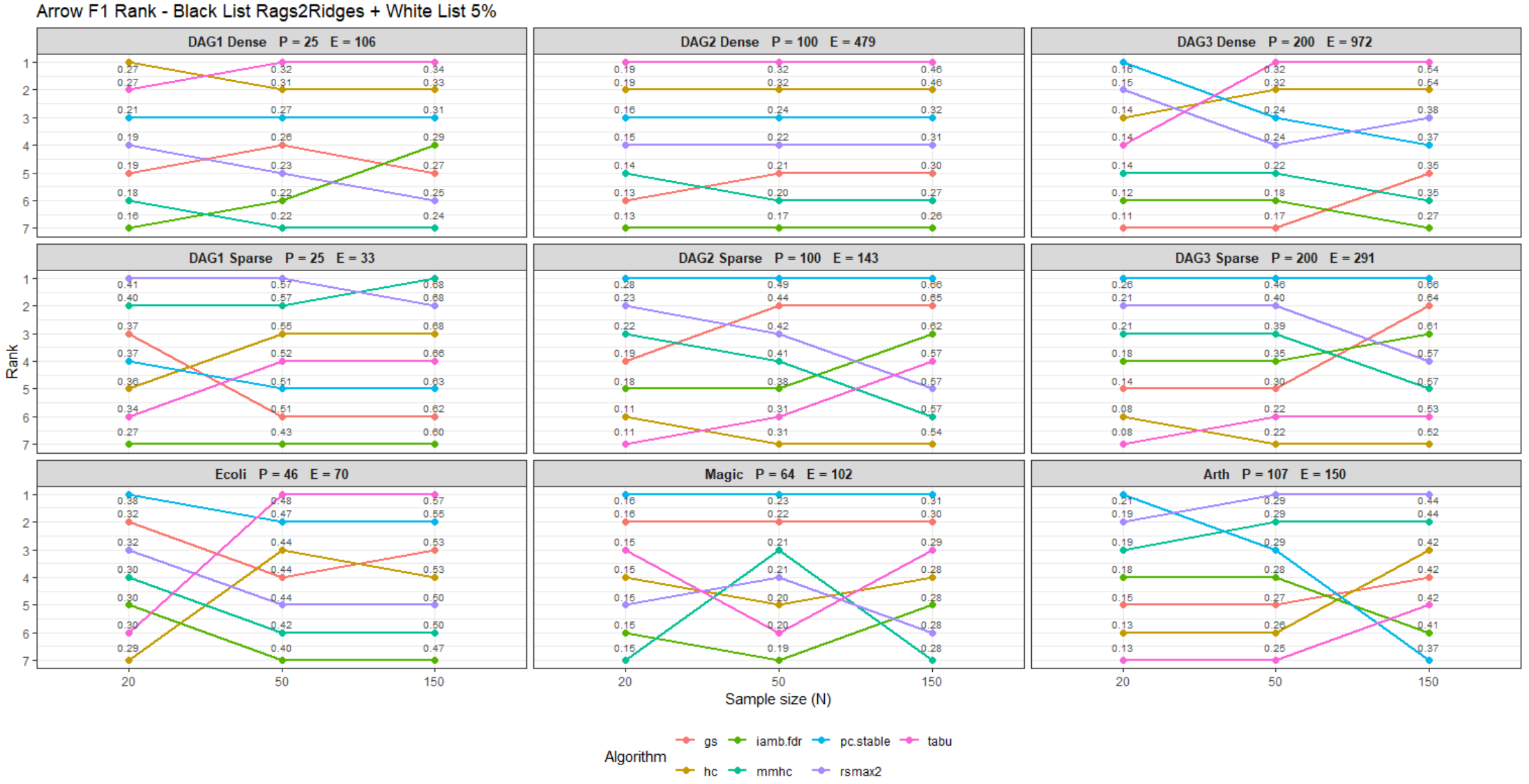


FIGURE A28. F1-arrow rank scores with Rags2Ridges blacklist + whitelist 5%.

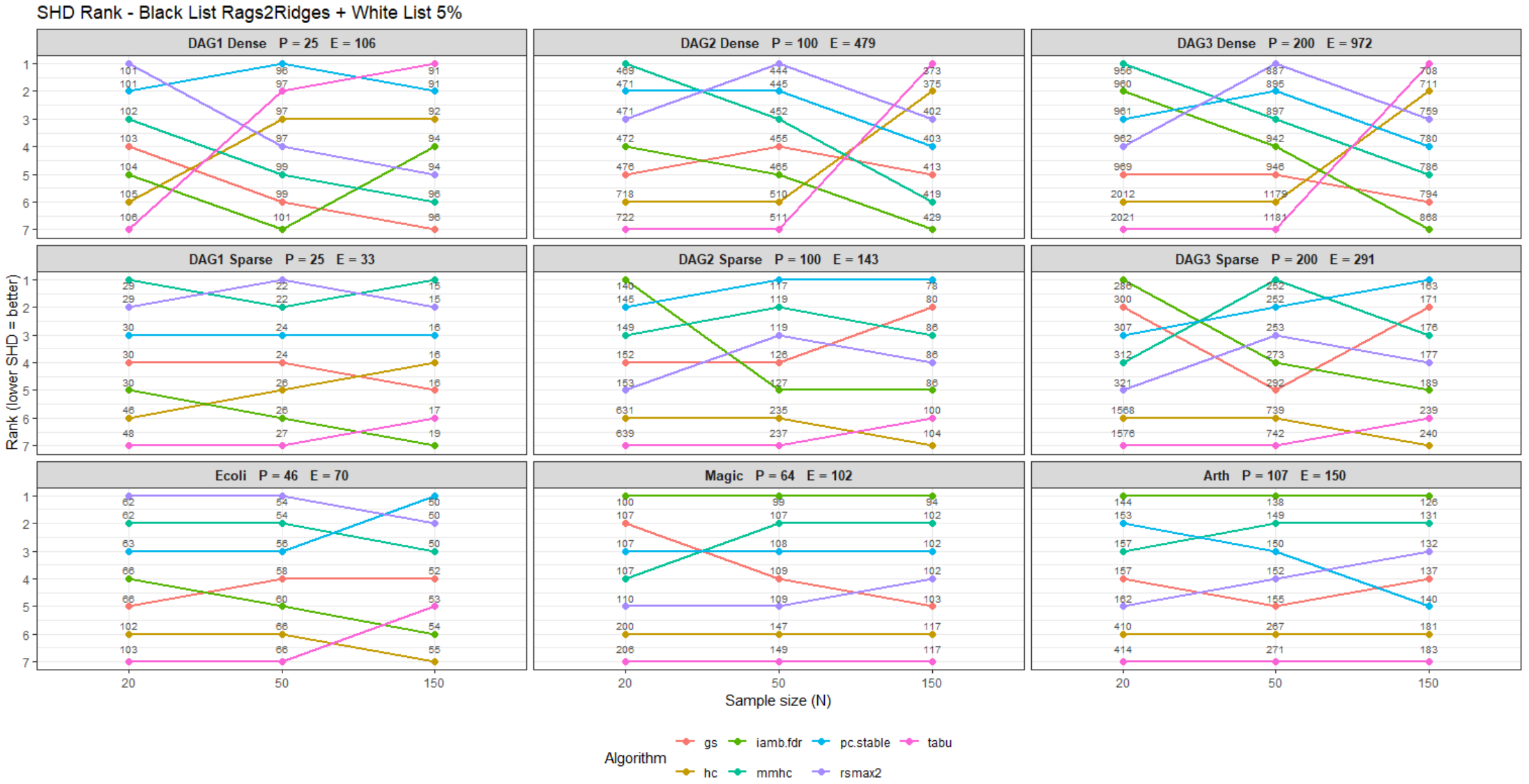


FIGURE A29. SHD rank scores with Rags2Ridges blacklist + whitelist 5%.

