FAST FLOW MATCHING BASED CONDITIONAL INDEPENDENCE TESTS FOR CAUSAL DISCOVERY

Shunyu Zhao,* **Yanfeng Yang***
Graduate University of Advanced Studies & The Institute of Statistical Mathematics
Tokyo, Japan
{zsydenny,yanfengyang0316}@gmail.com

Shuai Li
East China Normal University
Shanghai, China
shuaili.cq@foxmail.com

Kenji Fukumizu
The Institute of Statistical Mathematics
Tokyo, Japan
fukumizu@ism.ac.jp

ABSTRACT

Constraint-based causal discovery methods require a large number of conditional independence (CI) tests, which severely limits their practical applicability due to high computational complexity. Therefore, it is crucial to design an algorithm that accelerates each individual test. To this end, we propose the Flow Matching-based Conditional Independence Test (FMCIT). The proposed test leverages the high computational efficiency of flow matching and requires the model to be trained only once throughout the entire causal discovery procedure, substantially accelerating causal discovery. According to numerical experiments, FMCIT effectively controls type-I error and maintains high testing power under the alternative hypothesis, even in the presence of high-dimensional conditioning sets. In addition, we further integrate FMCIT into a two-stage guided PC skeleton learning framework, termed GPC-FMCIT, which combines fast screening with guided, budgeted refinement using FMCIT. This design yields explicit bounds on the number of CI queries while maintaining high statistical power. Experiments on synthetic and real-world causal discovery tasks demonstrate favorable accuracy-efficiency trade-offs over existing CI testing methods and PC variants.

1 INTRODUCTION

Conditional independence (CI) testing is a challenging task in statistics and machine learning. It has been widely used in causal discovery (Pearl, 1988; Spirtes et al., 2000) and variable selection (Dai et al., 2022; Yang et al., 2025). Assume we have p variables $X = (X_1, \ldots, X_p)$ with $X_i \in \mathbb{R}$ for all i. CI testing for causal discovery is to test the following hypotheses:

$$H_0 : X_i \perp\!\!\!\perp X_j | X_S \quad v.s. \quad H_1 : X_i \not\perp\!\!\!\perp X_j | X_S, \tag{1}$$

where X_i and X_j are arbitrary two variables among the p variables, and $S \subseteq [p] \setminus \{i, j\}$ is a (possibly empty) index set; we write $X_S := \{X_k : k \in S\}$. When S is empty, the test reduces to an unconditional independence test.

One of the most prominent applications of CI testing is constraint-based causal discovery. A representative and well-established example is the PC algorithm, which has served as a foundational method in this line of work and recovers a causal graph by repeatedly testing whether $X_i \perp\!\!\!\perp X_j \mid X_S$ for different triples (i, j, S) (Pearl, 1988; Spirtes et al., 2000; Kalisch & Bühlman, 2007). During its skeleton-learning stage, PC algorithm removes edges by searching over conditioning sets of increasing size, which can trigger a combinatorial number of CI tests and become the dominant computational bottleneck in high-dimensional settings (Kalisch & Bühlman, 2007; Colombo et al., 2014). As a result, designing computationally efficient CI tests is essential for making constraint-based causal discovery practical.

*Equal contribution.

Recently, many CI testing methods have been developed. The kernel-based methods measure the CI in reproducing kernel Hilbert spaces and conduct hypothesis testing accordingly (Fukumizu et al., 2007; Zhang et al., 2011; Shah & Peters, 2020; Scetbon et al., 2022; Guan & Kuang, 2025; He et al., 2025b; Miyazaki & Uematsu, 2025). However, kernel-based methods typically incur cubic time complexity with respect to the sample size, which severely limits their scalability and significantly slows down causal discovery on large datasets. Another line of work focuses on metric-based CI testing methods, which test CI relationships using carefully designed test statistics (Wang et al., 2015; Runge, 2018). However, their performance is often compromised in high-dimensional settings. Recently, Candès et al. (2018) proposed the conditional randomization test (CRT) framework, which is able to effectively control the type-I error under high-dimensional cases. However, a key limitation of this framework is that it requires knowledge of the conditional distribution of X_i given X_S, denoted by $P_{X_i|X_S}$ (Li et al., 2023; 2024). Fortunately, recent advances in generative models have rapidly progressed and effectively address this limitation (Ho et al., 2020; Song et al., 2021), giving rise to a line of CI tests based on generative models (Bellot & van der Schaar, 2019; Shi et al., 2021; Yang et al., 2025; Ren et al., 2025; He et al., 2025a).

Despite their flexibility, existing generative-based CI tests remain computationally expensive. Among these methods, the conditional diffusion model based conditional independence test (CDCIT) demonstrates strong performance on individual CI tests (Yang et al., 2025). However, its high computational cost makes causal discovery based on CDCIT impractical. More importantly, in causal discovery algorithms, both the target variable X_i and the conditioning set X_S vary across CI queries, often requiring conditional generative models to be retrained for different (i, S) pairs. This repeated retraining substantially increases the overall computational cost and severely limits scalability (Bellot & van der Schaar, 2019; Yang et al., 2025; Ren et al., 2025). Therefore, to make generative CI testing practical, it is crucial to both accelerate each individual test and avoid repeated model training across all CI queries in causal discovery algorithms.

To address the aforementioned challenges, we design a new CI testing method tailored for the PC algorithm, termed the Flow Matching based Conditional Independence Test (FMCIT). FMCIT leverages flow matching (FM) together with Picard sampling, enabling accurate conditional sampling and precise CI testing with substantially **reduced computation time**. Moreover, compared to previous generative-based CI testing approaches, our method requires **training the FM model only once**, which further leads to a significant speed-up of the overall causal discovery algorithm.

Beyond accelerating individual CI tests, we also aim to reduce the overall number of CI queries required by constraint-based causal discovery. To this end, we develop a guided PC-style skeleton learning framework that focuses CI testing on the most informative variable pairs and conditioning sets, yielding a practical accuracy–efficiency trade-off in high dimensions.

Our contributions are summarized as follows:

- **FMCIT:** We propose a novel and fast CI testing algorithm specifically designed for causal discovery. FMCIT leverages flow matching to learn the joint distribution of the data and employs Picard sampling for efficient sample generation. Moreover, FMCIT reformulates conditional sampling as an imputation task over the entire dataset given the conditioning variables, so the flow matching model is trained only once throughout the causal discovery process.

- **GPC-FMCIT:** a two-stage PC skeleton learning algorithm that combines fast screening with an edge-specific guided conditioning pool and a per-edge CI-query budget, using FMCIT as the main CI oracle.

