EFFICIENT BAYESIAN NETWORK STRUCTURE LEARNING WITH TSETLIN MACHINE

Anonymous authors
Paper under double-blind review

ABSTRACT

Constraint-based Bayesian network structure learning relies on conditional independence (CI) testing. However, as the number of variables increases, the number of possible conditioning sets grows combinatorially, thereby making CI testing computationally expensive and rendering it difficult to scale up to large networks. We propose Weighted Tsetlin Machine-Bayesian Network learning (WTM-BN), a method that makes use of a WTM to reduce the search space. For each target variable, a WTM is trained, and its clauses are utilised to score the relevance of every other variable. A relative-drop cutoff is thereafter applied so as to select a reduced candidate set for CI testing. The resulting skeleton is oriented with the help of collider detection and Meek's rules, so as to obtain a Completed Partially Directed Acyclic Graph (CPDAG). Across five Bayesian Networks (BNs) from the *bnlearn* benchmark and eight real-world datasets, WTM-BN matches or comes close to the accuracy of state-of-the-art methods, while achieving substantial reduction in runtime. These results demonstrate that WTM-BN offers a favourable accuracy–runtime trade-off for constraint-based BN structure learning.

1 INTRODUCTION

A Bayesian network (BN) is a probabilistic graphical model that uses a directed acyclic graph (DAG) to represent random variables and their conditional dependencies (Polotskaya et al., 2024). Each node is a variable and each edge is a direct dependency, which enables the factorization of the joint distribution into local conditional distributions. BNs are widely used in fields like biology, healthcare, engineering fault diagnosis and air pollution modeling (Kitson et al., 2023). However, their performance largely depends on the learned graph structure from the data.

BN structure learning is broadly divided into constraint-based and score-based approaches (Kitson et al., 2023). Constraint-based methods infer edges using conditional independence (CI) tests (Colombo & Maathuis, 2014), while score-based methods search the space of DAGs to optimize a scoring function such as BIC or BDeu. The number of DAGs grows super-exponentially with the number of variables . It is known that learning the optimal structure is NP-hard (Le et al., 2019), which motivates the development of efficient and scalable learning methods .

Optimisation-based methods. One line of work reframes DAG structure learning as a continuous optimisation problem. (Zheng et al., 2018) introduced the first fully differentiable formulation making use of a matrix-exponential acyclicity constraint, and DAGs via M-matrices for Acyclicity (DAGMA) (Bello et al., 2022) thereafter replaced this with a log-determinant characterisation over M-matrices along with a central-path optimisation scheme, thereby yielding better gradients and substantial speedups.

Order-based methods. A second line of work searches over variable orderings rather than graphs directly (Teyssier & Koller, 2005; Solus et al., 2021; Lam et al., 2022). Among these, Best Order Score Search (BOSS) (Andrews et al., 2023) makes use of grow-shrink trees so as to efficiently score DAGs from permutations, thereby achieving good accuracy and scalability on large, densely connected graphs, where earlier permutation-based methods had struggled.

Score-based methods. Greedy Equivalence Search (GES) (Chickering, 2002) greedily adds and removes edges so as to maximise a scoring criterion, offering strong guarantees in the infinite-data limit, but often converging to local optima in practice, more so on denser graphs. Extremely Greedy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Equivalent Search (XGES) (Nazaret & Blei, 2024) addresses this by favouring edge deletions earlier in the search, thereby reducing the chances of getting stuck in local optima, while preserving GES's theoretical guarantees and achieving considerable speedups.

Constraint-based methods. Recent constraint-based methods bound the conditioning set size (Wienöbst & Liskiewicz, 2020; Kocaoglu, 2023), but they still test every CI relation up to that bound. $\mathcal{C}$-PC (Lee et al., 2025) relaxes this by restricting the tests to a specified, conditionally closed set of conditioning sets, thereby defining $\mathcal{C}$-Markov equivalence so as to characterise the resulting graph class.

Despite these efforts, reducing CI tests remains important, particularly for high-dimensional BN structure learning. To address this, a Tsetlin Machine (TM) identifies highly correlated variables, and a relative-drop cutoff selects a reduced candidate set for CI testing. The resulting skeleton is then oriented using collider detection and Meek's rules to obtain a Completed Partially Directed Acyclic Graph (CPDAG).

The rest of the paper is organized as follows. Section 2 covers background on BNs, CI theory. Section 3 reviews the TM. Section 4 provide a detailed explanation of our method. Section 5 analyzes the computational complexity of the method. Section 6 reports evaluations and comparisons with state-of-the-art methods. Section 7 concludes.

2 Conceptual background

We formalise the key definitions, assumptions, and theorems underlying our constraint-based framework, following standard BN and CI theory.

2.1 Notations, definitions, and formalization

Definition 1 (Bayesian Network). Let U denote a finite set of random variables. A BN is defined as a triplet $\langle U, G, P\rangle$, where $G = (U, E)$ is a DAG over U, with directed edges $E \subseteq U \times U$, and P is a joint distribution over U such that

$$P(U) = \prod_{X \in U} P(X \mid \text{Pa}(X)), \tag{1}$$

where $\text{Pa}(X)$ denotes the set of parents of X in G.

Definition 2 (Conditional independence). Two variables $X, Y \in U$ are conditionally independent given a set $Z \subseteq U \setminus \{X, Y\}$, denoted as $X \perp\!\!\!\perp Y \mid Z$, if and only if

$$P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z). \tag{2}$$

Definition 3 (D-separation). A path between X and Y is blocked by $Z \subseteq U \setminus \{X, Y\}$ if it contains a non-collider in Z or a collider such that neither it nor its descendants are in Z. Variables X and Y are d-separated by Z in G (denoted $X \perp\!\!\!\perp_G Y \mid Z$) when all paths between them are blocked by Z.

Definition 4 (Causal Markov Condition). A BN $\langle U, G, P\rangle$ satisfies the Causal Markov Condition if every variable $X \in U$ is conditionally independent of its non-descendants in G given its parents $\text{Pa}(X)$. Consequently, d-separation in G implies conditional independence in P:

$$X \perp\!\!\!\perp_G Y \mid Z \implies X \perp\!\!\!\perp Y \mid Z. \tag{3}$$

Definition 5 (Faithfulness). A BN $\langle U, G, P\rangle$ is faithful if all conditional independencies in P correspond to d-separations in G:

$$X \perp\!\!\!\perp Y \mid Z \implies X \perp\!\!\!\perp_G Y \mid Z. \tag{4}$$

Under the Causal Markov Condition and Faithfulness, conditional independence in P and d-separation in G are logically equivalent ($X \perp\!\!\!\perp_G Y \mid Z \iff X \perp\!\!\!\perp Y \mid Z$).

