Probabilistic Semantics Guided Discovery of Approximate Functional Dependencies

Liang Duan[1,2] **Xinran Wu**[1,2] **Xinhui Li**[1,2] **Lixing Yu**[1,2] **Kun Yue**[1,2]

[1]Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, China
[2]School of Information Science and Engineering, Yunnan University, Kunming, China

Abstract

As the general description of relationships between attributes, approximate functional dependencies (AFDs) almost hold for a given dataset with a few violations. Most of existing methods for AFD discover are insufficient to balance the efficiency and accuracy due to the massive search space and permission of violations. To address these issues, we propose an efficient method of probabilistic semantics guided discovery of AFDs based on Bayesian network (BN). Firstly, we learn a BN structure and conduct conditional independence tests on the learned structure rather than the entire search space, such that candidate AFDs could be obtained. Secondly, we fulfill search space reduction and structure pruning by making use of probabilistic semantics of graphical models in terms of BN. Consequently, we provide a branch-and-bound algorithm to discover the AFDs with the highest smoothed mutual information scores. Experimental results illustrate that our proposed method is more effective and efficient than the comparison methods. Our code is available at https://github.com/DKE-Code/BNAFD.

1 INTRODUCTION

Functional dependency (FD) is a constraint between attributes of a relational database, indicating that the value of one attribute can be uniquely determined by other attributes. FDs are employed in the relational schema normalization to remove redundancy from databases [Wei and Link, 2021], facilitating tasks like query optimization [Gultchin et al., 2023] and data cleaning [Bounia and Koriche, 2023]. Since FDs are frequently unknown, numerous studies focus on automatic FD discovery without human intervention.

As the relaxed FD, approximate functional dependency (AFD) allows for a few violating rows in datasets [Mandros et al., 2017]. Since real-world data typically contains noise, the mining of AFDs is more useful than that of FDs, and the discovered AFDs exhibit enhanced performance in subsequent data management tasks [Zhang et al., 2020]. For instance, Table 1 illustrates that noise can make the true FDs overlooked by FD discovery methods, such as $\mathrm{City} \rightarrow \mathrm{State}$. Additionally, finite samples can result in obtaining spurious FDs, like $\mathrm{State}, \mathrm{Stauts} \rightarrow \mathrm{City}$, which could be addressed by AFD discovery methods [Jiang et al., 2023]. Nevertheless, the AFD discovery faces not only a massive search space but also the challenge of evaluating AFD's validity due to its relaxation.

City	State	Zip	Telephone	Status
MONROE	GA	30655	7702678461	OPEN
MONROE	GA	30655	7702678461	OPEN
ATLANTA	GA	30327	4043553788	CLOSE
DAYTONA BEACH	FL	32117	3862313907	OPEN
DAYTONA BEACH	KY	32117	3862316000	OPEN

Table 1: Noise and sample limitations in FD discovery.

Various methods have been proposed for AFD discovery from various perspectives. The threshold-based methods, such as traditional FD discovery methods [Flach and Savnik, 1999, Huhtala et al., 1999] and PYRO [Kruse and Naumann, 2018], search and validate all possible AFDs in dataset by setting the maximum number of violating rows (called error threshold). Due to the manually set fixed threshold, these methods often obtain a large number of spurious dependencies, which only hold in specific datasets but fail to reflect the true dependencies between attributes. The structure-based method FDX [Zhang et al., 2020] transforms the task of AFD mining into learning a sparse structure to improve the efficiency. However, it generates AFDs from a sparse structure, which typically results in obtaining only a few simple AFDs and low recall. Additionally, the score-based methods, RFI [Mandros et al., 2017] and smoothed mutual information estimator (SMI) [Pennerath et al., 2020], use in-

formation theoretic scores to validate AFDs, preventing the discovery of spurious dependencies. Nonetheless, they suffer from low efficiency due to the massive search space and might discover non-minimal dependencies, whose left-hand side contains redundant attributes.

As is known that Bayesian network (BN) [Pearl, 1988, Yu et al., 2021] is widely used for uncertain knowledge representation and inference. To address the challenges of the massive search space and non-minimal dependencies, we propose BNAFD, a BN-based AFD discovery method that leverages BN's dependence representation capabilities.

Specifically, we first learn a BN structure from input data and limit the search space to the local structures within the BN, in accordance with the properties of the Markov blanket. We then generate candidate AFDs (cAFDs) by considering the probabilistic semantics and conducting conditional independence (CI) tests to remove non-minimal dependencies. In view of the merit of SMI score to accurately measure the correlation between attribute sets and effectively avoid obtaining spurious dependencies [Pennerath et al., 2020], we use SMI scores to validate AFDs. We next provide an upper bound for the SMI score and design a branch-and-bound algorithm to efficiently search for the highest-scoring AFDs from candidates. Moreover, during the calculation of SMI scores, we reuse some intermediate results to further improve the efficiency.