- **Empirical validation:** extensive experiments on synthetic CI benchmarks and high-dimensional causal discovery, and real-world studies such as gene network inference, demonstrating favorable efficiency–accuracy trade-offs.

In Section 2.1, we propose FMCIT, which performs CRT using fast flow-matching-based conditional sampling with batched Picard–RePaint imputation. In Section 2.2, we integrate FMCIT into a guided and budgeted PC skeleton learning pipeline (GPC-FMCIT) to reduce the number of CI queries. Section 3 presents synthetic benchmarks for both CIT and causal discovery, and Section 4 evaluates our approach on real-world data. Background on CRT and the PC algorithm is provided in Appendix A.

2 METHODS

2.1 FLOW MATCHING BASED CONDITIONAL INDEPENDENCE TEST (FMCIT)

Our proposed CI testing method is based on the Conditional Randomization Test (CRT) which is detailed introduced in Appendix A.1. Despite the multiple advantages of CRT, it suffers from a fundamental limitation: the conditional distribution $P_{X_i|X_S}$ is typically unknown in practice. Previous work by Bellot & van der Schaar (2019) proposed estimating $P_{X_i|X_S}$ using generative adversarial networks (GANs); however, training a GAN is often unstable and prone to mode collapse, making it difficult to accurately learn the distribution Dhariwal & Nichol (2021). More recently, Yang et al. (2025) and Ren et al. (2025) proposed learning $P_{X_i|X_S}$ via score matching. However, due to the slow sampling speed of stochastic differential equation (SDE) solvers, both methods are computationally expensive. More importantly, when combined with the PC algorithm, these CI testing methods require retraining a generative model for each individual test, which dramatically increases the overall computational complexity.

To address the slow sampling speed of SDE solvers, we instead employ flow matching Lipman et al. (FM, 2023) to learn the conditional distribution $P_{X_i|X_S}$. The training process of FM is stable, and its ordinary differential equation (ODE) solver enables fast generation of samples that approximately follow $P_{X_i|X_S}$. Moreover, to reduce the number of model training procedures, we use FM to learn the joint distribution P_X of all variables only *once* across the whole PC algorithm. We view sampling from $P_{X_i|X_S}$ as imputing the missing values of X given X_S. We first generate a sample approximately following $P_{X|X_S}$ which is the distribution of X given X_S. Then, we only pick up the generated X_i and discard the other variables. We employ Picard sampler to generate X_i conditioned on X_S, which is a standard practice in generative missing value imputation and time series forecasting Hu et al. (2025).

Let $X(1) = X \in \mathbb{R}^p$ denote the target variable. FM connects a Gaussian noise $X(0)$ and $X(1)$ through an ordinary differential equation (ODE):

$$\frac{dX(t)}{dt} = X(1) - X(0), \quad X(0) \sim N(0, I_p), \quad t \in [0, 1], \tag{2}$$

which also implies that:

$$X(t) = tX(1) + (1-t)X(0). \tag{3}$$

When performing generation via the ODE, the target variable $X(1)$ is unknown in advance. Therefore, the vector field at the right-hand side of the ODE in (2) is replaced by the conditional expectation $\mathbb{E}[X(1) - X(0)|X(t)]$. In practice, this conditional expectation is also intractable. Therefore, we use a neural network v_θ to approximate the vector field by minimizing the following loss:

$$\mathbb{E}_{X(0)\sim N(0,I_p),\, X(1)\sim P_X,\, t\sim U(0,1)} \left\| X(1) - X(0) - v_\theta\left(tX(1) + (1-t)X(0),\, t\right) \right\|_2^2. \tag{4}$$

Consequently, the ODE used for data generation is given by:

$$\frac{dX(t)}{dt} = v_\theta(X(t), t), \quad X(0) \sim N(0, I_p), \quad t \in [0, 1]. \tag{5}$$

We view sampling from $P_{X|X_S}$ as imputing the entire X given the conditioning variables X_S. Here, $P_{X|X_S}$ denotes the conditional distribution of X given X_S, which includes the target conditional distribution $P_{X_i|X_S}$. To achieve this, we combine Picard sampling with RePaint Lugmayr et al. (2022); Hu et al. (2025) to inject the information from the conditioning set S throughout the whole generation process. Specifically, given a sampling schedule $0 = t_0 < t_1 < \cdots < t_L = 1$, we first draw $X(0) \sim N(0, I_p)$ as the initial point. In each time step $t_i < 1$, we apply one Picard sampling step to obtain an estimate of $X(1)$, denoted by $\widehat{X}(t_i)$:

$$\widehat{X}(t_i) = X(t_i) + \int_{t_i}^{1} v_\theta(X(t), t)dt \approx X(t_i) + (1-t_i)v_\theta(X(t_i), t_i). \tag{6}$$

Here, the $\approx$ is because we do not wish to expend substantial computational resources to evaluate the integral $\int_{t_i}^{1} v_\theta(X(t), t)\, dt$. Moreover, we note that the true vector field in the ODE (2) is linear, which allows the integral to be well approximated by $(1-t_i)\, v_\theta(X(t_i), t_i)$. Since the approximation is quite

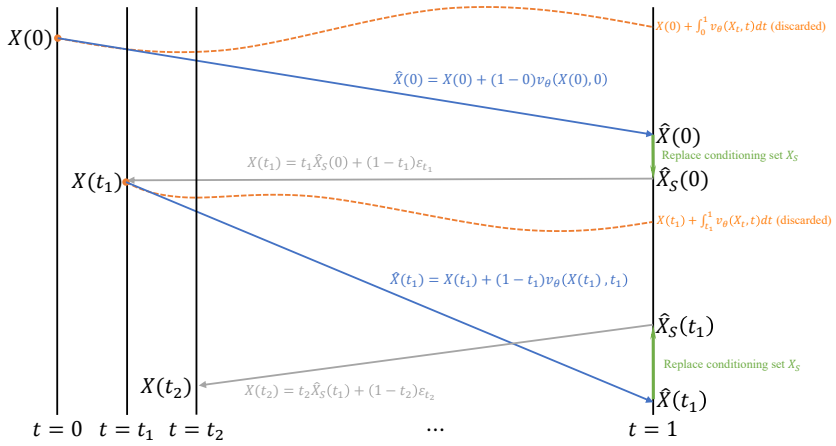

Figure 1: An illustration of Picard sampling and RePaint.

rough when t_i is small, we then replace the components of $\widehat{X}(t_i)$ corresponding to the conditioning set S with the given values of X_S, and map the modified estimate back to time $t = t_{i+1}$ using (3) together with a random noise $\varepsilon_{t_{i+1}} \sim N(0, I_d) : X(t_{i+1}) = t_{i+1}\widehat{X}(t_i) + (1 - t_{i+1})\varepsilon_{t_{i+1}}$. Inspired by RePaint Lugmayr et al. (2022), we use the random noise $\varepsilon_{t_{i+1}}$ rather than the initial noise X_0, which introduces stochasticity into the ODE sampling process and leads to higher-quality generated samples. In the last time step $t = t_{L-1}$, we repeat all processes without mapping it back.

