Out-of-Sample Tuning for Causal Discovery

Konstantina Biza, Ioannis Tsamardinos, and Sofia Triantafillou

***Abstract*— Causal discovery is continually being enriched with new algorithms for learning causal graphical probabilistic models. Each one of them requires a set of hyperparameters, creating a great number of combinations. Given that the true graph is unknown and the learning task is unsupervised, the challenge to a practitioner is how to tune these choices. We propose out-of-sample causal tuning (OCT) that aims to select an optimal combination. The method treats a causal model as a set of predictive models and uses out-of-sample protocols for supervised methods. This approach can handle general settings like latent confounders and nonlinear relationships. The method uses an information-theoretic approach to be able to generalize to mixed data types and a penalty for dense graphs to penalize for complexity. To evaluate OCT, we introduce a causal-based simulation method to create datasets that mimic the properties of real-world problems. We evaluate OCT against two other tuning approaches, based on stability and in-sample fitting. We show that OCT performs well in many experimental settings and it is an effective tuning method for causal discovery.**

***Index Terms*— Causal-based simulation, causal discovery, out-of-sample, tuning.**

I. Introduction

LEARNING causal graphical models from observational data has been an active area of research in the past decades, and a wide range of algorithms have been proposed in the literature. The algorithms may differ in their distributional assumptions, theoretical properties, search heuristics for the optimal structure, approximations, or other characteristics that make them more or less appropriate and effective for a given learning task. In addition, statistical hypothesis tests and scoring functions are continually being developed. As a result, the choice of algorithm and corresponding hyperparameters (hereafter, called a configuration) can have a sizable impact on the quality of the learned graph. Unfortunately, practitioners are faced with optimizing the configuration for the task at hand. Given that the problem is unsupervised, standard out-of-sample estimation methods used for supervised problems, such as cross-validation, cannot be directly applied.

We have recently developed **Out-of-sample Causal Tuning (OCT)** [1], a method that automatically selects an algorithmic configuration for learning a causal graph from an input dataset $\mathcal{D}$. The method is based on the observation that a causal network $\mathcal{G}$ induces a set of optimal predictive models. Specifically, if we are given a network $\mathcal{G}$, we identify the Markov boundary $\mathbf{MB}_{\mathcal{G}}(X)$ for each node X in $\mathcal{G}$ [2]. Under some conditions, $\mathbf{MB}_{\mathcal{G}}(X)$ is the minimal set of nodes that leads to an optimal predictive model for X [3]. Hence, a graph that is close to the true causal graph will lead to an optimal out-of-sample prediction power over all nodes. In our previous work [1], we showed that we can use this principle as a basis to select the optimal configuration for learning the causal structure and that our proposed approach improves causal structure learning. However, the method was initially developed for purely continuous and discrete data, assuming linear Gaussian relationships and multinomial distributions, and evaluated on synthetic data. We also proposed OCTs, a version of OCT that tries to avoid configurations leading to supersets of the correct Markov boundaries. OCTs performed well on continuous data, but not on discrete data.

In this work, we improve and extend our approach as follows.

1) We provide a common approach for continuous, discrete, and mixed data, using an information-theoretic approach. We also develop OCT, so as to be suitable for nonlinear data.
2) We introduce the *Sparsity Penalty*, a step inside OCT, to avoid graphs with false-positive members in the Markov boundaries.
3) We introduce the *causal-based simulation*, to resimulate ground-truth data that have properties (joint distributions) that are similar to real-world datasets.
4) We use both simulated and resimulated data to compare our approach against alternatives based on network stability [4] or scoring criteria for in-sample fitting [5]. We show that OCT performs on par or better, while it is easily applicable to settings where other approaches are not (e.g., causally insufficient systems or nonlinear distributions).

The rest of the document is organized as follows. In Section II, we discuss the problem of hyperparameter tuning and its connection to causal discovery. In Section III, we briefly refer to the theoretical framework of causality. In Section IV, we discuss two approaches for parameter optimization and how they can be applied to tune causal discovery. In Section V, we describe the OCT method and introduce

Manuscript received 2 August 2021; revised 15 March 2022; accepted 11 June 2022. This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement under Grant 617393, and in part by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to Support Faculty Members and Researchers and the Procurement of High-Cost Research Equipment Grant" under Project 1941. *(Corresponding author: Konstantina Biza.)*

Konstantina Biza and Ioannis Tsamardinos are with the Computer Science Department, University of Crete, 70013 Heraklion, Greece (e-mail: konbiza@gmail.com).

Sofia Triantafillou is with the Department of Mathematics and Applied Mathematics, University of Crete, 70013 Heraklion, Greece.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3185842.

Digital Object Identifier 10.1109/TNNLS.2022.3185842

its extensions. In Section VI, we detail our experimental setup and the causal-based simulation. In Section VII, we report and discuss the results. In Sections VIII and IX, we propose ideas for future research and summarize our conclusions. We use upper-case letters to denote single variables or nodes (e.g., X), bold upper-case letters to denote sets (e.g., **MB**), and the symbols $\mathcal{G}$, $\mathcal{D}$, $\mathcal{M}$ for graphs, datasets, and models, respectively. Lower-case letters denote scalars or indices.

II. Problem Definition

In an application of causal discovery to real data, a practitioner is faced with selecting the appropriate algorithm to use. In addition, each algorithm requires the choice of the values of a certain number of hyperparameters that influence its output, like, for example, the significance threshold for a conditional independence test. Hyperparameters differ from model parameters in the sense that the former are set by the user, while the latter are estimated from the data. The impact of the choice of the hyperparameter values for a given algorithm in causal discovery has been noted in several papers [6], [7]. Optimizing over both algorithm and hyperparameters has been coined the *Combined Algorithm Selection and Hyperparameter optimization problem* in the supervised learning literature [8], CASH for short, or "tuning." We adopt the same terminology in this work. Notice that the choice of algorithm can also be represented as a hyperparameter. An instantiation of all hyperparameter values (including the algorithm) is called a configuration.

A related problem in statistics is the problem of *model selection*. In both cases, one optimizes among a set of possible choices. However, there are conceptual differences in perspectives on the problem. Historically, in statistics, different models are fit and then the final model is selected among all the ones that fit. The choice is often manual by visualizing the model's fit and residuals. Principled methods for model selection typically score the tradeoff between model fitting and model complexity. Such a model selection criterion is the Bayesian information criterion (BIC) [or similarly, the Akaike information criterion (AIC)] scoring fitting using the *in-sample* data likelihood and penalizing for the model's degrees of freedom. The main observation in model selection is that all models are trained on all training data; selection is based on the in-sample data fit.

In contrast, the CASH perspective focuses on the learning algorithm, not the specific models (model instances to be precise). It is not the model that is selected, but the algorithm and its hyperparameters (the configuration) to be applied to all data. For example, during cross-validating an algorithm, several models are produced. None of them is the final model. They only serve to estimate how accurate the models produced by the algorithm are on average. The final model to return is the model trained on all data using the learning algorithm; all other models serve only for estimating performance purposes. Thus, CASH selects algorithms, not models, typically using out-of-sample estimation protocols (e.g., cross-validation).

It is not straightforward to apply the above techniques to the CASH problem in causal discovery, since the task is inherently unsupervised and the true causal network is unknown. Thus, there is no direct way of estimating how well a model approximates the underlying causal structure. Despite its obvious importance to practitioners, the problem of tuning has not been extensively studied in the context of causal discovery.

III. Preliminaries

Causal Bayesian networks (CBNs) consist of a directed acyclic graph (DAG) $\mathcal{G}$ and a set of probabilities P [2]. Nodes in the DAG represent variables (we use the terms node and variable interchangeably) and directed edges represent direct causality (in the context of nodes in the graph). Each node can represent a continuous or discrete variable. If $X \rightarrow Y$ in a DAG $\mathcal{G}$, we say that X is a parent of Y, and Y is a child of X in $\mathcal{G}$. Two nodes that share a common child are called spouses.

The graph and the distribution are connected through the causal Markov condition (CMC). CMC states that every variable is independent of its noneffects given its direct causes. Given the CMC, $P(\mathbf{V})$ can be factorized as $P(X_1, \ldots, X_n) = \prod_i P(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$, where $\mathbf{Pa}_{\mathcal{G}}(X_i)$ denote the parents of X_i in $\mathcal{G}$. Equivalently, the CMC entails a set of conditional independencies expected to hold in the joint probability distribution of variables in the graph. CBNs can only model causally sufficient systems, meaning that no pair of variables share an unmeasured common cause (confounder). Extensions of CBNs such as maximal ancestral graphs (MAGs) [9] model causal relationships in causally insufficient systems.