Definition 6 (Parents and Children set). For a variable $X \in U$, the Parents and Children set, denoted $\text{PC}(X)$, consists of all variables adjacent to X in G. Under the Causal Markov Condition

and Faithfulness, $Y \in \text{PC}(X)$ if and only if there exists no conditioning set $Z \subseteq U \setminus \{X, Y\}$ such that $X \perp\!\!\!\perp Y \mid Z$.

Definition 7 (V-structure and Unshielded Triples). An unshielded triple is a sequence of three nodes (A, M, B) such that A and M are adjacent, M and B are adjacent, but A and B are non-adjacent. An unshielded triple forms a v-structure (or unshielded collider) if the edges are oriented as $A \to M \leftarrow B$.

Definition 8 (Markov Equivalence and CPDAG). Two DAGs G_1 and G_2 over U belong to the same Markov Equivalence Class (MEC) if and only if they have the same underlying undirected skeleton and the same set of v-structures. A CPDAG represents an MEC; an edge in a CPDAG is directed if it retains the same orientation across all DAGs in the MEC, and undirected otherwise.

Definition 9 (Causal Sufficiency). A set of variables U is causally sufficient if no unobserved variable is a common cause of any pair of variables in U.

3 Overview of the Tsetlin machine

TMs are ensembles of Tsetlin Automata (TA) that perform propositional logic-based pattern recognition, learning interpretable clauses through game-theoretic feedback (Granmo, 2021). We summarise the operational concept below.

3.1 Tsetlin machine concept

For a Boolean input $\mathbf{x} = [x_1, \ldots, x_o] \in \{0,1\}^o$, the TM forms a literal set $L = \{l_1, \ldots, l_{2o}\}$ containing all features and their negations. Each literal has a TA with $2N$ states (N Exclude and N Include), and the TA's state decides whether the literal is included in a clause. A clause C_i takes the AND of its included literals and outputs 1 only when all of them are true. Each class has m clauses, split equally into positive clauses (C_i^+), which vote for the class, and negative clauses (C_i^-), which vote against it. The total vote $s(\mathbf{x})$ for an input is:

$$s(\mathbf{x}) = \sum_{i=1}^{m/2} C_i^+(\mathbf{x}) - \sum_{i=1}^{m/2} C_i^-(\mathbf{x}). \tag{5}$$

For binary classification, the prediction is given by $\hat{y} = u(s(\mathbf{x}))$, where $u(v) = 1$ if $v \geq 0$, else 0. For multi-class problems, each class k maintains its own clause set and its class sum $s_k(\mathbf{x})$. The predicted class is the one with the highest vote

$$\hat{y} = \arg\max_k \big(s_k(\mathbf{x})\big). \tag{6}$$

3.2 Tsetlin machine learning

During training, the TM uses Type I (Ia and Ib) and Type II feedback to update each literal's TA, based on the class sum and the hyperparameters S (specificity) and T (threshold). The type of feedback depends on the clause polarity and class label. Positive-polarity clauses C_i^+ receive Type I feedback when $y = 1$ and Type II feedback when $y = 0$; negative-polarity clauses C_i^- receive Type I feedback when $y = 0$ and Type II feedback when $y = 1$.

Type Ia feedback: This feedback reinforces correctly firing clauses by strengthening the patterns they match. It is applied to active positive-polarity clauses when $y = 1$ and active negative-polarity clauses when $y = 0$. For each literal l_t, if $l_t = 1$, the TA state is incremented towards Include with probability 1; if $l_t = 0$, it is decremented towards Exclude with probability $\frac{1}{S}$.

Type Ib feedback: This feedback combats false negatives and is applied to inactive (output 0) positive-polarity clauses when $y = 1$ and inactive negative-polarity clauses when $y = 0$. Each literal's TA is decremented by one state towards Exclude with probability $\frac{1}{S}$, weakening the clause so that it can learn a different pattern.

Type II feedback: This feedback suppresses false positives by correcting incorrectly firing clauses. For $l_t = 0$, the Exclude action is penalised with probability 1, moving the TA towards Include and forcing literals that can make the clause output 0.

3.2.1 THRESHOLD-BASED ACTIVATION

The feedback is applied only to a randomly chosen clause subset, controlled by threshold T. First clamp the sum:

$$c(\mathbf{x}) = \text{clamp}(s(\mathbf{x}), -T, T). \tag{7}$$

Then each clause receives feedback with probability:

$$p_i(\mathbf{x}) = \begin{cases} \dfrac{T + c(\mathbf{x})}{2T}, & y = 0, \\ \dfrac{T - c(\mathbf{x})}{2T}, & y = 1. \end{cases} \tag{8}$$

As $s(\mathbf{x})$ approaches $\pm T$, the update probability $p_i(\mathbf{x})$ decreases, slowing down the feedback frequency and eventually stopping. The randomised clause selection preserves diversity across clauses covering different sub-patterns.

3.3 WEIGHTED TSETLIN MACHINE

The Weighted Tsetlin machine (WTM), proposed by Abeyrathna et al. (2021), extends the TM by assigning a weight to each clause, resulting in a more compact model. Initially, all clause weights are set to 1. During training, Type Ia feedback increases the weights of correctly predicting clauses, while Type II feedback decreases the weights of clauses contributing to false positives. The final prediction is based on a weighted majority vote:

$$s(\mathbf{x}) = \sum_{i=1}^{m/2} w_i^+ C_i^+(\mathbf{x}) - \sum_{i=1}^{m/2} w_i^- C_i^-(\mathbf{x}). \tag{9}$$

4 PROPOSED METHOD : WTM-BN

This section describes our approach for learning a BN structure. For each target variable $Y \in U$, a WTM identifies the most correlated variables. A relative-drop technique selects a small candidate set, and CI tests on these candidates give the undirected skeleton. Edges are then oriented using collider detection and Meek's rules to obtain the CPDAG.

4.1 TRAINING THE TSETLIN MACHINE AND SCORING VARIABLES

The input is one-hot encoded (OHE), turning each categorical X_j (b_j values) into binary variables $X_{j,1}, \ldots, X_{j,b_j}$. A WTM is trained on this data to predict Y. Since clause literals derive from OHE, each clause C_i is normalised back to its original variables:

$$\tilde{C}_i = \{X_j \mid \exists\, h :\ X_{j,h} \in C_i \text{ or } \neg X_{j,h} \in C_i\}. \tag{10}$$

So each X_j appears at most once in a normalised clause, regardless of how many of its literals the clause contains.

Let D_Y denote the set of unique original variables occurring in any normalised clause associated with Y. For each $X_j \in D_Y$, we count how many normalised clauses it appears in:

$$I(X_j) = \sum_i \mathbf{1}(X_j \in \tilde{C}_i), \tag{11}$$

where $\mathbf{1}(\cdot)$ is the indicator function. A variable that appears in many clauses is repeatedly used by the WTM to recognise patterns of Y, so its frequency reflects its importance for predicting Y.