Unlike Euler and Heun schemes that propagate local discretization errors step by step, Picard sampling performs a global fixed-point iteration Hartman (2002), yielding more stable and coherent imputations under observed-data constraints. RePaint introduces stochasticity into ODE-based sampling, making the imputed samples more realistic Lugmayr et al. (2022). Overall, the combination of Picard sampling and RePaint not only improves the quality of imputation but also avoids the high computational cost caused by repeated RePaint iterations. Our sampling method requires only 5–50 steps to generate high-quality samples, whereas SDE samplers used in diffusion models typically require more than 200 steps.

In conclusion, we reformulate the problem of sampling from the unknown conditional distribution $P_{X_i|X_S}$ in Section A.1 as an imputation task, where FM is used to impute the entire dataset conditioned on X_S. Notably, the CRT requires generating B samples, and we sample these B instances in parallel, which substantially reduces the computational cost of the CRT. The training procedure for flow matching, the sampling algorithm, and the overall FMCIT algorithm can be found in Algorithms 1, Algorithms 2, and Algorithms 3, respectively.

2.2 Guided PC skeleton learning with FMCIT

We briefly recall the PC-stable skeleton learning procedure in Appendix A.2. Our guidance is computed only from the screening outputs and is used to prioritize candidate conditioning variables when forming an edge-specific pool. It does not change the CI oracle in the refinement stage, which remains FMCIT. We integrate FMCIT into a two-stage skeleton learning procedure to accelerate PC by reducing both the *number* of CI queries and the *cost* per query. Let $G_{\text{scr}} = ([p], E_{\text{scr}})$ be a screening skeleton obtained by running PC-stable with FisherZ at level α_{scr} and maximum conditioning size $d_{\max}^{\text{scr}}$ Colombo et al. (2014). This screening stage removes evidently independent pairs cheaply and yields a sparse graph whose neighborhoods provide useful candidates for conditioning. The refinement stage initializes $G^{(0)} \leftarrow G_{\text{scr}}$ and then performs PC-style edge deletion only on edges in E_{scr}, using FMCIT as the CI oracle with significance α_{fm}.

In the refinement stage, at conditioning level $\ell \in \{0, 1, \ldots, d_{\max}\}$, for each current edge (i, j) we search for a separating set S with $|S| = \ell$ such that FMCIT accepts $X_i \perp\!\!\!\perp X_j | X_S$. If such an S is found, we remove the edge $i - j$ and record $\text{Sep}(i, j) \leftarrow S$ (and symmetrically $\text{Sep}(j, i) \leftarrow S$). A key computational challenge is that the number of candidate sets grows combinatorially with ℓ and

with the size of the adjacency sets. To control this growth, we restrict the candidate search space via an edge-specific *guided conditioning pool* of fixed size k and test only a budgeted number of conditioning sets per edge per level.

We first define the raw candidate set from the screening neighborhoods. Let $\mathrm{Adj}_{G_{\mathrm{scr}}}(i)$ denote the adjacency set of node i in G_{scr} and define

$$\mathcal{Q}_{ij} := \big(\mathrm{Adj}_{G_{\mathrm{scr}}}(i) \cup \mathrm{Adj}_{G_{\mathrm{scr}}}(j)\big) \setminus \{i, j\}.$$

Intuitively, $\mathcal{Q}_{ij}$ collects variables that are locally connected to i or j in the coarse screening graph and therefore are plausible candidates for separating i and j. We then construct a fixed-size pool

$$\mathcal{P}_{ij} \subseteq [p] \setminus \{i, j\}, \qquad |\mathcal{P}_{ij}| = k, \tag{7}$$

by selecting variables from $\mathcal{Q}_{ij}$ and completing the pool using a global ranking when $|\mathcal{Q}_{ij}| < k$. The global ranking can be any deterministic score that is cheap to compute from screening information (e.g., frequency/degree-based scores), and is used only to keep the pool size fixed across edges. Optionally, we compute a coarse component structure on $[p]$ (e.g., via stability frequencies across several screening levels, or via triangle support) and prioritize variables outside the components containing i and j, which tends to enrich the pool with variables that act as cross-region separators, in line with guided constraint-based ideas Shiragur et al. (2024). In our implementation, `stab` builds components from edge-selection frequencies across a small set of screening levels, while `tri` builds components from triangle-supported edges in G_{scr}.

In all cases, $\mathcal{P}_{ij}$ is constructed deterministically given the screening outputs and a random seed (if sampling is used), ensuring reproducibility.

Given $\mathcal{P}_{ij}$, at level ℓ we test only a budgeted family of conditioning sets

$$\mathcal{S}_{ij}^{(\ell)} \subseteq \binom{\mathcal{P}_{ij}}{\ell}, \qquad |\mathcal{S}_{ij}^{(\ell)}| \le M, \tag{8}$$

where M is the maximum number of tested sets per edge per level. If $\binom{k}{\ell} \le M$, we enumerate all sets in $\binom{\mathcal{P}_{ij}}{\ell}$; otherwise we randomly sample M distinct sets uniformly (with a fixed seed). For each $S \in \mathcal{S}_{ij}^{(\ell)}$, we compute the FMCIT p-value $p_{ij|S}$ and accept conditional independence when $p_{ij|S} > \alpha_{\mathrm{fm}}$. We remove the edge as soon as the first accepted separating set is found (early stopping), which further reduces the average number of CI queries per edge.

Each FMCIT query is implemented as a CRT with conditional resampling based on a flow-matching model. Importantly, in our pipeline the flow-matching model is trained once on the full joint distribution P_X (for fixed hyperparameters and normalization) and then reused across all refinement queries. For a given (i, j, S), FMCIT generates CRT resamples by imputing the missing coordinates of X conditioned on $\mathbf{X}_S$ using the Picard–RePaint sampler, and extracting the resampled coordinate $\mathbf{X}_i^{(b)}$ for $b = 1, \ldots, B$. This global-model reuse avoids retraining conditional samplers across CI queries and is crucial for scalability in a PC-style procedure.

We further allow the CRT repeat count in FMCIT to depend on the conditioning level,

$$B(\ell) = \begin{cases} B_0, & \ell = 0, \\ B_1, & \ell \ge 1, \end{cases} \qquad B(\ell) \le B_{\max},$$

which yields a practical speed–power trade-off: unconditional tests are typically cheaper and can use fewer randomizations, while higher-order conditional tests are noisier and benefit from more repeats. Optionally, one may also use a level-dependent statistic (e.g., a more expressive nonlinear statistic at $\ell = 0$ and a cheaper statistic for $\ell \ge 1$) without changing the overall procedure.