There are two main approaches for learning causal structure: constraint-based and score-based. Constraint-based methods apply tests of conditional independence to a dataset and then try to identify all causal graphs that are consistent with these (in)dependencies. Hyperparameters of these methods may include the type of conditional independence test, the conditioning set size, and the significance threshold for rejecting the null hypothesis. Score-based methods try to identify the graphical model that leads to a factorization that is closest to the one estimated by the observational data [10]. Hyperparameters of score-based methods may include the scoring function, sampling, and structure priors. Hybrid algorithms combine the above techniques, applying both hypothesis testing and scoring to learn the graph.

In most cases, the causal structure cannot be uniquely identified from the data. Instead, a set of DAGs will entail the same independence relationships (or equivalent factorizations). These graphs are called Markov equivalent and share the same edges and some orientations. Both constraint- and score-based algorithms typically return a partially DAG (PDAG) that summarizes the invariant features of Markov equivalent graphs. Similarly, in the case of causal insufficiency, algorithms will return a partial ancestral graph (PAG) that summarizes invariant features of a family of Markov equivalent MAGs.

A causal graph (DAG or MAG) induces a Markov boundary $\mathbf{MB}(X)$ for each node of the graph. The $\mathbf{MB}(X)$ is the minimal set that renders X conditionally independent of any other node. It is unique for distributions faithful to the graph and it is invariant among all graphs in the same Markov equivalence class. The Markov boundary of a node in a DAG or MAG

consists of its adjacent nodes and the nodes that are reachable through a collider path [11], [12].

IV. Related Work

Despite recent advances in causal discovery, the problem of tuning causal discovery algorithms has received little attention. We now describe two main approaches that can be used for causal tuning.

A. Network Stability and the StARS Algorithm

Stability Approach to Regularization Selection (StARS) [4] is an algorithm for tuning the $\Lambda = 1/\lambda$ hyperparameter of the graphical lasso [13]. The goal of the method is to select the Λ that leads to the most stable network with respect to small perturbations of the input data. For the task of learning undirected graphical networks, the method is shown to outperform BIC and AIC in high-dimensional settings.

StARS has also been applied to tune causal discovery algorithms [6]. Specifically, StARS is used to tune the significance level threshold for constraint-based algorithms, and the penalty discount for the score-based algorithms. A modification of StARS, called Stable Edge-specific Penalty Selection (StEPS), was developed to improve the performance of the method to mixed data by introducing different sparsity parameters for different types of edges (connecting pairs of continuous, pairs of discrete, or continuous to discrete variables) [14].

For tuning causal configurations instead of a single hyperparameter, we modified StARS as follows. For a given configuration a, we first estimate the probability p_{XY} of the presence of each edge in the network, using subsampling (learning multiple networks using the same configuration on resamples of the data without replacement). The instability of the edge is then defined as $\xi_{XY} \equiv 2p_{XY} \cdot (1 - p_{XY})$, that is, it is twice the variance of a Bernoulli distribution with parameter p_{XY}. It is low when p_{XY} is close to 0 or 1, and high when it is close to 0.5. The network instability $N(a)$ is the average instability for configuration a over all edges. We note that in graphical lasso, lower values of Λ lead to sparser graphs, and the StARS algorithm selects based on both stability and sparsity of the graph. Therefore, we compute the number of edges $Q_{a,s}$ of the graph (regardless of orientation) estimated with the configuration a and subsample s. We average over all subsamples and order the instability metric $N(a)$ by increasing density. We then "monotonize" $N(a)$, that is, we define $\overline{N}(a_j) = \max_{i \leq j} N(a_i)$. Subsequently, we select the configuration $a^* = \arg\max_{a_i}\{\overline{N}(a_i) | \overline{N}(a_i) \leq \beta\}$, where β is a hyperparameter of the StARS method. The pseudo-code is presented in Algorithm 1. We set the value of β to be 0.05 as suggested in [4].

StARS selects configurations that are robust to a few outliers in the data but does not evaluate how the model fits the data. Thus, a configuration that makes the same systematic error will be favored. We also note that StARS does not account for instability with respect to edge orientations.

Algorithm 1 Tuning With StARS

Input: Dataset $\mathcal{D}$ over nodes $\mathbf{V}$, Configurations $\mathbf{A}$, Subsamples $\mathbf{S}$, Threshold β
Output: Configuration a^*

for $a \in \mathbf{A}$ **do**
  for $s \in \mathbf{S}$ **do**
    ▷graph estimation
    $\mathcal{G}_{a,s} \leftarrow \text{causalAlg}_a(\mathcal{D}_s)$
    $Q_{a,s} \leftarrow$ number of edges in $\mathcal{G}_{a,s}$
  ▷density estimation (average number of edges)
  $Q_a \leftarrow \overline{Q}_{a,s}$ over $\mathbf{S}$
  ▷edge instability
  for each pair of variables X, Y **do**
    $p_{a,XY} \leftarrow$ frequency of edge (X, Y) in $\{G_{a,s}\}_{s \in \mathbf{S}}$
    $\xi_{a,XY} = 2\ p_{a,XY}(1 - p_{a,XY})$
  ▷network instability
  $N(a) \leftarrow \overline{\xi}_{a,XY}$ over all edges
▷order network instability
Rank $N(a)$ by increasing Q_a
▷monotonize network instability
$\overline{N}(a_j) \leftarrow max_{i \leq j} N(a_i)$
▷select configuration
$a^* = \arg\max_{a_i}\{\overline{N}(a_i) | \overline{N}(a_i) \leq \beta\}$
return a^*

B. Balancing Fitting With Model Complexity

Another principle for selecting the model and corresponding configuration is to select based on the best tradeoff between *in-sample* fitting of the data and the model complexity. A specific instantiation of the principle for causal discovery tuning appeared in [5] and is based on the BIC [15]. BIC scores a causal model based on the likelihood of the data given the causal model and penalizes it with the degrees of freedom of the model. The BIC score has been proposed to tune the Peter and Clark (PC) algorithm [5]. For each output PDAG, the BIC score for an equivalent DAG is computed as follows:

$$\text{BIC} = \log(n)k - 2\text{LL} \tag{1}$$

where LL is the log-likelihood of the causal model, k is the degrees of freedom, and n is the number of samples. When the model is a CBN, the likelihood of the data is computed based on the corresponding factorization

$$P(\mathcal{D}|\mathcal{G}) = \prod_{ij} P(x_{ij}|\mathcal{G}) = \prod_{ij} P(x_{ij}|\mathbf{Pa}_{\mathcal{G}}(i)) \tag{2}$$

where P is the probability or probability density function, x_{ij} is the value of the ith variable of the jth sample, and $\mathbf{Pa}_{\mathcal{G}}(i)$, the parents of the variable i in $\mathcal{G}$. In order to compute $P(x_{ij}|\mathbf{Pa}_{\mathcal{G}}(i))$ a parametric, statistical model needs to be fit with outcome each variable i given its parents $\mathbf{Pa}_{\mathcal{G}}(i)$.

Algorithm 2 describes a tuning method for DAGs that selects a configuration by minimizing the BIC score. We also apply the same method using the AIC, which is proved to be asymptotically equivalent to leave-one-out cross-validation [16]. For mixed data, we use an approximation for the BIC score presented in [17] and [18]. The authors propose three scoring functions: the conditional Gaussian (CG) score, the mixed variable polynomial (MVP) score, and the

Algorithm 2 Tuning With Likelihood-Based Score (BIC)

Input: Dataset $\mathcal{D}$ over nodes $\mathbf{V}$, Configurations $\mathbf{A}$
Output: Configuration a^*
for $a \in \mathbf{A}$ **do**
  ▷graph estimation
  $\mathcal{G}_a \leftarrow \text{causalAlg}_a(\mathcal{D})$
  $\mathcal{G}'_a \leftarrow \text{pdagToDag}(\mathcal{G}_a)$
  ▷ score computation
  $LL_a \leftarrow 2 \log P(\mathcal{D}|\mathcal{G}'_a)$
  $Score_a \leftarrow \log(n)k - LL_a$
▷ select configuration
$a^* = \arg\min_a Score_a$
return a^*

degenerate Gaussian (DG) score. The scores have been developed for score-based structure learning, and assume Gaussian and multinomial distributions. MVP score does not assume linearity and it is not score equivalent. We use the CG and DG scores as tuning methods for DAGs with mixed data. For MAGs with continuous nodes, we compute the BIC score as in [19] assuming linear Gaussian distributions. Currently, there are no other approximations of BIC for MAGs with discrete and mixed data.