4.2 RELATIVE-DROP TECHNIQUE FOR VARIABLE SELECTION

Once every $X_j \in D_Y$ has a score $I(X_j)$, we rank them in descending order: $I(X_{(1)}) \geq I(X_{(2)}) \geq \cdots \geq I(X_{(|D_Y|)})$, where $X_{(r)}$ is the variable ranked r-th. In practice, only a small subset carries

substantially higher importance than the rest, while the remaining variables show a low, roughly flat score. We locate the cutoff by measuring, at each rank r, the *relative* drop to the next-ranked score:

$$\Delta(r) = \frac{I(X_{(r+1)}) - I(X_{(r)})}{I(X_{(r)})}, \qquad r = 1, \ldots, |D_Y| - 1. \tag{12}$$

Since scores are sorted in descending order, $\Delta(r)$ is negative; the cutoff is the rank at which this proportional collapse is steepest:

$$r^* = \arg\min_r \Delta(r). \tag{13}$$

All variables ranked at or above r^* are retained, forming the reduced candidate set $D'_Y \subseteq D_Y$. To guard against too few variables being selected, we enforce a minimum candidate count $r_{\min}$, always retaining the top $\max(r^*, r_{\min})$ variables.

4.3 Conditional independence testing

Given the reduced candidate set $D'_Y \subseteq D_Y$ for Y, we perform CI testing to determine which candidates are genuinely connected to Y.

We first perform a zero-order (marginal) test: for each $X_j \in D'_Y$, we test $X_j \perp\!\!\!\perp Y \mid \emptyset$. Independent candidates are removed, with $\emptyset$ recorded as their separating set.

Remaining candidates are then tested in increasing order of $I(X_j)$ – the weakest survivors first, since these are more likely to represent spurious associations mediated by other variables. For each candidate X_j, we test $X_j \perp\!\!\!\perp Y \mid Z$, where Z is built from combinations of the strongest remaining candidates, taken in decreasing order of $I(X_j)$ (Yu et al., 2018). $|Z|$ is increased from 1 up to a predefined maximum conditioning-set size q. Once a separating set Z is found, X_j is removed and Z is recorded as its separating set.

The remaining candidates form $\hat{\text{PC}}(Y)$. Repeating this for every target and adding an edge $Y - X_j$ if $X_j \in \hat{\text{PC}}(Y)$ or $Y \in \hat{\text{PC}}(X_j)$ gives the skeleton $\hat{G}$.

4.4 Collider finding

For every unshielded triple $A - M - B$ in $\hat{G}$ – i.e. M adjacent to both A and B, while A and B are not directly adjacent, we determine whether it forms a collider $A \to M \leftarrow B$.

We first obtain a separating set for the pair (A, B): if one was already recorded during CI testing (Section 4.3), it is reused directly. Since CI testing is target-centred and does not test every pair, most (A, B) pairs at this stage will not have been tested previously; in that case, a separating set is searched for on demand from subsets of A's other neighbours, excluding B and M, up to size q. If none is found, the triple is discarded as an invalid collider candidate.

Given a separating set Z for (A, B): if $M \in Z$, the triple is a chain or fork and is discarded. If $M \notin Z$, we verify the collider with an additional test, $A \perp\!\!\!\perp B \mid Z \cup \{M\}$. The triple is confirmed as an unshielded collider only if this independence breaks i.e. A and B become dependent once M is added to the conditioning set. Confirmed colliders are oriented $A \to M \leftarrow B$, with A, B recorded as collider.

4.5 Edge Orientation via Meek's Rules

From confirmed colliders, we propagate orientations using Meek's R1–R3 (no new colliders, no cycles, no collider via triangle) until convergence; R4 is not required, as it is only needed when background knowledge is incorporated. Contradictory or cycle-inducing orientations are left undirected. The output is the estimated CPDAG.

5 Complexity

In this section, we analyse the computational complexity of the proposed method. Let n denote the number of samples, $|U|$ the number of variables, o the number of binary features after one-hot

encoding, N_{ep} the number of training epochs, m the number of clauses per class, n_c the maximum number of classes (values) of any target, q the maximum conditioning-set size, $K = \max_{Y \in U} |D'_Y|$ the maximum number of candidates retained for any target, and d the maximum node degree in the learned skeleton $\hat{G}$. We assume a single CI test costs $\mathcal{O}(n)$. Clause normalisation, scoring, the relative-drop cut-off and Meek's rules require no CI tests or passes over the data samples, so we leave them out.

WTM training. We train one WTM for each target, giving $|U|$ WTMs in total. In each epoch, a WTM goes through all n samples. For each sample, it evaluates m clauses for each of the n_c classes of the target, and each clause checks at most $2o$ literals. Assuming each variable has a bounded number of values, $o = \mathcal{O}(|U|)$. Therefore, one WTM costs $\mathcal{O}(N_{\text{ep}}\, n\, m\, n_c\, |U|)$, and all $|U|$ WTMs cost $\mathcal{O}(N_{\text{ep}}\, n\, m\, n_c\, |U|^2)$.

CI testing on D'. For each target, every candidate in D'_Y is tested against conditioning sets drawn from the other $|D'_Y| - 1$ candidates, up to size q. This requires at most $K \sum_{z=0}^{q} \binom{K-1}{z} = \mathcal{O}(K^{q+1})$ tests per target, so the total cost is $\mathcal{O}(|U|\, K^{q+1} n)$. In practice, fewer tests are needed, because the zero-order tests remove many candidates before any conditioning is done.

Collider search. Every unshielded triple $A - M - B$ is formed by a pair of M's neighbours, so each node M lies in the middle of at most $\binom{d}{2} = \mathcal{O}(d^2)$ triples. For each triple without a recorded separating set, we test conditioning sets drawn from A's other neighbours (at most $d - 1$) up to size q, which takes $\sum_{z=0}^{q} \binom{d-1}{z} = \mathcal{O}(d^q)$ tests, plus one test to confirm the collider. Repeating this for all $|U|$ nodes gives $\mathcal{O}(|U|\, d^{q+2}\, n)$.

Overall. The overall computational complexity of WTM-BN is bounded by:

$$\mathcal{O}\big(N_{\text{ep}}\, n\, m\, n_c\, |U|^2 + |U|\, K^{q+1} n + |U|\, d^{q+2} n\big) = \mathcal{O}\big(N_{\text{ep}}\, n\, m\, n_c\, |U|^2\big). \tag{14}$$

The cost is linear in n. In our experiments $q = 3$, so the CI testing and collider search terms become $O(|U|\, K^4 n)$ and $O(|U|\, d^5 n)$. When $K^4 + d^5 = O(N_{ep}\, n\, m\, n_c\, |U|)$, WTM training is the dominant term and the cost is quadratic in $|U|$. K becomes large when the relative-drop cut-off retains many candidates, or when no cut-off is found for some target and all $|U| - 1$ variables are retained. In that case CI testing can dominate, with worst case $O(|U|^5 n)$ when $K = d = |U| - 1$.