The guided conditioning pool restriction yields an explicit bound on the number of CI queries. Let $E^{(\ell)}$ be the edge set at the beginning of level ℓ in the refinement stage. Since refinement only deletes edges, we have $E^{(\ell)} \subseteq E_{\mathrm{scr}}$ for all ℓ. Moreover, by construction each edge tests at most M conditioning sets at level ℓ, hence the number of CI queries at level ℓ satisfies

$$N^{(\ell)} \le M|E^{(\ell)}| \le M|E_{\mathrm{scr}}|.$$

Summing over $\ell = 0, 1, \ldots, d_{\max}$ gives the total bound

$$N_{\mathrm{total}} \le (d_{\max} + 1)M|E_{\mathrm{scr}}|.$$

Table 1: Post-nonlinear CIT benchmark ($n = 1000$, $\alpha = 0.05$, 100 trials). We report empirical type-I error and power.

d_z	scale s	FMCIT		NNLSCIT		KCIT		EKCI		CDCIT	
		Type-I ↓	Power ↑	Type-I ↓	Power ↑	Type-I ↓	Power ↑	Type-I ↓	Power ↑	Type-I ↓	Power ↑
10	0.50	0.08	1.00	0.09	0.90	0.24	1.00	0.47	1.00	0.07	0.94
	0.75	0.05	1.00	0.08	0.55	0.24	0.97	0.45	0.98	0.04	0.74
	1.00	0.04	1.00	0.08	0.32	0.23	0.88	0.44	0.91	0.04	0.62
50	0.50	0.05	1.00	0.11	0.88	0.14	1.00	0.20	1.00	0.08	0.97
	0.75	0.07	1.00	0.06	0.52	0.14	1.00	0.23	1.00	0.07	0.80
	1.00	0.05	0.98	0.05	0.30	0.15	0.99	0.24	0.99	0.05	0.69

Therefore, the overall refinement budget is controlled by three transparent quantities: the screening sparsity $|E_{\mathrm{scr}}|$, the per-edge budget M, and the maximum conditioning size $d_{\max}$. Algorithm 4 summarizes the full procedure.

3 Simulation Study

3.1 Synthetic conditional independence testing benchmark

Data generating process. We evaluate the performance of the proposed FMCIT using a post-nonlinear data generation mechanism, which has been widely adopted in previous CI testing studies Scetbon et al. (2022); Li et al. (2023; 2024); Yang et al. (2025); Ren et al. (2025). Let $Z = (Z_1, \ldots, Z_{d_z})$ be a random vector whose d_z components are independently drawn from $N(0,1)$ and define $m := \frac{1}{d_z}\sum_{k=1}^{d_z} Z_k$. We generate X, Y as

$$\begin{aligned} H_0 &: X = f_1(m + s\varepsilon_x), & Y &= f_2(m + s\varepsilon_y), \\ H_1 &: X = f_1(m + s\varepsilon_x) + 0.5e_b, & Y &= f_2(m + s\varepsilon_y) + 0.5e_b, \end{aligned}$$

where s is a hyperparameter controlling noise strength, $\varepsilon_x, \varepsilon_y, \varepsilon_b \overset{i.i.d.}{\sim} N(0,1)$, and f_1, f_2 are randomly chosen from $\{X, X^2, \cos(X), X^3, \tanh(X)\}$. Then, we test whether $X \perp\!\!\!\perp Y | Z$. We consider two conditioning dimensions $d_z \in \{10, 50\}$, sample size $n = 1000$, and scale $s \in \{0.50, 0.75, 1.00\}$.

Metrics. For each configuration, we run independent tests 100 times and report the empirical type-I error and power when the significance level $\alpha = 0.05$. In each trial, we split samples into a training set and a testing set with ratio $0.5/0.5$. Implementation details and hyperparameter settings for FMCIT are provided in Appendix C.1.

Baselines. We compare FMCIT with several existing conditional independence testing methods, including NNLSCIT Li et al. (2024), which combines k-nearest neighbors with the CRT framework; KCIT Zhang et al. (2011) and its accelerated variant EKCI Guan & Kuang (2025), both based on kernel methods; and CDCIT Yang et al. (2025), which integrates diffusion models with the CRT framework.

Results. Table 1 reports the empirical type-I error and power. Across both $d_z = 10$ and $d_z = 50$, FMCIT maintains type-I error close to the nominal level ($\alpha = 0.05$) while achieving consistently high power. KCI and EKCI perform poorly in controlling the type-I error. As the scale s increases, the injected noise terms $s\varepsilon_x$ and $s\varepsilon_y$ become relatively stronger, effectively reducing the problem to an independence test. In this regime, all other CITs exhibit a substantial loss in power, whereas FMCIT remains robust. This is because the other CITs only learn the conditional distribution of X given Z, while FMCIT learns the joint distribution of (X, Y, Z), which effectively prevents performance degradation when the conditioning information becomes weak and the conditional independence test degenerates into an independence test.

3.2 Causal discovery on high-dimensional post-nonlinear SEMs

Data generating process. We evaluate skeleton learning on random post-nonlinear structural equation models (SEMs) with heavy-tailed noise. We first sample a random DAG over p nodes

Table 2: Skeleton discovery on post-nonlinear SEMs with heavy-tailed noise ($n = 500$, $p = 200$, $\texttt{df} = 3$, $\texttt{exp_edges} = 300$, $\texttt{mcs} = 2$). We report mean $\pm$ std over 5 runs (seeds 0–4).

Method	P ↑	R ↑	F1 ↑	SHD ↓	Runtime(s) ↓
FisherZ-PC	0.506 ± 0.036	0.688 ± 0.039	0.583 ± 0.036	295.6 ± 28.6	3.9 ± 0.6
EPC-ECIT	0.956 ± 0.021	0.431 ± 0.023	0.594 ± 0.024	176.6 ± 8.2	27.7 ± 1.6
GPC-FMCIT (ours)	0.604 ± 0.025	0.641 ± 0.032	0.621 ± 0.023	234.2 ± 14.1	110.6 ± 1.5

by selecting $\texttt{exp_edges}$ directed edges consistent with a fixed topological order (i.e., edges only from lower-index to higher-index nodes). Each variable is then generated sequentially as

$$X_j = g_j\left(\sum_{i \in \mathrm{Pa}(j)} w_{ij} f_{ij}(X_i) + \varepsilon_j\right),$$

where $\mathrm{Pa}(j)$ denotes the parent set, and each edge weight is sampled as $w_{ij} = s_{ij}\, u_{ij}$ with $u_{ij} \sim \mathrm{Unif}(0.7, 1.3)$ and $s_{ij} \in \{+1, -1\}$ uniformly at random. For each parent-child pair, f_{ij} is randomly chosen from a set of nonlinear functions (e.g., $\tanh(\cdot)$, $\sin(\cdot)$, quadratic, signed square-root, leaky-ReLU), and each g_j is randomly chosen from another set (e.g., identity, $\tanh(\cdot)$, clipped $\sinh(\cdot)$, signed $\log(1 + |\cdot|)$). The noise term ε_j follows a Student-t distribution with degrees of freedom $\texttt{df}$ (heavy-tailed when $\texttt{df}$ is small). We report results on the undirected skeleton of the ground-truth DAG.