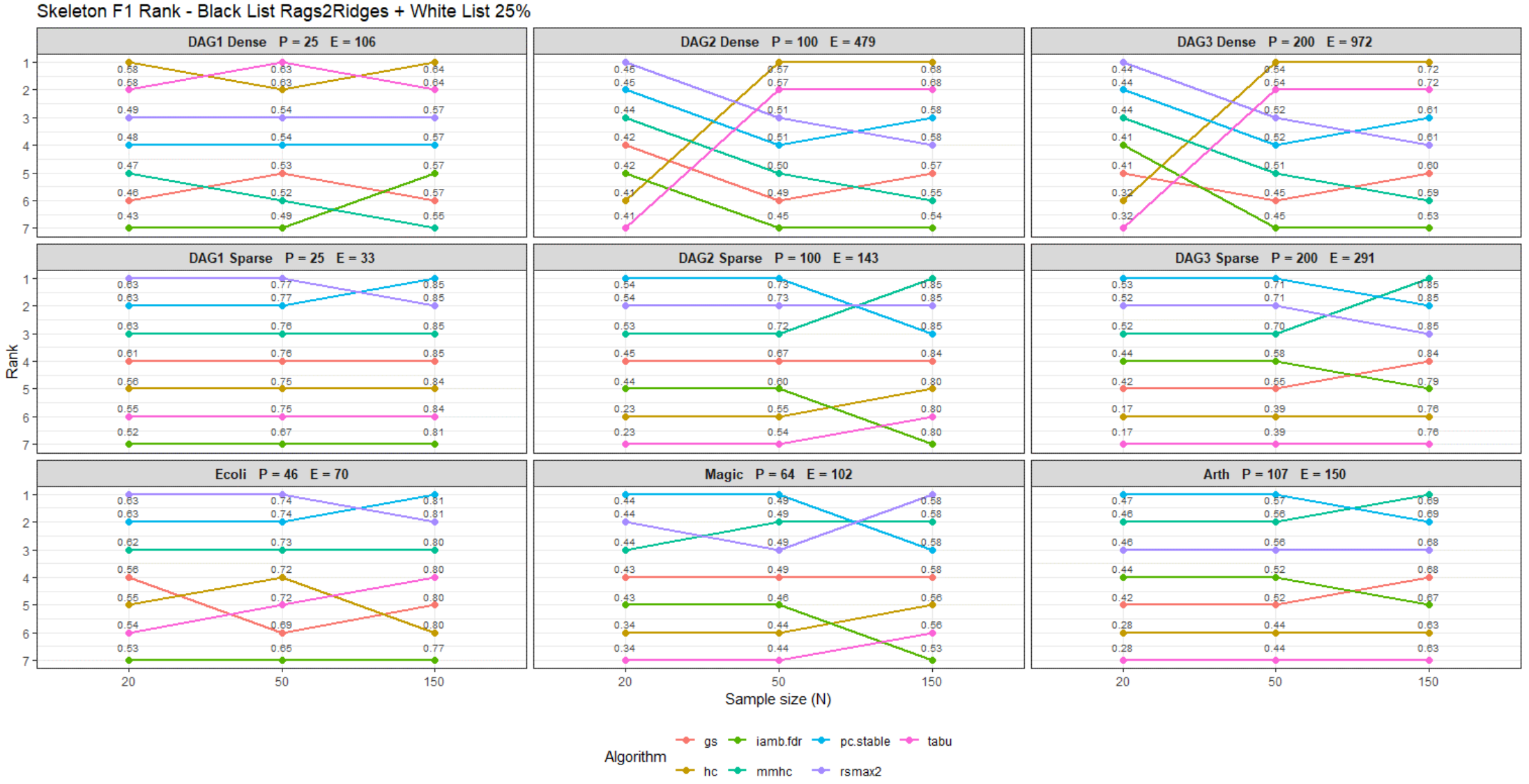


FIGURE A30. F1-skeleton rank scores with Rags2Ridges blacklist + whitelist 25%.

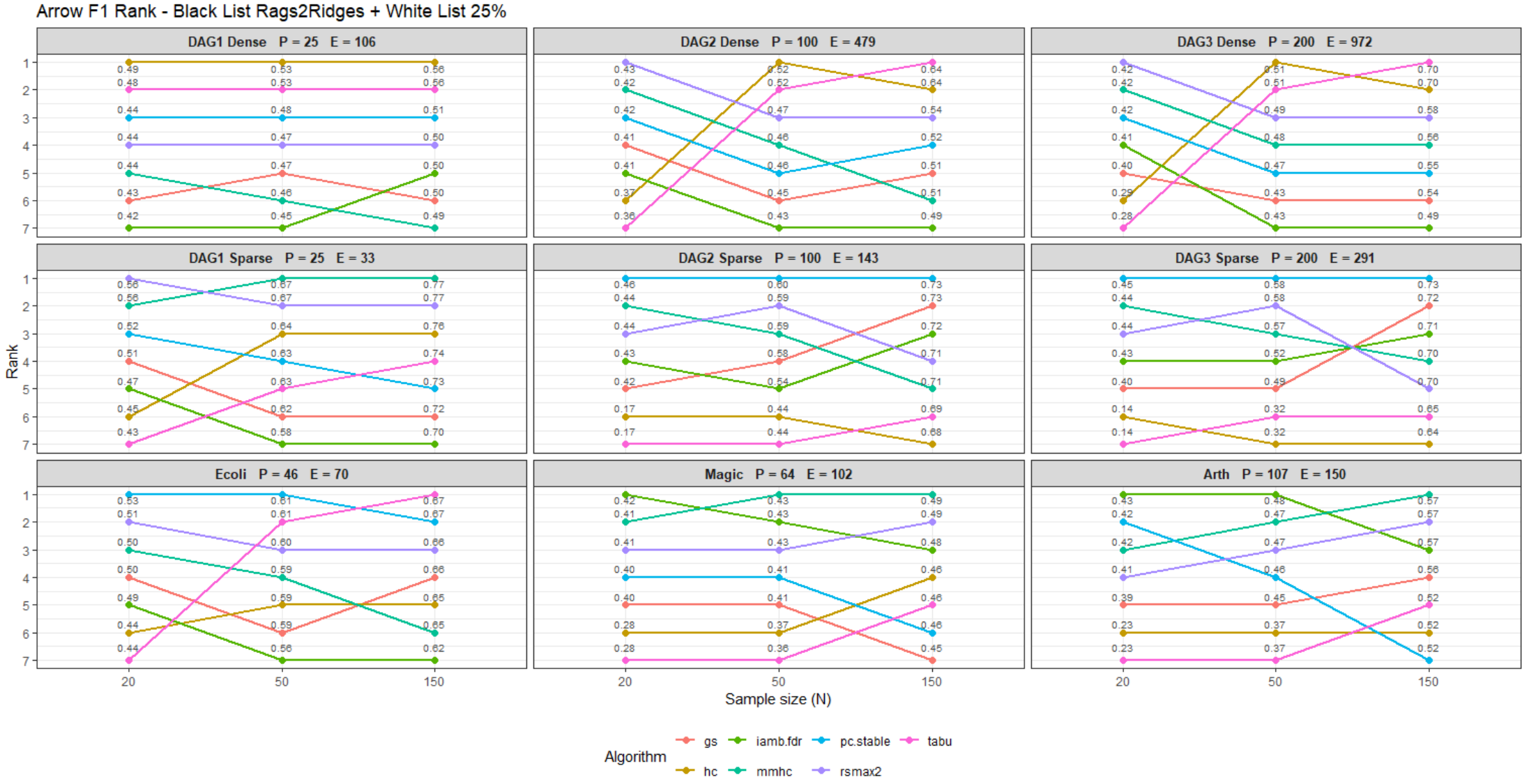


FIGURE A31. F1-arrow rank scores with Rags2Ridges blacklist + whitelist 25%.

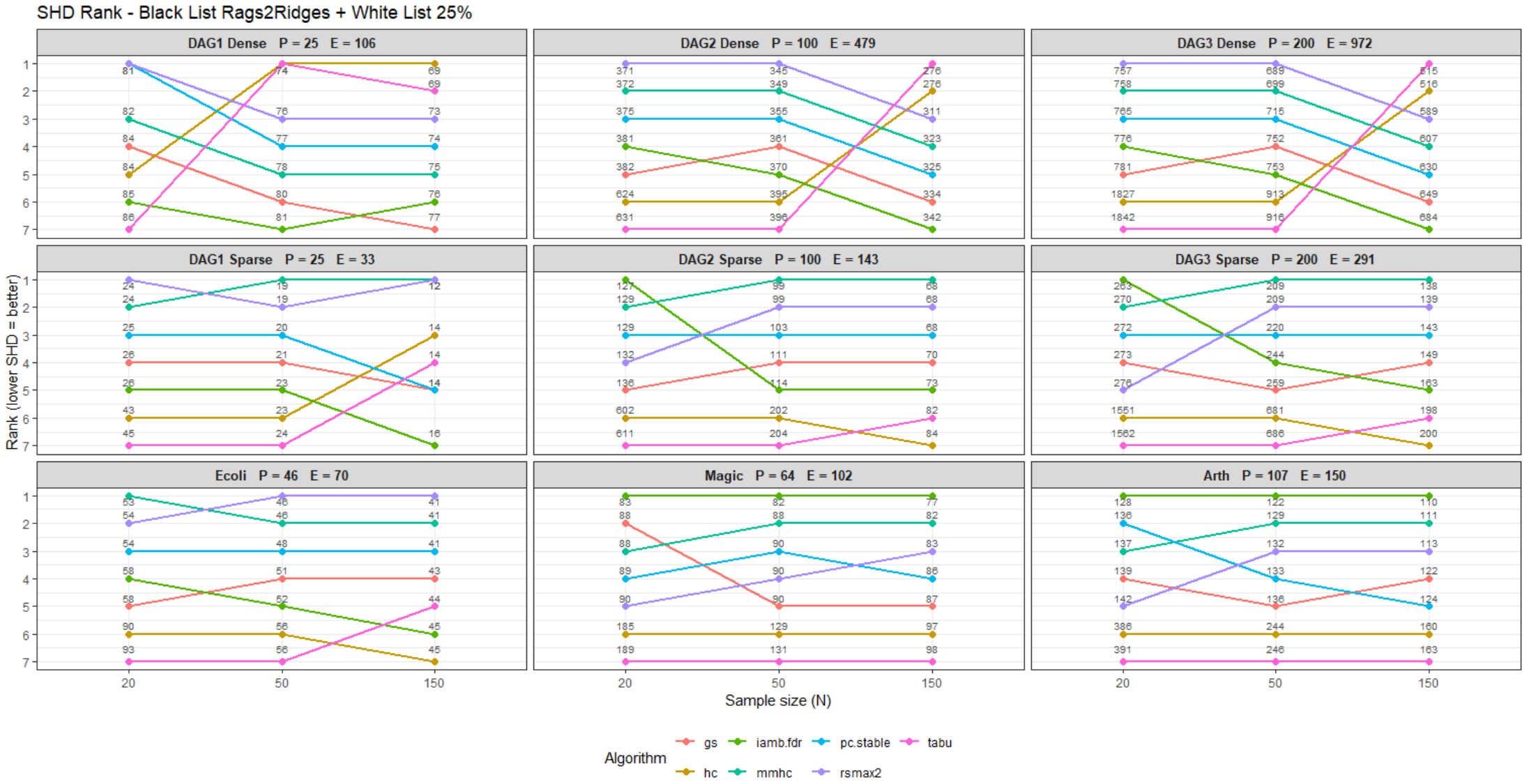


FIGURE A32. SHD rank scores with Rags2Ridges blacklist + whitelist 25%.

