Universiteit Leiden

Master Computer Science

An Evolutionary Approach to Causal Structure Learning

Name: Nima Mehrafar
Student ID: 3848051
Date: 31/07/2025

Specialisation: Artificial Intelligence

1st supervisor: Mitra Baratchi
2nd supervisor: Saber Salehkaleybar

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Abstract

Learning the causal structure between variables from observational data and representing these in a Directed Acyclic Graph (DAG) is an important challenge within the field of Causal Structure Learning (CSL). Gradient-based methods, such as NOTEARS, optimize a continuous weight matrix. These methods, however, are prone to local optima entrapment, leading to long running times or subpar structural accuracy. In this thesis, we propose a family of Evolutionary Strategy (ES) algorithms to find the DAG that aims at matching the observational data the closest with continuous edge weights that emphasize global search to boost convergence and accuracy. Three algorithms have been introduced, namely a method making exclusive use of ES called Pure ES, and two hybrid methods that combine global exploration of ES with the strength of the gradient-based NOTEARS. We benchmark our methods against established methods such as NOTEARS, TOPO-NOTEARS, GES, PC, and DirectLiNGAM on synthetic and real-world data. Our findings show that the hybrid methods achieve similar SHD and edge weight estimation, while offering a significant reduction in running time, especially in larger graphs. Pure ES showed its ability to improve the SHD on the real-world data as well. In general, our results showcase the strength of evolutionary-based algorithms, in particular when used in combination with a gradient-based method.

1 Introduction

Causal Discovery (CD) is the act of identifying relationships between variables from observational data [1]. An important part of CD is learning the causal structure between variables by representing them in a Directed Acyclic Graph (DAG), called Causal Structure Learning (CSL) [2]. CD has applications in fields such as medicine and finance, by helping to answer questions such as what the effect of a medicine is, or what makes a stock price go up [3, 4]. The task of finding this true underlying DAG remains an NP-hard problem, nonetheless [5]. This is mainly because of the large search space, where the number of possible graphs grows exponentially with the number of variables, as well as constraints such as maintaining acyclicity, which does not permit the occurrence of cycles in the graph. Previous studies have aimed at constructing true DAGs primarily through domain knowledge, however, this task quickly becomes cumbersome with the growing amount of data [6]. Consequently, the current CD methods assume Structural Equation Models (SEMs) to define how variables are related to each other in a DAG, which by construction satisfy the Causal Markov Property, stating that all variables are independent of their non-descendants in the graph [7]. They then assume faithfulness, which ensures that the causal graph exactly represents the conditional independencies observed in the data. Subsequently, causal sufficiency ensures that there are no hidden confounders influencing the variables [8, 9].

Several algorithms have been designed within the field of Causal Discovery with respect to our prior assumptions, which can be mainly divided into four categories [10, 2]. Firstly, there are constraint-based methods, which prune edges from a fully connected graph to construct an optimal DAG by performing conditional independence tests [11]. A popular baseline constraint-based method is the Peter-Clark (PC) Algorithm [12]. The second category, score-based methods, assesses the quality of candidate DAGs with a certain score function, such as the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC), which return the DAG with the best score [13, 14, 15]. A well-known example of such an algorithm is called Greedy Equivalent Search (GES) [16]. Thirdly, there are the functional-based methods, which learn the relationship between variables through the usage of functions and statistical independence assumptions [10]. A method representing this group is DirectLiNGAM, which assumes a linear SEM with non-Gaussian, independent noise terms. [17]. Finally, the last category is gradient-based methods, which use a continuous optimization through gradient descent [2]

A prominent CD method, called NOTEARS, used a score-based approach for continuous optimization, thereby effectively combining both methods [18]. They formulated the problem of Causal Structure Learning as a differentiable optimization task by introducing a smooth acyclicity constraint. This constraint adds a penalty term to the loss and is set at exactly zero if the graph contains no directed cycles. This smooth formulation allows for the use of gradient-based solvers to optimize the DAG structure. This has a main advantage that it allows for optimization of all edge weights simultaneously, while maintaining the required acyclicity constraint. Edge weights enable the accurate estimation of how strong the causal effects are, which has applications in fields such as medicine for clinical decisions of treatments [19]. NOTEARS' results showcase significant improvements over baseline algorithms such as GES or PC on the syntactically generated data schemes called Erdo˝s–Rényi (ER) and Scale-Free (SF) [12, 16]. Downsides of their method, however, include that although the optimization formulation of the loss and smooth acyclicity constraint is differentiable, the problem remains non-convex, which has been shown to be more prone to getting stuck in poor local minima, possibly leading to suboptimal solutions and long running times [20, 21, 22]. Furthermore, NOTEARS suffers from scale sensitivity, meaning that a change in the units of the data can lead to significant changes in performance, especially in synthetic data. This is because NOTEARS' objective function will favour a larger number of variables disproportionately, making them dominate the optimization process. Normalizing this data has been shown to heavily damage their reported results [23]. This downside could lead to a lack of robustness when measurement units change. Finally, the authors in the NOTEARS paper make use of a matrix exponential with a high computational complexity of $O(d^3)$. This is because the acyclicity constraint involves computing the trace of the matrix exponential, which requires repeated matrix multiplications [18]. We will dive into the exact workings of the NOTEARS algorithm in the next section.

In this thesis, we introduce an Evolutionary Strategy (ES) method to find optimal DAGs while, like

NOTEARS, focusing on continuous real-valued search. ES is a type of Evolutionary Algorithm, which optimizes based on stochastic variation inspired by natural evolution, designed for continuous, real-valued optimization tasks, unlike other EA's such as Genetic Algorithms that encode solutions in binary [24, 25]. This contributes an evolutionary method that focuses on continuous global population-based search without the need for gradients. Furthermore, ES has the benefit of evaluating many solutions in parallel, providing a balance between exploration and exploitation. A large, diverse population is maintained to provide global exploration. Exploitation occurs as the best solutions are selected and recombined, making the population move towards optimizing for a set objective function. In general, ES has been proven effective for global optimization and searching widely in the search space, even in non-convex landscapes, thereby minimizing the local entrapment [26, 27]. Secondly, the usage of ES potentially opens the door to a wider range of possibilities, such as the usage of non-smooth functions, like BDe, which enables interesting experimentation that is not possible with gradient-based methods. The integration of domain knowledge is another powerful capability, which can aid the causal structure learning process in real-life cases by introducing simple optimization constraints relevant to the domain in question. Finally, ES does not suffer from scale sensitivity, as opposed to NOTEARS, as it does not make use of derivatives.

We will introduce two types of our ES method, a pure ES, and two ES-hybrids with NOTEARS. We aim to provide a method for global exploration of the search space in continuous DAG optimization through ES, while providing opportunities to integrate domain-specific constraints. The pure ES variant does this by exclusively applying evolutionary search on the continuous adjacency weight matrix, which contains the edge weights for every pair of variables, to find the DAG that matches the observational data the closest. This variant has the additional advantage of bypassing the $O(d^3)$ computational complexity of the matrix exponential that comes with NOTEARS' smooth acyclicity constraint, by instead using a depth-first search to find cycles and prune cycles away if needed. This decreases the complexity to $O(d^2)$ as we are working with an adjacency matrix [28]. The Hybrids, on the other hand, do preserve the precise matrix exponential method of NOTEARS by making use of two phases. In the first phase, the search space is explored through ES, and a solution is warm-started to be finished in the second phase by NOTEARS, which then refines the weights and removes possible cycles. Warm-starting is the task of using a solution from a previous model as initialization for the next model. This has been proven to often lead to accelerated convergence, as well as improved quality of the solutions, as the evolutionary exploration in the first phase can help against local entrapment [29]. In general, hybrids between evolutionary methods and gradient-based methods have been proven to be advantageous as well. Ali et al. discuss how a hybrid approach for solving linear equations can give faster and more accurate results compared to standard evolutionary algorithms and gradient-based methods [30]. Bashir et al. came to similar conclusions in their research on such hybrids, also called memetic algorithms, claiming positive effects on robustness for solving global optimization problems [31]. Although these papers did not focus on Causal Structure Learning, their contributions added to the literature on the hybridization of evolutionary and gradient-based methods within the field of vector-valued linear equations and global optimization, respectively.

In this research, we aim to provide answers to the following set of questions:

- **RQ1**: Can Pure Evolutionary Strategies (ES) serve as a competitive alternative to gradient-based Causal Structure Learning methods in terms of structural accuracy on synthetic and real-world data?
- **RQ2**: Can hybridizing evolutionary strategies with gradient-based optimization offer a balanced trade-off between structural accuracy and runtime on synthetic data, and higher structural accuracy on real-world datasets, compared to the state-of-the-art methods?
- **RQ3**: How does the performance of our hybrid methods compare to state-of-the-art methods across different settings regarding graph sizes, graph models, and data sizes, on synthetic data?
- **RQ4**: How do our methods compare in edge weight estimation in comparison to the state-of-the-art methods?

We hypothesize that by employing an Evolutionary Strategy for continuous CSL, a more global exploration of the search space can be conducted. Especially, in more challenging landscapes arising from larger and

denser graphs, thus leading to a more robust optimization of DAG estimation over gradient-based methods across both synthetic and real-world data. We believe this to be mostly true for the hybrid models, where the strength of both methods can be combined, aiding the search in more challenging optimization landscapes. The Pure ES method might suffer from longer running time due to its iterative nature, however, with potentially an improved structural accuracy due to global exploration abilities.

Our main contributions in this thesis are as follows:

- The introduction of an Evolutionary Strategy method for finding the optimized DAG structure with real-valued edge weights
- The development of ES-Hybrid methods, consisting of two phases that combine evolutionary and gradient-based methods
- A comprehensive experimental setup comparing our methods to other baseline methods on synthetic and real-world data across different graph sizes, data sizes, and graph models.

In the following sections, we will dive deeper into the required background and related work supporting our research. Next, our methods will be described, as well as our experimental setup, measuring the performance of our work against the state-of-the-art methods. Finally, the results will be presented, discussed, and concluded.

2 Background

In this section, we will lay out important concepts on which the remainder of the thesis is built.

2.1 DAGs

A *Directed Acyclic Graph* (DAG) is a graph $G = (V, E)$, where V denotes the set of nodes or variables, and E represents the set of directed edges between these nodes $E \subseteq V \times V$, with $\forall v \in V : (v, v) \notin E$ excluding the possibility of self-loops to ensure acyclicity [32]. DAGs represent relationships between variables in CD and are typically represented as adjacency matrices during the learning process. Given a data matrix $\mathbb{X} \in \mathbb{R}^{n \times d}$, consisting of observational data with n data samples and d variables, we aim to learn the DAG G from the discrete search space $\mathcal{D} = \{W \in \mathbb{R}^{d \times d} \mid W \text{ is acyclic}\}$. $\mathbb{W} \in \mathbb{R}^{d \times d}$ is the weighted adjacency matrix, which shows the causal effect of i on j in W_{ij}, called edge weights. The DAG G is learned by extracting the binary adjacency matrix A from W, setting $A_{ij} = 1$ if $|W_{ij}| > \tau$ and $A_{ij} = 0$ otherwise, where $\tau > 0$ is a predefined threshold. The data matrix $\mathbb{X}$ is modelled via a Structural Equation Model (SEM), which we will discuss in Section 2.2 [18].

2.2 Structural Equation Models

Structural Equation Modelling (SEM) is a statistical approach used to test hypotheses about the relationships between variables within CD. A SEM can help us to understand how observed variables relate via a system of equations [33]. In this thesis, we focus on linear Gaussian SEM, which describes how each variable is modelled as a linear combination of its parent nodes [34]. We learn the weight matrix $\mathbb{W} \in \mathbb{R}^{d \times d}$ for a DAG G with variables $\{X_1, \ldots, X_d\}$, with each node X_j being a linear combination of its parent nodes with the weights W_{ij}, signifying the strength between the connection of the nodes X_i to X_j. An added random noise z_j drawn from a Gaussian distribution is added: [35]:

$$X_j \;=\; \sum_{i=1}^{d} \mathbf{W}_{ij}\, X_i \;+\; z_j\,. \tag{1}$$

SEMs will play a pivotal role in the generation of observational data in our synthetic data experiments. We choose linear Gaussian SEMs because methods like NOTEARS and its variants derive their loss function from the linear Gaussian SEM formulation, ensuring that minimizing this loss is exactly maximum-likelihood estimation under the SEM's data generation [18, 2].

2.3 Evolutionary Algorithms

Evolutionary Algorithms (EA) describe population-based optimisation algorithms that mimic natural evolution [27]. It starts with an initial set of solutions, called the population. A selection mechanism will choose the best solutions, based on a domain-specific fitness function that measures the strength of each solution. The chosen solutions, also referred to as the parents, will go through a phase of recombination to produce offspring. Here, features of both parents are combined to form the child. Besides recombination, a phase of mutation is added to the individuals of the population to increase diversity in the so-called "gene pool". Similarly, to human evolution, where "survival of the fittest" applies, only the individuals with the best fitness scores are chosen to be part of the next generation and recombine together to create offspring. This evolutionary cycle of selection, recombination, and mutation is repeated over multiple generations, progressively improving the population's fitness. The usage of such methods is proven to find good solutions close to the global optimum as they have the advantage of combining good solutions to possibly obtain better ones [36]. In this thesis, we use Evolutionary Strategies, which is a type of EA, specialized in the optimization of continuous values. The recombination and mutation mechanisms in ES differ from other EA's, such as bit-string Genetic Algorithms, where each candidate solution is encoded in binary. Whereas in classical EAs, recombination happens by mixing the bit-strings of two parent solutions, in ES, the average parameters of

the parents are taken to create the offspring. Mutation, on the other hand, occurs by applying Gaussian noise to the solution parameters, instead of applying bit-flips.