6 Experiment

6.1 Experimental setup

Implementation and hardware. All methods were implemented in Python 3.13 and evaluated on 13 datasets (Scutari, 2010; Bischl et al., 2025) as shown in Table 5. BOSS used the causal-learn implementation (Zheng et al., 2024), while other methods used their official implementations. Experiments were run on a MacBook Pro with an Apple M3 processor and 16 GB RAM.

Preprocessing. Categorical/BN data were label-encoded then OHE for WTM-BN; $\mathcal{C}$-PC and WTM-BN used chi-square ($\alpha = 0.01$), BOSS used BDeu. Continuous data were quantile-binned (5 bins) and OHE for WTM-BN, with raw values kept for Fisher's-Z ($\alpha = 0.01$, $\mathcal{C}$-PC and WTM-BN); BOSS and XGES used BIC. WTM-BN used an 80/20 split throughout, with training-only resampling for class balance in the classification setting.

Evaluation protocol. For the BN benchmark datasets, we report precision, recall, F1-score, and runtime over 10 seeded runs. For the real-world datasets, we evaluated the predictive performance of the discovered neighbours using classification accuracy for categorical targets and R^2 for continuous targets.

6.2 Results

6.2.1 Bayesian network benchmarks

Table 1 reports precision, recall, F1-score and runtime of BOSS, $\mathcal{C}$-PC and WTM-BN on five *bnlearn* benchmarks.

Table 1: Structure recovery on BN benchmarks (mean ± std). NA: did not complete.

Dataset	Algorithm	Precision	Recall	F1-score	Time (s)
Insurance	BOSS	$95.84\% \pm 4.97\%$	$\mathbf{80.77\% \pm 5.16\%}$	$\mathbf{87.65\% \pm 5.09\%}$	50.82 ± 2.05
	$\mathcal{C}$-PC	$\mathbf{97.94\% \pm 1.92\%}$	$72.31\% \pm 1.54\%$	$83.18\% \pm 1.48\%$	$\mathbf{1.22 \pm 0.03}$
	WTM-BN	$83.74\% \pm 2.98\%$	$76.92\% \pm 3.65\%$	$80.14\% \pm 2.79\%$	17.81 ± 0.14
Hailfinder	BOSS	$87.68\% \pm 1.13\%$	$\mathbf{81.82\% \pm 0.00\%}$	$\mathbf{84.64\% \pm 0.53\%}$	141.86 ± 7.08
	$\mathcal{C}$-PC	$\mathbf{100.00\% \pm 0.00\%}$	$50.91\% \pm 0.74\%$	$67.47\% \pm 0.65\%$	$\mathbf{4.30 \pm 0.03}$
	WTM-BN	$74.11\% \pm 2.64\%$	$67.58\% \pm 2.27\%$	$70.68\% \pm 2.34\%$	38.39 ± 0.20
Hepar2	BOSS	$\mathbf{99.72\% \pm 0.56\%}$	$59.84\% \pm 2.09\%$	$\mathbf{74.78\% \pm 1.71\%}$	92.13 ± 2.57
	$\mathcal{C}$-PC	$99.06\% \pm 0.77\%$	$50.73\% \pm 1.51\%$	$67.08\% \pm 1.25\%$	$\mathbf{11.91 \pm 0.42}$
	WTM-BN	$92.60\% \pm 2.59\%$	$\mathbf{60.65\% \pm 1.22\%}$	$73.28\% \pm 1.46\%$	46.80 ± 0.65
Munin1	BOSS	$79.93\% \pm 1.34\%$	$\mathbf{66.66\% \pm 1.94\%}$	$\mathbf{72.69\% \pm 1.70\%}$	12836.74 ± 1772.91
	$\mathcal{C}$-PC	$\mathbf{99.58\% \pm 0.52\%}$	$33.70\% \pm 0.70\%$	$50.35\% \pm 0.74\%$	488.01 ± 19.08
	WTM-BN	$62.11\% \pm 1.96\%$	$50.92\% \pm 0.52\%$	$55.95\% \pm 0.92\%$	$\mathbf{168.75 \pm 8.94}$
Pigs	BOSS	NA	NA	NA	NA
	$\mathcal{C}$-PC	$\mathbf{99.55\% \pm 0.21\%}$	$\mathbf{100.00\% \pm 0.00\%}$	$\mathbf{99.77\% \pm 0.10\%}$	5405.45 ± 60.15
	WTM-BN	$98.78\% \pm 0.56\%$	$99.94\% \pm 0.08\%$	$99.36\% \pm 0.32\%$	$\mathbf{1076.72 \pm 87.24}$

On the three smaller networks (Insurance, Hailfinder and Hepar2), BOSS gives the highest F1-score (87.65%, 84.64% and 74.78%) but takes much more time, whereas $\mathcal{C}$-PC is the fastest but has low recall. WTM-BN lies between the two. It is 2–4$\times$ faster than BOSS, with an F1-score lower by 1.5–14 points, and gives higher recall than $\mathcal{C}$-PC on all three networks, with the best recall on Hepar2. It also gives higher F1 than $\mathcal{C}$-PC on Hailfinder and Hepar2, though not on Insurance. However, its precision is lower than that of the other two methods.

The runtime advantage of WTM-BN increases with network size. On Munin1, it is about 76$\times$ faster than BOSS and 3$\times$ faster than $\mathcal{C}$-PC, and gives higher F1 than $\mathcal{C}$-PC (55.95% vs. 50.35%), though lower than BOSS (72.69%). On Pigs, BOSS does not complete, whereas WTM-BN reaches an F1-score of 99.36%, within one point of $\mathcal{C}$-PC (99.77%), at about one-fifth of the runtime.

Overall, WTM-BN gives a reasonable balance between accuracy and runtime on smaller networks, and the best trade-off among the three methods on larger networks.

6.3 Edge orientation

Table 2 reports the precision, recall and F1-score of the learnt edge directions. An edge is counted as correct only if its direction matches the ground-truth DAG, so the scores are generally lower than in Table 1.

On the three smaller networks, BOSS gives the highest F1-score (70.31%, 66.54% and 64.21% on Insurance, Hailfinder and Hepar2), and WTM-BN comes second on all three (60.62%, 61.28% and 56.96%). The recall of WTM-BN is close to that of BOSS, and is even slightly higher on Hailfinder (55.45% vs. 53.03%). In contrast, $\mathcal{C}$-PC orients very few edges correctly. Its precision is sometimes high, but its recall is low, and on Hepar2 its F1-score falls to only 14.56%.