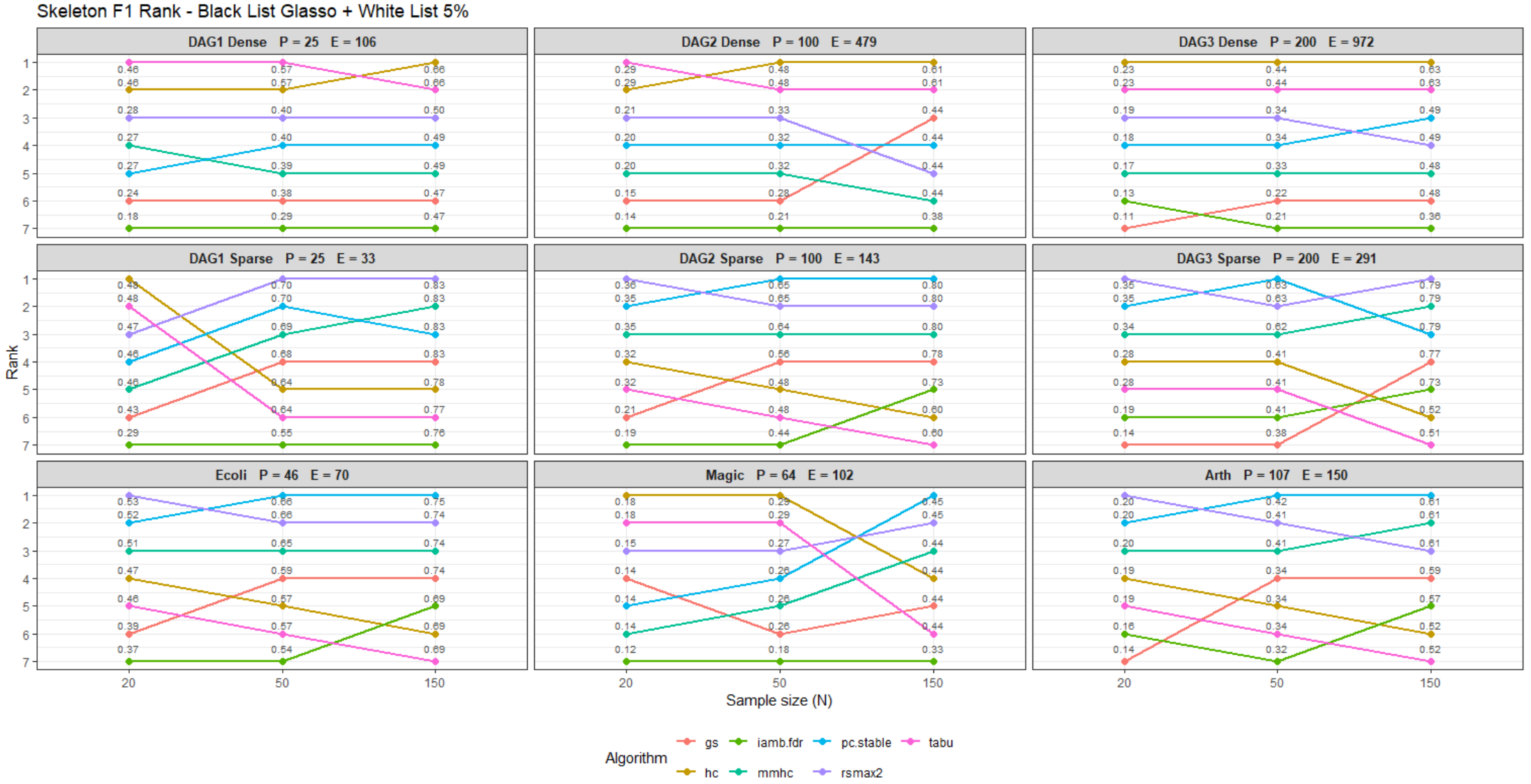


FIGURE A33. F1-skeleton rank scores with Glasso blacklist + whitelist 5%.

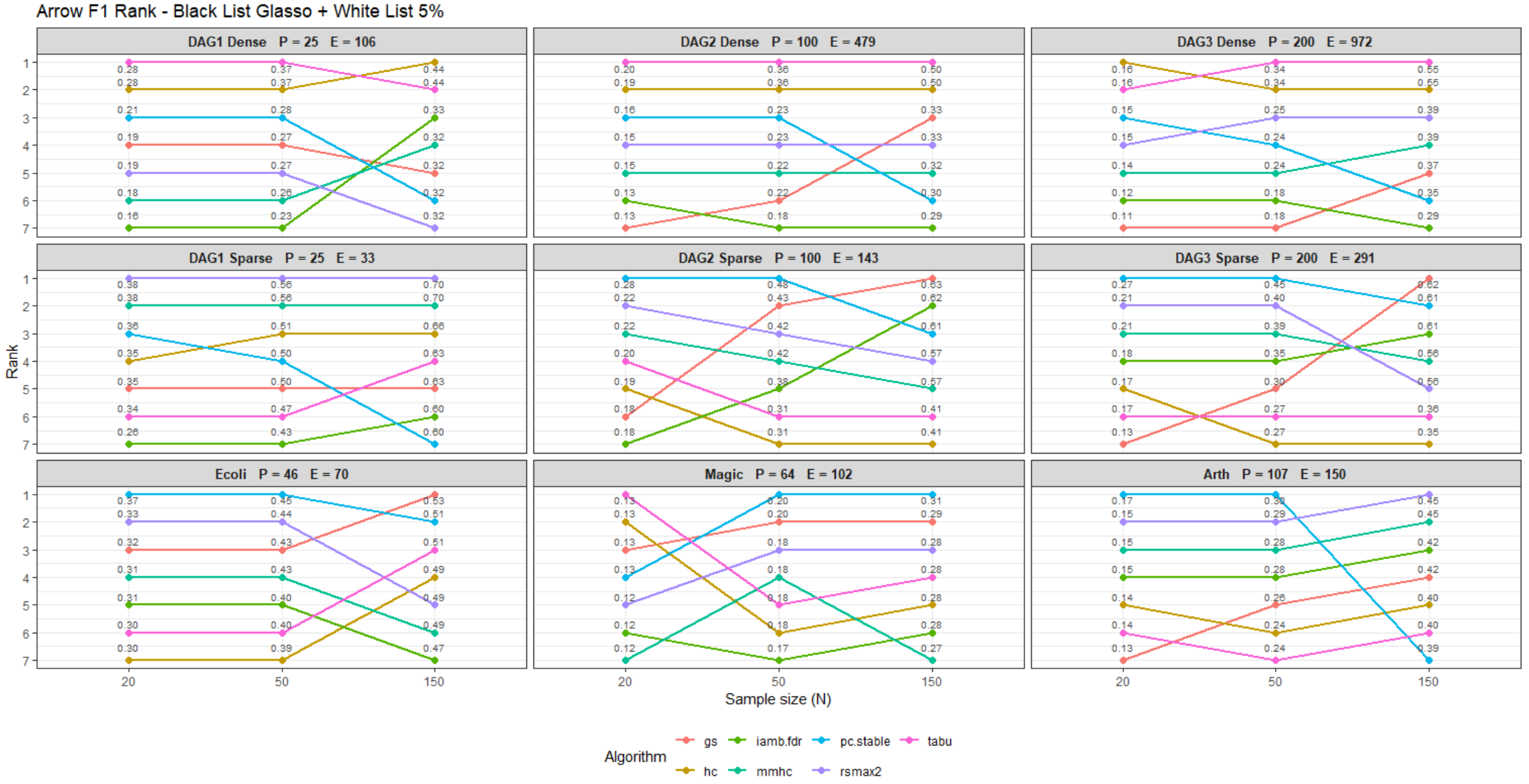


FIGURE A34. F1-arrow rank scores with Glasso blacklist + whitelist 5%.