Our contributions are summarized as follows:

- We propose an effective method to generate cAFDs by removing non-minimal dependencies, which is implemented by reduction of search space and CI tests on the BN structure learned from data.
- We provide a theoretically effective bound for SMI scores and design a branch-and-bound algorithm to search for AFDs with the highest SMI score. Moreover, a sample count reuse approach is provided to speed up the calculation of SMI scores.
- We conduct extensive experiments on public and synthetic datasets. The results demonstrate that our method achieves the best balance between the $F1$ score and efficiency for AFD discovery.

2 RELATED WORK AND PROBLEM STATEMENT

2.1 RELATED WORK

FD discovery. The methods for mining minimal non-trivial FDs check whether FDs are valid on the given dataset. Column-based methods: [Huhtala et al., 1999] propose TANE to validate minimal, non-trivial FD by dynamically pruning the search space. [Novelli and Cicchetti, 2001] and [Yao et al., 2002] adopt different traversal strategies and pruning rules. Row-based methods: Dep-Miner [Lopes et al., 2000] and FastFDs [Wyss et al., 2001] find attribute sets that agree on certain tuple pairs and build maximal or minimal sets to derive FDs. [Flach and Savnik, 1999] propose Fdep to initiate a set of FDs and specialize them by pair-wise comparisons of all tuples. Hybrid methods: [Papenbrock and Naumann, 2016] propose HYFD to generate candidate FDs and validate them. [Abu-El-Haija et al., 2023] offer potential improvements for FD discovery in noisy or large-scale datasets. [Skyler et al., 2024] introduce a hybrid approach for multi-objective heuristic search. [Chen et al., 2019] and [Núñez-Molina et al., 2024] consider discovering FDs on dynamic datasets. However, these methods for FD discovery only work on error-free datasets and often produce some spurious dependencies.

AFD discovery. The methods of AFD discovery can be classified into three categories according to various validation mechanisms. Threshold-based methods: [Kruse and Naumann, 2018] propose PYRO to use a separate-and-conquer approach to locate promising candidates via error estimations. Structure-based methods: [Zhang et al., 2020] propose FDX to transform the task into structure learning to improve the efficiency. Score-based methods: [Mandros et al., 2017] propose RFI to subtract the expected value from the normalized mutual information, and [Pennerath et al., 2020] propose SMI to employ smoothing technique to counteract the overestimation of mutual information. However, these methods may yield non-minimal dependencies with low precision due to the vast search space.

2.2 PROBLEM STATEMENT

Let D' be a dataset of a relational schema R that contains sufficient samples. Assume each attribute $A \in R$ has a domain $V(A)$ and $V(X) = V(A_1) \times \cdots \times V(A_{|X|})$ is the domain of an attribute set $X = \{A_1, \cdots, A_{|X|}\} \subseteq R$. For any $X \subseteq R$ and $Y \in R$, the conditional probability $P(Y = y|X = X)$ can be statistically derived from D' without sampling bias, since the sample size is large enough. Thus, an AFD that holds in D' is defined as follows:

Definition 1 (AFD). *An AFD $X \to Y$ holds, equivalent to there being a function f such that:*

$$\forall X \in V(X) : P(Y = y|X = X) = \begin{cases} 1 - \epsilon, \text{if } y = f(X) \\ \epsilon, \text{otherwise} \end{cases} \tag{1}$$

where ϵ is a positive real number representing relaxation.

This definition indicates that an AFD is a FD that holds for most tuples with a few of violations. Moreover, $X \to Y$ is *minimal* if there is no subset of X determining Y, and it is *non-trivial* if $Y \notin X$.