Experimental setting. Unless stated otherwise, we set $n = 500$, $p = 200$, $\texttt{exp_edges} = 300$, and $\texttt{df} = 3$, and use maximum conditioning set size $\texttt{mcs} = 2$ for skeleton learning. We report mean $\pm$ standard deviation over 5 runs with seeds $\{0, 1, 2, 3, 4\}$. See Appendix C.2 for full experimental settings and evaluation details.

Methods. We compare the following approaches:

1. **FisherZ-PC:** PC-stable skeleton learning with Fisher's Z test. We tune the significance level on a small grid and fix it to $\alpha = 0.03$ as the best-performing setting.
2. **EPC-ECIT:** the ensemble PC method (EPC) with an ensemble conditional independence test (ECIT) Guan & Kuang (2025) , using FisherZ as the base CI test and aggregating p-values across ensemble members (we use the default EPC settings with $k = 3$).
3. **GPC-FMCIT (ours):** our guided PC skeleton learning using *FMCIT* as the CI oracle. We first obtain a screening graph using FisherZ-PC with $\alpha_{\mathrm{scr}} = 0.15$, and then refine the skeleton by running a guided deletion stage driven by FMCIT with significance $\alpha_{\mathrm{fm}} = 0.07$. The guidance uses the stability rule with $\texttt{stab_alphas} = (0.08, 0.10, 0.12)$, $(\texttt{stab_low}, \texttt{stab_high}) = (0.35, 0.65)$, conditioning pool size $k = 10$, and at most $M = 2$ candidate conditioning sets per edge per level.

FMCIT configuration. We train a single flow-matching model on the full p-dimensional joint data once and reuse it across all CI tests. Conditional samples are generated with flow-matching sampling using $\texttt{sampling_steps} = 15$ and a train/test split ratio $\texttt{train_ratio} = 0.6$. We use adaptive CRT repeats $\texttt{crt_repeat}_{\ell=0} = 15$ and $\texttt{crt_repeat}_{\ell\geq 1} = 20$ (capped at 200). Following a hybrid choice of dependence measures, we use RDC at level $\ell = 0$ and correlation at levels $\ell \geq 1$.

Results. Table 2 summarizes the results. Compared to FisherZ-PC, **GPC-FMCIT** improves F1 while substantially reducing SHD, indicating improved robustness beyond linear-Gaussian CI assumptions. Compared to EPC-ECIT, **GPC-FMCIT** achieves a more balanced precision–recall trade-off, yielding higher F1 under this high-dimensional heavy-tailed post-nonlinear setting.

4 Real data analysis

We further evaluate conditional independence tests on the real-world flow-cytometry dataset introduced by Sachs et al. (2005), which contains 11 signaling proteins measured under multiple experimental conditions. Following standard practice in the literature, we consider a fixed set of 90

Table 3: Real-data CIT benchmark on the Sachs flow-cytometry dataset (90 fixed tasks, 10 rounds; mean±std; positive=dependent). Best settings (chosen by highest mean F1 across the swept hyperparameters): FisherZ α=0.02; RCIT α=0.005; FastKCI α=0.01; FMCIT α=0.10, B=200, RDC(k=10, s=1/3, n=1), sampling steps = 20.

Method	Precision	Recall	F1	Runtime (s)
FisherZ	0.7222 ± 0.0000	0.6500 ± 0.0000	0.6842 ± 0.0000	0.0369 ± 0.0008
RCIT	0.7695 ± 0.0335	0.5975 ± 0.0184	0.6724 ± 0.0204	2.0312 ± 0.3698
FastKCI	0.7247 ± 0.0337	0.6400 ± 0.0337	0.6789 ± 0.0226	166.2569 ± 2.5656
FMCIT (Ours)	0.7826 ± 0.0245	0.7875 ± 0.0270	$\mathbf{0.7845 \pm 0.0151}$	29.0012 ± 0.0196

conditional independence (CI) queries, consisting of 50 CI relations and 40 non-CI relations curated based on domain knowledge.

All methods are evaluated on the same set of CI queries across 10 independent rounds, and performance is reported in terms of precision, recall, and F1-score with respect to detecting dependence (i.e., rejecting conditional independence). Runtime is measured as the wall-clock time required to evaluate all CI queries in a single round.

Results. Table 3 reports the performance of different CI tests under their respective best hyperparameter settings. Classical tests such as FisherZ are computationally efficient but exhibit limited recall, reflecting their sensitivity to linearity and Gaussian assumptions. Kernel-based tests (RCIT and FastKCI) improve robustness to nonlinear dependencies, at the cost of substantially increased computational overhead, particularly for FastKCI.

FMCIT achieves the highest F1-score among all methods while maintaining a favorable precision–recall balance. Notably, FMCIT attains strong recall without sacrificing precision, indicating its ability to reliably detect conditional dependencies in complex real-world settings. In terms of runtime, FMCIT is significantly faster than kernel-based alternatives, while remaining more expressive than parametric tests.

Discussion. These results suggest that FMCIT provides an effective trade-off between statistical power and computational efficiency on real biological data. Importantly, FMCIT does not rely on explicit parametric assumptions about the data-generating process, making it well-suited for heterogeneous and nonlinear systems such as cellular signaling networks.

5 Conclusion

In this paper, we propose a fast conditional independence test, termed Flow Matching based Conditional Independence Test (FMCIT). FMCIT is specifically designed for accelerating constraint-based causal discovery methods. FMCIT employs flow matching to learn the joint distribution of the data and reformulates conditional sampling as an imputation task over the entire dataset given the conditioning variables. Throughout the causal discovery process, FMCIT requires training the model only once and enables fast sample generation, thereby substantially accelerating causal discovery. Extensive simulations and real data analyses demonstrate the strong empirical performance of FMCIT. In particular, on synthetic benchmarks, causal discovery methods performed by FMCIT achieve both favorable computational efficiency and competitive accuracy. In the future, we aim to establish theoretical guarantees for the type-I error control of FMCIT and to further improve its performance by incorporating double-robust techniques.