2.4 NOTEARS

The NOTEARS paper plays an important role in our research as we aim to build upon it and compare our results against it. Therefore, a thorough background knowledge of their method is required. NOTEARS estimated the structure of DAGs by formulating the learning process into a continuous optimization problem, rather than the traditional combinatorial approach [18]. This was realized by building upon the linear Gaussian SEM of Section 2.2 (see Eq. 1), as minimizing the Least Squares (LS) loss is equivalent to finding the maximum likelihood estimate of the weight matrix (W) under the linear Gaussian SEM. The formula of the LS loss is the following:

$$F(W) = \frac{1}{2n}\|X - XW\|_F^2 + \lambda\|W\|_1 \tag{2}$$

Here, each row of the data matrix $X \in \mathbb{R}^{n\times d}$ is multiplied by the weights matrix to produce the predicted values. Next, the residual matrix is calculated by subtracting the predicted values from the observed data, which quantifies the prediction errors made. The Frobenius norm and average are taken from the residual matrix to return the sum of squared errors across all n data samples and d variables. An l_1 penalty term is added to ensure sparsity. Besides the objective function, the assurance of acyclicity is crucial as well. For this reason, acyclicity is defined as a constraint, h, where every entry of the weight matrix is squared element-wise and the matrix exponential is taken. The trace of this matrix exponential sums up the total weight of all walks that start and end at the same node (see Eq. 3). Subsequently, the trivial loops are removed by subtracting the number of variables d from this sum. This leaves us with a number of directed loops, which have to be set to zero.

$$h(W) \;=\; \mathrm{tr}\big(e^{\,W\circ W}\big) \;-\; d. \tag{3}$$

Finally, this objective function F and acyclicity constraint h were combined to form the constrained optimization problem:

$$\min_{W} F(W) \quad \text{subject to} \quad h(W) = 0. \tag{4}$$

This problem was solved by making use of the Augmented Lagrangian Function of the following form:

$$L^p(W,\alpha) = F(W) + \frac{\rho}{2}h(W)^2 + \alpha h(W). \tag{5}$$

Iteratively, two optimization steps are taken each time. First W is updated through gradient descent to minimize the augmented Lagrangian $L^p(W,\alpha)$ by applying the quasi-Newton method for optimization called L-BFGS. Next, the Lagrange multiplier α is adjusted to ensure satisfaction of the acyclicity constraint, while the penalty weight ρ, which sets how strong violations of the acyclicity are penalised, remains fixed:

$$\alpha \leftarrow \alpha + \rho h(W^*). \tag{6}$$

The same process is repeated until convergence to a good data fit and satisfaction of the acyclicity constraint.

2.5 Evaluation Metrics

The CSL algorithms are evaluated on synthetically produced observation data through SEMs and real-world data. The synthetic data has the advantage of offering a ground truth as the exact causal structure and parameters of the data-generating model are fully known. The real-world data suffers more from a lack of ground truth. However, thanks to domain expertise and experimentation, a form of ground truth has been established [37]. The created DAGs and the ground-truth are compared to each other, and the following metrics are measured:

- **Structural Hamming Distance (SHD)**: The total number of different edges between the estimated graph and ground truth when only applying additions, deletions, and reversals.

- **True Positive Rate (TPR)**: The ratio of correctly estimated edges to the total number of true edges that exist in the ground truth DAG.
- **False Discovery Rate (FDR)**: The ratio of falsely identified edges in the estimated graph among all identified edges in the graph.
- **Running Time (RT)**: The time required for the CSL algorithm to find the optimal DAG, measured in seconds (s)
- **Mean Squared Error (MSE)**: The MSE quantifies the average squared difference between the predicted values and the actual observed values. It serves as a method to assess the edge weight estimation abilities of our methods.

These metrics will determine the strengths of the algorithms tested in the experimental setup later in this thesis.

3 Related work

In the following section, we review related literature to this research and discuss the strengths and weaknesses of each of these methods in the field of CSL. These works contribute to our current understanding of the field and help in the experimental design presented later in this thesis.

3.1 Causal Discovery methods

Important algorithms frequently used for comparisons as baseline algorithms within most CSL literature, that do not make use of continuous optimization, are Peter Clark (PC), Greedy Equivalence Search (GES), and DirectLiNGAM [12, 16, 17]. These papers are considered as baselines, because of their role as representatives of the constraint-, score-, and functional-based methods. The constraint-based method called PC starts with a complete undirected graph and uses conditional independence tests to prune edges, categorizing itself as a constraint-based method. PC returns a CPDAG, which contains both directed and undirected edges that do not indicate the direction of the edges. The directed edges show the orientation that holds in every DAG of the Markov equivalence class, which is the family of DAGs over the same variables that entail the same conditional-independence relation. A downside of the PC method is the worst-case running time, which is exponential to the number of variables. This makes the PC algorithm unsuitable for larger datasets [38].

GES, on the other hand, is a score-based method that starts by adding edges greedily to improve a score function such as BIC in the forward phase until no improvements can be made [16]. Afterwards, in the backward phase, edges are iteratively removed to further improve scores until no deletion can further improve the score. A downside of this method is that its greedy nature increases the odds of getting stuck in local optima [16, 39].

Finally, the functional-based method called DirectLiNGAM estimates the causal ordering of variables' non-Gaussianity, without the need for iterative search in the parameter space [17]. They do this by first identifying a variable that is not influenced by others. This variable is then removed, and the procedure is repeated until the causal structure has been found. A downside of this method is that the choice of hyperparameters can greatly impact performance [17].

In general, these algorithms form important baselines within the CSL literature, with many available implementations. Our Pure ES algorithm could be considered within this group that does not rely on continuous optimization, as well as a score-based method. Therefore, we will include these baseline methods together with Pure ES to provide comprehensive comparisons.

3.2 Continuous Optimization methods

Continuous optimization methods or gradient-based methods have the strength of estimating all edge weights at the same time through gradient-based solvers such as L-BFGS, often resulting in fast and accurate structure search. The aforementioned NOTEARS is an important method with a strong contribution to this group as it aims to learning DAGs through continuous differentiable optimization, with the addition of a smooth acyclicity constraint to combat combinatorial search [18]. In their experiments, they tested the performance of their method against baseline algorithms such as the Greedy Equivalent Search (GES), PC, and LiNGAM on synthetic and real-world data, as well as testing different parameter settings regarding regularization and sample sizes [40, 12, 41]. The results showed that NOTEARS outperformed the baseline algorithms on both synthetic and real-world data, achieving lower Structural Hamming Distance (SHD) and False Discovery Rate (FDR). The discussed downsides, however, include the local entrapment risk, especially experienced with non-convex optimization, as well as the high computational complexity carried by the matrix exponential [18]. The NOTEARS algorithm will form an important part of this thesis, as we aim to compare our methods against it, as well as build upon NOTEARS with our hybrids.

The NOTEARS algorithm formed the basis of research for other algorithms as well, which aim to extend upon its findings. One of these algorithms, called TOPO-NOTEARS, focuses on the lack of global

optimum guarantees NOTEARS provides due to the non-convexity of the optimization problem [42]. Therefore, the authors propose a bi-level optimization method. The outer level of this algorithm is responsible for performing topological swaps, where nodes defining the order of a DAG are iteratively substituted. The inner level takes the set topological order and aims at optimizing the objective function (see equation 2), without the need for acyclicity constraints, as this is already guaranteed in the outer level. In general, the algorithm iteratively improves the topological order, guided by KKT (Karush–Kuhn–Tucker) conditions, and re-optimizes the set topological order within the inner-level. Two versions of their method were created by the authors, one called Random TOPO, which starts with a random topological order, and the better-performing TOPO-NOTEARS, which starts with the initialization of NOTEARS and then performs the bi-level optimization afterwards. the latter has performed the best out of the two, compared against other state-of-the-art algorithms such as GOLEM, NOTEARS, and NOFEARS [22, 18, 43]. Their results show significant improvements in SHD over the other models. Downsides of this method include the significant increase in running time as the search space increases. We will include TOPO-NOTEARS in our experiments to compare performances.

Similarly to TOPO-NOTEARS, a method called CL-NOTEARS [2] builds upon NOTEARS by focusing on the mitigation of the data sampling noise and the susceptibility of NOTEARS to converging to local minima. The CL in their name stands for curriculum learning, as it treats the learning process, where the model starts by learning from simpler, lower-noise data samples before progressively being given more challenging data, similarly to how humans learn tasks. In their experiments against baseline methods, including NOTEARS, Gran-DAG, and DirectLiNGAM [18, 44, 17], they report lower SHD and FDR, highlighting the benefits of curriculum learning. A downside mentioned by the authors is the lack of a more rigorous noise-check per noisy samples to prevent the occurrence of highly noisy samples during its curriculum stages. Currently, the determine which samples are noisy by measuring the model's loss on each sample. Furthermore, despite its curriculum mechanism, CL-NOTEARS still inherits NOTEARS' non-convex optimization challenges.

Another algorithm that adds upon the identified shortcomings of NOTEARS is called GOLEM [22]. They examined the acyclicity and sparsity constraints in NOTEARS and stated that these might lead to difficulties, such as the penalty coefficient going to infinity. Therefore, the authors formulate the structure learning task into a log-likelihood-based function with the addition of fixed weight terms for sparsity and acyclicity. These softer penalties lead to easier solvable unconstrained optimization problems, unlike the hard constraints of acyclicity and sparsity in NOTEARS through the augmented Lagrangian. GOLEMs objective function incorporates a log-determinant meant to discourage the formation of cycles in DAGs. Their results showcase lower SHD in comparison to NOTEARS on the synthetic data and comparative results on real-world data. Downsides of this method include long running times, high sensitivity to hyperparameters, and a non-convex objective that the optimizer may get stuck in poor local minima [22]. Although GOLEM and CL-NOTEARS are not evaluated in our experiments, because of a lack of available implementations and the choice to only pick the most recent NOTEARS variants due to limited computational resources, their contributions remain relevant as they represent an alternative direction in the causal discovery literature.

This current thesis will build upon NOTEARS as well through the introduction of our hybrid models. Our main goal is to boost global exploration through the usage of ES and minimize the risk of falling into local minima, especially within non-convex landscapes, to accelerate running times and improve structural accuracy. This is vastly different than CL-NOTEARS and GOLEM, which focus on mitigating sampling noise and sparsity constraints, respectively. TOPO-NOTEARS does share a similar as our hybrid models, however, their methodology differs significantly from ours, as they focus on topological swaps, while our focus is on ES. Nonetheless, it does open up for an interesting comparison between our methods, which will be discussed in Section 6.

4 Methodology

In this section, we formalize our proposed methods based on Evolutionary Strategies and provide details of our implementation of our Pure ES method and the two hybrids with NOTEARS.

4.1 Pure Evolutionary Strategies

We propose an Evolutionary Strategies (ES) algorithm with the objective of learning DAGs given the provided observational data. ES is an evolutionary algorithm, capable of optimizing a set of continuous values through the generational cycle of mutation, recombination, and selection. An adaptive mutator is added as well to aid more direct search space exploration. In Pure ES, each individual is represented by a weight matrix $\mathbb{W} \in \mathbb{R}^{d\times d}$, with non-zero entries above a set threshold being defined as an edge between the two variables. Each individual is initialized with random entries. All individuals are evaluated on their fitness, indicating the strength of their solution in explaining the observed data as closely as possible. In this model, we make use of the Negative Log-Likelihood (NLL) as our fitness function, whose value has to be minimized. We choose this function because, under the linear-Gaussian SEM assumption, minimizing the NLL is equivalent to performing maximum likelihood estimation, which can estimate parameters efficiently [45]. NLL works by predicting for each existing node, the value X_j from its parents X_i in the data matrix $\mathbb{X} \in \mathbb{R}^{n\times d}$, where $i = 1, \ldots, n$ indicates samples and $j = 1, \ldots, d$ specifies variables. Parents are found by firstly binarizing the weight matrix $\mathbb{W} \in \mathbb{R}^{d\times d}$, where edges above a certain edge threshold are converted to 1, indicating a relationship between the variables. We then assume a linear-Gaussian model where each X_j is generated by a linear combination of its parents, an intercept β, and Gaussian noise with zero mean μ and variance σ_j^2 [46]. Therefore, the likelihood of observing X_j is defined as

$$p\big(\mathbf{X}_j \mid \beta, \sigma_j^2\big) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\,\sigma_j^2}} \exp\Big(-\frac{(X_{ij} - \hat{X}_{ij})^2}{2\,\sigma_j^2}\Big), \;\; \mathbf{X}_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^\top, \tag{7}$$

with $\hat{X}_{ij}$ being the model's prediction for the i-th sample of variable j, while X_{ij} is the actual observed value for that sample and variable. Subsequently, the negative logarithm is taken with the Sum of Squared Errors (SSE), with SSE defined as $\text{SSE} = \sum_{i=1}^{n}(X_{ij} - \hat{X}_{ij})^2$, quantifying the overall prediction error for a node:

$$\text{NLL}_j = \frac{n}{2}\,\log\left(2\pi\,\sigma_j^2\right) + \frac{\text{SSE}_j}{2\,\sigma_j^2} \tag{8}$$

Next, for each node j we fit an ordinary least-squares regression of X_j to its parents for estimating the regression coefficients β and to compute the residuals. These residuals aid the computation of the Maximum Likelihood Estimate (MLE) for the variance σ_j^2, which can then be plugged back into the Gaussian NLL formula to calculate the NLL [47]. This procedure is repeated for each node and summed up to obtain the total NLL to indicate the solutions' fit to the data. Our evaluation function to assess the quality of each solution is based on a combination of NLL and an added penalty, which is proportional to the number of edges, which is multiplied by a regularisation parameter λ to promote sparsity.