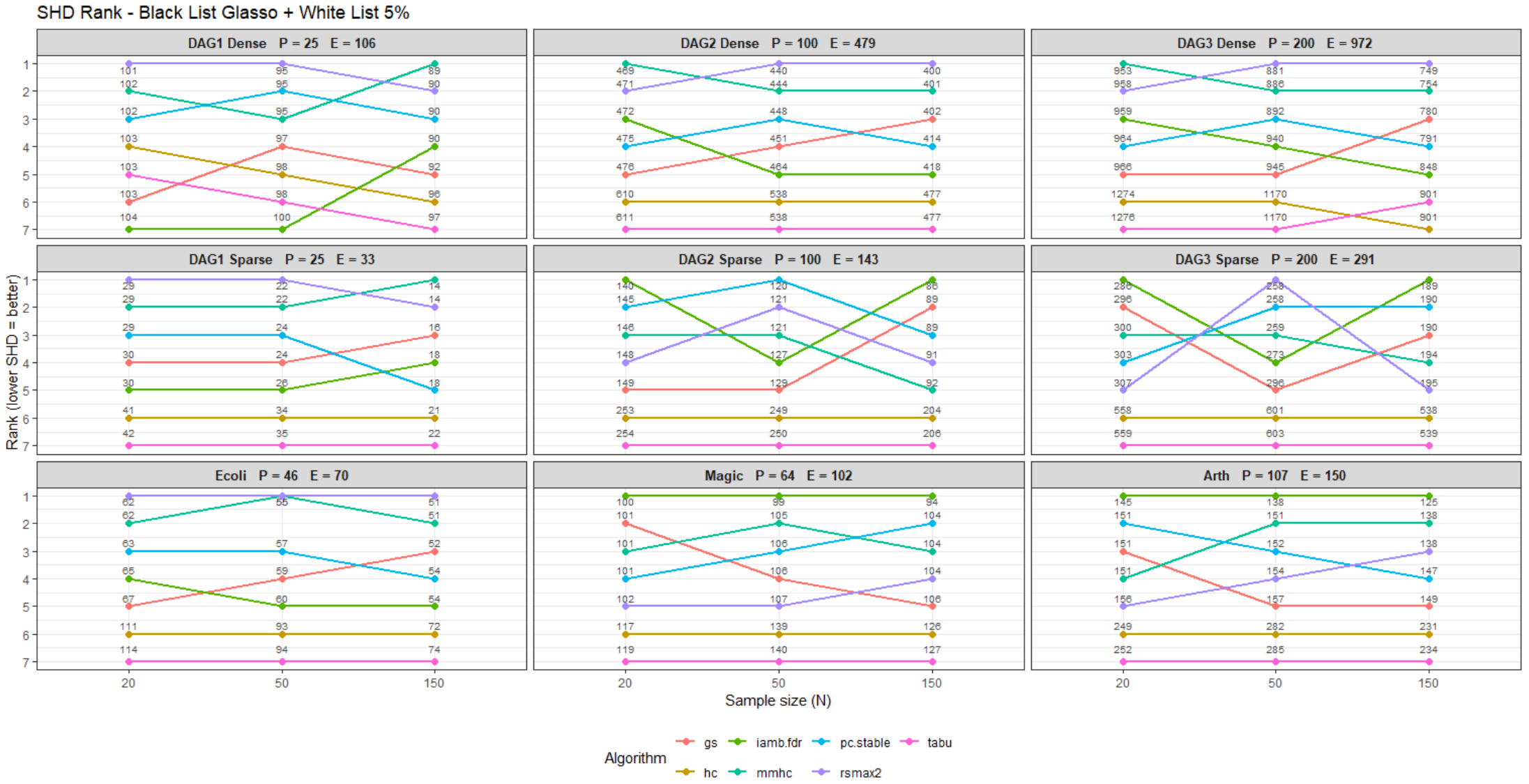


FIGURE A35. SHD rank scores with Glasso blacklist + whitelist 5%.

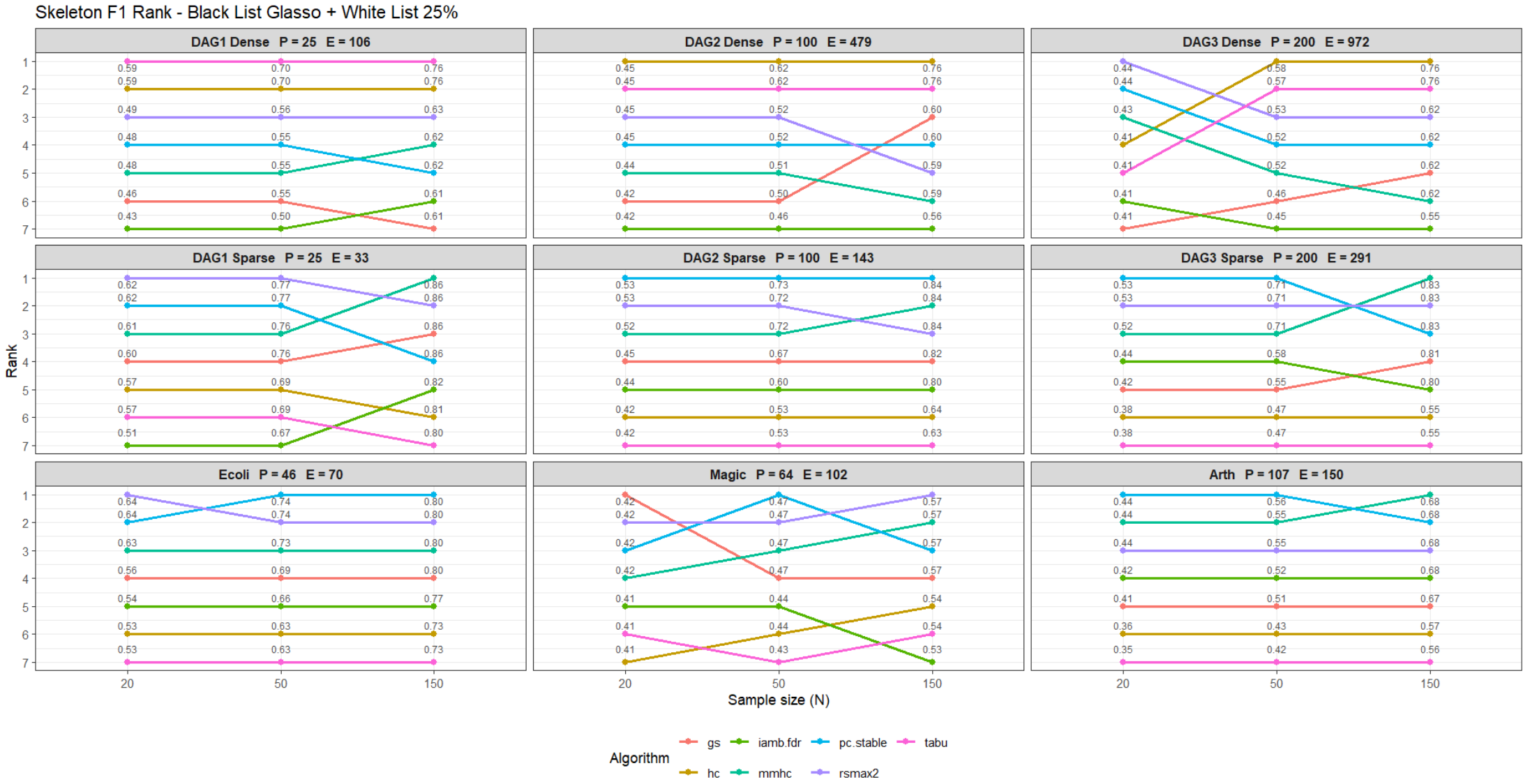


FIGURE A36. F1-skeleton rank scores with Glasso blacklist + whitelist 25%.