References

Alexis Bellot and Mihaela van der Schaar. Conditional independence testing using generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Emmanuel Candès, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold: 'model-x' knockoffs for high dimensional controlled variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(3):551–577, 2018.

Diego Colombo, Marloes H Maathuis, et al. Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

Ben Dai, Xiaotong Shen, and Wei Pan. Significance tests of feature relevance for a black-box learner. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. Kernel measures of conditional dependence. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, March 2012. ISSN 1532-4435.

Zhengkang Guan and Kun Kuang. Efficient ensemble conditional independence test framework for causal discovery. *arXiv:2509.21021*, 2025.

Philip Hartman. *Ordinary differential equations*. SIAM, 2002.

Chenxuan He, Yuan Gao, Liping Zhu, and Jian Huang. From conditional to unconditional independence: Testing conditional independence via transport maps. *arXiv:2504.09567*, 2025a.

Zheng He, Roman Pogodin, Yazhe Li, Namrata Deka, Arthur Gretton, and Danica J. Sutherland. On the hardness of conditional independence testing in practice. In *Neural Information Processing Systems*, 2025b.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Yang Hu, Xiao Wang, Zezhen Ding, Lirong Wu, Huatian Zhang, Stan Z. Li, Sheng Wang, Jiheng Zhang, Ziyun Li, and Tianlong Chen. FlowTS: Time series generation via rectified flow. *arXiv 2411.07506*, 2025.

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

Shuai Li, Ziqi Chen, Hongtu Zhu, Christina Wang, and Wang Wen. Nearest-neighbor sampling based conditional independence testing. In *AAAI Conference on Artificial Intelligence*, volume 37, pp. 8631–8639, 06 2023.

Shuai Li, Yingjie Zhang, Hongtu Zhu, Christina Wang, Hai Shu, Ziqi Chen, et al. K-nearest-neighbor local sampling based conditional independence testing. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

David Lopez-Paz, Philipp Hennig, and Bernhard Schölkopf. The randomized dependence coefficient. In *Neural Information Processing Systems*, pp. 1–9, 2013.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Computer Vision and Pattern Recognition*, pp. 11451–11461, 2022.

Christopher Meek. Causal inference and causal explanation with background knowledge. In *Uncertainty in Artificial Intelligence*, pp. 403–410, 1995.

Ryunosuke Miyazaki and Yoshimasa Uematsu. Testing conditional independence via the spectral generalized covariance measure: Beyond euclidean data. *arXiv:2511.15453*, 2025.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

Joseph Ramsey. Improving accuracy and scalability of the pc algorithm by maximizing p-value. *arXiv:1610.00378*, 2016.

Joseph Ramsey, Peter Spirtes, and Jiji Zhang. Adjacency-faithfulness and conservative causal inference. In *Uncertainty in Artificial Intelligence*, pp. 401–408, 2006.

Yixin Ren, Chenghou Jin, Yewei Xia, Li Ke, Longtao Huang, Hui Xue, Hao Zhang, Jihong Guan, and Shuigeng Zhou. Score-based generative modeling for conditional independence testing. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2410–2419, 2025.

Jakob Runge. Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information. In *International Conference on Artificial Intelligence and Statistics*, pp. 938–947, 2018.

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.

Meyer Scetbon, Laurent Meunier, and Yaniv Romano. An asymptotic test for conditional independence using analytic kernel embeddings. In *International Conference on Machine Learning*, pp. 19328–19346, 2022.

Rajen D. Shah and Jonas Peters. The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48(3):1514 – 1538, 2020.

Chengchun Shi, Tianlin Xu, Wicher Bergsma, and Lexin Li. Double generative adversarial networks for conditional independence testing. *Journal of Machine Learning Research*, 22(285):1–32, 2021.

Kirankumar Shiragur, Jiaqi Zhang, and Caroline Uhler. Causal discovery with fewer conditional independence tests. In *International Conference on Machine Learning*, volume 235, pp. 45060–45078, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Peter Spirtes, Christopher Meek, and Thomas Richardson. Causal inference in the presence of latent variables and selection bias. In *Uncertainty in Artificial Intelligence*, pp. 499–506, 1995.

Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT Press, 2000.

Xueqin Wang, Wenliang Pan, Wenhao Hu, Yuan Tian, and Heping Zhang. Conditional distance correlation. *Journal of the American Statistical Association*, 110(512):1726–1734, 2015.

Yanfeng Yang, Shuai Li, Yingjie Zhang, Zhuoran Sun, Hai Shu, Ziqi Chen, and Renming Zhang. Conditional diffusion models based conditional independence testing. In *AAAI Conference on Artificial Intelligence*, 2025.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Conference on Uncertainty in Artificial Intelligence*, pp. 804–813, 2011.

A PRELIMINARIES

In this section, we introduce the framework of CRT, PC algorithm, and its variants.

A.1 CONDITIONAL RANDOMIZATION TEST (CRT)

Note that the definition of CI in (1) is equivalent to $P_{X_i|X_j,X_S} = P_{X_i|X_S}$, where $P_{X_i|X_j,X_S}$ denotes the conditional distribution of X_i given (X_j, X_S). According to Bayes' rule, CI is also equivalent to $P_{X_i,X_j,X_S} = P_{X_i|X_S} P_{X_j,X_S}$, where P_{X_i,X_j,X_S} and P_{X_j,X_S} are the joint distributions of (X_i, X_j, X_S) and (X_j, X_S), respectively. Suppose we have n i.i.d. samples from the distribution P_X of X, denoted by $\mathbf{X} \in \mathbb{R}^{n\times p}$. Candès et al. (2018) reformulates the CI testing problem as a two-sample testing problem Gretton et al. (2012). Specifically, it compares the original samples $(\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_S)$ with the randomized samples $(\mathbf{X}_i^{(b)}, \mathbf{X}_j, \mathbf{X}_S)$ for $b = 1, \ldots, B$. Here, $\mathbf{X}_i$, $\mathbf{X}_j$, and $\mathbf{X}_S$ denote samples drawn from X_i, X_j, and X_S, respectively, while $\mathbf{X}_i^{(b)}$ are independent samples drawn from the conditional distribution $P_{X_i|X_S=\mathbf{X}_S}$. Let $T : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{n\times|S|} \to \mathbb{R}$ be a test statistic that measures the conditional dependence between the n samples of X_i and X_j given X_S, where larger values of T indicate stronger conditional dependence. When the null hypothesis H_0 holds, that is, when X_i and X_j are conditionally independent given X_S, the value of $T(\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_S)$ should be comparable to that of $T(\mathbf{X}_i^{(b)}, \mathbf{X}_j, \mathbf{X}_S)$. In contrast, when the alternative hypothesis H_1 holds, i.e., when X_i and X_j are conditionally dependent given X_S, the value of $T(\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_S)$ should be significantly larger than that of $T(\mathbf{X}_i^{(b)}, \mathbf{X}_j, \mathbf{X}_S)$, since $\mathbf{X}_i^{(b)}$ are drawn from the conditionally independent distribution $P_{X_i|X_S=\mathbf{X}_S}$. Finally, the p-value of the CRT is given by