On the larger networks, the same pattern continues. On Munin1, BOSS again gives the best F1-score (67.95%), but it takes about 76 times longer than WTM-BN. WTM-BN outperforms $\mathcal{C}$-PC by over 15 points (43.82% vs. 28.22%), mainly because of its higher recall. On Pigs, BOSS does not complete, while $\mathcal{C}$-PC and WTM-BN recover almost all edge directions (F1-scores of 100.00% and 99.07%).

Overall, BOSS gives the most accurate directions wherever it runs, but at a high computational cost. WTM-BN outperforms $\mathcal{C}$-PC in F1-score on four of the five networks, including Insurance, where $\mathcal{C}$-PC had a better skeleton, and it also scales to networks where BOSS fails.

6.3.1 Real datasets: categorical targets

Since ground truth is not available for real-world datasets, we evaluate structure recovery indirectly. For each dataset, the complete network is learned using BOSS, $\mathcal{C}$-PC and WTM-BN, and the local

Table 2: Learned directions on BN benchmarks (mean ± std). NA: did not complete.

Dataset	Algorithm	Precision	Recall	F1-score
Insurance	BOSS	$\mathbf{95.95\% \pm 1.35\%}$	$\mathbf{56.15\% \pm 9.99\%}$	$\mathbf{70.31\% \pm 8.94\%}$
	$\mathcal{C}$-PC	$72.45\% \pm 4.68\%$	$33.08\% \pm 5.88\%$	$45.25\% \pm 6.45\%$
	WTM-BN	$72.41\% \pm 8.56\%$	$53.85\% \pm 11.60\%$	$60.62\% \pm 7.93\%$
Hailfinder	BOSS	$89.29\% \pm 0.90\%$	$53.03\% \pm 0.00\%$	$\mathbf{66.54\% \pm 0.25\%}$
	$\mathcal{C}$-PC	$\mathbf{97.39\% \pm 2.14\%}$	$34.24\% \pm 1.55\%$	$50.66\% \pm 1.92\%$
	WTM-BN	$69.45\% \pm 4.48\%$	$\mathbf{55.45\% \pm 7.21\%}$	$61.28\% \pm 4.63\%$
Hepar2	BOSS	$\mathbf{93.47\% \pm 2.59\%}$	$\mathbf{48.94\% \pm 2.88\%}$	$\mathbf{64.21\% \pm 2.84\%}$
	$\mathcal{C}$-PC	$31.99\% \pm 4.50\%$	$9.43\% \pm 1.51\%$	$14.56\% \pm 2.25\%$
	WTM-BN	$77.20\% \pm 4.40\%$	$45.20\% \pm 2.33\%$	$56.96\% \pm 2.38\%$
Munin1	BOSS	$78.21\% \pm 1.04\%$	$\mathbf{60.07\% \pm 1.67\%}$	$\mathbf{67.95\% \pm 1.47\%}$
	$\mathcal{C}$-PC	$\mathbf{81.69\% \pm 4.37\%}$	$17.07\% \pm 0.89\%$	$28.22\% \pm 1.25\%$
	WTM-BN	$54.31\% \pm 1.78\%$	$36.92\% \pm 3.70\%$	$43.82\% \pm 2.59\%$
Pigs	BOSS	NA	NA	NA
	$\mathcal{C}$-PC	$\mathbf{100.00\% \pm 0.00\%}$	$\mathbf{100.00\% \pm 0.00\%}$	$\mathbf{100.00\% \pm 0.00\%}$
	WTM-BN	$98.77\% \pm 0.62\%$	$99.38\% \pm 0.52\%$	$99.07\% \pm 0.56\%$

neighbourhood of the target is given as input to Support Vector Classifier (SVC), Naive Bayes and K-Nearest Neighbors (KNN). Table 3 reports the accuracy, runtime and average neighbourhood size per node.

WTM-BN gives higher accuracy than BOSS on Mushroom and Tuandromd, while running much faster (16.92s vs. 107.19s and 163.84s vs. 42660.46s). On Spect and Neticusdroid, its accuracy is within one percentage point of BOSS, and it is even the best in KNN on Spect, at a fraction of the runtime (1.47s vs. 5.48s and 57.99s vs. 2131.34s). WTM-BN is also the fastest method on Neticusdroid and Tuandromd, whereas $\mathcal{C}$-PC is the fastest on Spect and Mushroom. However, $\mathcal{C}$-PC gives the lowest accuracy on all four datasets, and the gap is largest on Tuandromd (about 41% vs. 98–99% for WTM-BN).

BOSS and WTM-BN learn neighbourhoods of similar size, whereas $\mathcal{C}$-PC learns sparser neighbourhoods on Spect, Neticusdroid and Tuandromd and a larger one on Mushroom, without any gain in accuracy. These results show that WTM-BN recovers neighbourhoods with enough predictive information to match or exceed BOSS, at a much lower computational cost.

These results indicate that WTM-BN recovers neighbourhoods with sufficient predictive information to match BOSS's classification performance, at a fraction of its computational cost.

Table 3: Predictive performance of the discovered neighbours on real-world classification datasets.

Dataset	Algorithm	Acc. (SVC)	Acc. (Naive Bayes)	Acc. (KNN)	Time (s)	size/node
Spect	BOSS	$\mathbf{84.43\% \pm 0.53\%}$	$\mathbf{83.49\% \pm 0.52\%}$	$82.27\% \pm 1.05\%$	5.48	2.52 ± 1.02
	$\mathcal{C}$-PC	$73.27\% \pm 0.61\%$	$72.22\% \pm 0.71\%$	$71.95\% \pm 0.93\%$	**0.31**	$\mathbf{1.65 \pm 0.91}$
	WTM-BN	$84.30\% \pm 0.66\%$	$83.45\% \pm 0.78\%$	$\mathbf{82.54\% \pm 1.02\%}$	1.47	2.70 ± 1.04
Mushroom	BOSS	$76.50\% \pm 0.16\%$	$74.28\% \pm 0.31\%$	$76.67\% \pm 0.11\%$	107.19	$\mathbf{5.91 \pm 3.68}$
	$\mathcal{C}$-PC	$67.06\% \pm 0.13\%$	$66.27\% \pm 0.35\%$	$65.04\% \pm 0.16\%$	**2.15**	10.09 ± 5.76
	WTM-BN	$\mathbf{79.47\% \pm 0.24\%}$	$\mathbf{78.47\% \pm 0.03\%}$	$\mathbf{79.26\% \pm 0.23\%}$	16.92	6.45 ± 4.25
Neticusdroid	BOSS	$\mathbf{96.68\% \pm 0.10\%}$	$\mathbf{95.92\% \pm 0.10\%}$	$\mathbf{96.29\% \pm 0.12\%}$	2131.34	6.21 ± 4.85
	$\mathcal{C}$-PC	$87.31\% \pm 0.11\%$	$86.41\% \pm 0.14\%$	$86.94\% \pm 0.21\%$	64.06	$\mathbf{4.92 \pm 4.53}$
	WTM-BN	$96.49\% \pm 0.06\%$	$95.71\% \pm 0.07\%$	$96.19\% \pm 0.06\%$	**57.99**	5.54 ± 3.37
Tuandromd	BOSS	$79.81\% \pm 0.20\%$	$79.02\% \pm 0.19\%$	$79.72\% \pm 0.21\%$	42660.46	4.64 ± 6.66
	$\mathcal{C}$-PC	$41.31\% \pm 0.04\%$	$40.85\% \pm 0.07\%$	$41.21\% \pm 0.05\%$	803.44	$\mathbf{2.25 \pm 3.92}$
	WTM-BN	$\mathbf{98.85\% \pm 0.24\%}$	$\mathbf{98.27\% \pm 0.25\%}$	$\mathbf{98.70\% \pm 0.26\%}$	**163.84**	4.79 ± 3.60