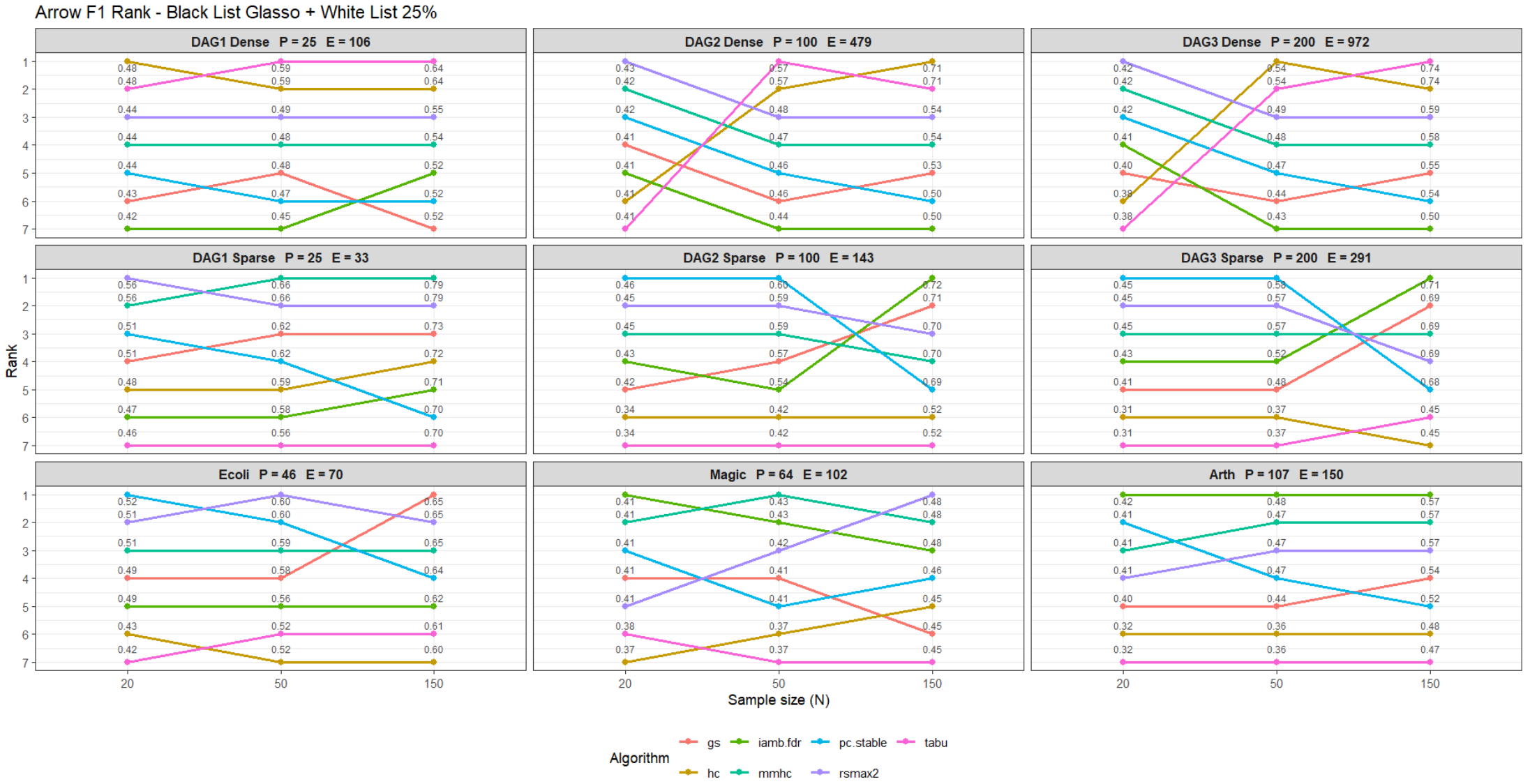


FIGURE A37. F1-arrow rank scores with Glasso blacklist + whitelist 25%.

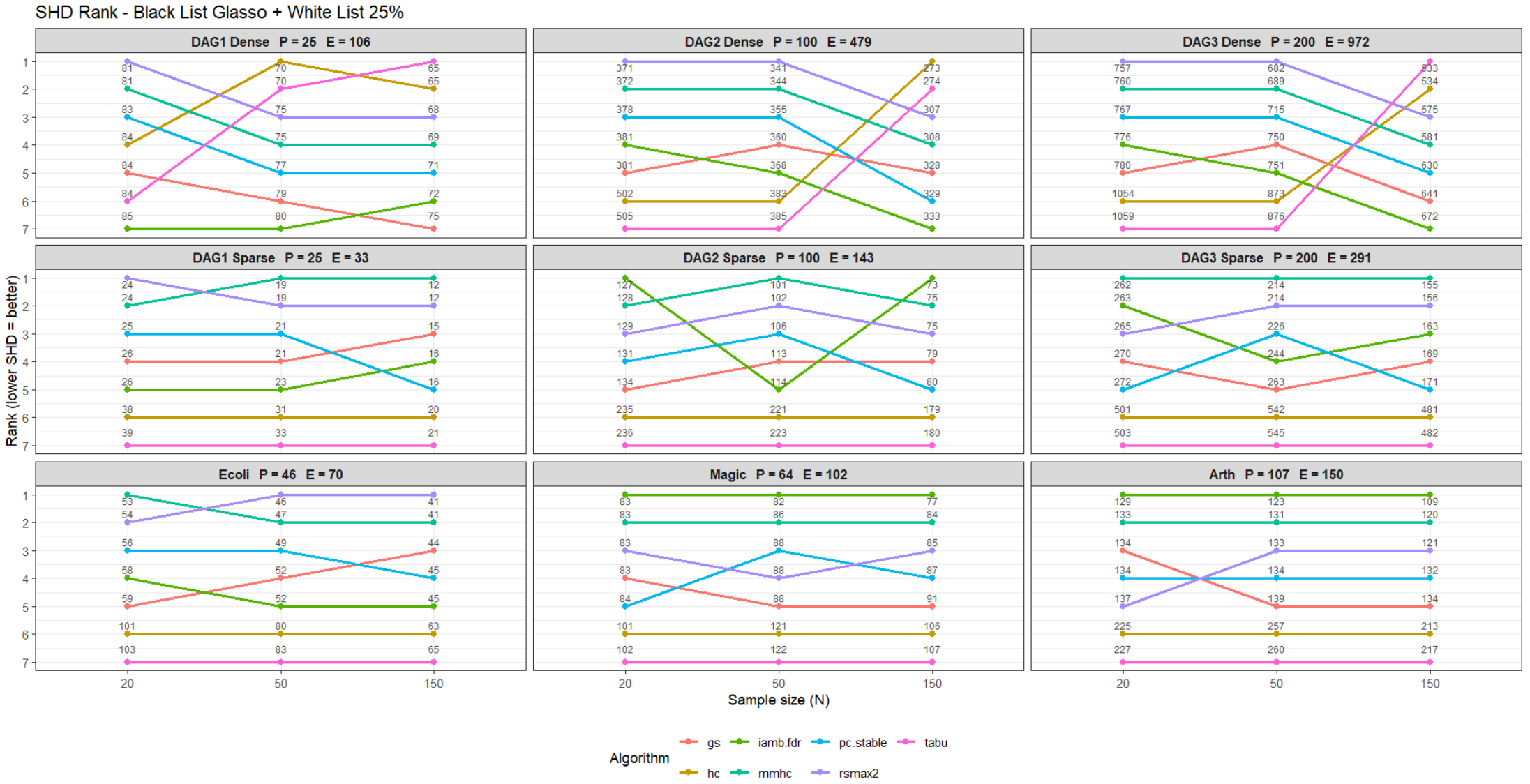


FIGURE A38. SHD rank scores with Glasso blacklist + whitelist 25%.