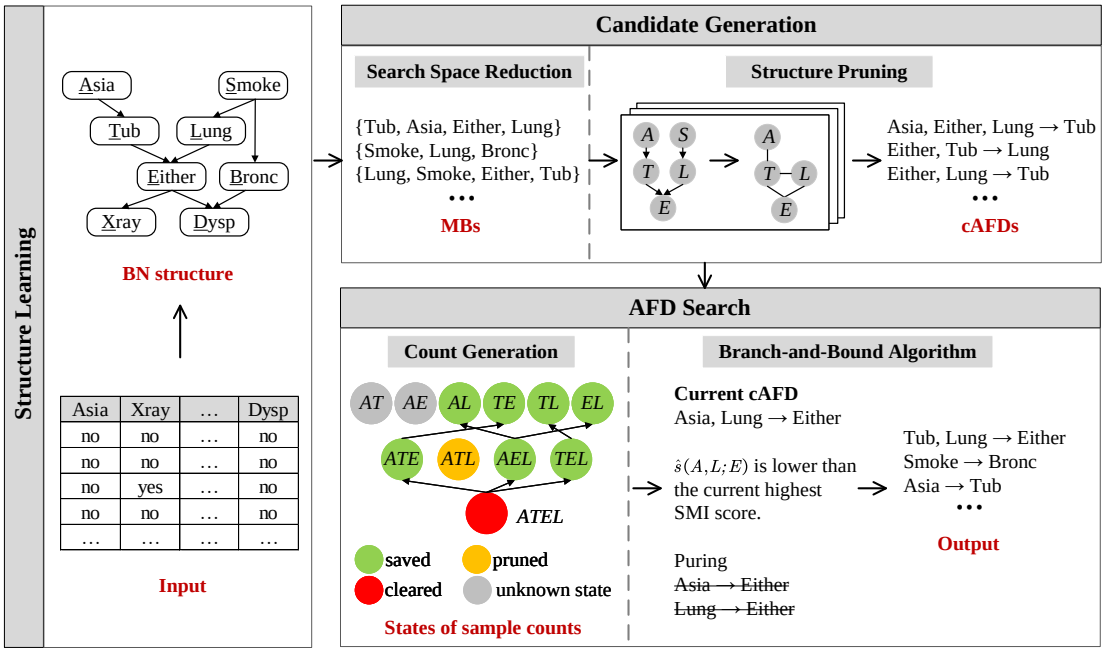


Figure 1: The framework of BNAFD, consisting of two phases: (1) candidate generation where the learned BN structure prunes the search space to generate cAFDs, and (2) AFD search that finds cAFDs with the highest SMI score.

Problem statement. Given a noisy dataset D of relational schema R with finite samples, the goal of AFD discovery is to learn a BN structure $\mathcal{G}$ from D and restrict the search scope using Markov blankets (MBs), ensuring that only the most relevant attributes X are considered for each target attribute Y. To achieve this, we conduct CI tests to remove non-minimal dependencies, ensuring that no subset $X' \subseteq X$ suffices to determine Y. Then, we introduce SMI scores and design a branch-and-bound search strategy to prune suboptimal AFDs using an upper bound of SMI and accelerate computations by reusing sample counts. Ultimately, we find all minimal non-trivial AFDs that hold on R utilizing D.

3 METHODOLOGY

3.1 OVERVIEW

Figure 1 provides an overview of our BNAFD method. The input to BNAFD is a noisy dataset D of relation schema R, and the output is a set of minimal non-trivial discovered AFDs. BNAFD consists of the following two stages:

- **Candidate generation.** We learn a BN structure $\mathcal{G}$ from D and generate multiple sub-search spaces by limiting the search space to the MBs within $\mathcal{G}$. Then, we obtain candidates by conducting CI tests to remove non-minimal dependencies with a smaller search space, thereby improving accuracy in AFD discovery.
- **AFD search.** We derive an upper bound for the SMI score and design a branch-and-bound algorithm to identify cAFDs with the highest SMI values as outputs. Furthermore, we reuse certain sample counts from the SMI calculation to reduce the computational overhead and improve overall efficiency.

3.2 CANDIDATE GENERATION

To mitigate search space explosion, we remove non-minimal dependencies to generate effective candidates.

Search space reduction. A random variable Y in BN is usually dependent on a subset of variables S rather than all variables in the BN. S contains all the useful information of Y and is called a Markov blanket [Pearl, 1988]. This indicates that we only need to consider the AFDs between the attributes in each MB, since other attributes are independent with the attributes in this MB. By limiting the search space to MBs in the context of BN, many non-minimal and invalid dependencies can be eliminated.

Inspired by existing efficient BN structure learning methods [Bello et al., 2022], we learn the BN structure $\mathcal{G}$ from the input dataset D and then limit the search space of each attribute to reduce the non-minimal dependencies. Specifically, we find the MBs of all attributes $\mathbb{S} = \{S_1, \cdots, S_n\}$, where each MB contains an attribute, its parents, its children and the parents of its children in $\mathcal{G}$. Consequently, the search

space is divided into multiple sub-search spaces, each containing all possible AFDs formed by attribute combinations within the corresponding MB. This reduces the time complexity from exponential in attribute count to exponential in the maximum MB (MMB) size.

Structure pruning. To remove the non-minimal dependencies, we first prove that CI tests can identify non-minimal dependencies through Theorem 1 and Theorem 2, and then we construct a graph structure for efficient pruning.

According to the minimality pruning rule, given any $X' \subseteq X$, if $X' \to Y$ holds, then $X \to Y$ is non-minimal [Huhtala et al., 1999]. In the process of FD discovery, this critical rule dynamically prunes non-minimal dependencies using known dependencies. However, the score-based methods determine the validity of AFD only at the final stage, which prevents us from using the previous pruning rule. Thus, we modify the above rule by incorporating CI tests.

Theorem 1. *Let X_1 and X_2 be arbitrary disjointed subsets of X with $X_1 \cup X_2 = X$. If Y and X_1 are conditional independent given X_2, denoted as $Y \perp X_1 | X_2$, then $X \to Y$ is not a minimal non-trivial AFD.*

Proof. *From $Y \perp X_1 | X_2$, we have*

$$P(Y|X_2) = P(Y|X_1, X_2) = P(Y|X) \tag{2}$$

According to Definition 1, the validity of both $X_2 \to Y$ and $X \to Y$ is equivalent. If $X_2 \to Y$ holds, then $X \to Y$ holds but is non-minimal. Conversely, if $X_2 \to Y$ not holds, then $X \to Y$ is invalid. In any situation, $X \to Y$ is not a minimal non-trivial AFD. We refer to such AFDs as excludable AFDs (eAFDs), while other AFDs are cAFDs.