The search for the optimum is executed through a search strategy of altering the population by changing the edge weights of each individual's weight matrix towards a direction that minimises the evaluation function. This is realised by firstly selecting two individuals for recombination, where a uniform crossover of the elements in both parents is passed on to the newly created candidate solution, called the child. This is done by randomly picking parameters from a parent to create a child that is a mix of both parents. Next, the children are mutated as each edge is chosen randomly by some probability ϵ. The chosen edges are altered by adding Gaussian noise to them, controlled by a parameter σ, which defines the standard deviation of the Gaussian noise. All realised mutations are checked for their influence on the acyclicity within the individual, as they are required to produce valid DAGs that do not contain any cycles. For this reason, a Depth-First-Search is performed on the binary matrix to check the existence of any added cycles and possibly remove the observed cycle with the smallest edge weight. This binary matrix is acquired by thresholding the continuous weight matrix, where edge weights that exceed the threshold are set to 1 and others to zero.

This binary matrix then reveals the DAG structure of this individual, which then facilitates the search for possible cycles. Both the mutation and recombination phases are responsible for diversifying the population. The population with increased diversity is subsequently ranked by the evaluation function, and the worst-ranked individuals are removed, with only the "elites", as well as the best children, being saved for the next generation, where the same procedure repeats. Elitism is a strategy in which a fixed number of the overall best solutions are carried over to the next generation, and not replaced by the children [48]. In general, this evolutionary process is constructed such that the population moves towards optimising the fitness function, with the addition of diverse individuals making it possible to explore the landscape more efficiently, enabling the possibility of getting out of local optima.

The search mechanism is guided by a set of hyperparameters controlling which edges are mutated ϵ, and the mutation strength, determined by the amount of Gaussian noise added to them σ. To control the size of the mutations, the mutation strength is updated dynamically across generations through an adaptive mutator, which can increase or decrease depending on the added Gaussian noise. A set hyperparameter τ sets the standard deviation of the Gaussian distribution we sample from, as we stochastically update the mutation strength σ by multiplying it by $\exp(\tau)$ to increase diversity in the step sizes for a more global exploration. The idea of self-adaptation is a common strategy in Evolutionary Strategies [49].

Finally, the explained evolutionary process is cut into two phases. A short exploration phase, which makes up 5 percent of the total number of generations, where the hyperparameters responsible for the mutations σ and ϵ, as well as the population size, are set higher to encourage more occupation of the search space. Afterwards, the population size and mutation hyperparameters are scaled down to control the search more with shorter steps towards the optimum. The full pseudocode is described in Algorithm 1. In this pseudocode, we can see the previously described inputs of the method listed as requirements. The algorithm starts by initialising the population of weight matrices, evaluating this and storing the best found DAG with the lowest score. Next, for the first 5 percent of the maximum number of generations, set by κ, we will start an exploration phase where we use a larger population size, and higher values for σ and ϵ. This promotes additional randomness to start with a diverse population. Afterwards, we move to the refinement phase where the population size, σ and ϵ are set lower and only the best solution candidates are taken to this phase. Within each phase, we go through the steps of recombination, mutation, and selection (see lines 15-24). In lines 15 and 16, the population at time step t, goes through recombination and mutation to create the children, P"(t). Notice how line 16 uses σ and ϵ to signify the strength of the mutation. Lines 17-20 show the DFS check that is done after the mutation, with the possible repair. Next, the fitness of each individual is checked through the fitness function. Afterwards, the self-adaption parameter τ log-updates σ in line 23. We end by selecting the next generation in line 24, where a combination is taken of the top α parents ('elites') in P(t), together best individuals from the union of the offspring and other parents. Finally, we update the best DAG found and go to the next generation until termination or until the maximum number of generations is reached.

Algorithm 1 Pure Evolutionary Strategy (Pure ES)

Require: *popSize*, *maxGenerations*, mutation strength σ, mutation chance ϵ, step-size adaptation rate τ, edge threshold τ_{edge}, phase-change κ set at 0.05, elite size α
fitnessFunction$(\cdot)$: Gaussian NLL with sparsity penalty λ; uses τ_{edge} to binarise W
DAG_Repair$(\cdot)$: cut weakest edge to break cycles
cycleCheck$(\cdot)$: depth-first search (DFS) for cycle detection
Ensure: bestDAG: best DAG found

$t \leftarrow 0$
$P(t) \leftarrow$ *Initialize(popSize)*
Evaluate $P(t)$ using *fitnessFunction*
Update **bestDAG**

while not termination criterion **and** $t <$ *maxGenerations* **do**
  // Phase 1: Exploration
  if $t < \kappa \times$ *maxGenerations* **then**
    Use larger population and higher mutation rate
  end if
  // Phase 2: Refinement
  if $t \geq \kappa \times$ *maxGenerations* **then**
    Use smaller population and lower mutation rate
  end if

  $P'(t) \leftarrow$ *Recombine*$(P(t))$
  $P''(t) \leftarrow$ *Mutate*$(P'(t), \sigma(t), \epsilon)$ ▷ self-adaptive mutation
  for all offspring $i \in P''(t)$ **do**
    if *cycleCheck*$(iDAG)$ **then**
      $iDAG \leftarrow$ *DAG_Repair*$(iDAG)$
    end if
    $iscore \leftarrow$ *fitnessFunction*$(iDAG)$
  end for

  $\sigma(t{+}1) \leftarrow \sigma(t)\,\exp\big(\tau\, N(0,1)\big)$ ▷ self-adaptating mutator
  $P(t{+}1) \leftarrow$ *Select*$\big(topP(t) * \alpha \cup (P(t) \cup P''(t))\big)$ ▷ elitism keeps top performers
  Update **bestDAG** with best in $P(t{+}1)$
  $t \leftarrow t + 1$
end while
  return bestDAG

4.2 Hybrid

In this research, we define hybrid approaches that are designed to combine the global exploration capabilities of Evolutionary Strategies with the refinement methods proposed by NOTEARS. We propose two hybrids, namely the Pure ES-Hybrid and the Vanilla ES-Hybrid. The former is meant to explore the abilities of a sophisticated evolutionary method with NOTEARS, where the first phase is designed to independently produce suitable, valid DAGs, only for NOTEARS to fine-tune the solutions. On the other hand, the Vanilla ES Hybrid performs reliably on NOTEARS and is designed to aid NOTEARS in broader search space investigation by employing a relatively simple evolutionary search. Vanilla ES is solely designed for exploration and does not perform acyclicity checks, making it dependent on NOTEARS in the second phase.

4.2.1 Pure ES-Hybrid

The hybrid that combines the Pure ES method with NOTEARS aims to explore the search space briefly in the evolutionary phase, initializing the solution to a valid DAG that aims to return the true underlying DAG. Pure ES is expected to give an already acceptable solution, before handing it over to NOTEARS for edge weight refinement and further acyclicity removals where needed. This is essentially the idea of warm-starting, which refers to the act of using a previously computed solution as a starting point for an algorithm, with the aim of improving convergence and performance [50]. The pseudocode of this method is shown in Algorithm 2. Here, the same Evolutionary Strategies is applied as in Algorithm 1, but with a smaller population size and fewer generations, as the goal is to warm-start the solution and not to run the entirety of the algorithm. Next, this solution is refined by NOTEARS, which runs with the same settings as in the original paper. The intuition is that the warm-starting phase could improve performance, as well as running time, especially in landscapes where local optima entrapment plays a larger role.

Algorithm 2 Pure ES-Hybrid

Require: *popSize*, *maxGenerations*, *See other hyperparameters in Algorithm1*
Ensure: **bestDAG**: Final DAG

//Phase 1: ES
    bestDAG ← Run(Algorithm 1 with smaller popSize & fewer generations) ▷ ES explores

Phase 2: Final Refinement with *NOTEARS*
    bestDAG ← *NOTEARS*(**bestDAG**, NOTEARS parameters)

return bestDAG

4.2.2 Vanilla ES-Hybrid

The Vanilla ES Hybrid makes use of a simpler evolutionary method, fully focusing on exploring the search space without regard to acyclicity constraints. This algorithm starts with an all-zero initialized weight matrix. Next, a population of weight matrices is made by applying small Gaussian perturbations to the initialized weight matrix, after which, through elitist selection, the best-performing individuals are retained. The evaluation of the individuals is executed based on the Least Squares Loss, inspired by NOTEARS. Finally, the centroid of the best-performing individuals is taken as the new weight matrix to move more in the direction of a more promising region according to the loss function. Vanilla ES-Hybrid takes inspiration from current research on Evolutionary Algorithms, where elitism and the intermediate recombination of all parents are applied [49]. The pseudocode can be found in Algorithm 3. It shows how a weight matrix W is first initialized to zeros, and the number of elites is determined by the α parameter. Next, for generations, the population is perturbed with σ controlling the mutation strength. The evaluation takes place by using the Least Squares Loss with an added sparsity penalty, controlled by λ. Finally, the mean of the elite solutions is taken to create the new weight matrix W, ready for the next iteration. The final solution is refined by NOTEARS. An important difference between Pure ES Hybrid and this hybrid is that this hybrid integrates ES within the NOTEARS framework, whereas Pure ES is an independent method.

Algorithm 3 Vanilla ES-Hybrid

Require: population size P, elite fraction α, mutation scale σ, number of generations G
Ensure: **bestDAG**: Final DAG

//Phase 1: ES

$W \leftarrow \mathbf{0}_{d \times d}$ ▷ Initialize weight matrix
$\mu \leftarrow \lfloor \alpha \times P \rfloor$ ▷ Number of elites
for gen $\leftarrow 1$ **to** G **do**
  for $i \leftarrow 1$ **to** P **do**
    $\text{candidate}_i \leftarrow W + \sigma\,\mathcal{N}(\mathbf{0}, I_{d \times d})$ ▷ Sample around W
    $\text{score}_i \leftarrow \text{Loss}\big(\text{candidate}_i, X\big) \;+\; \lambda_1 \,\|\text{candidate}_i\|_1$ ▷ ES objective
  end for
  $\text{eliteIndices} \leftarrow \text{argsort}\big(\{\text{score}_i\}\big)[1:\mu]$ ▷ Pick top μ
  $\text{elites} \leftarrow \{\text{candidate}_j \mid j \in \text{eliteIndices}\}$
  $W \leftarrow \frac{1}{\mu} \sum_{\text{cand} \in \text{elites}} \text{cand}$ ▷ Average the elites
end for

//Phase 2: NOTEARS
bestDAG $\leftarrow$ *NOTEARS*(W) ▷ Refinement and cycle removal
return bestDAG

5 Experimentation

In the following section, we outline our experimental setup in which we aim to evaluate our Pure ES and hybrid methods against other established methods within the relevant CSL literature, to test our hypothesis and answer our research questions. The performance measurements will allow comparisons between the trade-offs between structural accuracy and running time among the methods. Furthermore, the edge weight accuracy of the methods focusing on optimizing continuous values will be measured.

5.1 Datasets

Our experiments will be conducted on two types of data. Firstly, the synthetic data, which has been generated through modelling and has the availability of the ground truth as the main benefit. Secondly, we have real-world data to prove the practical applications of our causal discovery models.

5.1.1 Synthetic Data

Synthetic data will be generated through the usage of two canonical random graph models, called Erdős–Rényi (ER) and Scale-Free (SF). The ER model constructs DAGs by connecting each pair of nodes with a uniform probability drawn from a Bernoulli distribution with edge probability p [51]. Here, a random topological ordering of the variables is sampled to guarantee acyclicity. Next, each potential edge among the d variables is added with a probability of p, such that the has an expected number of k edges [52]. Scale-Free (SF), on the other hand, follows a power law to determine the distribution of edges and nodes in the network [53]. Both of these graphs are constructed with a specified expected number of edges kd to simulate different levels of network connectivity. Next, a linear SEM is used where each variable in the graph is matched with an equation that indicates how that variable depends on its parent nodes. Subsequently, the observational data is simulated by sampling from the SEMs, as for each variable a value is computed with respect to its parents, according to the equation (see Eq. 1). The added noise term drawn from a probability distribution will be Gaussian in our study [54]. In this thesis, we will test our models on ER-2 and SF-4, indicating an expected number of edges of 2 * d and 4 * d, respectively. Details about the amount and types of graphs that will be used in this research will be discussed in Section 5.2.

5.1.2 Real-world Data

The real-world data used in our experiments is the Sachs Protein Data [37]. This data defines the relationship between 11 proteins to knock-outs and spiking among them. The data consists of measurements of protein levels in human cells with 7466 observations and 20 edges. Experimental works have created a network, known as the *golden standard model*, to be used as ground-truth for model validation. As mentioned before, one of the advantages of ES is its ability to integrate domain knowledge into the optimization process of Causal Structure Learning. For this reason, the Pure ES methods have an extra constraint in their search, restricting the in-degree among the nodes to more than 2. Relevant literature within the community has shown that in biology, the protein networks are typically small, with most genes being influenced by a small number of regulators, making the resulting DAG typically sparse [55, 56].