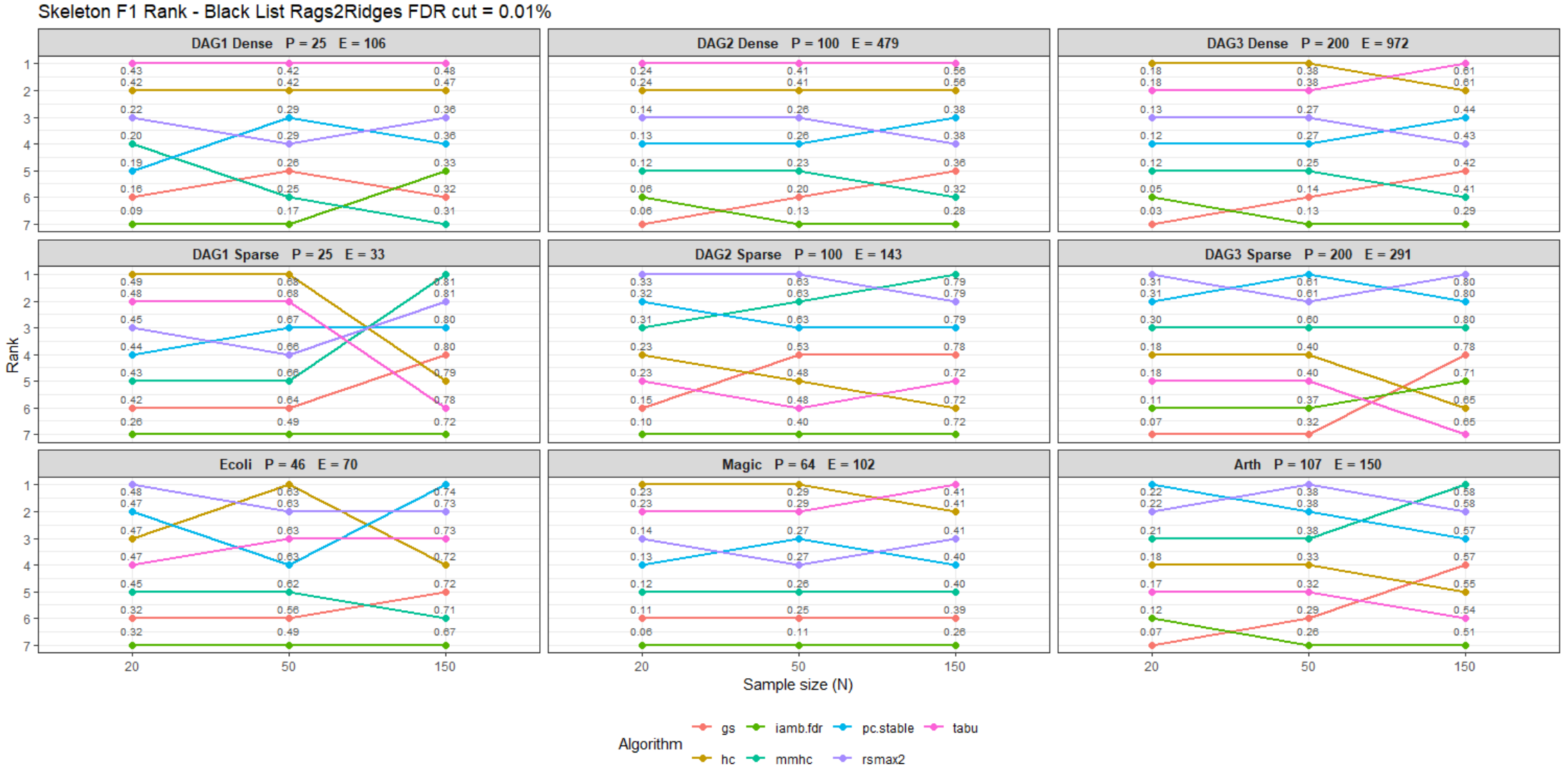


FIGURE A39. F1-skeleton rank scores with Rags2Ridges blacklist, FDR cut = 0.01

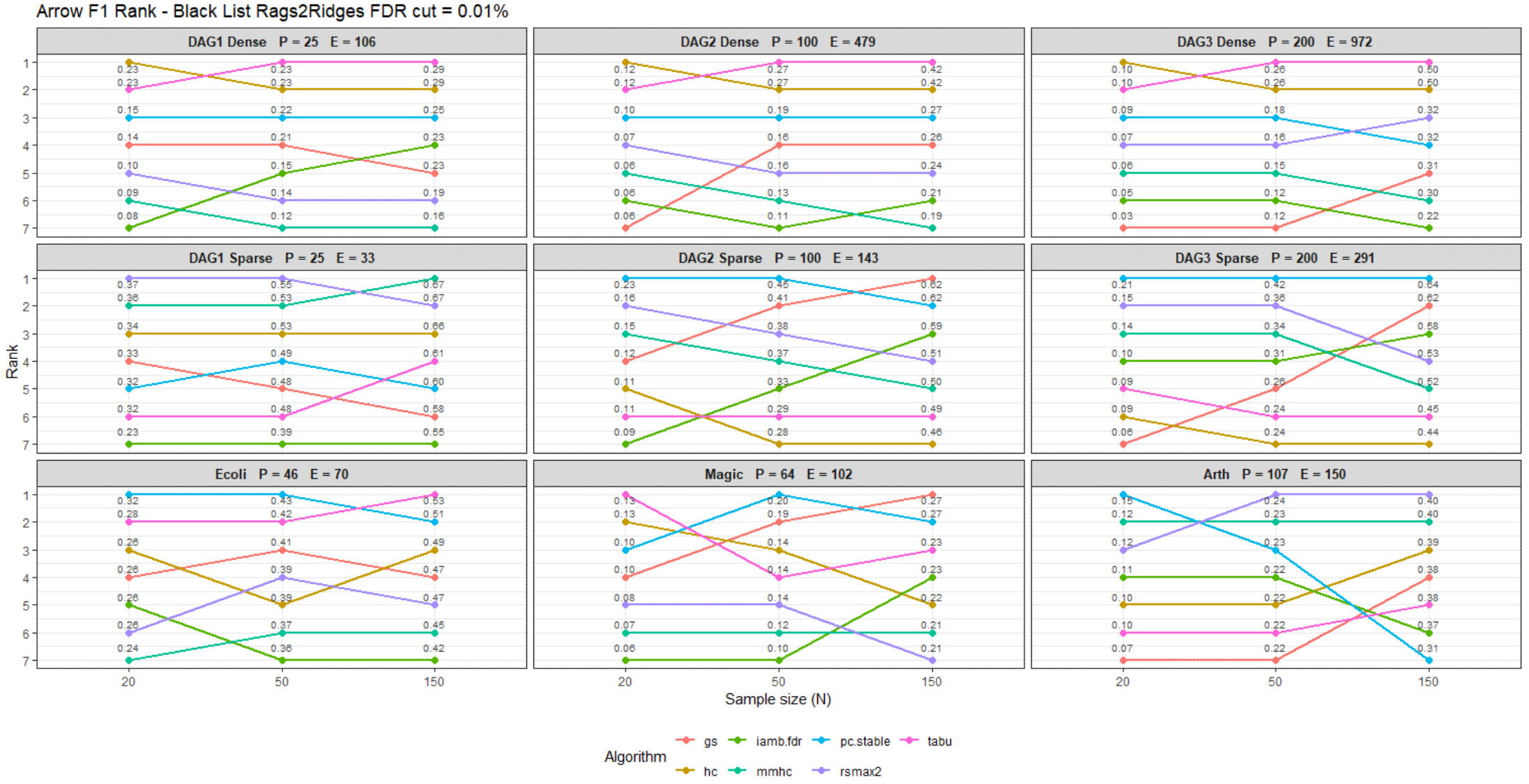


FIGURE A40. F1-arrow rank scores with Rags2Ridges blacklist, FDR cut = 0.01.

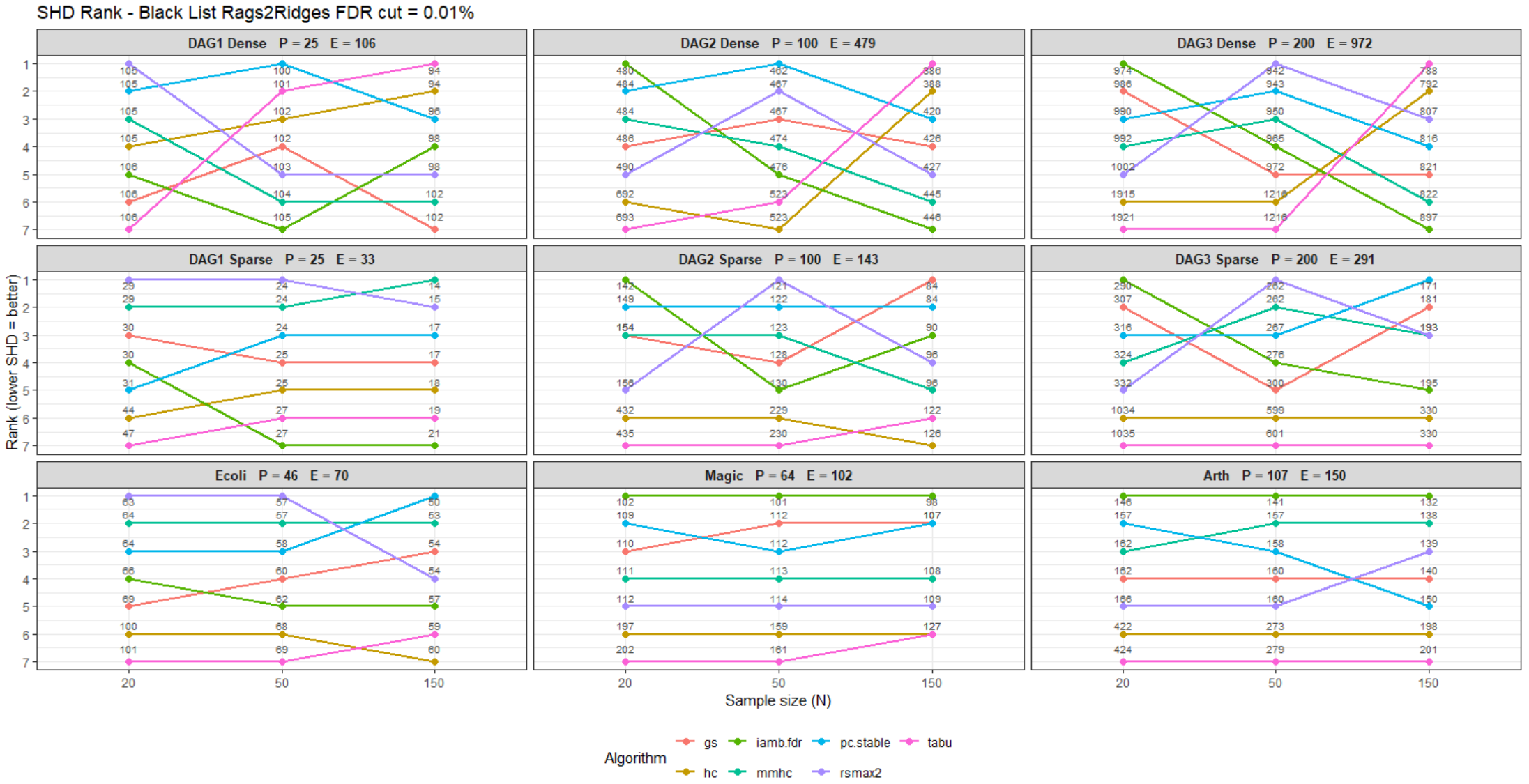


FIGURE A41. SHD rank scores with Rags2Ridges blacklist, FDR cut = 0.01.