Theorem 1 allows us to identify eAFDs using CI tests. However, it only necessitates that X_1 and X_2 serve as two partitions of X, containing $(|X|^2)-2$ potential partitioning ways. Conducting CI tests for all potential partitions is computationally infeasible. Therefore, we introduce the following theorem to decrease the number of CI tests to $|X|$.

Theorem 2. *Let X_1 and X_2 be arbitrary disjointed subsets of X with $X_1 \cup X_2 = X$. $\forall A \in X_1$, $Y \perp X_1 | X_2 \iff Y \perp A | X \setminus \{A\}$.*

Proof. ($\Longleftarrow$) *Since the right side is a special case of the left side, holds clearly.*
($\Longrightarrow$) *According to the decomposition property of CI [Pearl, 1988], we have*

$$\begin{aligned} Y \perp X_1 | X_2 &\Longrightarrow Y \perp X_1 - A | X_2 \\ &\Longrightarrow P(Y|X_2) = P(Y|X \setminus \{A\}) \end{aligned} \tag{3}$$

Since $Y \perp X_1 | X_2 \Longrightarrow P(Y|X_2) = P(Y|X)$, we obtain

$$P(Y|X) = P(Y|X \setminus \{A\}) \Longrightarrow Y \perp A | X \setminus \{A\}. \tag{4}$$

Theorem 2 indicates that to determine whether $X \to Y$ is excludable, we simply need to verify whether there exists $A \in X$ such that $Y \perp A | X \setminus \{A\}$ holds. To reduce the time complexity of existing CI test methods, we propose a novel graph structure, called dependency exclusion graph (DEG), which makes it possible to determine if multiple AFDs are excludable by building the DEG once. Given the attribute set $Z \subseteq R$, its DEG $\mathcal{G}_d(Z)$ is built by the following steps:

- **Construct the ancestral graph** $\mathcal{G}_a(Z)$: Trim $\mathcal{G}$ onto $Z \cup an(Z)$, where $an(Z)$ represents the ancestor nodes of Z.
- **Construct the moral graph** $\mathcal{G}_m(Z)$: Add undirected edges between parents sharing a common child in $\mathcal{G}_a(Z)$, then convert all directed edges to undirected.
- **Construct the DEG** $\mathcal{G}_d(Z)$: delete each attribute in $an(Z)$ sequentially and add undirected edges between its neighboring nodes.

Theorem 3 gives the idea to determine whether each $\{Z \setminus \{Y\} \to Y \mid Y \in Z\}$ is an eAFD using $\mathcal{G}_d(Z)$.

Theorem 3. *Let $Z \subseteq R$, and $Y \in Z$. If Y is not fully connected to other nodes in $\mathcal{G}_d(Z)$, then $Z \setminus \{Y\} \to Y$ is an eAFD.*

Proof. *First, according to Theorem 1 and Theorem 2, if there exists $A \in Z \setminus \{Y\}$ such that $Y \perp A | Z \setminus \{Y, A\}$ holds, then $Z \setminus \{Y\} \to Y$ is an eAFD.*

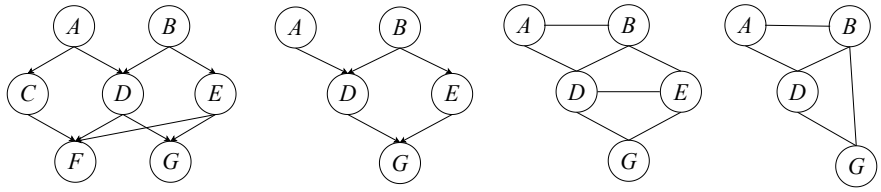
Second, we construct three graphs: (1) $\mathcal{G}_m(Z)$, the moral graph of Z; (2) $\mathcal{G}_m{}'(Z)$, obtained by pruning $\mathcal{G}_m(Z)$ onto $Y \cup A \cup an(Z)$. $\mathcal{G}_m{}'(Z)$ only retains $Y \cup A \cup an(Z)$ and their edges; (3) $\mathcal{G}_d(Z)$, the DEG, created by adding edges between nodes connected through $an(Z)$ and removing $an(Z)$ in $\mathcal{G}_m(Z)$.

Third, based on M-separation, if there is no path between Y and A in $\mathcal{G}_m{}'(Z)$, then $Y \perp A | Z \setminus \{Y, A\}$ holds. In this case, Y and A are not directly connected and are not indirectly connected through $an(Z)$ in $\mathcal{G}_m(Z)$. That is, Y and A are not directly connected in $\mathcal{G}_d(Z)$. Since A can be any node in $Z \setminus \{Y\}$, if Y is not directly connected with all the other nodes in $\mathcal{G}_d(Z)$, then $Z \setminus \{Y\} \to Y$ is an eAFD.

Last, the DEG remains consistent for each $Y \in Z$. We can deduce that for every $Y \in Z$, if Y is not fully connected to other nodes in $\mathcal{G}_d(Z)$, then $Z \setminus \{Y\} \to Y$ is an eAFD.