5.2 Experimental set-up

We will conduct our experiments on the synthetic data against a group of baseline algorithms on four different variable sizes d = {10,20,50,100}, data sizes N = {1000, 20}, and two graph models G = {ER-2, SF-4}. This experimental setup is standard in the continuous optimization CSL literature and enables a thorough comparison between the methods [2, 18, 42]. Furthermore, this set-up facilitates the analysis of the performance of the methods on increasingly challenging problems, where we search for larger DAGs with more variables d, or denser graphs. Denser graphs are defined as graphs with a larger in-degree, indicating more edges between the nodes. For example, SF-4 is considered to be denser than ER-2 with its expected in-degree of 4. The availability of abundant data N is an important indicator of performance as well, as data might be expensive to gather in some domains. Finally, the experimentation on both types of graph models G provides as test on the generalization of the methods, as these graph models are generated uniquely, with

SF-4 being extra challenging as in these graphs a few nodes often hold an extra high in-degree, meaning they serve as hubs with many connections [53]. ER-2, on the other hand, has a uniform distribution of in-degree among the nodes [51]. For all experiments, we make use of the Gaussian SEM, where each variable is a linear combination of its parents plus independent Gaussian noise. This is consistent with experiments in related papers, such as NOTEARS [18]. Since our focus is on evaluating performance across different variables and sample sizes, we restrict our experiments to Gaussian SEMs to avoid large increases in experimental complexity that have no direct purpose for our main research objectives.

The algorithms will mainly be evaluated by measuring the Structural Hamming Distance (SHD), as well as the Running Time (RT). The latter can vary per device. Therefore, it is important to mention that all experiments were conducted on a laptop running Windows 11 Pro, equipped with an 11th Gen Intel Core i7-1185G7 CPU at 3.00 GHz and 32 GB of RAM. No GPU acceleration was used. Furthermore, each experiment was executed 10 times on different seeds, and the results were averaged.

Besides the synthetic data, the methods will also be evaluated on their performance on the real-world protein dataset Sachs, where their SHD is measured [37].

5.3 Baselines

The algorithms used in this experimental setup include:

The gradient-based methods

- NOTEARS, serving as our main comparison paper, as it's the paper the hybrids are based on [18].
- TOPO-NOTEARS, a variant of NOTEARS, making use of bi-level optimization, where the outer level performs topological swaps and the inner level optimizes the objective function with this set order [42].

The classical baseline methods

- GES, a greedy score-based method that adds and removes edges iteratively to improve the score [16]
- PC, a constraint-based method that prunes edges from a complete undirected graph through conditional independence tests [12]
- DirectLiNGAM, a baseline algorithm that recovers a causal ordering in linear non-Gaussian SEMs by iteratively extracting variables whose regression residuals are independent of the rest [17].

Our methods

- Pure ES, our method that makes exclusive use of Evolutionary Strategies to recover the correct DAG
- Pure ES - Hybrid, a combination of Pure ES and NOTEARS
- Vanilla ES - Hybrid, a simple ES method for global exploration in combination with NOTEARS.

The goal is to provide a mix of established baseline algorithms with PC, GES, and DirectLiNGAM, together with NOTEARS and a recently published peer-reviewed variant.

5.4 Hyperparameters

The selection of hyperparameters plays a pivotal role in the performance of algorithms. The aforementioned hyperparameters that are used in Pure ES (see Algorithm 1), and in Vanilla ES Hybrid (see Algorithm 3) play a significant role in changing the search dynamics as they allow for a controlled balance between exploration and exploitation. Hyperparameter tuning in evolutionary algorithms remains a challenge in evolutionary strategies due to the non-stationary, stochastic nature of the search [57]. For this reason, searching for optimal hyperparameters is considered a niche of research by itself, within the field of Automated Machine Learning (AutoML) and is therefore out of the scope of this research [58, 59]. The optimal hyperparameter settings

most likely differ per landscape. Nonetheless, the decision was made to use a fixed set of hyperparameters for this research. This set was selected based on experimentation across a small set of runs and is believed to produce good results. The tuning of the hyperparameters for each landscape would most likely produce better results. However, this was not realized because of the size of this search.

5.4.1 NOTEARS and TOPO-NOTEARS

The hyperparameters in the NOTEARS include the maximum number of iterations, set at 200. The convergence tolerance on the acyclicity constraint, standard, is set at 1e-8. The upper-bound on the augmented-Lagrangian penalty parameter, standard at 1e+16, and its edge threshold is at 0.3. Finally, the authors introduce two variants of their NOTEARS version, one with the sparsity penalty set at 0, and the other set at 0.1. The NOTEARS version used in this research is the one with the sparsity penalty set at 0.1, as this was the best-performing version of NOTEARS. All hyperparameters were kept at the default values specified by Zheng et al. in the original NOTEARS implementation.

TOPO-NOTEARS inherits the same hyperparameters used in their NOTEARS initialization. The other important TOPO-NOTEARS hyperparameters include:

- **size_small**: the number of top-ranked node-pair swaps to evaluate in each iteration
- **size_large**: the size of the larger swap pool used in case of getting stuck
- **no_large_search**: how many times to sample one candidate from that large pool if you get stuck

All of these are computed automatically depending on the variable and data sizes to get a problem-size–aware search without the need for manual tuning.

5.4.2 Pure ES

The hyperparameters used in Algorithm 1 are displayed in Table 1. The edge-threshold, mentioned previously, is responsible for converting the edge weights to binary values to indicate whether an edge is present in the DAG, depending on whether it exceeds the Edge-threshold. This conversion to binary values was important in the calculation of the NLL score, as well as in the end for calculating the accuracy of the final binary matrix in comparison to the ground truth. The chosen threshold was chosen intuitively based on a short run of experiments, however, we observed that generally changes to the edge threshold did not lead to significant performance differences.

Hyperparameter	**Value**
Pop. size (stage 1)	100
Pop. size (stage 2)	50
Max. generations	100
Mutation chance ϵ (high)	0.10
Mutation chance ϵ (low)	0.03
Mutation strength σ (high)	0.10
Mutation strength σ (low)	0.04
Step-size adaptation rate, τ	0.20
Elite set size	20
Edge-threshold	0.10
Sparsity penalty, λ_{L1}	0.10

Table 1: Pure ES hyperparameter settings

5.4.3 Pure ES - Hybrid

The hybrid version of Pure ES has similar hyperparameter settings as the Pure ES version, with some minor changes in population size and number of generations. The population sizes for both stages have decreased from 100 and 50 to 30 and 10, respectively. The elite set size has also been decreased from 20 to 5. The intuition behind these changes was that only a smaller chunk of the evolutionary search would be needed, as there would be a refinement stage afterwards. The evolutionary search in the first phase of the hybrid would still offer its capabilities of exploring the search space, also with fewer individuals to lower redundant search and running time. The refinement phase of NOTEARS was left untouched with the same hyperparameters.

5.4.4 Vanilla ES - Hybrid

The Vanilla ES - Hybrid made use of the following hyperparameters in the evolutionary phase:

- Number of Generations: set at 200
- Population size: set at 30
- Fraction of population set as elite: 0.2
- Sparsity penalty: set at 0.1

Similar to the other methods, these hyperparameters were chosen based on small empirical tests. Although for Vanilla ES-Hybrid, we found that changes in the hyperparameter settings did not lead to significant differences in performance. The hyperparameters from the NOTEARS refinement were left unchanged.

5.5 Evaluation

5.5.1 Multi-objective evaluation

The performances regarding the trade-offs between Structural Accuracy, measured through SHD, and Running Time (RT) possibly constitute a multi-objective problem. In this case, algorithms conflict with each other in their performance on the two objectives, namely, low SHD and low RT. As long as one algorithm does not dominate the other, that is, it does not perform better on both objectives, we can consider a trade-off between the performances on the objectives. A common method to combine both objective functions into a single evaluation metric is called the Weighed Sum Scalarization (WSS) [60, 61]. Here, each objective function, f_1 for SHD and f_2 for RT, takes the learned DAG x as input. After both objectives are normalized, their weighted sum produces the scalar score $f(x)$, which represents the overall performance of the algorithm as we balance both objectives, scaled by a weight parameter α which sums to 1. The weight determines the importance of each objective, with a higher weight giving more importance to objective 1, while a lower weight gives more importance to objective 2:

$$f(x) = \alpha\, f_1(x) + (1 - \alpha)\, f_2(x) \tag{9}$$

This metric will play an important role in the evaluation of our experimental results as it helps us to capture the trade-offs between the methods on the different objectives.

5.5.2 Significance tests

We use statistical tests to confirm the significance of our results and to combat randomness. In this thesis, we make use of Welch t-tests, which are a parametric statistical test used to compare the means of two independent groups without assuming equal variances [62]. This fits our case as we are dealing with normally distributed data, measured through a Shapiro-Wilk Test, as well as unequal variances [63].

6 Results

The results are organized in the following way, where the performance of the algorithms on the synthetic data will be assessed on their ability to retrieve the correct DAG structure and their ability to estimate the edge weights as accurately as possible. Afterwards, all algorithms are assessed jointly on their performance on the real-world data.

6.1 Structure Learning

The Structure Learning results are divided between two groups, with a distinction made between the differentiable methods and the classical baseline algorithms, which do not use any gradient-based learning. The reason for this is that these methods share similar objectives and therefore achieve performances that are closer to each other. For a more interesting view on the results, these two types are compared with each other. The differentiable methods are assessed by measuring RT and SHD, while with the baselines, RT is replaced by TPR, as RT plays a less significant role for the smaller graphs that the baseline methods solve.

Table 2: Structure Estimation Performance comparison on ER-2 and SF-4

Nodes	**Algorithm**	$N = 1000$				$N = 20$			
		ER-2		**SF-4**		**ER-2**		**SF-4**	
		RT (↓)	**SHD (↓)**	**RT (↓)**	**SHD (↓)**	**RT (↓)**	**SHD (↓)**	**RT (↓)**	**SHD (↓)**
$d = 10$	NOTEARS	1.7s ± 0.5s	1.2 ± 1	**1.3s ± 0.5s**	8.9 ± 3.5	1.7s ± 0.6s	9.3 ± 5.2	1.5s ± 0.4s	12.8 ± 4.3
	TOPO-NOTEARS	1.7s ± 0.7s	**0 ± 0**	1.6s ± 0.6s	**0 ± 0**	1.9s ± 0.7s	14.1 ± 2.6	1.7s ± 0.4s	**10.2 ± 3**
	Pure ES - Hybrid	5.2s ± 3.1s	3.9 ± 2.6	3.6s ± 0.4s	10.8 ± 5.2	3.7s ± 0.7s	10.9 ± 3.9	3.5s ± 0.3s	13.9 ± 3.6
	Vanilla ES - Hybrid	**1.7s ± 0.3s**	2 ± 2.6	1.4s ± 0.1s	8.7 ± 3.8	**1.5s ± 0.3s**	**9 ± 3.6**	**1.4s ± 0.2s**	13.8 ± 2.6
$d = 20$	NOTEARS	23.1s ± 26.4s	9.6 ± 9.1	**5.6s ± 1.4s**	10.5 ± 6	10s ± 8.7s	**28.6 ± 8.3**	10.3s ± 2.7s	**40.8 ± 13**
	TOPO-NOTEARS	24.1s ± 25.3s	**0 ± 0**	11.2s ± 7.7s	**0 ± 0**	16.2s ± 14.4s	111.5 ± 12.6	14.1s ± 4.9s	85.6 ± 7.5
	Pure ES - Hybrid	13.8s ± 7.2s	10.1 ± 7.6	12s ± 6.4s	16.7 ± 8.5	17.4s ± 4.9s	30.8 ± 8.3	18.3s ± 4.3s	40.9 ± 9.7
	Vanilla ES - Hybrid	**8.4s ± 4.4s**	7.1 ± 5.1	5.8s ± 1s	16.7 ± 11.1	**6.4s ± 2.8s**	30 ± 8.1	**5.4s ± 1s**	44 ± 11.4
$d = 50$	NOTEARS	131.9s ± 83s	19.3 ± 14	89.2s ± 50.6s	44.3 ± 27.5	256.3s ± 34.4s	118.6 ± 15.9	270.4s ± 91.4s	178.3 ± 16.5
	TOPO-NOTEARS	310.2s ± 192.6s	**0.3 ± 0.5**	146.5s ± 55.8s	**18.4 ± 15.6**	248.7s ± 124.6s	474.8 ± 30.9	195.8s ± 37.8s	506.6 ± 25.9
	Pure ES - Hybrid	151.9s ± 59.2s	19.2 ± 15.4	120.8s ± 42.1s	56.8 ± 29.6	206.6s ± 23.6s	**111.5 ± 16.5**	183.6s ± 27.2s	**173 ± 17.4**
	Vanilla ES - Hybrid	**101.2s ± 29.2**	20.1 ± 18.7	**48.9s ± 12.1s**	54.7 ± 36.2	**87s ± 29.6s**	121.3 ± 19.2	**71.9s ± 13s**	188.2 ± 15.2
$d = 100$	NOTEARS	1043.6s ± 331.9s	31.2 ± 25.3	930.9s ± 369.6s	48 ± 32.4	1660.6s ± 334.0s	**259.9 ± 18.8**	1812.7s ± 305.6s	435.1 ± 23.5
	TOPO-NOTEARS	2292.8s ± 951.7s	**4.5 ± 2**	2061.6s ± 1381.6s	**9.1 ± 19.5**	2294.1s ± 1164.2s	538 ± 90.3	2270.8s ± 671.2s	762.3 ± 54.2
	Pure ES - Hybrid	1288.3s ± 212.2s	31.3 ± 12.8	991.7s ± 170.7s	53.2 ± 42	3355.2s ± 665.8s	261.8 ± 18.3	3026.9s ± 385.4s	**423.2 ± 28.1**
	Vanilla ES - Hybrid	**642s ± 164.5s**	30.1 ± 19.9	**506.4s ± 189.2s**	60.5 ± 47.4	**852.2s ± 78.9s**	292.3 ± 25.0	**856.7s ± 231.8s**	451.7 ± 24.2

Table 3: Welch t–test p-values comparing each method against our methods against the others and each other. Labels: (–) no significant difference, (V) Vanilla ES-Hybrid significantly better, (N) NOTEARS significantly better, (T) TOPO-NOTEARS significantly better. The significance threshold is set at 0.05.