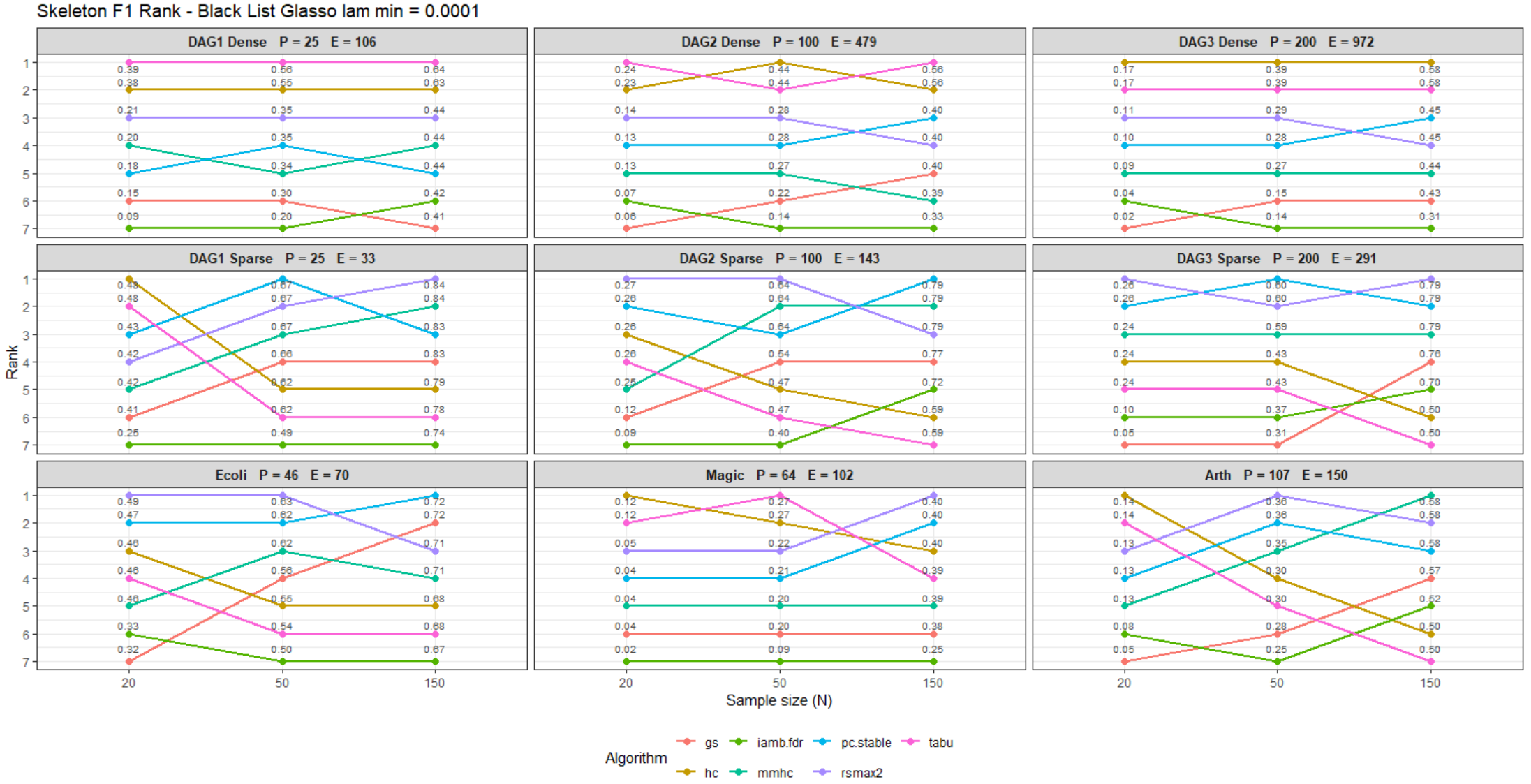


FIGURE A42. F1-skeleton rank scores with Glasso blacklist, lam.min = 0.0001

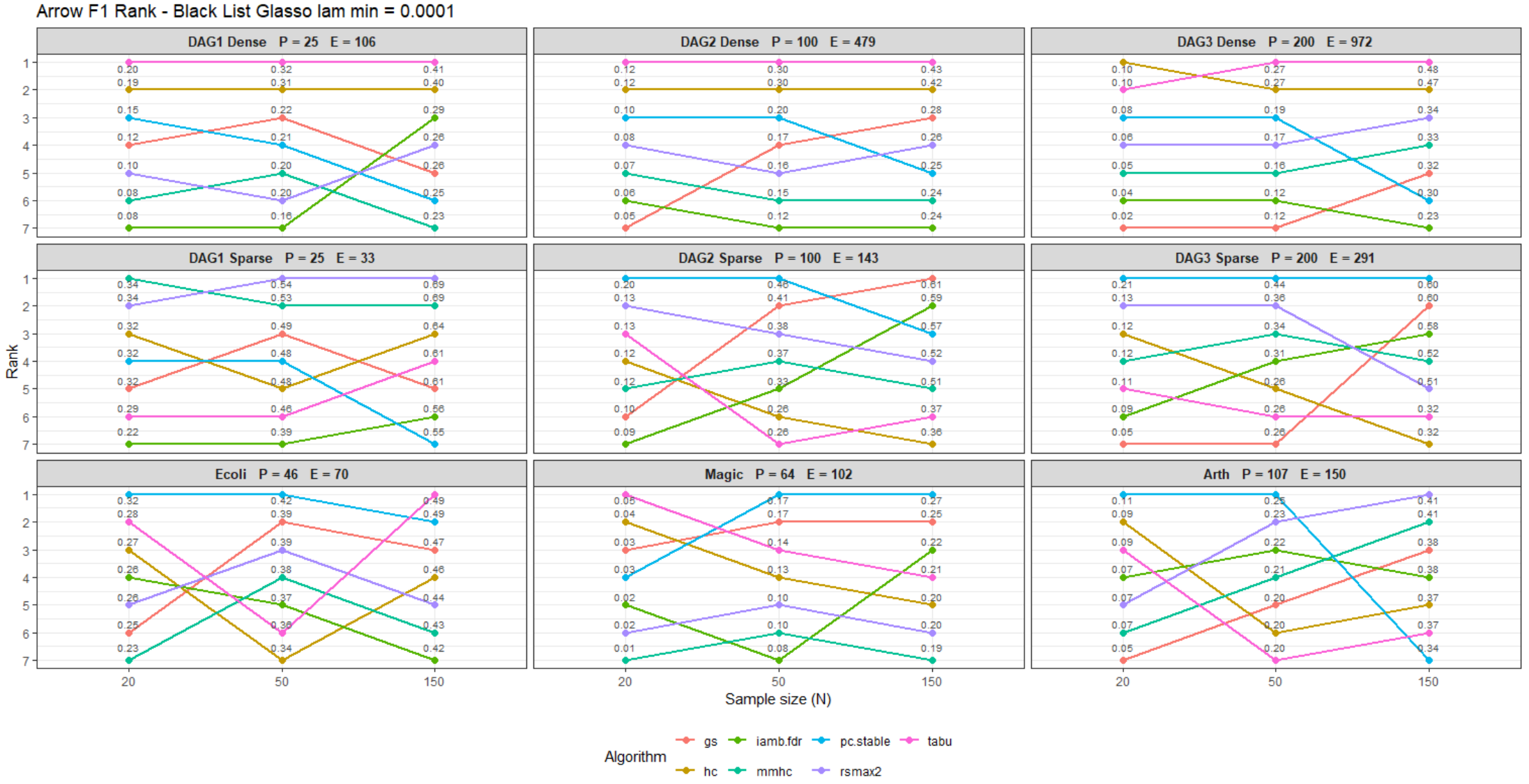


FIGURE A43. F1-arrow rank scores with Glasso blacklist, lam.min = 0.0001.