For example, Figure 2 provides an example of identifying eAFDs using DEG, where $Z = \{A, B, D, G\}$ and Figure 2 (d) shows the DEG $\mathcal{G}_d(Z)$. Since A and G are not fully connected to other nodes, $\{B, D, G\} \to A$ and $\{A, B, D\} \to G$ are excludable.

For each MB in $\mathcal{G}$, we generate all attribute subsets with more than one attribute and construct a DEG for each subset. Then, we prune the non-minimal dependencies and invalid



(a) BN structure (b) Ancestral graph (c) Moral graph (d) DEG

Figure 2: An example of identifying eAFDs using DEG.

Algorithm 1 Structure Pruning

Input: $\mathcal{G}$, the BN structure; $\mathbb{S}$, the MBs in $\mathcal{G}$.
Output: $\mathbb{C}$, the cAFDs in the search space.

```
ℂ ← {}, ℤ' ← {} // ℤ' is the set of processed attributes.
for each S in {𝕊} do
    ℂ[S] ← [ ]
    Generate a Z ⊆ S in descending order of attribute
     count and store them in ℤ
    for each Z in ℤ do
        if |Z| > 1 and Z ⊈ ℤ' then
            ℤ' ← ℤ' ∪ {Z}
            build DEG 𝒢_d(Z)
            for each B in Z do
                if B is fully connected to other nodes in
                𝒢_d(Z) then
                    ℂ[S].append(Z \ {Y} → Y)
                end if
            end for
        end if
    end for
end for
return ℂ
```

dependencies (see Algorithm 1) based on Theorem 3. The time complexity of constructing $\mathcal{G}_d(Z)$ is $O(m^2)$ in the worst case, and thus the time complexity of determining whether a possible AFD can be excluded is just $O(m^2/|Z|)$. The time complexity of SMI score calculation is at least $O(n \times |Z|)$, where n is the sample count. By excluding a large number of possible AFDs with a lower computational cost, we can obtain candidate dependencies.

3.3 AFD SEARCH

We next search the cAFDs with the highest SMI score as outputs. Considering the expensive process of SMI score calculation, we propose a branch-and-bound algorithm and count generations to respectively reduce the number of SMI score calculations and lower the single calculation cost.

Branch-and-bound algorithm for AFD discovery. First, we derive an upper bound for the SMI score. Given the attribute set $X \subseteq R$ and attribute $Y \in R$, the SMI score between any subset $X' \subseteq X$ and Y is always not larger than the upper bound. That is

$$\forall X' \subseteq X \implies \widehat{s}(X;Y) \geq s(X';Y) \tag{5}$$

where $s(X';Y)$ is the SMI score of $X' \to Y$, and $\widehat{s}(X;Y)$ is the upper bound, defined as follows

$$\begin{aligned}\widehat{s}(X;Y) = &-\sum_{y \in V(Y)} \mathrm{x}\log\mathrm{x}\left(\frac{n_y + N_X\alpha}{n + N_X N_Y \alpha}\right) \\ &- \frac{n}{n + N_X N_Y \alpha} H(Y|X)\end{aligned} \tag{6}$$

where $\mathrm{x}\log\mathrm{x}(\cdot)$ is a simplified representation of $(\cdot)\log(\cdot)$, n is the sample size, n_y is the sample count with a value of y on Y, N_X is the size of $V(X)$, $H(Y|X)$ is the conditional entropy, and α is the pseudocount in the SMI score.

Since the sample counts required for $s(X';Y)$ and $\widehat{s}(X;Y)$ are consistent, $\widehat{s}(X;Y)$ requires almost no additional computational cost.

Then, we prove the validity of the upper bound, where the formula of $s(X';Y)$ is as follows

$$\begin{aligned}s(X';Y) &= \tilde{H}_{X'Y}(Y) - \tilde{H}_{X'Y}(Y|X') \\ \tilde{H}_{X'Y}(Y) &= -\sum_{y \in V(Y)} \mathrm{x}\log\mathrm{x}\left(\tilde{P}(y)\right) \\ -\tilde{H}_{X'Y}(Y|X') &= \sum_{X'y \in V(X'Y)} \tilde{P}(X'y)\log\tilde{P}(y|X') \\ \tilde{P}(y) &= (n_y + N_{X'}\alpha)/(n + N_{X'}N_Y\alpha) \\ \tilde{P}(X'y) &= (n_{X'y} + \alpha)/(n + N_{X'}N_Y\alpha) \\ \tilde{P}(y|X') &= (n_{X'y} + \alpha)/(n_{X'} + N_Y\alpha).\end{aligned}$$

We calculate the partial derivative of $\tilde{H}_{X'Y}(Y)$ w.r.t. $N_{X'}$ as follows

$$\begin{aligned}\frac{\partial \tilde{H}_{X'Y}(Y)}{\partial N_{X'}} = &\frac{\alpha}{(n + N_{X'}N_y\alpha)^2} \sum_{y \in V(Y)} [(n - N_Y n_y) \\ &\times \left(\log\left(\frac{1}{\tilde{P}(y)}\right) + \frac{1}{\ln 2}\right)]\end{aligned}$$

Algorithm 2 AFD Discovery

Input: D, the dataset; $\mathbb{C}$, the cAFDs; $\mathbb{S}$, the MBs.
Output: $\mathcal{F}$, the discovered AFDs.