			$N = 1000$		$N = 20$	
	d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
Vanilla ES-Hybrid vs. NOTEARS	50	ER-2	0.9151 (–)	0.2930 (–)	0.7361 (–)	$< 10^{-4}$ **(V)**
	50	SF-4	0.4794 (–)	**0.0342 (V)**	0.1800 (–)	**0.0001 (V)**
	100	ER-2	0.9152 (–)	**0.0044 (V)**	**0.0045 (N)**	$< 10^{-4}$ **(V)**
	100	SF-4	0.5011 (–)	**0.0063 (V)**	0.1371 (–)	$< 10^{-4}$ **(V)**

			$N = 1000$		$N = 20$	
	d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
Vanilla ES-Hybrid vs. TOPO-NOTEARS	50	ER-2	**0.0085 (T)**	**0.0397 (V)**	$< 10^{-4}$ **(V)**	**0.0025 (V)**
	50	SF-4	**0.0128 (T)**	**0.0003 (V)**	$< 10^{-4}$ **(V)**	$< 10^{-4}$ **(V)**
	100	ER-2	**0.0028 (T)**	**0.0004 (V)**	$< 10^{-4}$ **(V)**	**0.0035 (V)**
	100	SF-4	**0.0081 (T)**	**0.0061 (V)**	$< 10^{-4}$ **(V)**	**0.0001 (V)**

			$N = 1000$		$N = 20$	
	d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
Pure-ES Hybrid vs. Vanilla ES-Hybrid	50	ER-2	0.9078 (-)	**0.0302 (V)**	0.2370 (-)	$< 10^{-4}$ **(V)**
	50	SF-4	0.8887 (-)	**0.0003 (V)**	0.0523 (-)	$< 10^{-4}$ **(V)**
	100	ER-2	0.8747 (-)	$< 10^{-4}$ **(V)**	**0.0065 (P)**	$< 10^{-4}$ **(V)**
	100	SF-4	0.7198 (-)	$< 10^{-4}$ **(V)**	**0.0260 (P)**	$< 10^{-4}$ **(V)**

			$N = 1000$		$N = 20$	
	d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
Pure ES-Hybrid vs. NOTEARS	50	ER-2	0.9880 (-)	0.5436 (-)	0.3402 (-)	**0.0017 (P)**
	50	SF-4	0.3409 (-)	0.1469 (-)	0.4935 (-)	**0.0156 (P)**
	100	ER-2	0.9913 (-)	0.0679 (-)	0.8214 (-)	$< 10^{-4}$ **(N)**
	100	SF-4	0.7603 (-)	0.6448 (-)	0.3183 (-)	$< 10^{-4}$ **(N)**

			$N = 1000$		$N = 20$	
	d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
Pure ES-Hybrid vs. TOPO-NOTEARS	50	ER-2	**0.0037 (T)**	0.8361 (-)	$< 10^{-4}$ **(P)**	0.3194 (-)
	50	SF-4	**0.0028 (T)**	0.2613 (-)	$< 10^{-4}$ **(P)**	0.4194 (-)
	100	ER-2	**0.0001 (T)**	**0.0087 (P)**	$< 10^{-4}$ **(P)**	**0.0249 (T)**
	100	SF-4	**0.0102 (T)**	**0.0372 (P)**	$< 10^{-4}$ **(P)**	**0.0078 (T)**

6.1.1 Differentiable Methods

The differentiable methods in our research include NOTEARS, TOPO-NOTEARS, Pure ES-Hybrid, and Vanilla ES-Hybrid. The results measuring the RT and SHD of all methods under the different settings are shown in Table 2. An important note is that the added standard deviations in the Table 2 capture the variance among different problems that are generated by each seed. This can result in large standard deviations, as the more challenging problems affect performance the most. In general, these results show the ability of Vanilla ES-Hybrid, also referred to as just Vanilla ES, to deliver faster running times than NOTEARS, while having comparable SHD values. This difference becomes most apparent as the number of variables grows. Vanilla ES finds larger graphs with 100 variables considerably faster than NOTEARS, with speed-ups up to 2 times being realized, while the accuracy trade-off is restricted. There are also cases where Vanilla ES fully dominates NOTEARS, such as at {*ER-2, D=100, N=1000*}. In other cases, the trade-off results in a

deterioration of accuracy at a maximum of 12 percent.

The comparison between Vanilla ES and TOPO-NOTEARS offers an interesting trade-off as well. In settings where the number of samples is set at 1000, we can notice a trade-off existing between the improved running time that Vanilla ES offers and the improved SHD of TOPO-NOTEARS. This trade-off is the most relevant for the larger 100-variable graphs, as the RT and SHD difference increases. The performances of TOPO-NOTEARS at N=20 seem to lag in both SHD and running time. The SHD values of TOPO-NOTEARS at N=20 at different variable sizes, upward from 20, are worse than randomly selecting the expected number of edges with a hypergeometric distribution. This showcases its diminished ability to estimate the true underlying DAG when there is little available observational data.

Pure ES-Hybrid seems to perform less well than our other method, Vanilla ES-Hybrid. This is especially true regarding the running time, which is significantly worse, often without any SHD improvement as a trade-off. The comparison between Pure ES-Hybrid and NOTEARS seems to often results in similar performances regarding both SHD and RT, except for some RT results at N=20 in larger graphs from $d \geq 50$, where Pure ES-Hybrid runs for almost twice as long at D=100. Finally, Pure ES-Hybrid is fully dominated by TOPO-NOTEARS regarding SHD values when data samples are abundant. This often comes at the price of longer running times, especially in larger graphs. Similarly to before, TOPO-NOTEARS performs less at N=20, with Pure ES-Hybrid performing better in comparison, in terms of SHD, nonetheless, with longer running times.

We further substantiate our findings in Table 3, where Welsh t-tests are performed to measure the statistical significance between the results of our methods, as we further zoom in on the comparison between Vanilla ES-Hybrid, NOTEARS, TOPO-NOTEARS, and Pure ES-Hybrid. These tests calculate the p-value, with values below 0.05 indicating significance. As the aforementioned objective was to search difficult landscapes efficiently through the global exploration of Evolutionary Strategies, we therefore focus on problems with graph sizes from 50 variables onward. In these settings, the differences in observed RT and SHD increase, resulting in more interesting comparisons. The first comparison between Vanilla ES-Hybrid and NOTEARS clearly shows how Vanilla ES (V) significantly improves the Running Time (RT) in 7 out of 8 cases over NOTEARS (N). In all cases, but one, it manages to achieve this without sacrificing significant losses in SHD.

Similarly, in the second comparison between Vanilla ES-Hybrid and TOPO-NOTEARS, Vanilla ES offers significant improvements in RT, however, at the cost of significantly improved SHD performances from TOPO-NOTEARS with N=1000. As noticed before, TOPO-NOTEARS struggles when N is set at 20, resulting in significant RT and SHD improvements from Vanilla ES-Hybrid.

Next, the significance tests confirm our observation that Vanilla ES-Hybrid often performs better than Pure ES-Hybrid. Pure ES-Hybrid does, nevertheless, perform similarly to NOTEARS, with improvements over each other often being insignificant.

Finally, TOPO-NOTEARS significantly improves over Pure ES-Hybrid at N=1000. This comes at a trade-off of RT in the larger 100-variable graph, but such a trade-off does not exist at 50 graphs. Lastly, due to TOPO-NOTEARS' lower performances at N=20, Pure ES-Hybrid scores a higher SHD. This does, for the first time in this setting, not come with an improved RT time, too, as TOPO-NOTEARS even scores significantly better in two out of four settings. The remaining results of the significance tests are displayed in the Appendix in Tables 7 till 11.

Overall, from the results in Tables 2 and 3, it can be observed that we are frequently dealing with a multi-objective optimization problem where a trade-off exists between RT and SHD. The Weighed Sum Scalarization (see Eq. 9) helps us to combine the multiple objectives into a single scalar value, such that we can better understand the strength of each method, considering the trade-off. This single metric could make the cost of sacrificing one objective for the other visible, as a method that runs quickly but yields poor accuracy is penalized with low scores, even when the RT is heavily weighted, while approaches that achieve a balanced improvement in both RT and SHD stand out with high scores. This helps us understand how

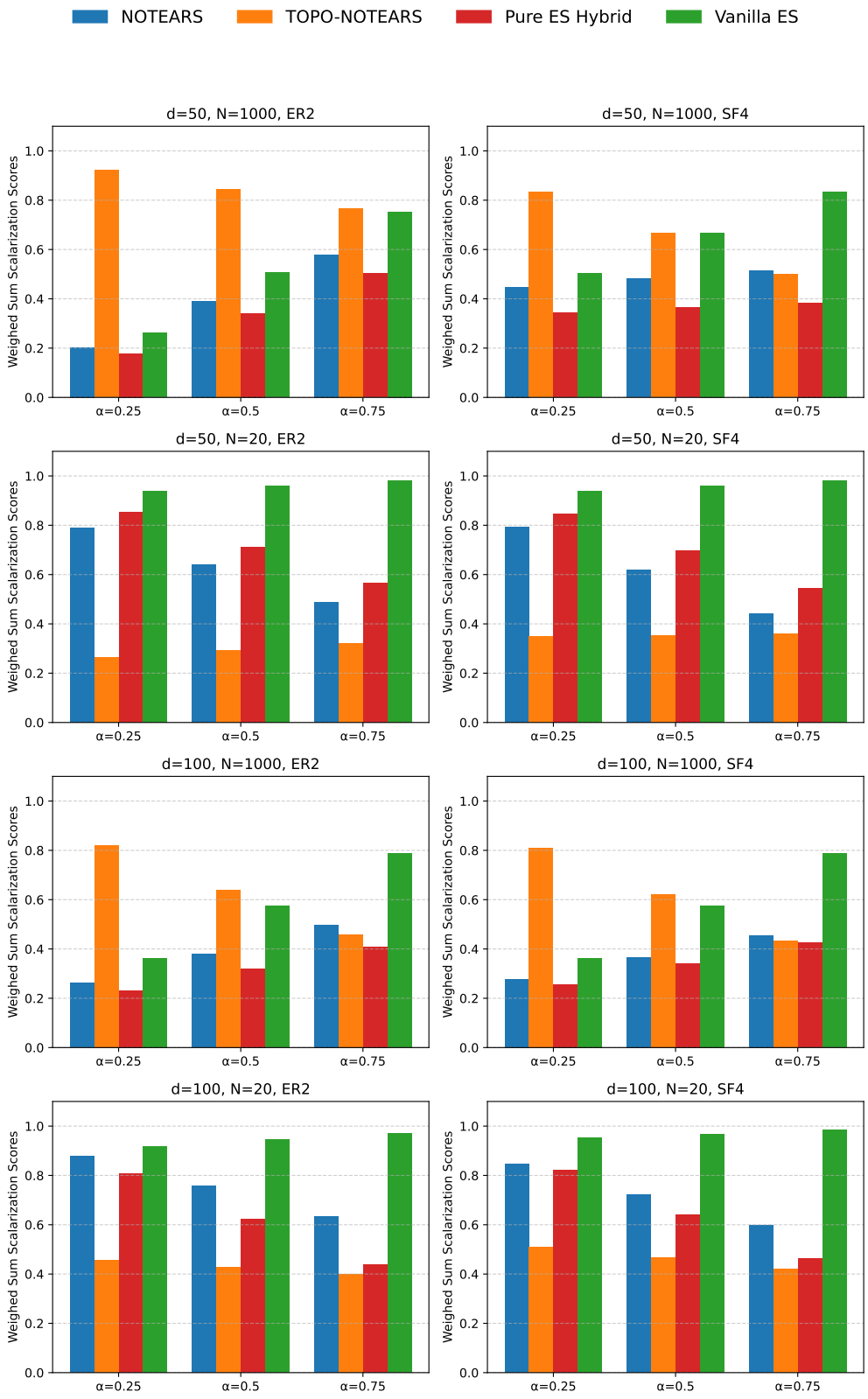

Figure 1: Weighted Sum Scalarized scores comparing all differentiable methods, with a higher α giving more importance to RT ($d \geq 50$)

strong that SHD–RT trade-off is. The weight α aids us in controlling how much importance we assign to one of the two objectives. In the case where a low SHD is preferred over faster RT, a lower α will highlight this more. The opposite is true with a higher α. Figure 1 compares the Weighted Sum Scalarized Scores from the differentiable methods with each other on the larger graphs ($d \geq 50$), with the different graph models, and data sample sizes, at different levels of α.