$$\frac{1 + \sum_{b=1}^{B} \mathbf{I}\{T(\mathbf{X}_i^{(b)}, \mathbf{X}_j, \mathbf{X}_S) \geq T(\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_S)\}}{1 + B}, \tag{9}$$

where $\mathbf{I}\{\cdot\}$ is the indicator function. In our CI testing procedure, we adopt the Randomized Dependence Coefficient (RDC) as the test statistics T Lopez-Paz et al. (2013).

A.2 PC ALGORITHM AND ITS VARIANTS

PC algorithm Spirtes et al. (2000) is a constraint-based causal discovery method that estimates a Markov equivalence class of DAGs, represented as a CPDAG, from CI relations inferred from data.

Assumptions. PC algorithm is typically presented under the *Causal Markov* and *Causal Faithfulness* assumptions with respect to an underlying DAG, together with i.i.d. sampling. In addition, the vanilla PC algorithm assumes *causal sufficiency* (no latent confounders); extensions such as FCI relax this assumption (see "Variants" below).

Skeleton learning (edge deletion). PC algorithm starts from the complete undirected graph $G^{(0)}$ over $[p]$ and iteratively removes edges using conditional independence tests. Let $\mathrm{Adj}_G(i)$ denote the current adjacency set of node i in an undirected graph G. At conditioning level $\ell = 0, 1, 2, \ldots$, for each adjacent pair (i, j) in the current graph, PC searches for a separating set

$$S \subseteq \mathrm{Adj}_G(i) \setminus \{j\}, \qquad |S| = \ell, \tag{10}$$

such that the CI test accepts

$$X_i \perp\!\!\!\perp X_j | X_S. \tag{11}$$

If such an S is found, the edge $i - j$ is removed and the separating set is stored as $\mathrm{Sep}(i, j) \leftarrow S$ (and symmetrically $\mathrm{Sep}(j, i) \leftarrow S$). The algorithm increases ℓ until a predefined maximum conditioning size is reached or no further edges can be removed. Since the search over conditioning sets is combinatorial, the skeleton stage is typically the computational bottleneck in practice, especially in high dimensions Kalisch & Bühlman (2007).

Orientation (v-structures and propagation). After skeleton learning, PC orients edges in two steps. First, for every unshielded triple $i - k - j$ (i.e., i adjacent to k, k adjacent to j, but i and j non-adjacent), PC orients a v-structure

$$i \to k \leftarrow j$$

whenever $k \notin \mathrm{Sep}(i, j)$. Second, PC applies sound orientation propagation rules (e.g., Meek rules) to orient as many remaining edges as possible without introducing directed cycles or new v-structures.

Order-independence and variants. A well-known practical issue is that the adjacency search in skeleton learning can depend on the order in which CI tests are performed. The PC-stable variant Colombo et al. (2014) addresses this by using an order-independent deletion scheme (often referred to as a "stable" PC skeleton stage). Other variants include conservative PC (CPC) Ramsey et al. (2006) and PC-Max Ramsey (2016), which modify v-structure orientation rules to reduce false orientations, and FCI Spirtes et al. (1995), which extends the framework to settings with latent confounding and selection bias. In this work, we focus on accelerating the skeleton-learning stage, and standard orientation procedures can be applied afterwards using the recorded separating sets Meek (1995).

B ALGORITHM OF FMCIT

In this section, we introduce the algorithm of training a FM model, sampling from a trained FM model, and the whole algorithm of FMCIT.

Algorithm 1 Training a flow matching model

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$.
Ensure: A trained neural network v_θ.
1: Initialize a neural network v_θ
2: **while** Not converge **do**
3: Draw $t \sim U(0, 1)$
4: Draw $\mathbf{X}(0) \sim N(0, I)$
5: Calculate $L_{FM} = \|\mathbf{X} - \mathbf{X}(0) - v_\theta(t\mathbf{X} + (1-t)\mathbf{X}(0), t)\|^2$
6: Calculate $\nabla_\theta L_{FM}$ and update the parameters of v_θ
7: **end while**
8: **Output** v_θ

Algorithm 2 Sampling from a trained FM model, FM-IMPUTE($\mathbf{X}_S$)

Require: Samples of the conditioning set $\mathbf{X}_S$, trained vector field v_θ, sampling schedule $0 = t_0, t_1, \ldots, t_L = 1$.
Ensure: Samples approximate $P_{X|X_S=\mathbf{x}_S}$.
1: Draw $\mathbf{X}(0) \sim N(0, I)$
2: **for** $i = 0, \ldots L-1$ **do**
3: Calculate $\widehat{\mathbf{X}}(t_i) = X(t_i) + (1 - t_i)v_\theta(X(t_i), t_i)$
4: Replace the corresponding components of $\widehat{\mathbf{X}}(t_i)$ with $\mathbf{X}_S$.
5: **if** $i \neq L - 1$ **then**
6: Draw $\varepsilon_{t_{i+1}} \sim N(0, I)$
7: $\mathbf{X}(t_{i+1}) = t_{i+1}\widehat{\mathbf{X}}(t_i) + (1 - t_{i+1})\varepsilon_{t_{i+1}}$
8: **end if**
9: **end for**
10: **Output** $\widehat{\mathbf{X}}(t_{L-1})$.

Algorithm 3 Flow matching based conditional independence test, FMCIT$(\mathbf{X}_i \perp\!\!\!\perp \mathbf{X}_j | \mathbf{X}_S)$

Require: Samples of whole dataset $\mathbf{X}$, variable index (i, j), conditioning set S and conditioning samples $\mathbf{X}_S$, CRT sampling times B, test statistics $T(\cdot, \cdot, \cdot)$, significance level α.

Ensure: CI or non-CI.