6.3.2 REAL DATASETS: CONTINUOUS TARGETS

Table 4 reports results on four continuous-target datasets. For each learned graph, the local neighbourhood of the target is used as input to linear regression (LR), KNN and Support Vector Regression (SVR), and the R^2 values are averaged over 10 random 80/20 train–test splits.

XGES gives the best LR performance on all four datasets, but its runtime is very high, especially on Musk (57699s) and ECG5000 (22268s). BOSS is second-best in LR throughout, within about two percentage points of XGES, and also gives the best SVR on Superconductivity and Musk; however, it is much slower than $\mathcal{C}$-PC and WTM-BN on the three larger datasets. DAGMA learns the sparsest neighbourhoods (1.1–2.0 variables per node), but gives the lowest R^2 on California, Superconductivity and Musk.

WTM-BN gives the best KNN performance on Superconductivity and Musk, and is the fastest method on Musk and ECG5000, the two largest datasets. On ECG5000, however, its R^2 is the lowest among all methods. $\mathcal{C}$-PC gives the best KNN and SVR on ECG5000 and is the fastest on Superconductivity.

Overall, WTM-BN gives competitive accuracy at a much lower runtime than XGES and BOSS, and this advantage increases with the number of variables.

Table 4: Predictive performance R^2 of the discovered neighbours on real-world regression datasets.

Dataset	Algorithm	R^2 (LR)	R^2 (KNN)	R^2 (SVR)	Time (s)	size/node
California	XGES	$\mathbf{76.04\% \pm 0.47\%}$	$\mathbf{76.88\% \pm 1.06\%}$	$\mathbf{77.72\% \pm 1.82\%}$	2.58	5.56 ± 1.42
	DAGMA	$69.09\% \pm 0.42\%$	$67.73\% \pm 1.18\%$	$69.05\% \pm 1.13\%$	1.96	$\mathbf{1.11 \pm 0.57}$
	BOSS	$74.36\% \pm 0.53\%$	$74.47\% \pm 1.15\%$	$75.30\% \pm 1.77\%$	**0.01**	4.44 ± 1.50
	$\mathcal{C}$-PC	$71.82\% \pm 0.34\%$	$72.27\% \pm 1.49\%$	$72.51\% \pm 1.32\%$	0.02	2.22 ± 0.63
	WTM-BN	$73.18\% \pm 0.44\%$	$73.44\% \pm 0.92\%$	$73.77\% \pm 1.68\%$	1.25	2.89 ± 0.74
Superconductivity	XGES	$\mathbf{97.96\% \pm 0.05\%}$	$94.40\% \pm 0.24\%$	$94.00\% \pm 0.57\%$	6157.18	36.83 ± 6.59
	DAGMA	$85.07\% \pm 0.27\%$	$88.60\% \pm 0.23\%$	$87.06\% \pm 0.26\%$	25.64	$\mathbf{1.59 \pm 0.75}$
	BOSS	$97.80\% \pm 0.05\%$	$95.27\% \pm 0.22\%$	$\mathbf{98.09\% \pm 0.23\%}$	66.05	24.32 ± 5.48
	$\mathcal{C}$-PC	$88.71\% \pm 0.20\%$	$93.37\% \pm 0.28\%$	$91.83\% \pm 0.30\%$	**16.71**	2.61 ± 1.23
	WTM-BN	$93.33\% \pm 0.12\%$	$\mathbf{96.93\% \pm 0.13\%}$	$95.52\% \pm 0.20\%$	18.78	5.59 ± 1.90
ECG5000	XGES	$\mathbf{96.67\% \pm 0.17\%}$	$91.16\% \pm 0.35\%$	$87.90\% \pm 1.15\%$	22268.51	24.23 ± 12.59
	DAGMA	$95.13\% \pm 0.25\%$	$93.82\% \pm 0.21\%$	$91.11\% \pm 1.07\%$	62.30	$\mathbf{1.96 \pm 0.26}$
	BOSS	$96.58\% \pm 0.19\%$	$92.67\% \pm 0.28\%$	$89.41\% \pm 1.15\%$	223.66	14.23 ± 8.11
	$\mathcal{C}$-PC	$95.81\% \pm 0.25\%$	$\mathbf{94.32\% \pm 0.26\%}$	$\mathbf{91.20\% \pm 1.14\%}$	147.24	3.70 ± 1.26
	WTM-BN	$91.62\% \pm 0.39\%$	$90.20\% \pm 0.44\%$	$87.06\% \pm 0.96\%$	**30.82**	5.70 ± 1.59
Musk	XGES	$\mathbf{94.52\% \pm 0.09\%}$	$96.65\% \pm 0.20\%$	$95.91\% \pm 0.14\%$	57699.20	36.42 ± 6.28
	DAGMA	$76.31\% \pm 0.05\%$	$81.99\% \pm 0.09\%$	$79.65\% \pm 0.15\%$	272.53	$\mathbf{1.66 \pm 0.88}$
	BOSS	$93.85\% \pm 0.09\%$	$97.34\% \pm 0.22\%$	$\mathbf{97.64\% \pm 0.11\%}$	258.81	24.78 ± 5.46
	$\mathcal{C}$-PC	$86.58\% \pm 0.05\%$	$97.20\% \pm 0.22\%$	$94.06\% \pm 0.15\%$	150.13	3.18 ± 1.08
	WTM-BN	$87.60\% \pm 0.09\%$	$\mathbf{97.73\% \pm 0.24\%}$	$95.20\% \pm 0.21\%$	**38.80**	5.64 ± 1.67

7 CONCLUSION

In this work, we have proposed WTM-BN, a BN structure learning method that combines WTM–based variable scoring with CI testing. Clause-based importance scores select a small candidate set for each variable, so CI testing is done only on these candidates. Collider detection and Meek's rules are then applied to obtain a complete CPDAG. On the bnlearn benchmarks, WTM-BN is 2–76 times faster than BOSS, gives better F1 than $\mathcal{C}$-PC on larger networks, and is the fastest method on Munin1 and Pigs. Its runtime advantage increases with network size. On Pigs, where BOSS could not complete, it matches the accuracy of $\mathcal{C}$-PC with nearly five times lower runtime. On real-world datasets, it gives competitive predictive performance, with the best results on Mushroom and Tuandromd. On continuous-target datasets, WTM-BN gives the best KNN performance on Superconductivity and Musk, and is the fastest method on Musk and ECG5000, the two largest datasets. However, it does not guarantee a valid separating set, so skeleton recovery is not guaranteed to be complete.