In general, we notice that with a higher α Vanilla ES dominates in all of these settings, highlighting Vanilla ES's ability to estimate the DAG at lower running times, while providing competitive SHD values. This competitiveness in SHD is further showcased in the fact that with a balanced α of 0.5 at {N=1000}, it often matches or comes close to the WSS scores of TOPO-NOTEARS, except for at {N=1000, d=50, ER-2}. TOPO-NOTEARS dominates all other methods on the N=1000} settings with a lower α, where its strong structural accuracy performances are preferred. As discussed prior, these performances drop significantly with a N=20}, where Vanilla ES comes on top at every α setting, often followed by NOTEARS and Pure ES-Hybrid in second place.

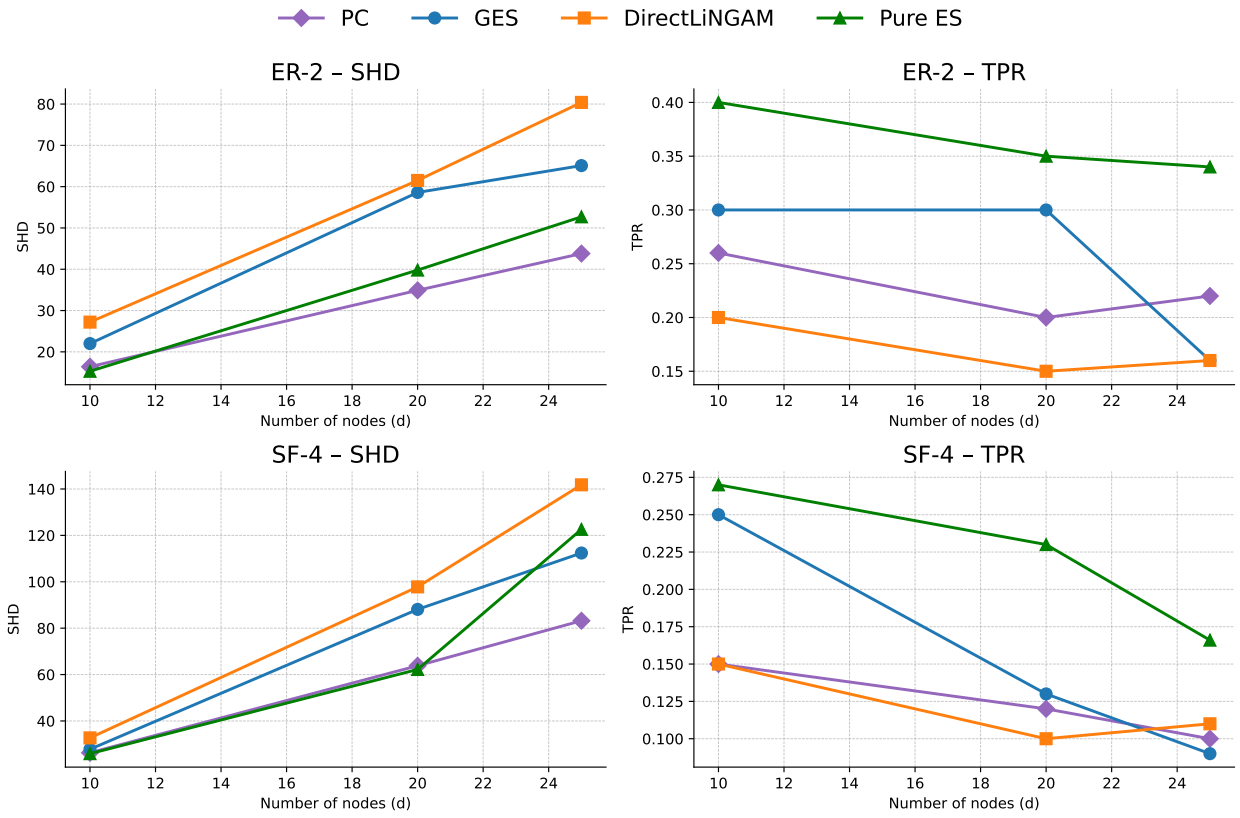

Figure 2: Performance of GES, DirectLiNGAM, and Pure ES across different d (ER-2 and SF-4 with $N = 1000$).

6.1.2 Classical Baselines

The baseline algorithms have been evaluated on their TPR and SHD across different variable sizes up to 25 on both ER-2 and SF-4 graph models. We limited our experiments to $d \leq 25$ because of the high computational costs for larger graphs with the baselines. The number of observational data samples has only been set at 1000, because the data size of 20 would cause trouble within PC and DirectLiNGAM, as the number of variables (d) would be less than or equal to the number of data samples (N). This would create problems as PC's Fisher Z-test requires more data points than variables for its conditional independence test, while for DirectLiNGAM's Least Squares Regression, we also require $n > d$ to solve the regression coefficients [64, 17].

The results are plotted in Figure 2, which visualizes Pure ES's ability to generate DAGs with a high TPR, yielding better results than the other baseline methods on both ER-2 and SF-4. On the other hand, based on the SHD metric, Pure ES still offers competitive results, only being beaten by the PC algorithm for DAGs with more nodes. We further confirm our findings in Table 4, where the results of the significance tests are displayed between each method and Pure ES. More details about these results can be found in the Appendix in Table 6.

Table 4: Welch t–test p-values comparing each baseline method against Pure ES. Labels: (–) no significant difference, (PC) PC Algorithm significantly better, (ES) Pure ES significantly better, (D) DirectLiNGAM significantly better, (G) GES significantly better. The significance threshold is set at 0.05.

Pair	**d**	**SHD**		**TPR**	
		ER-2	SF-4	ER-2	SF-4
Pure ES vs PC	10	0.4009	0.8287	**0.0351 (ES)**	**0.0500 (ES)**
	20	**0.0458 (PC)**	0.3547	**0.0009 (ES)**	$< 10^{-4}$ **(ES)**
	25	**0.0047 (PC)**	$< 10^{-4}$ **(PC)**	**0.0086 (ES)**	**0.0023 (ES)**
Pure ES vs GES	10	**0.0002 (ES)**	0.0559 (-)	0.1426 (-)	0.2981 (-)
	20	**0.0010 (ES)**	$< 10^{-4}$ **(ES)**	**0.0035 (ES)**	**0.0008 (ES)**
	25	**0.0030 (ES)**	0.1836 (-)	$< 10^{-4}$ **(ES)**	**0.0363 (ES)**
Pure ES vs DirectLiNGAM	10	**0.0207 (ES)**	**0.0046 (ES)**	**0.0052 (ES)**	**0.0326 (ES)**
	20	$< 10^{-4}$ **(ES)**	**0.0002 (ES)**	**0.0001 (ES)**	$< 10^{-4}$ **(ES)**
	25	$< 10^{-4}$ **(ES)**	**0.0002 (ES)**	**0.0003 (ES)**	**0.0085 (ES)**

6.2 Edge Weight Estimation

The experimentation of the edge weight estimations in our research included all methods that enable optimization with real-valued edge weights, namely NOTEARS, TOPO-NOTEARS, Pure ES-Hybrid, and Vanilla ES. Besides these differentiable methods, Pure ES can also optimize edge weights. However, this has not been added to the graphs for two reasons. Firstly, due to the computational complexity, Pure ES was only run for at most 25 variables. Secondly, as noticed before, Pure ES performed significantly worse than the gradient-based methods. Adding their results to the figure would distort the view of the results of the other methods. The results of the methods are displayed in Figure 3. These figures show the normalized MSE scores that have been measured by averaging ten runs per setting at different variable sizes, graph models, and sample sizes. The normalized MSE is simply the MSE multiplied by the edge density of each graph, defined as $\frac{d^2}{s}$, where d is the number of variables and s the expected number of edges. This scaling provides a clearer overview of per-edge error at each variable size. On the whole, these edge weight scores seem to resemble the structural accuracy scores closely in terms of ordering of the methods, with at $N = 1000$ NOTEARS, Vanilla ES, and Pure ES-Hybrid closely following each other, while TOPO-NOTEARS scores better. TOPO-NOTEARS struggles, just like in the structural accuracy results, on the $N = 20$ graphs. For this reason, TOPO-NOTEARS has also not been included in these graphs, as their performance was substantially worse, causing distortions to the graph.

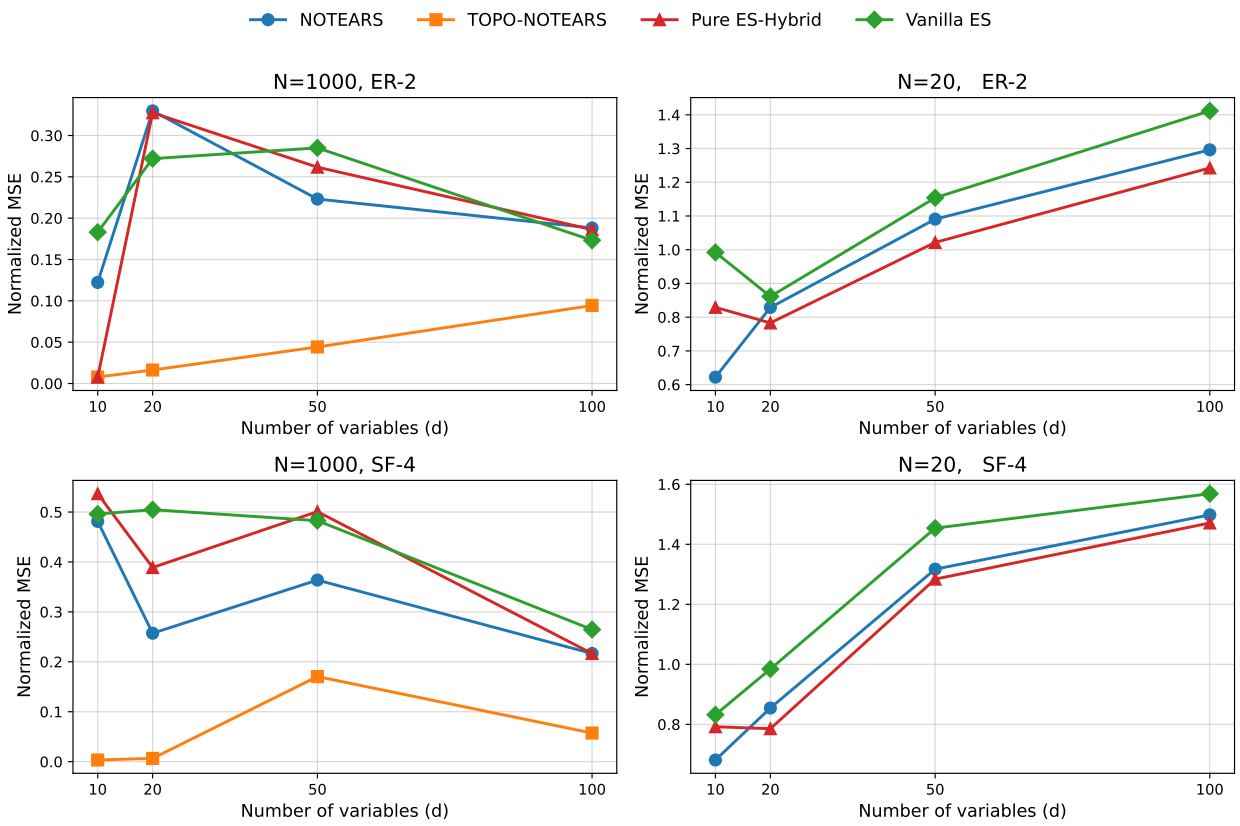

Figure 3: Normalized Mean Squared Error (MSE) of differentiable methods measured across different graph sizes, graph models, and data sizes

6.3 Real-World data

The performance of each method on the real-world Sachs Protein data is displayed in Table 5 [37]. It is illustrated how Pure ES performs the most optimally in terms of low FDR and high SHD, outperforming the other methods. Pure ES has made use of the integrated domain knowledge that limits the max-in-degree to two, as this is typical in protein networks [55, 56]. This has boosted its performance significantly from 31 to 20 SHD. The other method that enables this constraint integration was the Pure ES-Hybrid, however, the constraint seemed to have no effect for this method.

Table 5: Real-World data performance

Algorithm	**TPR**	**FDR**	**SHD**
NOTEARS	0.28	0.75	22
TOPO-NOTEARS	0.28	0.82	30
Pure ES - Hybrid	0.28	0.78	26
Vanilla ES - Hybrid	0.34	0.73	23
DirectLiNGAM	0.05	0.9	21
GES	**0.44**	0.70	25
PC	0.34	0.75	21
Pure ES	0.39	**0.63**	**20**

7 Discussion

In this section, we will interpret the results and look back at our research questions and hypotheses to answer these with an in-depth analysis, discussing possible reasons for certain occurrences. Afterwards, we discuss the limitations of this study and address future work.

7.1 Research Questions

7.1.1 RQ1

Can Pure Evolutionary Strategies (ES) serve as a competitive alternative to gradient-based Causal Structure Learning methods in terms of structural accuracy on synthetic and real-world data?