Repeat $\mathbf{X}_S$ for B times, and let the repeated sample be $\mathbf{X}_S^{(1:B)} \in \mathbb{R}^{nB \times |S|}$

Generate $\mathbf{X}^{(1:B)}$ = FM($\mathbf{X}_S^{(1:B)}$)

Separate $\mathbf{X}^{(1:B)}$ into B samples $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(B)}$

Let the i-th and j-th variable in $\mathbf{X}$ be $\mathbf{X}_i$ and $\mathbf{X}_j$

$T = T(\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_S)$

for $b = 1, \ldots B$ **do**

  Let the i-th variable in $\mathbf{X}^{(b)}$ be $\mathbf{X}_i^{(b)}$

  Calculate $T^{(b)} = T(\mathbf{X}_i^{(b)}, \mathbf{X}_j, \mathbf{X}_S)$

end for

Calculate $p = \frac{1+\sum_{b=1}^{B} \mathbf{I}\{T^{(b)} \geq T\}}{1+B}$

Return p

Algorithm 4 GPC-FMCIT (Guided PC skeleton learning with FMCIT)

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ with variables $(X_1, \ldots, X_p)$. Screening level α_{scr} and screening max conditioning size $d_{\max}^{\mathrm{scr}}$. Main max conditioning size $d_{\max}$ and main test level α_{fm}. FMCIT parameters: FM sampling steps L, dependence statistic $T(\cdot)$, and adaptive CRT repeats $B(\ell)$. Guided conditioning pool parameters: pool size k, budget M (max # conditioning sets per edge per level), a guidance rule `stab` (stability-based) or `tri` (triangle-based), and random seed s.

Ensure: Estimated undirected skeleton $\widehat{G}$ and separating sets $\mathrm{Sep}(\cdot, \cdot)$.

Stage I: Screening skeleton.
Compute screening skeleton $G_{\mathrm{scr}} \leftarrow$ PC-STABLE$(\mathbf{X}; \alpha_{\mathrm{scr}}, d_{\max}^{\mathrm{scr}})$ using FisherZ.
Initialize working graph $G \leftarrow G_{\mathrm{scr}}$ and $\mathrm{Sep} \leftarrow \emptyset$.
Stage II: Guided conditioning pool construction.
Define screening neighborhoods $\mathrm{Adj}_{G_{\mathrm{scr}}}(i)$ for all $i \in [p]$.
if `rule` = `stab` **then**
  For each $\alpha \in \mathcal{A}$, run PC-STABLE$(\mathbf{X}; \alpha, d_{\max}^{\mathrm{scr}})$ and compute edge frequencies freq_{ij}.
  Build a stability graph and extract connected components $\{C_1, \ldots, C_m\}$; compute a global node order π.
else if `rule` = `tri` **then**
  Build a triangle-supported graph from G_{scr} and extract components $\{C_1, \ldots, C_m\}$; compute a global node order π.
else
  Set a single component $C_1 = [p]$ and define an arbitrary global order π.
end if
Let $\mathrm{cid}(i)$ denote the component index of node i.
Stage III: Guided skeleton refinement with FMCIT.
for $\ell = 0, 1, \ldots, d_{\max}$ **do**
  Initialize removal list $\mathcal{R} \leftarrow \emptyset$.
  for each current edge (i, j) in G **do**
    Form raw candidates $\mathcal{Q}_{ij} \leftarrow \big(\mathrm{Adj}_{G_{\mathrm{scr}}}(i) \cup \mathrm{Adj}_{G_{\mathrm{scr}}}(j)\big) \setminus \{i, j\}$.
    Construct the guided conditioning pool $\mathcal{P}_{ij} \subseteq [p] \setminus \{i, j\}$ with $|\mathcal{P}_{ij}| = k$ by selecting from $\mathcal{Q}_{ij}$ (optionally prioritizing nodes with $\mathrm{cid}(\cdot) \notin \{\mathrm{cid}(i), \mathrm{cid}(j)\}$) and completing using the global order π if needed.
    Construct budgeted conditioning sets $\mathcal{S}_{ij}^{(\ell)} \subseteq \binom{\mathcal{P}_{ij}}{\ell}$ with $|\mathcal{S}_{ij}^{(\ell)}| \leq M$ (enumerate if $\binom{k}{\ell} \leq M$, else sample M unique sets with seed s).
    for each $S \in \mathcal{S}_{ij}^{(\ell)}$ **do**
      Compute $p_{ij|S} \leftarrow$ FMCIT$(X_i \perp\!\!\!\perp X_j | X_S;\ B(\ell), L, T)$.
      if $p_{ij|S} > \alpha_{\mathrm{fm}}$ **then**
        Add (i, j, S) to $\mathcal{R}$ and **break**.
      end if
    end for
  end for
  for each $(i, j, S) \in \mathcal{R}$ **do**
    Remove edge (i, j) from G.
    Set $\mathrm{Sep}(i, j) \leftarrow S$ and $\mathrm{Sep}(j, i) \leftarrow S$.
  end for
end for
Output. Return $\widehat{G} \leftarrow G$ and Sep.

C Experimental settings

C.1 Synthetic conditional independence benchmark: implementation details

Preprocessing. Unless otherwise stated, each feature is standardized to zero mean and unit variance using the training split statistics, and the same transformation is applied to the test split.

Flow-matching model. For FMCIT, the vector field v_θ is parameterized by a multilayer perceptron (MLP) with two hidden layers, each of width 256. We train v_θ for 200 epochs using Adam with learning rate 10^{-3} and batch size 1024. Model selection is done by the smallest training loss.

Conditional sampler. Conditional resamples are generated by the Picard–RePaint flow-matching imputation sampler with a uniform time grid $0 = t_0 < t_1 < \cdots < t_L = 1$ and $L = 50$ sampling steps.

CRT configuration and test statistic. Each CI query is evaluated using CRT with $B = 100$ randomizations and the p-value is computed as in Eq. (9). We use the randomized dependence coefficient (RDC) as the dependence statistic inside CRT. Unless otherwise stated, we use RDC hyperparameters $k = 10$, $s = 1/3$, and $n = 1$ (and fix the projection seed for reproducibility).

C.2 Causal discovery experiments

Experimental setting. Unless stated otherwise, we set the sample size to $n = 500$ and the number of variables to $p = 200$. The ground-truth graph is generated to have $\texttt{exp_edges} = 300$ edges, and the noise distribution is Student-t with $\texttt{df} = 3$ (heavy-tailed). For all skeleton learning methods, we use the maximum conditioning set size $\texttt{mcs} = 2$.

We run 5 independent repetitions with random seeds $\{0, 1, 2, 3, 4\}$. For each seed, we generate a fresh dataset (including a new random graph and new i.i.d. samples) and run each method once. We report mean $\pm$ standard deviation over the 5 runs. All evaluations are performed on the undirected skeleton: an edge is counted as correctly recovered if it appears in the ground-truth skeleton regardless of direction. We report Precision, Recall, F1 score, structural Hamming distance (SHD) on the undirected skeleton, and wall-clock runtime in seconds. All experiments were conducted on one NVIDIA H100 GPU.