AI USE STATEMENT

In this work, we used generative AI tools only for the tasks described below. We used generative AI tools to assist in implementing the state-of-the-art methods used for comparison in our experiments. All generated code was carefully reviewed and tested by the authors to ensure it correctly follows the original methods. We also used generative AI tools to edit the manuscript to improve readability, limited to correcting grammar and spelling and polishing the wording of text written by the authors.

We did not use generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims or assisting with proofs, proposing or refining hypotheses, designing research methodology or experiments, translation, cleaning or reformatting datasets, qualitative or thematic data analysis, or interpreting results. The generative AI tools did not contribute research ideas, results, or claims.

ETHICS STATEMENT

This work proposes a method for Bayesian network structure learning and does not involve human subjects, user studies, or the collection of new data. All experiments use publicly available datasets: five benchmark networks from the *bnlearn* repository and eight real-world datasets from *openml*. These are used under their respective licenses and in line with their intended research use. To the best of our knowledge, none of these datasets contain personally identifiable or sensitive information, so no IRB approval was required. The authors declare no conflicts of interest or sponsorship that could have influenced this work.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of our results. The full WTM-BN procedure is described in Section 4. This includes WTM training for each target variable, clause-based relevance scoring, the relative-drop cutoff for candidate selection, CI testing, and orientation via collider detection and Meek's rules. The experimental setup, including the benchmark networks, datasets, baselines, and evaluation metrics, is detailed in Section 6. Hyperparameter selection for WTM-BN and all baseline methods is reported in Appendix A.1. The complete source code and all datasets used in our experiments are provided in the supplementary material, together with instructions for reproducing the reported results. All datasets are also publicly available from the *bnlearn* repository and *openml*.

REFERENCES

K. Darshana Abeyrathna, Ole-Christoffer Granmo, and Morten Goodwin. Extending the tsetlin machine with integer-weighted clauses for increased interpretability. *IEEE Access*, 9:8233–8248, 2021. doi: 10.1109/ACCESS.2021.3049569.

Bryan Andrews, Joseph Ramsey, Rubén Sánchez-Romero, Jazmin Camchong, and Erich Kummerfeld. Fast scalable and accurate discovery of dags using the best order score search and grow-shrink trees. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Kevin Bello, Bryon Aragam, and Pradeep Ravikumar. DAGMA: Learning DAGs via M-matrices and a Log-Determinant Acyclicity Characterization. In *Advances in Neural Information Processing Systems*, 2022.

Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, Lennart Purucker, Sahithya Ravi, Jan N van Rijn, Prabhant Singh, Joaquin Vanschoren, Jos van der Velde, and Marcel Wever. Openml: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7):101317, 2025. doi: 10.1016/j.patter.2025.101317. URL https://www.cell.com/patterns/fulltext/S2666-3899(25)00165-5.

David Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 01 2002. doi: 10.1162/153244303321897717.

Diego Colombo and Marloes H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, January 2014. ISSN 1532-4435.

Ole-Christoffer Granmo. The Tsetlin Machine – A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic, 2021. URL https://arxiv.org/abs/1804.01508.

Neville Kenneth Kitson, Anthony C. Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of bayesian network structure learning. *Artif. Intell. Rev.*, 56(8):8721–8814, January 2023. ISSN 0269-2821. doi: 10.1007/s10462-022-10351-w. URL https://doi.org/10.1007/s10462-022-10351-w.

Murat Kocaoglu. Characterization and learning of causal graphs with small conditioning sets. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Wai-Yin Lam, Bryan Andrews, and Joseph Ramsey. Greedy relaxations of the sparsest permutation algorithm. In James Cussens and Kun Zhang (eds.), *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 1052–1062. PMLR, 01–05 Aug 2022. URL https://proceedings.mlr.press/v180/lam22a.html.

Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast pc algorithm for high dimensional causal discovery with multi-core pcs. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 16(5):1483–1495, September 2019. ISSN 1545-5963. doi: 10.1109/TCBB.2016.2591526. URL https://doi.org/10.1109/TCBB.2016.2591526.

Kenneth Lee, Bruno Ribeiro, and Murat Kocaoglu. Constraint-based causal discovery from a collection of conditioning sets. In *Proceedings of the Forty-First Conference on Uncertainty in Artificial Intelligence*, UAI '25. JMLR.org, 2025.

Achille Nazaret and David Blei. Extremely greedy equivalence search. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, UAI '24. JMLR.org, 2024.

Kristina Polotskaya, Carlos S. Muñoz-Valencia, Alejandro Rabasa, Jose A. Quesada-Rico, Domingo Orozco-Beltrán, and Xavier Barber. Bayesian networks for the diagnosis and prognosis of diseases: A scoping review. *Machine Learning and Knowledge Extraction*, 6(2):1243–1262, 2024. ISSN 2504-4990. doi: 10.3390/make6020058. URL https://www.mdpi.com/2504-4990/6/2/58.

M. Scutari. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3):1–22, 2010. URL http://www.jstatsoft.org/v35/i03/.

L Solus, Y Wang, and C Uhler. Consistency guarantees for greedy permutation-based causal inference algorithms. *Biometrika*, 108(4):795–814, 12 2021. ISSN 0006-3444. doi: 10.1093/biomet/asaa104. URL https://doi.org/10.1093/biomet/asaa104.

Marc Teyssier and Daphne Koller. Ordering-based search: a simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pp. 584–590, Arlington, Virginia, USA, 2005. AUAI Press. ISBN 0974903914.

Marcel Wienöbst and Maciej Liskiewicz. Recovering causal structures from low-order conditional independencies. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(06): 10302–10309, April 2020. ISSN 2159-5399. doi: 10.1609/aaai.v34i06.6593. URL http://dx.doi.org/10.1609/aaai.v34i06.6593.

Kui Yu, Lin Liu, Jiuyong Li, and Huanhuan Chen. Mining markov blankets without causal sufficiency. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6333–6347, 2018. doi: 10.1109/TNNLS.2018.2828982.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Dags with no tears: Continuous optimization for structure learning, 2018. URL https://arxiv.org/abs/1803.01422.

Yujia Zheng, Biwei Huang, Wei Chen, Joseph Ramsey, Mingming Gong, Ruichu Cai, Shohei Shimizu, Peter Spirtes, and Kun Zhang. Causal-learn: Causal discovery in python. *Journal of Machine Learning Research*, 25(60):1–8, 2024.