There are several advantages to BIC scoring for model selection. First, several algorithms use BIC internally to search and score the best possible causal model, proving its effectiveness. Using (2), one needs to fit models for each node X_i from only its parents in the graph $\mathbf{Pa}(X_i)$. In comparison, the proposed method below employs models for each node given its Markov boundary. The latter is a superset of the parents and thus requires more samples to be fit accurately. There are also some disadvantages, a major one being that it requires the computation of likelihood and the degrees of freedom of the causal model. This is typically possible only with statistical, parametric models such as Gaussian and multinomial.

V. Tuning Based on Predictive Performance

In this work, we propose to tune causal discovery algorithms using the predictive performance of the learned graphs. The main principle of our proposed method is *to treat a causal model as a set of predictive models*. Specifically, a causal model $\mathcal{G}$ induces a Markov boundary $\mathbf{MB}(X)$ for each node of the graph. The $\mathbf{MB}(X)$ is the minimal set that renders X conditionally independent of any other node. It is unique for distributions faithful to the graph and it is invariant among all graphs in the same Markov equivalence class. Under some conditions on the appropriateness of the learning algorithm and the metric of performance, $\mathbf{MB}(X)$ is the minimal set of nodes that is necessary and sufficient for optimal prediction of X [3]. Thus, a successful learning algorithm will learn a causal graph that leads to optimal predictive models for each node.

This allows us to evaluate the configurations producing causal models using *out-of-sample performance estimation protocols* such as cross-validation. Notice that out-of-sample performance estimation is not generally possible for causal discovery, since the ground truth is generally not known. Therefore, we evaluate causal discovery algorithms based on a set of relevant predictive tasks. A similar approach has been suggested for other unsupervised learning tasks, such as dimensionality reduction with the principal component analysis (PCA) algorithm [20] and clustering [21].

Algorithm 3 OCT

Input: Dataset $\mathcal{D}$ over nodes $\mathbf{V}$, Configurations $\mathbf{A}$, Folds $\mathbf{K}$, Significance threshold s
Output: Configuration a^*
for $a \in \mathbf{A}$ **do**
  for $k \in \mathbf{K}$ **do**
    ▷ graph estimation
    $\mathcal{G}_{a,k} \leftarrow \text{causalAlg}_a(\mathcal{D}_k^{train})$
    ▷ predictive modeling
    for $X \in \mathbf{V}$ **do**
      $\mathbf{MB}_{a,k,X} \leftarrow \text{markovBoundary}(X,\ \mathcal{G}_{a,k})$
      $\mathcal{M}_{a,k,X} \leftarrow \text{fitModel}(X,\ \mathbf{MB}_{a,k,X},\ \mathcal{D}_k^{train})$
      $\hat{X}_{a,k} \leftarrow \text{predict}(\mathcal{M}_{a,k,X},\ \mathcal{D}_k^{test})$
  ▷ predictive performance evaluation
  for $X \in \mathbf{V}$ **do**
    $\hat{X}_a \leftarrow$ pool $\hat{X}_{a,k}$ over $\mathbf{K}$
    $I_{a,X} \leftarrow \text{mutualInfo}(X,\ \hat{X}_a)$
  $I_a \leftarrow \overline{I_{a,X}}$ over $\mathbf{V}$
▷ average size of Markov boundaries
$MB_a \leftarrow \overline{|\mathbf{MB}_{a,k,X}|}$ over $\mathbf{K}$ and $\mathbf{V}$
▷ configuration that maximizes predictive performance
$a^* = \arg\max_a I_a$
▷ select configuration
$a^* = \text{SparsityPenalty}(\mathbf{A},\ a^*,\ X,\ \hat{X},\ MB,\ I,\ s)$
return a^*

We call this approach OCT. As shown in Algorithm 3, OCT selects the configuration resulting in the best set of predictive models (one for each node), using an out-of-sample protocol. The method takes as input a dataset over variables $\mathbf{V}$ and a set of configurations of causal discovery algorithms $\mathbf{A}$. It also takes as input the folds $\mathbf{K}$ for the cross-validation and a significance threshold s. For each configuration a and each fold k, we estimate a causal graph by running the corresponding configuration causalAlg_a on the training dataset $\mathcal{D}_k^{\text{train}}$. Subsequently, OCT identifies the Markov boundary $\mathbf{MB}(X)$ of each variable X, builds a predictive model for X based on $\mathbf{MB}(X)$ (any supervised machine learning method can be used in this step), and predicts the target variable using the test set $\mathcal{D}_k^{\text{test}}$. We then pool the predictions of all folds and evaluate the predictive performance.

To evaluate the predictive performance of the learned model, we need to compare the true values of each X in the dataset with the pooled predictions $\hat{X}$ from the cross-validation procedure. Popular choices for this evaluation are the root mean squared error for continuous targets and (multiclass) Area Under Curve (AUC) for discrete targets. However, these metrics are on different scales. This creates problems in applying Algorithm 3 to data with both continuous and discrete

variables since the performance of the causal configuration is judged based on the predictive performance over all nodes.

To make the predictive performance for discrete and continuous variables comparable, we compare the true values of the target node X with the pooled predictions $\hat{X}$ using mutual information. Higher values of mutual information indicate that the predictions hold the expected amount of information for the true values. For continuous variables, the mutual information is defined as

$$I(X, \hat{X}) = \int_x \int_{\hat{x}} p(x, \hat{x}) \log \frac{p(x, \hat{x})}{p(x)p(\hat{x})} \tag{3}$$

where $p(x)$ and $p(\hat{x})$ are the marginal densities of X and $\hat{X}$ and $p(x, \hat{x})$ is their joint density. For Gaussian distributions, $I(X, \hat{X})$ can be computed as

$$I(X, \hat{X}) = -\frac{1}{2} \log(1 - \rho^2) \tag{4}$$

where ρ is the correlation coefficient of X and $\hat{X}$. Mutual information can also be approximated in a nonparametric way using k-nearest neighbors [22]. However, due to the computational cost, in this work, we assume that the variables follow Gaussian distributions.

For discrete variables, the integrals in (4) are replaced by the sum, so the marginal and joint distributions are computed by counting the number of samples falling in each class c

$$I(X, \hat{X}) = \sum_{c_x \in \mathbf{C}} \sum_{c_{\hat{x}} \in \mathbf{C}} P(c_x, c_{\hat{x}}) \log \frac{P(c_x, c_{\hat{x}})}{P(c_x)P(c_{\hat{x}})}. \tag{5}$$

After computing mutual information for every node, the overall performance of a configuration causalAlg_a is the average mutual information of all of the predictive models (one for each variable).

Asymptotically, the true causal graph will be among the models that achieve the best performance.

Theorem 1: Assuming that the following conditions hold: 1) data are generated by a CBN $\mathcal{G}_{\text{true}}$ over variables $\mathbf{V}$; 2) the learning algorithm can exactly learn the conditional distribution of each node $X \in \mathbf{V}$ given its $\mathbf{MB}$; and 3) the learning algorithm uses a proper scoring criterion, that is, a function that is maximum when the algorithm's probabilistic predictions report the true probability distribution [23]. Then any DAG $\mathcal{G}$ for which $\mathbf{MB}_{\mathcal{G}}(X) \supseteq \mathbf{MB}_{\mathcal{G}_{\text{true}}}(X)\ \forall\ X \in \mathbf{V}$ will asymptotically have the maximum score.