Our results indicate that on synthetic data, Pure ES does not serve as a competitive alternative to the gradient-based methods, but rather to the baseline methods, making use of score-based, functional-based, or constraint-based methods. We hypothesize several reasons for this observation. First of all, we believe that the stochastic nature of ES on its own leads to the evaluation of many less promising regions in the landscape. In a way, the strength of ES within the hybrids, namely exploring the search space more globally, can limit itself as well by less effectively exploiting feasible regions once its found and being clouded by noise. Empirically, this stochastic noise was also discovered as we experienced large performance differences with varied hyperparameter settings. Another reason for the decline in performance could be the added acyclicity check with DFS. Currently, once a cycle is found, the weakest link with the smallest edge weight is removed. This is a rather greedy manner that does not account for whether this move might be non-optimal later in the optimization process. Furthermore, by abruptly cutting an edge, sudden jumps might arise in the NLL calculations, further damaging the optimization process. Besides the lowered structural accuracy, the RT takes longer as well due to the influence exerted by evaluating each candidate with the fitness function, and running DFS cycle checks for each added mutation, which carries a computational complexity of $O(d^2)$. Finally, we did notice that while Pure ES does not offer a competitive alternative to the gradient-based methods in the synthetic data, it does improve upon these methods with the Real-World Data through the introduction of a simple domain knowledge constraint. This highlights the adaptable strength of evolutionary methods towards the domain it is used.

7.1.2 RQ2

Can hybridizing evolutionary strategies with gradient-based optimization offer a balanced trade-off between structural accuracy and runtime on synthetic data, and higher structural accuracy on real-world datasets, compared to the state-of-the-art methods?

Overall, we noticed that the evolutionary gradient hybrids offered a balanced trade-off between SHD and RT on the ER-2 and SF-4 data, as they offered competitive results to the gradient-based methods. Vanilla ES is the stronger method between the two, improving the gradient-based methods significantly regarding running time, especially in larger graphs. We highlighted that in comparison to NOTEARS, Vanilla ES improves the RT with little significant sacrifice being made regarding structural accuracy (see Table 3). A more balanced trade-off arises between Vanilla ES and TOPO-NOTEARS with abundant data, where TOPO-NOTEARS scores lower SHD scores, but at the cost of longer running times. The calculation of the WSS scores in Figure 1 amplified this trade-off, where in most of such cases we could see balanced scores with α set at 0.5, while lower α favoured TOPO-NOTEARS and higher α values Vanilla ES. Pure ES - Hybrid mostly lagged behind Vanilla ES, but also offered slight improvements in structural accuracy in settings with lower N and higher d. Furthermore, Pure ES-Hybrid did offer a trade-off in RT gains with TOPO-NOTEARS as well at {N=1000, d=100} (see Table 3). However, the WSS scores in this case showed a clear preference towards TOPO-NOTEARS within this trade-off.

We theorize that the running time improvements by Vanilla ES are largely caused by the improved robustness that an evolutionary search offers in more challenging landscapes, leading to more global exploration and

faster convergence as a result. This matches our earlier hypothesis about hybrid methods as a warm-starting method, as well as the aforementioned theory about hybrid methods [31, 30]. The evolutionary starting phase could conduct a broad, population-based exploration of the search space and, within a short time, identify a feasible region that is already proximate to the true DAG. This would, in turn, make the refinement phase go faster as the algorithm then requires far fewer gradient iterations to converge to its estimation. At the same time, we suggest that the difference between Vanilla ES and Pure ES-Hybrid is due to similar reasons that Pure ES lagged. Vanilla ES did not suffer from these limitations and was allowed to be more focused on global exploration, without needing to worry about constraints or a surplus of hyperparameters.

The performances of our hybrid methods on the real-world data match, for the most part, with all other methods. Vanilla ES - Hybrid scored the best out of our hybrids, only slightly behind NOTEARS. Possible reasons why, in general, the SHD of all the methods score similarly, and relatively high compared to the number of variables, could be because the Sachs Protein Data includes non-linear relationships that exist between some proteins in the data. Moreover, the Sachs Data is not one homogeneous observational sample, as it pools measurements from different laboratory experiments together with their probability distribution [37]. This might lead to optimization difficulties for CSL algorithms that rely on independent and identically distributed observations [18].

7.1.3 RQ3

How does the performance of our hybrid methods compare to state-of-the-art methods across different settings regarding graph sizes, graph models, and data sizes, on synthetic data?

In the experimentation among the hybrid methods with other gradient-based methods, we have noticed the increasing strength of the hybrids in larger graphs regarding improved running times (see Table 2. More specifically, from 50 variables on in both ER-2 and SF-4, the saved time of mostly Vanilla ES improves significantly, while keeping strong SHD scores. We do notice that the SHD scores for all methods, including our hybrids, are better for ER-2 than SF-4. This is logical, since SF-4 is a denser graph with twice the number of edges, which constitutes a more challenging CSL problem. Subsequently, our results indicated overall longer running times and higher SHD scores for smaller sample sizes, due to a lack of data, making the optimization process noisier and less accurate. At *(N=20)*, we observed that the RT difference realized by Vanilla ES becomes more apparent in comparison to NOTEARS and TOPO-NOTEARS. NOTEARS still scores slightly better than SHD scores to Vanilla ES, while TOPO-NOTEARS struggles on both RT and SHD. A possible reason for the struggling convergence of the gradient-based methods could be attributed to the sample covariance matrix used in gradient-based methods. A lower number of data samples could lead to a highly variable sample covariance, which in turn causes noisy gradients and unstable convergence [65]. The faster convergence by Vanilla ES is likely due to the same reasons mentioned at *(N=1000)*, namely that the exploratory warm-start in Vanilla ES limits the number of steps still required to converge in the final phase. The same cannot be said for Pure ES, as it appears that the warm-starting phase offers fewer gains in convergence. Possibly due to the longer running time of ES, as well as a lower quality warm-started solution, leading to even more NOTEARS iterations needed. Nonetheless, Pure ES-Hybrid does seem to perform slightly better regarding SHD than the other methods at *(N=20)* from 50 variables on. TOPO-NOTEARS, on the other hand, seems to fully fail at *(N=20)*. As TOPO-NOTEARS inherits its initialization from NOTEARS, it takes additional running time. Next, TOPO-NOTEARS' noisy gradients might make all topological swaps seem optimal, ending on a solution that is worse than random.

7.1.4 RQ4

How do our methods compare in edge weight estimation in comparison to the state-of-the-art methods?

The scores of the edge weights estimation of our hybrid methods remained mostly parallel to the Structural Hamming Distance results in terms of ordering, indicating competitive edge weight estimation of our method with the other state-of-the-art methods. Overall, at *(N=1000)*, TOPO-NOTEARS outperformed the other methods significantly, as it did with the structural accuracy experiments. The other three methods

follow more closely as expected. A noticeable observation is the higher Normalized MSE values and lower variable sizes for *(N=1000)*. A possible reason for this is that with lower variable sizes, each edge weight error contributes a larger fraction. This would mean that a single larger error can distort the overall MSE more. This becomes less of an issue at *(N=20)* where the many edges make large errors spread over all variable sizes, making the curve go up more linearly.

7.2 Limitations and future work

The overall limitations of our method involve the subpar performance of the Pure ES method. On the whole, this algorithm did not perform well enough to challenge the state-of-the-art gradient-based methods, due to our earlier identified reasons regarding stochasticity, the acyclicity check, and the hyperparameters. A possible improvement over this could have been the addressing of a less greedy manner to remove cycles. This could, for example, have been an approach similar to the one used in GOLEM, where they gradually lower the edge threshold until the cycle disappears by itself [22]. The lack of an extensive hyperparameter tuning phase can be considered a limitation as well. The large role of hyperparameters in ES could have been utilized better by making use of adaptive hyperparameters, instead of a fixed set. These adaptive hyperparameters would change concerning the graph size, graph density. The integration of automated hyperparameter optimization techniques, such as Bayesian Optimization, could have mitigated this problem as well. Nonetheless, the search for optimal hyperparameters is an active field of research and could, for this reason, be considered outside of the scope of this research [58, 59]. Finally, we could have considered experimenting with larger variable sizes and denser graphs as well. These larger graphs would have implications for real-world applications, as problems exist within fields such as genetics that deal with problems up to more than 500 connected variables [66]. We decided on our current set of settings because of the added computational complexity, and that most papers within the field use similar experimental settings, which enables more accurate comparisons. Overall, we hope that future works can build upon our work and address the limitations we have mentioned.

8 Conclusion

This thesis aimed to discover the abilities of Evolutionary Strategies (ES) as a competitive method against established gradient-based methods within the field of Causal Structure Learning with real values. We hypothesized that the stochastic nature of ES could provide global exploration of the more challenging non-convex landscapes where gradient-based methods might fall into local entrapment. Furthermore, ES would allow the integration of domain-knowledge constraints that would not be feasible for smooth methods. To test our hypothesis, we introduced two types of methods: a Pure ES method and two hybrid methods. The Pure ES method worked independently from any gradient-based methods and utilized its own methods for evaluation and acyclicity constraints, while the hybrids combined the strengths of ES and the gradient-based method called NOTEARS in two phases [18]. We have conducted experiments on synthetic and real-world data with varying numbers of variables and sample sizes and compared our methods to other established methods such as NOTEARS, TOPO-NOTEARS, PC, GES, and DirectLiNGAM [18, 42, 12, 16, 17]. Our findings show that while a Pure ES algorithm does not hold up to the gradient-based methods based on performances on synthetic data, it does challenge other baseline methods by improving overall TPR and performing well on SHD, only being beaten by PC as the graph grows larger. Pure ES also performed the best on the real-world data, mainly due to the introduction of the domain-knowledge constraint. Moreover, we observed the ability that hybrid methods offer by offering a balanced trade-off between structural accuracy and running time, as our best-performing hybrid, called Vanilla ES, significantly improved convergence times while maintaining strong performances on structural accuracy. This phenomenon was especially true as the graph sizes grew with the number of variables, resulting in more challenging landscapes where the global exploration of ES in the first phase of the hybrid was proven worthy. Additionally, we found the same to be true with lower numbers of data samples. Finally, the measurement of edge weight estimation through MSE proved to be mostly parallel with the structural accuracy results, showcasing the ability of our hybrid methods to refine the edge weights between variables. We hope our work adds to the current literature on Causal Structure Learning and sets a path for future work for utilizing the full potential of evolutionary algorithms and hybrid methods within the field.

For implementation details, please refer to `https://github.com/nimamehrafar/ThesisProject.git`

References

[1] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

[2] Kaiyue Liu, Lihua Liu, Kaiming Xiao, Xuan Li, Hang Zhang, Yun Zhou, and Hongbin Huang. Cl-notears: Continuous optimization algorithm based on curriculum learning framework. *Mathematics*, 12(17):2640, 2024.

[3] Lisa M Anderson, Kelvin O Lim, Erich Kummerfeld, Ross D Crosby, Scott J Crow, Scott G Engel, Lauren Forrest, Stephen A Wonderlich, and Carol B Peterson. Causal discovery analysis: A promising tool in advancing precision medicine for eating disorders. *International Journal of Eating Disorders*, 56(11):2012–2021, 2023.

[4] Meike Nauta, Doina Bucur, and Christin Seifert. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):19, 2019.

[5] David Maxwell Chickering. Learning bayesian networks is np-complete. In *Learning from data: Artificial intelligence and statistics V*, pages 121–130. Springer, 1996.

[6] Fangting Zhou, Kejun He, and Yang Ni. Causal discovery with heterogeneous observational data. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 2383–2393. PMLR, August 2022.

[7] Stephan Bongers, Patrick Forré, Jonas Peters, and Joris M Mooij. Foundations of structural causal models with cycles and latent variables. *The Annals of Statistics*, 49(5):2885–2915, 2021.

[8] Antti Hyttinen, Frederick Eberhardt, and Patrik O Hoyer. Experiment selection for causal discovery. *The Journal of Machine Learning Research*, 14(1):3041–3071, 2013.

[9] Peter Spirtes and Kun Zhang. Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics*, volume 3, pages 1–28. Springer, 2016.

[10] Lei Wang, Shanshan Huang, Shu Wang, Jun Liao, Tingpeng Li, and Li Liu. A survey of causal discovery based on functional causal model. *Engineering Applications of Artificial Intelligence*, 133:108258, 2024.

[11] Honghao Li, Vincent Cabeli, Nadir Sella, and Hervé Isambert. Constraint-based causal structure learning with consistent separating sets. *Advances in neural information processing systems*, 32, 2019.

[12] Naftali Harris and Mathias Drton. Pc algorithm for nonparanormal graphical models. *The Journal of Machine Learning Research*, 14(1):3365–3383, 2013.

[13] Biwei Huang, Kun Zhang, Yizhu Lin, Bernhard Schölkopf, and Clark Glymour. Generalized score functions for causal discovery. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1551–1560, 2018.

[14] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20:197–243, 1995.

[15] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[16] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[17] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, Kenneth Bollen, and Patrik Hoyer. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research-JMLR*, 12(Apr):1225–1248, 2011.

[18] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

[19] Roy S Zawadzki, Joshua D Grill, Daniel L Gillen, and for the Alzheimer's Disease Neuroimaging Initiative. Frameworks for estimating causal effects in observational settings: comparing confounder adjustment and instrumental variables. *BMC Medical Research Methodology*, 23(1):122, 2023.

[20] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

[21] Marcus Kaiser and Maksim Sipos. Unsuitability of notears for causal graph discovery when dealing with dimensional quantities. *Neural Processing Letters*, 54(3):1587–1595, 2022.

[22] Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

[23] Alexander Reisach, Christof Seiler, and Sebastian Weichwald. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34:27772–27784, 2021.

[24] Xin Yao. Global optimisation by evolutionary algorithms. In *Proceedings of IEEE International Symposium on parallel algorithms architecture synthesis*, pages 282–291. IEEE, 1997.

[25] Zhenhua Li, Xi Lin, Qingfu Zhang, and Hailin Liu. Evolution strategies for continuous optimization: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 56:100694, 2020.