$\mathcal{F} \leftarrow \{\}, \mathcal{I} \leftarrow \{\}$ //$\mathcal{I}$ is the highest SMI score currently
for each A in R **do**
  count n_A from D // $n_A = \{n_a | \forall a \in V(A)\}$
  $\mathcal{I}[A] \leftarrow 0$
end for
for each S in $\mathbb{S}$ **do**
  count n_S from D
  $\mathcal{N}_0[S] \leftarrow n_S, \mathcal{N}_1 \leftarrow \{\}, l \leftarrow |S|$
  while $\mathbb{C}[S] \neq \emptyset$ **do**
    pop the first cAFD $X \rightarrow Y$ from $\mathbb{C}[S]$ with the most attributes
    $n_{XY}, n_X \leftarrow$ *generate_counts*$(X, Y, \mathcal{N}_0, \mathcal{N}_1, l)$
    calculate $s(X;Y)$ and $\widehat{s}(X;Y)$ using n_{XY}, n_X and n_Y
    if $\widehat{s}(X;Y) \leq \mathcal{I}[Y]$ **then**
      pop all cAFDs $X' \rightarrow Y$ from $\mathbb{C}[S]$ $(X' \subseteq X)$
      continue
    end if
    if $s(X;Y) > \mathcal{I}[Y]$ **then**
      $\mathcal{I}[Y] \leftarrow s(X;Y)$
      $\mathcal{F}[Y] \leftarrow X$
    end if
  end while
end for
return $\mathcal{F}$

where $\sum_{y \in V(Y)} (n - N_Y n_y) = 0$. Since $\log\left(\frac{1}{\tilde{P}(y)}\right) < (\text{or} >) \log N_Y$ when $n_y > (\text{or} <) \frac{n}{N_Y}$, and $\sum_{y \in V(Y)} (n - N_Y n_y) \log N_Y = 0$, we have

$$\frac{\partial \tilde{H}_{X'Y}(Y)}{\partial N_{X'}} \geq 0 \tag{7}$$

If $X' = X$, then $N_{X'}$ reaches its maximum value N_X. Based on the monotonicity, we can obtain

$$\forall X' \subseteq X, \tilde{H}_{X'Y}(Y) \leq \tilde{H}_{XY}(Y) \tag{8}$$

$\forall X'y \in V(X'Y), \tilde{P}(X'y) \log \tilde{P}(y|X') \leq 0$, we have

$$\begin{aligned} &-\tilde{H}_{X'Y}(Y|X') \leq \sum_{X'y \in V'(X'Y)} \tilde{P}(X'y) \log \tilde{P}(y|X') \\ &= \sum_{X' \in V'(X')} \frac{n_{X'} + N_Y \alpha}{n + N_{X'} N_Y \alpha} \sum_{y \in V'_{X'}(Y)} \mathrm{x} \log \mathrm{x} \left(\tilde{P}(y|X')\right) \end{aligned} \tag{9}$$

where $V'(X')$ denotes the value of samples on X', and $V'_{X'}(Y)$ denotes the value of samples on Y when X equals to X'. Similar to Equation 9, we have

$$\begin{aligned} &-\tilde{H}_{X'Y}(Y|X') \\ &\leq -\sum_{X' \in V'(X')} \frac{n_{X'} + N_Y \alpha}{n + N_{X'} N_Y \alpha} H(Y|X' = X') \\ &\leq -\frac{n}{n + N_{X'} N_Y \alpha} H(Y|X') \end{aligned} \tag{10}$$

Algorithm 3 Count Generation

Input: D, the dataset; $\mathbb{C}[S]$, the cAFDs corresponding to S; n_S, the count of S; X, the set of attributes; Y, an attribute; l, the current search level; $\mathcal{N}_0$, $\mathcal{N}_1$, the sample count for the attribute sets containing l and $l-1$ attributes.
Output: n_{XY}, the sample count for XY ; n_X, the sample count for X.

if $|XY| \neq l$ **then**
  $l \leftarrow |XY|$
  for each $X' \rightarrow Y'$ in $\mathbb{C}[S]$ **do**
    if $l > |X'Y'|$ **then**
      break
    end if
    if $n_{X'Y'} \notin \mathcal{N}_1$ **then**
      if $\exists n_Z \in \mathcal{N}_0$ $(or\ \mathcal{N}_1)$ $s.t.\ X'Y' \subseteq Z$ **then**
        count $n_{X'Y'}$ from n_Z
      else
        count $n_{X'Y'}$ from n_S
      end if
      $\mathcal{N}_1[X'Y'] \leftarrow n_{X'Y'}$
    end if
  end for
  $\mathcal{N}_0 \leftarrow \mathcal{N}_1, \mathcal{N}_1 \leftarrow \{\}$
end if
$n_{XY} \leftarrow \mathcal{N}_0[XY]$
if $n_X \notin \mathcal{N}_1$ **then**
  count n_X from n_{XY}
  $\mathcal{N}_1[X] \leftarrow n_X$
else
  $n_{\boldsymbol{X}} \leftarrow \mathcal{N}_1[\boldsymbol{X}]$
end if
return n_{XY}, n_X

According to the monotonicity, we have

$$\forall X' \subseteq X, -\tilde{H}_{X'Y}(Y|X') \leq -\frac{n}{n + N_X N_Y \alpha} H(Y|X) \tag{11}$$

Based on Equation 8 and Equation 11, we obtain the validity of the upper bound.