A APPENDIX

Table 5: Datasets used in the evaluation.

Dataset	Samples	Variables
Insurance	5,000	27
Hailfinder	5,000	56
Hepar2	5,000	70
Munin1	5,000	186
Pigs	5,000	441
Mushroom	3,000	22
Neticusdroid	3,000	86
Tuandromd	3,000	241
Spect	267	22
California Housing	3,000	8
Superconductivity	3,000	81
Musk	3,000	166
ECG5000	3,000	140

A.1 HYPERPARAMETER SELECTION

We tuned the WTM-BN clause (C) and specificity (S) using a grid search, while fixing the number of epochs to 5 and the threshold to $T = C$. The search was performed separately for categorical and continuous targets using one representative dataset for each case, Insurance for categorical targets and Superconductivity for continuous targets. We considered $C \in \{10, 50, 100\}$ and $S \in \{1, 5, 10\}$, and evaluated the resulting F1-score/accuracy and time, as shown in Figures 1 and 2. For the continuous case, accuracy was obtained by training a linear regression probe over 10 runs and reported as mean ± standard deviation, whereas time reflects a single WTM-BN run per configuration and is therefore reported as a single value.

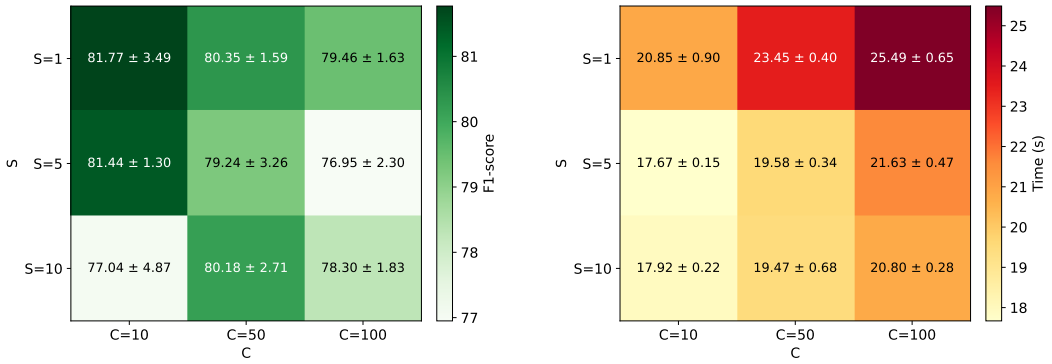
For the categorical case, the configuration $C = 10$, $S = 5$ achieved a near-best F1-score ($81.44\% \pm 1.30\%$) with low variance and lower time compared with configurations having a higher clause count or specificity. The higher-C and higher-S settings did not provide any consistent improvement.

For the continuous case, the differences in accuracy across values of S were small at higher clause counts. However, at $C = 10$, $S = 1$ substantially outperformed $S = 5$ and $S = 10$, achieving $93.19\% \pm 0.13\%$, compared with 86.37% and 86.63%, respectively. It was also the fastest configuration. Since this configuration achieved accuracy comparable to that obtained with larger clause counts, it was preferred for computational efficiency.

Based on these results, we fixed $C = 10$, $T = C$, and 5 epochs across all datasets. We used $S = 5$ for categorical datasets and $S = 1$ for continuous datasets. The same hyperparameters were maintained across all experiments to ensure a fair and consistent comparison.

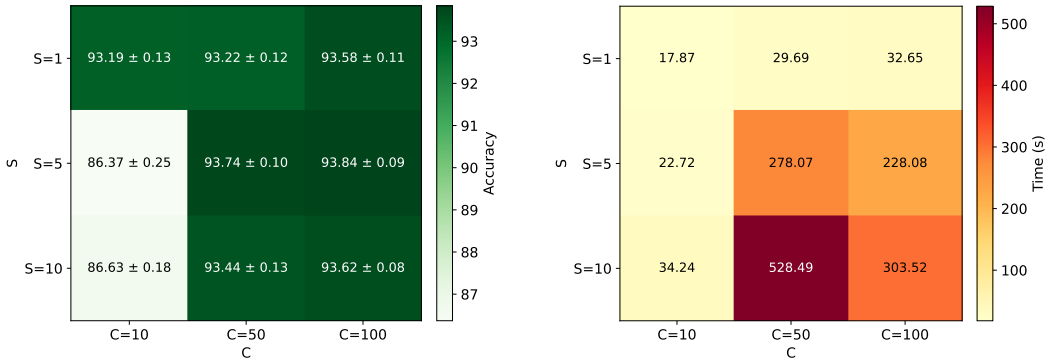
A.2 CANDIDATE SET SIZE UNDER THE RELATIVE-DROP

Table 6 presents the average number of candidate variables retained for each target by the relative-drop technique, that is, the size of the reduced candidate set D' on which CI testing is carried out. On six datasets, namely ECG5000 (5.10), Musk (5.22), California Housing (5.25), Pigs (5.35), Hailfinder (5.71) and Superconductivity (6.00), the average value lies very close to the minimum candidate count $r_{\min} = 5$, with small standard deviations (at most 0.72). This indicates that the steepest relative drop generally occurs within the first few ranks. Insurance (6.84) and Hepar2 (7.25) retain slightly larger candidate sets. On Spect (12.93), Mushroom (17.41) and Neticusdroid (17.96), the candidate sets are moderately larger, indicating that the steepest relative drop occurs at a later rank. The largest candidate sets are observed on Munin1 (34.15) and Tuandromd (112.84), which also show the highest variability across targets ($\pm$3.66 and $\pm$7.43 respectively). The restriction of the conditioning search space to a small candidate set on most datasets contributes to the runtime savings achieved by WTM-BN.



(a) F1-score. (b) Time.

Figure 1: Grid search over clause C and specificity S on Insurance dataset at fixed $T = C$, reporting (a) F1-score and (b) Time.



(a) Accuracy. (b) Time.

Figure 2: Grid search over clause C and specificity S on Superconductivity dataset at fixed $T = C$, reporting (a) Accuracy and (b) Time.

Table 6: Mean number of candidate variables retained per target by the relative-drop technique.

Dataset	**Mean Cut-off**
Insurance	6.84 ± 0.54
Hailfinder	5.71 ± 0.72
Hepar2	7.25 ± 0.56
Munin1	34.15 ± 3.66
Pigs	5.35 ± 0.08
Mushroom	17.41 ± 0.91
Neticusdroid	17.96 ± 1.84
Tuandromd	112.84 ± 7.43
Spect	12.93 ± 1.86
California Housing	5.25 ± 0.29
Superconductivity	6.00 ± 0.19
Musk	5.22 ± 0.10
ECG5000	5.10 ± 0.07