[26] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in neural information processing systems*, 31, 2018.

[27] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):178–195, 2014.

[28] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[29] Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

[30] AYAD RAMADHAN ALI and BAYDA GHANIM FATHI. Hybridization gradient based methods with genetic algorithm for solving systems of linear equations. *Journal of Duhok University*, 25(2):41–49, 2022.

[31] Hassan A Bashir and Richard S Neville. Hybrid evolutionary computation for continuous optimization. *arXiv preprint arXiv:1303.3469*, 2013.

[32] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[33] Natasha K Bowen and Shenyang Guo. *Structural equation modeling*. Oxford University Press, 2011.

[34] Rex B Kline. *Principles and practice of structural equation modeling*. Guilford publications, 2023.

[35] Marco F Eigenmann, Preetam Nandy, and Marloes H Maathuis. Structure learning of linear gaussian structural equation models with weak edges. *arXiv preprint arXiv:1707.07560*, 2017.

[36] Alain Hertz and Daniel Kobler. A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126(1):1–12, 2000.

[37] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[38] Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast pc algorithm for high dimensional causal discovery with multi-core pcs. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(5):1483–1495, 2016.

[39] Achille Nazaret and David Blei. Extremely greedy equivalence search. *arXiv preprint arXiv:2502.19551*, 2025.

[40] Tom Claassen and Ioan G Bucur. Greedy equivalence search in the presence of latent confounders. In *Uncertainty in Artificial Intelligence*, pages 443–452. Pmlr, 2022.

[41] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

[42] Chang Deng, Kevin Bello, Bryon Aragam, and Pradeep Kumar Ravikumar. Optimizing notears objectives via topological swaps. In *International Conference on Machine Learning*, pages 7563–7595. PMLR, 2023.

[43] Dennis Wei, Tian Gao, and Yue Yu. Dags with no fears: A closer look at continuous optimization for learning bayesian networks. *Advances in Neural Information Processing Systems*, 33:3895–3906, 2020.

[44] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

[45] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[46] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[47] Bryon Aragam and Qing Zhou. Concave penalized estimation of sparse gaussian bayesian networks. *The Journal of Machine Learning Research*, 16(1):2273–2328, 2015.

[48] Dumitru Dumitrescu, Beatrice Lazzerini, Lakhmi C Jain, and Alexandra Dumitrescu. *Evolutionary computation*. CRC press, 2000.

[49] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1:3–52, 2002.

[50] Bo-Yu Chu, Chia-Hua Ho, Cheng-Hao Tsai, Chieh-Yen Lin, and Chih-Jen Lin. Warm start for parameter selection of linear classifiers. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 149–158, 2015.

[51] Alex Loftus Eric Bridgeford and Joshua Vogelstein. Ers. `https://docs.neurodata.io/graph-stats-book/appendix/ch12/ers.html`, n.d. Accessed: 2025-04-07.

[52] M Kalisch, A Hauser, MH Maathuis, and Martin Mächler. An overview of the pcalg package for r. 2020.

[53] Albert-László Barabási. Scale-free networks: a decade and beyond. *science*, 325(5939):412–413, 2009.

[54] Judea Pearl. *Causality*. Cambridge university press, 2009.

[55] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64–68, 2002.

[56] Uri Alon. *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC, 2019.

[57] R Hinterding, Zbigniew Michalewicz, and A Eiben. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[58] Matthias Feurer and Frank Hutter. *Hyperparameter optimization*. Springer International Publishing, 2019.

[59] Gerlise Chan, Tom Claassen, Holger Hoos, Tom Heskes, and Mitra Baratchi. Autocd: Automated machine learning for causal discovery algorithms. 2024.

[60] Cristina Bazgan, Stefan Ruzika, Clemens Thielen, and Daniel Vanderpooten. The power of the weighted sum scalarization for approximating multiobjective optimization problems. *Theory of Computing Systems*, pages 1–21, 2022.

[61] Vira Chankong and Yacov Y Haimes. *Multiobjective decision making: theory and methodology*. Courier Dover Publications, 2008.

[62] Bernard L Welch. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

[63] Zofia Hanusz, Joanna Tarasinska, and Wojciech Zielinski. Shapiro–wilk test with known mean. *REVSTAT-statistical Journal*, 14(1):89–100, 2016.

[64] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

[65] Alvin C Rencher. A review of "methods of multivariate analysis, ", 2005.

[66] Y Wang, DJ Miller, and R Clarke. Approaches to working in high-dimensional data spaces: gene expression microarrays. *British journal of cancer*, 98(6):1023–1028, 2008.

A Additional Results

Table 6: Structure Estimation Performance (RT ↓, TPR ↑, SHD ↓) on ER and SF graphs for $N = 1000$

		ER-2			SF-4		
d	Algorithm	RT	TPR	SHD	RT	TPR	SHD
10	PC	**0.16s ± 0.03s**	0.26 ± 0.08	16.4 ± 1.5	0.17s ± 0.03s	0.15 ± 0.06	26.3 ± 2.4
	GES	4.2s ± 2.2s	0.3 ± 0.16	22 ± 4.3	2s ± 0.3s	0.25 ± 0.13	27.8 ± 2.7
	DirectLiNGAM	0.25s ± 0.25s	0.20 ± 0.08	27.2 ± 13.3	**0.16s ± 0.16s**	0.15 ± 0.06	32.7 ± 2.9
	Pure ES	7s ± 0.6s	**0.40 ± 0.17**	**15.3 ± 3.7**	15.1s ± 6.4s	**0.27 ± 0.16**	**25.9 ± 5.2**
20	PC	**0.65s ± 0.08s**	0.2 ± 0.04	**34.9 ± 1.7**	**0.59s ± 0.07s**	0.12 ± 0.03	63.8 ± 2.4
	GES	95.3s ± 74.9s	0.3 ± 0.13	58.6 ± 11	73.2s ± 42.5s	0.13 ± 0.07	88.1 ± 8.3
	DirectLiNGAM	1.9s ± 1.1s	0.15 ± 0.03	61.5 ± 6.6	1s ± 0.1s	0.1 ± 0.03	97.8 ± 10.6
	Pure ES	29.5s ± 11.1s	**0.35 ± 0.1**	39.8 ± 6.6	34.2s ± 4.6s	**0.23 ± 0.05**	**62.2 ± 4.7**
25	PC	**0.98s ± 0.21s**	0.22 ± 0.08	**43.8 ± 4.2**	**0.84s ± 0.19s**	0.1 ± 0.02	**83.2 ± 2.5**
	GES	154.4s ± 57.2s	0.16 ± 0.07	65.1 ± 11.2	291.6s ± 168.4s	0.09 ± 0.06	112.4 ± 10.1
	DirectLiNGAM	2.3s ± 0.8s	0.16 ± 0.04	80.4 ± 3	1.9s ± 0.25s	0.11 ± 0.03	141.8 ± 10.6
	Pure ES	45.6s ± 15.3s	**0.34 ± 0.1**	52.7 ± 7.3	48.4s ± 16.2s	**0.17 ± 0.05**	122.6 ± 6.2

Table 7: Vanilla ES-Hybrid vs. NOTEARS. Labels: (–) no significant difference, (V) Vanilla ES-Hybrid significantly better, (N) NOTEARS significantly better.

		$N = 1000$		$N = 20$	
d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
10	ER-2	0.3823 (-)	1.0000 (-)	0.8826 (-)	0.3627 (-)
10	SF-4	0.9039 (-)	0.5494 (-)	0.5387 (-)	0.4918 (-)
20	ER-2	0.4610 (-)	0.1147 (-)	0.7071 (-)	0.2391 (-)
20	SF-4	0.1428 (-)	0.7179 (-)	0.5658 (-)	**0.0002 (V)**
50	ER-2	0.9151 (–)	0.2930 (–)	0.7361 (–)	$< 10^{-4}$ **(V)**
50	SF-4	0.4794 (–)	**0.0342 (V)**	0.1800 (–)	**0.0001 (V)**
100	ER-2	0.9152 (–)	**0.0044 (V)**	**0.0045 (N)**	$< 10^{-4}$ **(V)**
100	SF-4	0.5011 (–)	**0.0063 (V)**	0.1371 (–)	$< 10^{-4}$ **(V)**

Table 8: Vanilla ES-Hybrid vs. TOPO-NOTEARS. Labels: (–) no significant difference, (V) Vanilla ES-Hybrid significantly better, (T) TOPO-NOTEARS significantly better.

		$N = 1000$		$N = 20$	
d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
10	ER-2	**0.0378 (T)**	1.0000 (-)	**0.0022 (V)**	0.1222
10	SF-4	$< 10^{-4}$ **(T)**	0.3242 (-)	**0.0104 (T)**	0.0533 (-)
20	ER-2	**0.0017 (T)**	0.0834 (-)	$< 10^{-4}$ **(V)**	0.0617 (-)
20	SF-4	**0.0010 (T)**	0.0545 (-)	$< 10^{-4}$ **(V)**	**0.0003 (V)**
50	ER-2	**0.0085 (T)**	**0.0397 (V)**	$< 10^{-4}$ **(V)**	**0.0025 (V)**
50	SF-4	**0.0128 (T)**	**0.0003 (V)**	$< 10^{-4}$ **(V)**	$< 10^{-4}$ **(V)**
100	ER-2	**0.0028 (T)**	**0.0004 (V)**	$< 10^{-4}$ **(V)**	**0.0035 (V)**
100	SF-4	**0.0081 (T)**	**0.0061 (V)**	$< 10^{-4}$ **(V)**	**0.0001 (V)**

Table 9: Pure-ES Hybrid vs. Vanilla ES-Hybrid. Labels: (–) no significant difference, (V) Vanilla ES-Hybrid significantly better, (P) Pure-ES Hybrid significantly better.

		$N = 1000$		$N = 20$	
d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
10	ER-2	0.1196 (-)	**0.0060 (V)**	0.2726	$< 10^{-4}$ **(V)**
10	SF-4	0.3174 (-)	$< 10^{-4}$ **(V)**	0.9441	$< 10^{-4}$ **(V)**
20	ER-2	0.3156 (-)	0.0613 (-)	0.8298 (-)	$< 10^{-4}$ **(V)**
20	SF-4	1.0000 (-)	**0.0136 (V)**	0.5210 (-)	$< 10^{-4}$ **(V)**
50	ER-2	0.9078 (–)	**0.0302 (V)**	0.2370 (–)	$< 10^{-4}$ **(V)**
50	SF-4	0.8887 (–)	**0.0003 (V)**	0.0523 (–)	$< 10^{-4}$ **(V)**
100	ER-2	0.8747 (–)	$< 10^{-4}$ **(V)**	**0.0065 (P)**	$< 10^{-4}$ **(V)**
100	SF-4	0.7198 (–)	$< 10^{-4}$ **(V)**	**0.0260 (P)**	$< 10^{-4}$ **(V)**

Table 10: Pure ES-Hybrid vs. NOTEARS. Labels: (–) no significant difference, (P) Pure-ES Hybrid significantly better, (N) NOTEARS significantly better.

		$N = 1000$		$N = 20$	
d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
10	ER-2	**0.0102 (N)**	**0.0060 (N)**	0.4472 (-)	$< 10^{-4}$ **(N)**
10	SF-4	0.3523 (-)	$< 10^{-4}$ **(N)**	0.5431 (-)	$< 10^{-4}$ **(N)**
20	ER-2	0.8954 (-)	0.3070 (-)	0.5608 (-)	**0.0342 (N)**
20	SF-4	0.0776 (-)	**0.0116 (N)**	0.9847 (-)	**0.0002 (N)**
50	ER-2	0.9880 (–)	0.5436 (–)	0.3402 (–)	**0.0017 (P)**
50	SF-4	0.3409 (–)	0.1469 (–)	0.4935 (–)	**0.0156 (P)**
100	ER-2	0.9913 (–)	0.0679 (–)	0.8214 (–)	$< 10^{-4}$ **(N)**
100	SF-4	0.7603 (–)	0.6448 (–)	0.3183 (–)	$< 10^{-4}$ **(N)**

Table 11: Pure ES-Hybrid vs. TOPO-NOTEARS. Labels: (–) no significant difference, (P) Pure-ES Hybrid significantly better, (T) TOPO-NOTEARS significantly better.

		$N = 1000$		$N = 20$	
d	Graph	p_{SHD}	p_{RT}	p_{SHD}	p_{RT}
10	ER-2	**0.0011 (T)**	**0.0060 (T)**	**0.0001 (P)**	$< 10^{-4}$ **(T)**
10	SF-4	**0.0001 (T)**	$< 10^{-4}$ **(T)**	**0.0228 (T)**	$< 10^{-4}$ **(T)**
20	ER-2	**0.0023 (T)**	0.2427 (-)	$< 10^{-4}$ **(P)**	0.8076 (-)
20	SF-4	**0.0002 (T)**	0.8035 (-)	$< 10^{-4}$ **(P)**	0.8076 (-)
50	ER-2	**0.0037 (T)**	0.8361 (–)	$< 10^{-4}$ **(P)**	0.3194 (–)
50	SF-4	**0.0028 (T)**	0.2613 (–)	$< 10^{-4}$ **(P)**	0.4194 (–)
100	ER-2	**0.0001 (T)**	**0.0087 (P)**	$< 10^{-4}$ **(P)**	**0.0249 (T)**
100	SF-4	**0.0102 (T)**	**0.0372 (P)**	$< 10^{-4}$ **(P)**	**0.0078 (T)**