Proof: If a proper scoring rule is used, then the highest performance for each variable X can only be obtained by the true probability distribution $P(X|\mathbf{V} \setminus X) = P(X|\mathbf{Z})$ for any set $\mathbf{Z}$ that is a superset of the Markov boundary. Hence, any DAG $\mathcal{G}$ for which $\mathbf{MB}_{\mathcal{G}}(X) \supseteq \mathbf{MB}_{\mathcal{G}_{\text{true}}}(X)\ \forall\ X \in \mathbf{V}$, $\mathcal{G}$ will asymptotically achieve optimal performance. □

Intuitively, causal models that miss members of a $\mathbf{MB}(X)$ will achieve a lower predictive performance than possible, as they lack informational predictors. Causal models that add false-positive members of a $\mathbf{MB}(X)$ may result in overfitting in finite samples. However, in the large sample limit, graphs that entail Markov boundaries that are supersets of the true Markov boundaries will also achieve the maximum score and could be selected based on predictive performance. Therefore, selecting solely on predictive performance does not penalize for false-positive edges and could end up selecting very dense graphs.

Algorithm 4 Sparsity Penalty

Input: Configurations $\mathbf{A}$ and a^*, True variables X, Predicted variables $\hat{X}$, Size of Markov boundaries MB, Performances I, Significance threshold s
Output: Configuration a^*

for $a \in \mathbf{A} \setminus a^*$ **do**
  ▷ observed difference in performance
  $T^a_{obs} = (I_{a^*} - I_a)$
  ▷ permutation test
  for $p = 1$ to 1000 **do**
    for $X \in \mathbf{V}$ **do**
      $\hat{X}'_{a^*}, \hat{X}'_a \leftarrow \text{swap}(\hat{X}_{a^*}, \hat{X}_a)$
      $I'_{a^*,p,X} \leftarrow \text{mutualInfo}(X, \hat{X}'_{a^*})$
      $I'_{a,p,X} \leftarrow \text{mutualInfo}(X, \hat{X}'_a)$
    $I'_{a^*,p} \leftarrow \overline{I'_{a^*,p,X}}$ over $\mathbf{V}$
    $I'_{a,p} \leftarrow \overline{I'_{a,p,X}}$ over $\mathbf{V}$
    $T^a(p) = (I'_{a^*,p} - I'_{a,p})$
  $p_{val}(a) = \frac{|T^a \geq T_{obs}|}{P}$
▷ select configuration
if $\exists\ a \in \mathbf{A}$ s.t. $p_{val}(a) > s$ and $a = \arg\min_a MB_a$ **then**
  $a^* = a$
return a^*

To address this issue, we use a postprocessing step that selects the model with the smallest Markov boundaries, among all equally performing configurations, up to statistical indistinguishability. We call this procedure *Sparsity Penalty* (see Algorithm 4). The method looks for configurations that have a statistically indistinguishable performance from the configuration that maximizes predictive performance (a^*) and returns the one with the smallest Markov boundaries, on average. Specifically, the method goes through all configurations $a \in A$ and tests whether their predictions are equal to the predictions of the optimal configuration a^*. The null hypothesis assumes that the difference in performance is zero, on average. Under the null hypothesis, we can randomly swap half of the predictions obtained from a^*, with the ones obtained by the configuration a to test. We create 1000 permuted sets and we swap the same samples for all nodes. We compute the test statistic $T^a(p)$ for the configuration a and the permuted set p. This is the difference in performance between the swapped predictions of a^* and a in the set p. The T^a_{obs} is the observed difference in performance between a^* and a. The p-value of a is the number of times T^a is greater than or equal to T^a_{obs} divided by the number of permuted sets. We find the configurations that have p-value above a significance level $s = 0.05$, that is, their performance is not significantly different from a^*. Among them, we select the one with the smallest Markov boundaries, averaged over all nodes and folds. If no configuration meets the above conditions, Sparsity Penalty will not change the configuration a^*.

One of the advantages of OCT is that it does not inherently need to make parametric assumptions about the data distribution; one could potentially employ any applicable modeling method in machine learning or statistics, and it will asymptotically select the optimal configuration with respect to prediction (assuming the conditions in Theorem 1 hold). In addition, it can be applied to any data type, including mixed variables. In this work, we propose to use Random Forests [24] as predictive learning algorithms, since they do not make assumptions about the data distribution and can handle mixed variables. In addition, we examine the mutual information as a metric of predictive performance, in order to average the results over continuous and discrete nodes. OCT can be also applied to both DAGs and MAGs.

On the other hand, there are some limitations of OCT. The first is the choice of the predictive modeling algorithm. If the algorithm cannot approximate the true distribution, the procedure can underperform. In addition, in this work, we compute the mutual information assuming Gaussian distribution for the true and predicted values of continuous nodes. A nonparametric computation would have obvious impacts on the computational cost, and the permutation testing inside OCT would be unfeasible. Furthermore, like StARS, OCT also requires a hyperparameter, the significance threshold s. Finally, predictive performance based on Markov boundaries can fail to distinguish among graphs that can be distinguished by scoring criteria like BIC. For example, $V_1 \rightarrow V_2 \leftarrow V_3$ and any full graph over $\{V_1, V_2, V_3\}$ entail the exact same Markov boundaries for all variables, but they are not Markov equivalent. Asymptotically, a scoring criterion like BIC will select the configuration that leads to the correct graph, while OCT cannot distinguish between them. Both of them will have the same performance and Markov boundary sizes, and OCT will consider the relevant configurations equivalent.

In addition, mutual information measures predictive performance for discrete and continuous variables in the same units, but the range could grow without bound. For a discrete variable X, this happens as the size d of the domain of X increases. For a continuous variable X, this happens as the linear correlation ρ, between X and its predicted values $\hat{X}$, increases in absolute value. Theoretically, it could be the case that a variable with very high $I(X, \hat{X})$ dominates the sum of mutual information over all variables; OCT would then select the causal algorithm that optimizes the Markov boundary of that specific variable, effectively ignoring the rest of the variables. However, the growth of mutual information is logarithmic to d and ρ and it does not occur for typical scenarios in practice. In our experiments, the mutual information among variables in the same network never differs by more than one order of magnitude. We do warn the user of OCT, however, that in the extreme case of deterministic relations where $\rho = 1$, $I(X, \hat{X}) = \infty$ and OCT will fail.

VI. Experimental Setup

A. Causal Configurations

For our experiments, we use a variety of causal discovery algorithms. If all variables are measured, we include PC variants (PC [25], Conservative PC (CPC) [26], PC-stable (PCstable), Conservative PC-stable (CPCstable) [27]), Fast Greedy Equivalence Search (FGES) [28], [29], and Linear Non-Gaussian Acyclic Model (LiNGAM) [30]. If we assume that latent variables exist, we use Fast Causal Inference (FCI) variants (FCI [25], Fast Causal Inference-Max (FCI-Max) [31], Really FCI (RFCI) [32]), and Greedy FCI (GFCI) [33]. PC variants, FGES, and FCI variants can be applied to any data type (continuous, discrete, or mixed), together with the appropriate independence test and/or scoring function. LiNGAM algorithm is appropriate for continuous variables with non-Gaussian error terms. We also include the hybrid algorithms Max-Min Hill-Climbing (MMHC) [34] and MAG Max–Min Hill-Climbing (M3HC) [19] for discrete DAGs and continuous MAGs, respectively. The causal discovery algorithms are not limited to the above options. New causal discovery algorithms are continually being developed (see [35]).

Most of the above algorithms require an independence test and/or scoring function. For discrete data, we use the chi-square (Chi2) and G-square (G2) tests and the Bayesian Dirichlet equivalent uniform (BDeu) and BIC scores. For continuous data, we use the Fisher-Z and conditional correlation independence (CCI) tests. The scoring function is the BIC score. The CCI [36] test is proposed for nonlinear, non-Gaussian data. The kernel conditional independence (KCI) [37] is also an efficient test for nonlinear, non-Gaussian cases; however, due to its computational cost, we do not include it in our experiments. For mixed data, we use the CG-BIC and DG-BIC scores [17], [18]. These scores compute the log-likelihood and the degrees of freedom and approximate the BIC score. They are also adapted as independence tests (CG-LRT, DG-LRT). We can apply them to purely continuous or discrete data, as well. For the rest of this article, we use the abbreviations CG and DG to refer to both CG-BIC and DG-BIC scores and CG-LRT and DG-LRT independence tests, respectively.