Ultimately, we develop a branch-and-bound algorithm for finding minimal non-trivial AFDs. Within each sub-search space, we calculate the SMI scores and upper bounds of the cAFDs in descending order of the attribute count. If the current upper bound $\widehat{s}(X;Y)$ is lower than the known highest SMI score, then we prune all $X' \rightarrow Y$ $(X' \subseteq X)$ in the search space (see Algorithm 2). Since the SMI scores and their upper bounds share the same sample counts, Algorithm 2 incurs almost no additional cost. Compared to traversing all cAFDs, this pruning technique significantly reduces the number of SMI score calculations.

Count generation. To reduce the cost of counting the samples from data for calculating SMI scores, we store

and reuse the previously obtained sample counts to prevent redundant counting. Specifically, we use $\mathcal{N}_0$ and $\mathcal{N}_1$ to alternately store the sample counts. The attribute count of current cAFD is defined as the search level, represented by l. Here, $\mathcal{N}_0$ stores the sample count for the attribute sets containing l attributes, and $\mathcal{N}_1$ stores the sample count for the attribute sets containing $l-1$ attributes. We prioritize reusing previously stored sample counts to regenerate them when needed (see Algorithm 3). By trading a small amount of additional memory usage, our approach avoids redundant counting and thus reduces the overall computational cost of calculating the SMI score.

4 EXPERIMENTAL STUDY

4.1 EXPERIMENTAL SETUP

Datasets. We select 6 public and 10 synthetic datasets with ground-truth AFDs to evaluate the effectiveness of our method. The statistics of these datasets are summarized in Table 2.

Dataset	# Attributes	# Edges	# Samples	# AFDs
Earthquake	5	4	5,000	3
Cancer	5	4	5,000	3
Asia	8	8	5,000	5
Insurance	27	52	5,000	18
Water	32	66	5,000	18
Alarm	37	46	5,000	24

Table 2: Statistics of public datasets.

For public datasets, since the dependencies between attributes are well-defined, we select six datasets including Cancer, Earthquake, Asia, Insurance, Water, and Alarm. We generate 5,000 samples as the dataset by forward sampling on each BN.

For synthetic datasets, we first set the number of attributes and edges, and then construct a Erdős-Rényi (ER) random graph [Erdos and Rényi, 1960]. The attributes without incoming edges are assigned by random values ranging from 0 to 4. We finally assign values to the remaining attributes following the topological order, with the values being primarily determined by a function of their parents, $p(A = f(\boldsymbol{o})|pa(A) = \boldsymbol{o}) = 0.8$, where A and $pa(A)$ is an attribute and its parents, respectively. The AFDs can be obtained from the ER random graph corresponding to the situation of each dataset. Each synthetic dataset contains 5,000 samples with different attributes and edges, i.e., {a10e10, a15e15, a20e20, a25e25, a30e30, a60e60, a65e65, a70e70, a75e75, a80e80}, where a10e10 means that the dataset including 10 attributes and 10 edges.

To evaluate the impact of dataset size on the efficiency, we generate synthetic datasets by varying MMB size and the number of attributes. Specifically, we first randomly generate BN structures along with their conditional probability tables, and then select the BNs whose MMB size and attribute numbers meet the requirements. We also generate 5,000 samples as the dataset by forward sampling on each synthetic BN. Actually, the MMB sizes on these datasets are {3, 5, 7, 9, 11} and the attribute numbers are {5, 10, 15, 20, 25}.

Comparison methods. We carefully choose the following five methods for comparison:

- FDX [Zhang et al., 2020] is a structure-based method and transforms FD discovery into a structure learning problem over a linear structured equation model.
- RFI [Mandros et al., 2017] is a score-based method and finds AFDs using the score that adjusts the normalized mutual information by subtracting expected values under the hypothesis of independence.
- SMI [Pennerath et al., 2020] is a score-based method and discovers AFDs using the score that corrects the mutual information through Laplacian smoothing.
- PYRO [Kruse and Naumann, 2018] is a threshold-based method and combines a separate-and-conquer search strategy with sampling-based guidance to quickly detect and verify candidates.
- TANE [Huhtala et al., 1999] is a classical FD discovery method and can be used to find AFDs by setting an error threshold.

Metrics and implementation. The effectiveness of the AFD discovery method is measured by precision (P), recall (R), and $F1$ score.