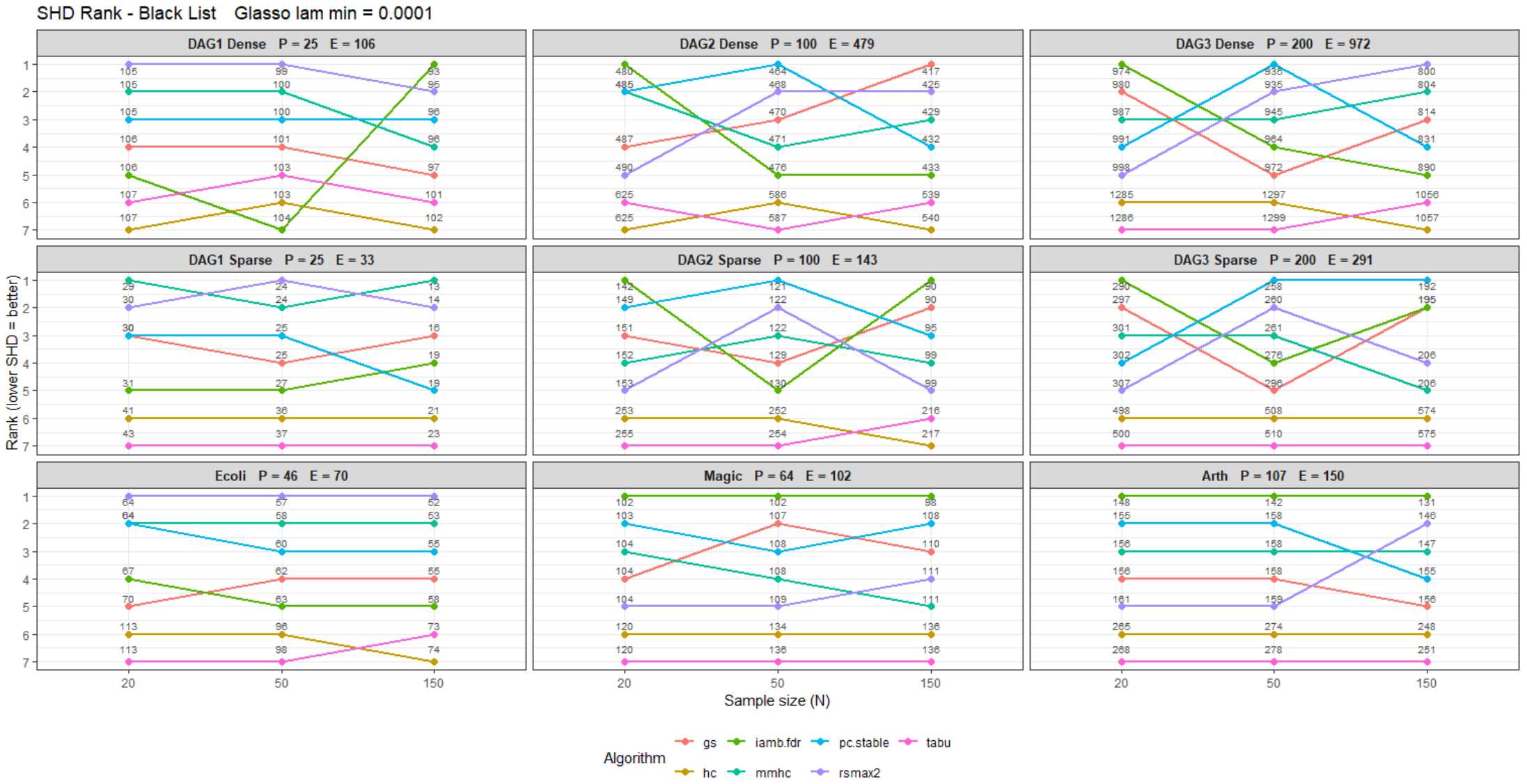


FIGURE A44. SHD rank scores with Glasso blacklist, lam.min = 0.0001.

Appendix C. Algorithms

C.0.1. ***GS***.

Algorithm A1: Grow–Shrink (GS) algorithm [63]

1. [Compute Markov Blankets]
For all $X \in P$, compute the Markov blanket $B(X)$.

2. [Compute Graph Structure]
For all $X \in P$ and $Y \in B(X)$, determine Y to be a direct neighbor of X if X and Y are dependent given S for all $S \subseteq T$, where T is the smaller of $B(X) - \{Y\}$ and $B(Y) - \{X\}$.

3. [Orient Edges]
For all $X \in P$ and $Y \in N(X)$, orient $Y \to X$ if there exists a variable $Z \in N(X) - N(Y) - \{Y\}$ such that Y and Z are dependent given $S \cup \{X\}$ for all $S \subseteq T$, where T is the smaller of $B(Y) - \{X, Z\}$ and $B(Z) - \{X, Y\}$.

4. [Remove Cycles]
Do the following while there exist cycles in the graph:

- Compute the set of edges $C = \{X \to Y \mid X \to Y \text{ is part of a cycle}\}$.
- Remove from the current graph the edge in C that is part of the greatest number of cycles, and put it in R.

5. [Reverse Edges]
Insert each edge from R in the graph in reverse order of removal in Step 4, reversed.

6. [Propagate Directions]
For all $X \in P$ and $Y \in N(X)$ such that neither $Y \to X$ nor $X \to Y$, execute the following rule until it no longer applies: if there exists a directed path from X to Y, orient $X \to Y$.

C.0.2. ***Iamb.fdr***.

Algorithm A2: Iamb.fdr algorithm [64]

1 $MB = \emptyset$
2 **for** i in $1..q-1$ **do**
3 $\quad p_i = pvalue(X \perp X_{(i)} \mid MB \setminus X_{(i)})$
4 **for** i in $q..1$ **do**
5 $\quad$ **if** $X_{(i)} \in MB$ **then**
6 $\quad\quad$ **if** $p_{(i)} \cdot \frac{q-1}{i} \cdot \sum_{k=1}^{q-1} \frac{1}{k} > \alpha$ **then**
7 $\quad\quad\quad MB = MB \setminus X_{(i)}$
8 $\quad\quad$ **go to** line 2
9 **for** i in $1..q-1$ **do**
10 $\quad$ **if** $X_{(i)} \notin MB$ **then**
11 $\quad\quad$ **if** $p_{(i)} \cdot \frac{q-1}{i} \cdot \sum_{k=1}^{q-1} \frac{1}{k} \leq \alpha$ **then**
12 $\quad\quad\quad MB = MB \cup X_{(i)}$
13 $\quad\quad$ **go to** line 2
14 **return** MB

C.0.3. ***PC.stable***.

Algorithm A3: PC.stable algorithm [40]

Form the complete undirected graph C on the vertex set V
Let $\ell = -1$;
repeat
  Let $\ell = \ell + 1$;
  for all vertices X_i in C **do**
    Let $a(X_i) = \mathrm{adj}(C, X_i)$
  end for
  repeat
    Select a (new) ordered pair of vertices (X_i, X_j) that are adjacent in C and satisfy $|a(X_i) \setminus \{X_j\}| \geq \ell$, using $\mathrm{order}(V)$;
    repeat
      Choose a (new) set $S \subseteq a(X_i) \setminus \{X_j\}$ with $|S| = \ell$, using $\mathrm{order}(V)$;
      if X_i and X_j are conditionally independent given S then
        Delete edge $X_i - X_j$ from C;
        Let $\mathrm{sepset}(X_i, X_j) = \mathrm{sepset}(X_j, X_i) = S$;
      end if
    until X_i and X_j are no longer adjacent in C or all $S \subseteq a(X_i) \setminus \{X_j\}$ with $|S| = \ell$ have been considered
  until all ordered pairs of adjacent vertices (X_i, X_j) in C with $|a(X_i) \setminus \{X_j\}| \geq \ell$ have been considered
until all pairs of adjacent vertices (X_i, X_j) in C satisfy $|a(X_i) \setminus \{X_j\}| \leq \ell$
return C, sepset.

C.0.4. ***HC***.

Algorithm A4: Hill-Climbing (HC) algorithm [66]

algorithm HC is
input: dataset D
output: DAG G
$G :=$ empty DAG ; `// DAG, initially empty`
repeat
  `// possible changes mustn't create a cycle, and can delete`
  `// or reverse arcs currently in` G`, or add an arc to` G
  `// variable` δ `holds score change for each possible arc change`
  for each possible arc change **in** G **do**
    if δ[change] needs calculating or recalculating **then**
      δ[change] $:= \delta(G,$ change, $D)$
  if $\max(\delta[\text{change}]) > 0$ **then**
    change := change corresponding to $\max(\delta[\text{change}])$
    $G := G +$ change
until $\max(\delta[\text{change}]) \leq 0$
return G

C.0.5. ***Tabu***.

Algorithm A5: TABU algorithm [66]

algorithm TABU is
input: dataset D
output: DAG G

$G :=$ empty DAG ; `// DAG, initially empty`
tabu_list $:= [\,]$; `// fixed length list of last DAGs visited`
repeat
`// possible changes as for HC except also the change`
`// cannot result in a DAG currently in tabu_list`
  for each possible arc change **in** G **do**
    if δ[change] needs calculating or recalculating **then**
      δ[change] $:= \delta(G, \text{change}, D)$
  change := change corresponding to $\max(\delta[\text{change}])$
  $G := G +$ change
  add G **to** *tabu_list*
until stop_condition ; `// e.g., limit on number of score decreases`
return G

C.0.6. ***MMHC***.

Algorithm A6: *MMHC* Algorithm [67]

procedure MMHC($\mathcal{D}$)
Input: data $\mathcal{D}$
Output: a DAG on the variables in $\mathcal{D}$
% Restrict
for every variable $X \in \mathcal{V}$ **do**
  $\text{PC}_X = \text{MMPC}(X, \mathcal{D})$
end for
% Search
Starting from an empty graph perform Greedy Hill-Climbing with operators *add-edge*, *delete-edge*, *reverse-edge*. Only try operator *add-edge* $Y \to X$ if $Y \in \text{PC}_X$.
Return the highest scoring DAG found
end procedure

C.0.7. ***Rsmax2***.

Algorithm A7: General 2-Phase Restricted Maximization (Rsmax2) algorithm [68]

Input:

A data set $D = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$

An initial network B_0

A decomposable score $\mathrm{Score}(B \mid D) = \sum_{i=1}^{P} \mathrm{Score}\big(X_i \mid \mathrm{pa}_B(X_i), D\big)$

A parameter k

Output: A network B

Loop for $n = 1, 2, \ldots$ until convergence

Restrict

Based on D and B_{n-1}, select for each variable X_i a set C_i of candidate parents with $|C_i| \leq k$

This defines a directed graph $H_n = (V, E)$, where $E = \{X_j \to X_i \mid X_j \in C_i \text{ for some } i\}$

(Note that H_n is usually cyclic.)

Maximize

Find a network $B_n = (G_n, \Theta_n)$ maximizing $\mathrm{Score}(B_n \mid D)$ among networks that satisfy $G_n \subseteq H_n$ (i.e. $\mathrm{pa}_{G_n}(X_i) \subseteq C_i$ for all $i = 1, \ldots, P$)

Return B_n