The independence tests and scoring functions require, in turn, a choice of hyperparameter values. In this work, we set two different levels of significance for the independence tests (0.01, 0.05). We also change the penalty discount (1, 2) and the structure prior (0, 1, and 2) of the scores. The penalty discount penalizes the model complexity, and the structure prior corresponds to the expected number of parents of each node (zero value means that it is not included in computation). All of the above control the sparsity of the estimated graph. Any other hyperparameter that may be required has the default value. In the case of MAGs, we set penalty discount 1 and structure prior 1 to avoid many combinations with GFCI.

Table I summarizes the possible choices for causal discovery and the tuning methods that we use in this work. On the left part, we show the algorithms, independence tests, and scoring functions for each graph type (continuous, discrete, mixed DAGs or MAGs). A configuration is a combination of an algorithm, an independence test and/or a scoring function, and a set of values for the rest of the hyperparameters (significance level, penalty discount, and structure prior). Depending on the graph type, we have a variety of possible configurations ranging from 20 to 56. On the right columns, we show the

TABLE I
CAUSAL CONFIGURATIONS AND TUNING METHODS

Graph Type	Causal Hyper-parameters			# Config.	Tuning Methods					
	Algorithms	Independence Tests	Scores		OCT	BIC	AIC	CG	DG	StARS
Continuous DAG	PC variants FGES LiNGAM	Fisher-Z, CCI, CG, DG	BIC, CG, DG	51	✓	✓	✓			✓
Discrete DAG	PC variants FGES MMHC	Chi2, G2, CG, DG G2	BIC, BDeu, CG, DG BDeu	55	✓	✓	✓			✓
Mixed DAG	PC variants FGES	CG, DG	CG, DG	28	✓			✓	✓	✓
Continuous MAG	FCI variants GFCI M3HC	Fisher-Z, CCI, CG, DG Fisher-Z, CCI, CG, DG Fisher-Z	BIC, CG, DG BIC	50	✓	✓				✓
Discrete MAG	FCI variants GFCI	Chi2, G2, CG, DG Chi2, G2, CG, DG	BIC, BDeu, CG, DG	56	✓					✓
Mixed MAG	FCI variants GFCI	CG, DG CG, DG	CG, DG	20	✓					✓

TABLE II
NONLINEAR FUNCTIONS

Abbr	Causal Functional Relations	Abbr	Causal Functional Relations
$tanh$	$Y = \tanh(\sum_i(\beta_i X_i) + \epsilon)$	X^2	$Y = \sum_i(\beta_i X_i^2) + \epsilon$
$1/X$	$Y = \sum_i(\beta_i X_i^{-1}) + \epsilon$	$\|X\|$	$Y = \sum_i(\beta_i \|X_i\|) + \epsilon$
$exp0.5$	$Y = \sum_i(\beta_i \operatorname{sign}(X_i)\|Xi\|^{0.5}) + \epsilon$	$exp1.5$	$Y = \sum_i(\beta_i \operatorname{sign}(X_i)\|Xi\|^{1.5}) + \epsilon$
log	$Y = \sum_i(\beta_i \log\|X_i\|) + \epsilon$	$logcosh$	$Y = \sum_i(\beta_i \log(\cosh(X_i))) + \epsilon$
$prod$	$Y = \prod_i(\beta_i X_i) + \epsilon$	$prode$	$Y = (\prod_i(\beta_i X_i))\epsilon$

TABLE III
REAL-WORLD DATA

Name	Samples	Nodes	Continuous	Discrete	Edges
Iris	150	5	4	1	6
Wine	178	14	13	1	19
Heart Disease	297	14	5	9	14
Wine quality	1599	12	12	0	36
Breast Cancer Wisconsin	683	10	9	1	20
Car Evaluation	1728	7	0	7	5
Abalone	4177	9	8	1	26
Forest Fires	517	13	11	2	15
Student Performance	395	33	16	17	20

tuning methods that we apply in this work. OCT and StARS are applicable to any graph and data type, while the scores are appropriate only for specific cases.

We use the tetrad project for the simulation of mixed data, causal discovery algorithms, BIC, AIC, CG, and DG scores (https://github.com/cmu-phil/tetrad). The code for our experiments, including the implementation of OCT, can be found in the OCT folder in https://github.com/mensxmachina. In the same link is also the code for MMHC, M3HC, and BIC scores for continuous MAGs (CausalExplorer, M3HC folders).

B. Data Simulation

We simulate three types of data: continuous, discrete, and mixed. For continuous data, we use a linear Gaussian model, as well as nonlinear relations, as shown in Table II and appeared in [36]. Every variable in Table II is a function of its parents and a noise variable, which follows a uniform distribution, $\mathcal{U}(-1, 1)$. In all cases, the absolute coefficients β range between 0.1 and 0.9. The nonlinear, non-Gaussian data have also been used in [36] to evaluate the performance of the CCI and KCI tests. For discrete data, the conditional probability tables are sampled randomly from a Dirichlet distribution with $a = 0.5$. For each discrete variable, the number of categories ranges from 2 to 5. We create mixed data with the conditional Gaussian [17] and the Lee and Hastie model [38]. The continuous and discrete variables follow Gaussian and multinomial distributions, respectively. Both models have been used to evaluate the CG and DG scores in [18]. We simulate the data from known DAGs. For the experiments on MAGs, we randomly set as latent a percent of the nodes and we remove it from the dataset. We then convert the DAG to an MAG.

C. Causal-Based Simulation

We aim to evaluate the tuning methods on real data, as well. For this task, we apply a resimulation method, which is based on [39]. The idea is to estimate a graph $\mathcal{G}'$ from a real dataset $\mathcal{D}$ and then to simulate data from this graph. Our goal is to create data with similar properties to real-world problems.

In this work, we resimulate data with the *causal-based simulation* method, described in Algorithm 5. First, we use the FGES algorithm to learn a causal DAG $\mathcal{G}'$ over the variables in dataset $\mathcal{D}$. Then, for each variable X, we need a model that gives us $\hat{X} = P(X|\mathbf{Pa}_X)$ for discrete variables and $\hat{X} = E[X|\mathbf{Pa}_X]$ for continuous variables. The TRAIN function employs a 10-fold cross-validation to find the best model among all input predictive configurations $\mathbf{B}$, trains this model on all data, and returns it. We include support vector machines (SVMs) with linear and Gaussian kernels and Random Forests with default hyperparameters, for multiclass and regression problems ($\mathbf{B}^d$, $\mathbf{B}^c$). For continuous variables, we add noise to the predicted expected value, by adding a residual, selected uniformly at random (lines 16 and 17, Algorithm 5). We note that by doing this, we effectively assume that the variance of the error term is independent of the values of the predictive variables.

Table III shows the real data that we use for resimulation. We select nine datasets among the most popular from the University of California, Irvine (UCI) machine learning repository [40]. The first four columns show the number of samples, the number of nodes, and the number of continuous and discrete variables. We estimate the initial graph with the FGES

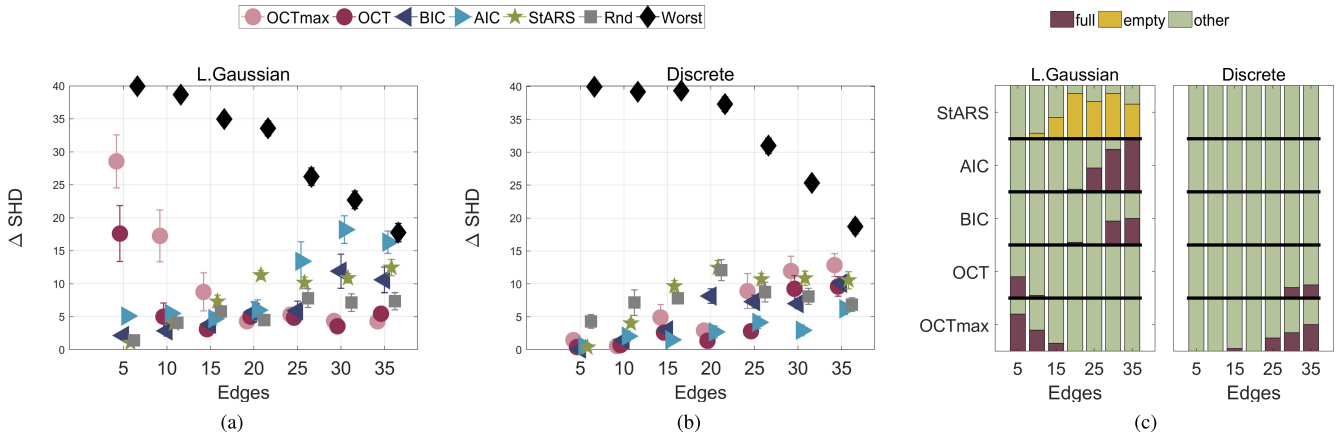


Fig. 1. **Sparsity Penalty is necessary for OCT.** (a) and (b) Tuning performance over increasing edges on linear Gaussian and discrete DAGs with ten nodes. We include the full and the empty graphs among the configurations. OCT performs well with respect to SHD in both cases. (c) Selected configurations: full, empty, and any other causal configuration. OCT avoids the full and the empty graph in most of the cases.

Algorithm 5 Causal-Based Simulation

Input: Real-data $\mathcal{D}$ over $\mathbf{V}$, Sample size n', Predictive Configurations $\mathbf{B}^d, \mathbf{B}^c$
Output: Dataset $\mathcal{D}'$

$\mathcal{D}' \leftarrow \varnothing$
▷ graph estimation
$\mathcal{G}', \mathbf{Pa} \leftarrow \text{causalAlg}(\mathcal{D})$
for $X \in \mathbf{V}$ **do**
  ▷ simulation
  if $\mathbf{Pa}_X$ is empty **then**
    $X' \leftarrow$ random samples from X
  else
    ▷ predictive modeling
    if X is discrete **then**
      $P(X|\mathbf{Pa}_X) \leftarrow \text{TRAIN}(X, \mathbf{Pa}_X, \mathbf{B}^d)$
    else
      $E[X|\mathbf{Pa}_X] \leftarrow \text{TRAIN}(X, \mathbf{Pa}_X, \mathbf{B}^c)$
      $R_X \leftarrow$ compute residuals for X
    ▷ assign predictions
    for $i = 1$ to n' **do**
      if X is discrete **then**
        $X'(i) \leftarrow$ draw from $P(X|\mathbf{Pa}_X(i))$
      else
        $r \leftarrow$ random residual from R_X
        $X'(i) \leftarrow E[X|\mathbf{Pa}_X(i)] + r$
  $\mathcal{D}' \leftarrow \mathcal{D}' \cup X'$
return $\mathcal{D}'$

algorithm, (CG-)BIC score, and default hyperparameters. The last column shows the number of edges in the estimated DAG. We then fit the models only once and resimulate 20 datasets of 1000 samples.

D. Tuning Performance

We evaluate the performance of the tuning methods with the following metrics: structural Hamming distance (SHD) [34], structural interventional distance (SID) [41], adjacency precision (AP), and adjacency recall (AR). SHD counts the number of steps needed to reach the true PDAG from the

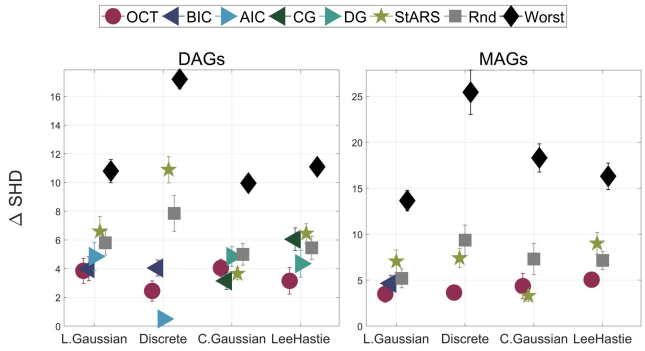
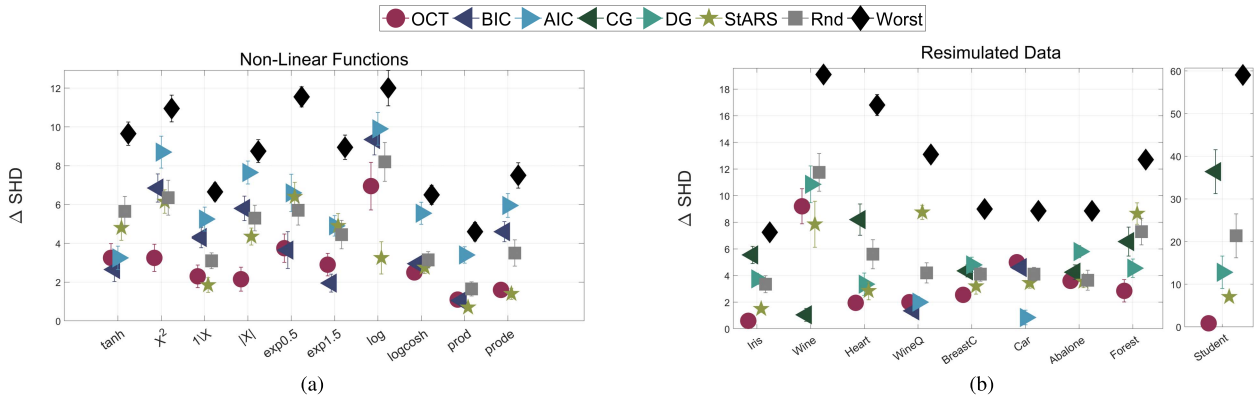


Fig. 2. **DAGs and MAGs with various data types.** Tuning performance over different data types on DAGs and MAGs. OCT performs well in most of the cases.

estimated PDAG. These modifications include edge removal, addition, and changes in orientation. Similarly, we use the SHD to compare the estimated PAG with the true PAG as in [42]. SID counts the number of pairs for which the intervention distribution is falsely estimated on the learned graph. SID upper (SIDu) and SID lower (SIDl) are the upper and lower limits of SID in the equivalence class of the network. AP computes the number of correctly estimated adjacencies divided by the number of all estimated adjacencies, and AR computes the number of correctly estimated adjacencies divided by the number of the true adjacencies. SHD and SID consider both the skeleton and orientations, while AP and AR only the skeleton. We compute each metric on the graph estimated on the entire dataset with a selected configuration.

VII. Results

We comparatively evaluate the performance of OCT against the other tuning methods. We report the distance of the selected graph from the oracle configuration. The oracle configuration achieves the best tuning performance value and corresponds to 0 difference, so lower is better for all metrics. For each experimental setting, a plot depicts the distance with respect to SHD achieved by each tuning method (y-axes). We also include the performance of a random selection of a configuration with uniform probability (Rnd). The black points show the performance with the worst configuration. Each point



Average over Non-Linear Functions

	ΔSHD	ΔSIDu	ΔSIDl	ΔAP	ΔAR
OCT	**2.98** ±0.51	12.81 ±1.28	15.22 ±1.25	**0.12** ±0.03	0.13 ±0.03
BIC	4.31 ±0.78	**9.91** ±0.85	14.47 ±1.42	0.23 ±0.05	0.15 ±0.04
AIC	6.12 ±0.68	10.80 ±0.76	**13.71** ±1.28	0.33 ±0.04	**0.09** ±0.02
StARS	3.65 ±0.63	15.79 ±2.41	21.71 ±3.12	0.17 ±0.04	0.25 ±0.03
Rnd	4.71 ±0.61	13.56 ±1.42	18.67 ±1.87	0.21 ±0.03	0.17 ±0.02

Average over Mixed Data Sets

	ΔSHD	ΔSIDu	ΔSIDl	ΔAP	ΔAR
OCT	**3.09** ±1.10	**15.98** ±4.89	**20.39** ±6.24	**0.06** ±0.01	**0.06** ±0.02
CG	9.48 ±4.56	20.69 ±3.89	22.90 ±7.91	0.21 ±0.05	0.08 ±0.02
DG	6.56 ±1.41	18.15 ±3.19	24.58 ±4.84	0.15 ±0.03	**0.06** ±0.02
StARS	4.95 ±1.07	18.13 ±3.78	28.61 ±6.09	0.11 ±0.04	0.16 ±0.07
Rnd	8.16 ±2.46	24.51 ±5.49	30.34 ±7.96	0.16 ±0.03	0.09 ±0.01

Fig. 3. **Nonlinear and real-world data.** (a) Tuning performance over nonlinear, non-Gaussian data. OCT is on average the best tuning method with respect to SHD, while BIC and AIC perform better with respect to SID. (b) Tuning performance over resimulated data. OCT performs well with respect to SHD on continuous and mixed data, and AIC on discrete data.

is the mean value over 20 networks. On the x-axis, we vary one of the following: data types, nonlinear functions, resimulated datasets, number of latent variables, number of nodes, number of edges, and number of samples. The default experimental setting contains graphs of ten nodes, 15 edges, and data with 1000 samples.

A. Sparsity Penalty Is Necessary for OCT

In the first experiment (see Fig. 1), we show the behavior of OCT with and without the Sparsity Penalty step. We call OCTmax the version that selects the configuration with the maximum predictive performance (without Sparsity Penalty). We focus on linear Gaussian and discrete DAGs with ten nodes and we increase the number of edges. We also include the full and the empty graph among the configurations. OCT performs often better than OCTmax with respect to SHD [see Fig. 1(a) and (b)]. In Fig. 1(c), we show the corresponding selected configurations. OCTmax may select the full graph in very sparse or dense graphs, while OCT can avoid the full graph more often. In addition, none of them select the empty graph. Based on the above, we suggest OCT instead of OCTmax and we will use it for the rest of our experiments. Regarding the other tuning methods, BIC performs better with linear Gaussian data, and AIC with discrete data. Both of them select the full graph on linear Gaussian data as the number of edges increases. StARS may select the empty graph. This happens when the other configurations are unstable, so the empty graph is the sparsest and most stable among them.

B. DAGs and MAGs With Various Data Types

Fig. 2 shows the tuning performance on DAGs and MAGs over four different data types: linear Gaussian, discrete, mixed from the conditional Gaussian model, and mixed from the Lee and Hastie model. We examine graphs of ten nodes, 15 edges, and four latent nodes (in the case of MAGs). OCT performs well, not only on purely continuous and discrete data, but also on mixed data. For mixed DAGs, CG performs well on the conditional Gaussian model, and DG scores on the Lee and Hastie model. This result is compatible with the results in [18], where CG and DG scores are evaluated with respect to structure learning. In the case of MAGs, we can compare against StARS for all data types and against BIC only for linear Gaussian data. OCT is suitable for all graphs and data types.

C. Nonlinear and Real-World Data

We then examine the performance over nonlinear, non-Gaussian data and resimulated data (see Fig. 3). The table below each plot depicts the average results over the x-axis with respect to SHD, SIDu, SIDl, AP, and AR. In Fig. 3(a), we focus on continuous DAGs and we simulate data with ten nonlinear functions (see Table II) and uniform error terms. OCT performs well in most of the cases and it is the best tuning method, on average, with respect to SHD. BIC and AIC scores perform better with respect to the SID metric. BIC and AIC favor more false-positive edges than OCT, which does not seem to affect the performance with respect to SID. This can be explained if these additional edges create supersets of the correct parent sets. Errors in orientation may also affect differently the SHD and SID metrics. Fig. 3(b) shows the results of resimulated data. No tuning method performs well on all datasets with respect to SHD. As before [see Figs. 1(b) and 2], AIC performs best on discrete data (car dataset). Overall, OCT performs well with respect to both SHD and SID on resimulated data.

D. Increasing Nodes, Samples, and Latents

In Fig. 4, we compare the performance of tuning algorithms on linear Gaussian data with an increasing number of nodes, samples, and latent variables. First, we increase the number of nodes in DAGs (10, 20, 50) with $1.5|V|$ edges (15, 30, 75).

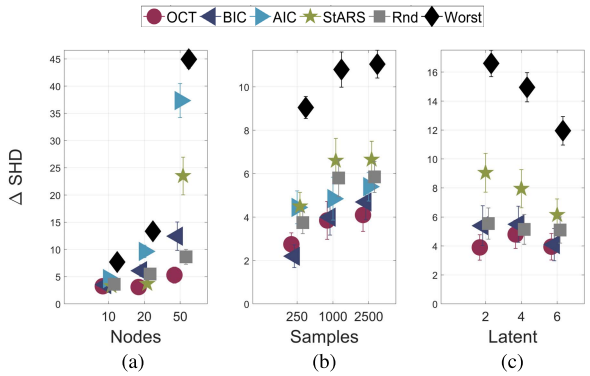


Fig. 4. **Increasing nodes, samples, and latents.** Tuning performance over increasing (a) number of nodes, (b) sample size, and (c) latent variables. OCT performs on par or better than the other tuning methods in most of the cases.

Due to computational cost, we do not include the CCI test and LiNGAM algorithm, so we tune over 42 configurations. As the number of nodes increases, OCT is the most suitable tuning method with respect to SHD. Then we increase the number of samples in DAGs (250, 1000, 2500). Fewer samples do not affect the performance of the tuning methods. On average, OCT and BIC perform similarly with respect to SHD. Finally, we increase the number of latent variables in MAGs (2, 4, 6). The initial DAGs have 12, 14, and 16 nodes and 20 edges. OCT performs on par or better than BIC with respect to SHD.

E. Computational Cost

Regarding the computational cost, OCT requires more time than the scoring functions and the StARS algorithm. This is because OCT runs each configuration ten times (due to the 10-fold cross-validation) and fits a random forest for each node of each configuration and fold. StARS runs each configuration 20 times on smaller subsets than OCT, and it does not require any other computationally costly procedure. In general, the computational cost of OCT is reasonable. For example, for a continuous DAG of ten nodes, 15 edges, and 1000 samples, OCT and StARS need around 73 and 25 min, respectively, to select among 51 configurations.

VIII. Future Research

We now present some ideas for future research to scale up OCT. OCT could be trivially parallelized at a granular level, since each configuration on each cross-validation fold can run independent of all others. Parallelizing the causal discovery algorithm is also possible, but algorithm-dependent. For high-dimensional data, to reduce the computational cost, one could opt to perform predictive modeling only on a subsample of the variables. How large this subsample is should require further research. In addition, one could drop configurations that cannot scale up and exceed a reasonable time-out limit. The time-out could be determined as a multiple of the execution time of a prototypical configuration with known properties and time complexity. Essentially, since OCT turns the problem into a supervised learning task, numerous ideas from the Automated Machine Learning (AutoML) literature, such as Bayesian Optimization [43], Meta-Learning [43], and Early Dropping [44], are available to try.

IX. Conclusion

We develop OCT to select the best combination of a causal discovery algorithm and its hyperparameters (causal configuration) for a given dataset. The main characteristic of OCT is the evaluation of causal models with respect to their predictive power, using as predictors the Markov boundaries and employing out-of-sample performance estimation protocols. OCT evaluates the predictive models with an information-theoretic approach and selects the causal configuration taking into account both the predictive performance and the sparsity of the output graph. In this work, we employ Random Forests as a predictive learning algorithm and we compute the mutual information. The above properties make OCT suitable for mixed data, variables with nonlinear relations, and causal sufficient or insufficient systems. However, an inappropriate choice of the learning algorithm or scoring criterion might affect OCT's tuning performance. We evaluate OCT in many experimental settings; DAGs and MAGs with purely continuous, discrete and mixed variables, and DAGs with nonlinear relationships. In addition, we introduce a causal-based simulation method to evaluate OCT on data that have similar properties to real-world problems. We also compare OCT against the StARS algorithm and several scoring functions: BIC, AIC, CG, and DG. OCT performs on par or better than the scoring functions that are often used internally during causal structure learning, but are not widely applicable to causally insufficient systems. AIC is the appropriate tuning method for discrete DAGs. StARS is a suitable tuning method for any data and graph type, like OCT, but further modifications are required to improve its performance in causal tuning. OCT performs well in many experimental settings with respect to the SHD metric and avoids false-positive adjacencies. It cannot always minimize the SID metric; however, it achieves low values. Overall, OCT is an effective tuning method for causal discovery, suitable for both DAGs and MAGs, graphs with mixed variables, and nonlinear distributions.

References

[1] K. Biza, I. Tsamardinos, and S. Triantafillou, "Tuning causal discovery algorithms," in *Proc. 10th Int. Conf. Probabilistic Graph. Models*, vol. 138, 2020, pp. 17–28.

[2] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[3] I. Tsamardinos and C. F. Aliferis, "Towards principled feature selection: Relevancy, filters and wrappers," in *AISTATS*, 2003, pp. 1–8.

[4] H. Liu, K. Roeder, and L. A. Wasserman, "Stability approach to regularization selection (stars) for high dimensional graphical models," in *Proc. NIPS*, vol. 24, 2010, pp. 1432–1440.

[5] M. H. Maathuis, M. Kalisch, and P. Bühlmann, "Estimating high-dimensional intervention effects from observational data," *Ann. Statist.*, vol. 37, no. 6A, pp. 3133–3164, Dec. 2009.

[6] V. K. Raghu, A. Poon, and P. V. Benos, "Evaluation of causal structure learning methods on mixed data types," *Proc. Mach. Learn. Res.*, vol. 92, pp. 48–65, Aug. 2018.

[7] J. Ramsey and B. Andrews, "A comparison of public causal search packages on linear, Gaussian data with no latent variables," *arXiv*, vol. abs/1709.04240, 2017.

[8] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2012, pp. 847–855.

[9] T. Richardson and P. Spirtes, "Ancestral graph Markov models," *Ann. Statist.*, vol. 30, no. 4, pp. 962–1030, 2002.

[10] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, no. 4, pp. 309–347, 1992.
[11] G. Borboudakis and I. Tsamardinos, "Forward-backward selection with early dropping," *J. Mach. Learn. Res.*, vol. 20, pp. 1–8, Jan. 2019.
[12] J. Pellet and A. Elisseeff, "Finding latent causes in causal networks: An efficient approach based on Markov blankets," in *Proc. NIPS*, 2008, pp. 1–8.
[13] J. H. Friedman, T. J. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 41–432, 2008.
[14] A. J. Sedgewick, I. Shi, R. M. Donovan, and P. V. Benos, "Learning mixed graphical models with separate sparsity parameters and stability-based model selection," *BMC Bioinf.*, vol. 17, no. S5, pp. 307–318, Dec. 2016.
[15] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, Jan. 1978.
[16] M. Stone, "An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion," *J. Roy. Statist. Soc. B, Methodol.*, vol. 39, no. 1, pp. 44–47, Sep. 1977.
[17] B. Andrews, J. Ramsey, and G. F. Cooper, "Scoring Bayesian networks of mixed variables," *Int. J. Data Sci. Analytics*, vol. 6, no. 1, pp. 3–18, Aug. 2018.
[18] B. Andrews, J. Ramsey, and G. F. Cooper, "Learning high-dimensional directed acyclic graphs with mixed data-types," *Proc. Mach. Learn. Res.*, vol. 104, pp. 4–21, Jul. 2019.
[19] K. Tsirlis, V. Lagani, S. Triantafillou, and I. Tsamardinos, "On scoring maximal ancestral graphs with the max–min Hill climbing algorithm," *Int. J. Approx. Reasoning*, vol. 102, pp. 74–85, Nov. 2018.
[20] P. O. Perry, "Cross-validation for unsupervised learning," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2009.
[21] W. Fu and P. O. Perry, "Estimating the number of clusters using cross-validation," *J. Comput. Graph. Statist.*, vol. 29, no. 1, pp. 162–173, Jan. 2020.
[22] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, Jun. 2004, Art. no. 066138.
[23] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *J. Amer. Stat. Assoc.*, vol. 102, no. 477, pp. 359–378, Mar. 2007.
[24] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[25] P. Spirtes, C. Glymour, S. N., and Richard, *Causation, Prediction, and Search*. Cambridge, MA, USA: MIT Press, 2000.
[26] J. Ramsey, J. Zhang, and P. L. Spirtes, "Adjacency-faithfulness and conservative causal inference," 2012, *arXiv:1206.6843*.
[27] D. Colombo and M. H. Maathuis, "Order-independent constraint-based causal structure learning," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3741–3782, 2014.
[28] D. M. Chickering, "Optimal structure identification with greedy search," *J. Mach. Learn. Res.*, vol. 3, pp. 507–554, Nov. 2002.
[29] J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour, "A million variables and more: The fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images," *Int. J. Data Sci. Anal.*, vol. 3, no. 2, pp. 121–129, Mar. 2017.
[30] S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen, "A linear non-Gaussian acyclic model for causal discovery," *J. Mach. Learn. Res.*, vol. 7, pp. 2003–2030, Dec. 2006.
[31] V. K. Raghu *et al.*, "Comparison of strategies for scalable causal discovery of latent variable models from mixed data," *Int. J. Data Sci. Anal.*, vol. 6, no. 1, pp. 33–45, Aug. 2018.
[32] D. Colombo, M. H. Maathuis, M. Kalisch, and T. S. Richardson "Learning high-dimensional directed acyclic graphs with latent and selection variables," *Ann. Statist.*, vol. 40, no. 1, pp. 294–321, 2012.
[33] J. M. Ogarrio, P. Spirtes, and J. Ramsey, "A hybrid causal search algorithm for latent variable models," in *Proc. 8th Int. Conf. Probabilistic Graph. Models*, vol. 52, Sep. 2016, pp. 368–379.
[34] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Mach. Learn.*, vol. 65, no. 1, pp. 31–78, 2006.
[35] O. Goudet, D. Kalainathan, P. Caillou, I. Guyon, D. Lopez-Paz, and M. Sebag, "Learning functional causal models with generative neural networks," 2017, *arXiv:1709.05321*.
[36] J. D. Ramsey, "A scalable conditional independence test for nonlinear, non-Gaussian data," 2014, *arXiv:1401.5031*.
[37] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf, "Kernel-based conditional independence test and application in causal discovery," in *Proc. 27th Conf. Uncertainty Artif. Intell.*, 2011, pp. 804–813.
[38] J. D. Lee and T. J. Hastie, "Learning the structure of mixed graphical models," *J. Comput. Graph. Statist.*, vol. 24, no. 1, pp. 230–253, 2015.
[39] A. Statnikov and C. F. Aliferis, "Analysis and computational dissection of molecular signature multiplicity," *PLOS Comput. Biol.*, vol. 6, no. 5, pp. 1–9, 2010.
[40] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[41] J. Peters and P. Bühlmann, "Structural intervention distance for evaluating causal graphs," *Neural Comput.*, vol. 27, no. 3, pp. 771–799, Mar. 2015.
[42] S. Triantafillou and I. Tsamardinos, "Score-based vs constraint-based causal learning in the presence of confounders," in *Proc. CFA@UAI*, 2016, pp. 1–9.
[43] F. Hutter, L. Kotthoff, and J. Vanschoren, "Automated machine learning: Methods, systems, challenges," in *Automated Machine Learning*. Cham, Switzerland: Springer, 2019.
[44] I. Tsamardinos, E. Greasidou, and G. Borboudakis, "Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation," *Mach. Learn.*, vol. 107, no. 12, pp. 1895–1922, Dec. 2018.

Konstantina Biza received the B.Sc. degree in biology from the National and Kapodistrian University of Athens, Athens, Greece, in 2016, and the M.Sc. degree in computer science from the University of Crete, Heraklion, Greece, in 2020, where she is currently pursuing the Ph.D. degree in computer science. Her research interests include causal discovery and machine learning.

Ioannis Tsamardinos received the B.Sc. degree from the University of Crete, Heraklion, Greece, in 1995, and the Ph.D. degree from the Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, USA, in 2001.

He joined the Department of Biomedical Informatics, Vanderbilt University, Nashville, TN, USA, as an Assistant Professor. In 2006, he joined the University of Crete, where he currently serves as a Professor of machine learning in the Computer Science Department. He has authored more than 140 peered-reviewed articles in the field of artificial intelligence, machine learning, and bioinformatics, on topics of causal discovery, feature selection, and applications in biomedicine.

Dr. Tsamardinos was a recipient of several international awards, including the ERC Consolidator and PoC Grants.

Sofia Triantafillou received the Diploma degree from the National Technical University of Athens, Athens, Greece, in 2006, and the Ph.D. degree in computer science from the University of Crete, Heraklion, Greece, in 2015.

She was a Post-Doctoral Researcher with Northwestern University, Evanston, IL, USA, and the University of Pennsylvania, Philadelphia, PA, USA. From 2018 to 2021, she was an Assistant Professor with the University of Pittsburgh, Pittsburgh, PA, USA. In 2021, she joined the University of Crete as an Assistant Professor of statistics at the Department of Mathematics and Applied Mathematics, where she teaches undergraduate and graduate statistics courses. Her current research interests include causal inference and the development of statistical methods for the analysis of multiple heterogeneous datasets.