- **Precision** measures the accuracy of AFD discovery and is the mean proportion of the correct attributes in the left-hand side of the discovered AFDs, defined as

$$P = E_d\left(\frac{|X \cap X^*|}{|X|}\right) \tag{12}$$

  where X represents the left-hand side of the discovered AFDs, whose ground-truth is X^* and $E_d(\cdot)$ denotes the mean value for all discovered AFDs.
- **Recall** measures the completeness of AFD discovery and is the mean proportion of the discovered attributes in the left-hand side of ground-truth AFDs, defined as

$$R = E_t\left(\frac{|X \cap X^*|}{|X^*|}\right) \tag{13}$$

  where $E_t(\cdot)$ denotes the mean value for all the ground-truth AFDs.
- $F1$ **score** is defined as $2PR/(P+R)$.

Dataset	Metric	BNAFD	FDX	RFI	SMI	PYRO	TANE
Earthquake	P	0.5000	0.5000	0.3667	0.2000	0.1667	0.0000
	R	1.0000	1.0000	1.0000	1.0000	0.3333	0.0000
	$F1$	**0.6667**	**0.6667**	0.5366	0.3333	0.2222	0.0000
	# AFDs	5	4	5	5	6	0
Cancer	P	0.5000	0.5000	0.4000	0.2000	0.3333	0.0000
	R	1.0000	0.6667	1.0000	1.0000	0.1667	0.0000
	$F1$	**0.6667**	0.5714	0.5714	0.3333	0.2222	0.0000
	# AFDs	5	4	5	5	3	0
Asia	P	0.4286	0.2381	0.2917	0.3036	0.1296	0.5000
	R	0.6000	0.4000	0.6000	1.0000	0.4000	0.1000
	$F1$	**0.5000**	0.2985	0.3925	0.4658	0.1958	0.1667
	# AFDs	7	7	8	8	27	2
Insurance	P	0.3944	0.4375	-	0.3327	0.0287	0.0708
	R	0.6944	0.1759	-	0.7407	0.8796	0.3704
	$F1$	**0.5031**	0.2509	-	0.4592	0.0556	0.1189
	# AFDs	27	16	-	27	743346	5327
Water	P	0.2776	0.3854	-	-	0.0294	-
	R	0.3491	0.1380	-	-	0.6852	-
	$F1$	**0.3092**	0.2032	-	-	0.0563	-
	# AFDs	26	16	-	-	462797	-
Alarm	P	0.4093	0.4236	-	-	-	-
	R	0.8576	0.4167	-	-	-	-
	$F1$	**0.5541**	0.4201	-	-	-	-
	# AFDs	36	24	-	-	-	-

Table 3: Comparison of effectiveness on public BN datasets. The best results are highlighted in **boldface**.

Additionally, the running time of each method is recorded to evaluate the efficiency. For public datasets, we use the state-of-the-art exact BN structure solver GOBNILP [Cussens et al., 2017] to learn BN structures with the convergence parameter set to 0.01. For synthetic datasets, we use the efficient continuous optimization method DAGMA [Bello et al., 2022] to achieve better scalability.

All experiments are conducted on a machine with Intel i9 13900KF CPU and 128GB RAM, running Windows 11 operation system.

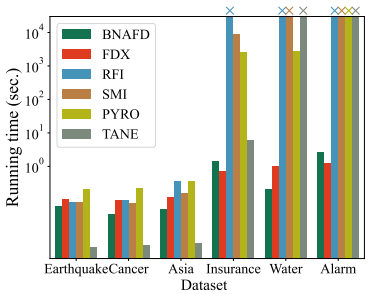


Figure 3: Comparison of efficiency on public datasets.

4.2 EXPERIMENTAL RESULTS

Effectiveness and efficiency evaluation on public datasets. We evaluate the effectiveness and efficiency of our BNAFD by comparing with other methods. For fairness, each method receives identical inputs and the running time is limited to 30,000 seconds. Table 3 reports the precision, recall, $F1$ score, and the number of AFDs discovered by each method, and Figure 3 shows the corresponding running time. Overall, BNAFD consistently outperforms other methods in terms of effectiveness and efficiency.

Effectiveness evaluation on synthetic datasets. We evaluate the effectiveness of our BNAFD by comparing with other methods. For large datasets, we only present the results of BNAFD and FDX, since the running time of the other methods exceed 30,000 seconds. For each configuration, we conduct the tests for 5 times and report the average result and variance. The precision, recall, and $F1$ score results are reported in Table 4 and Table 5, which demonstrate that our method BNAFD achieves the best overall performance on synthetic datasets.

Robustness evaluation on noisy datasets. We evaluate the robustness of our BNAFD by comparing with other methods under different noise rates. Table 6 and Table A.1 summarize the $F1$ score of each method. The results demon-

Dataset	Metric	BNAFD	FDX	RFI	SMI	PYRO	TANE
a10e10	P	**0.5911** $\pm$0.0013	0.5111$\pm$0.0054	0.3267$\pm$0.0072	0.4300$\pm$0.0066	0.1240$\pm$0.0001	0.1048$\pm$0.0019
	R	0.8714$\pm$0.0126	0.6339$\pm$0.0230	0.5000$\pm$0.0158	0.5964$\pm$0.0159	**1.0000**$\pm$0.0000	0.5679$\pm$0.0016
	$F1$	**0.7026**$\pm$0.0032	0.5635$\pm$0.0106	0.3944$\pm$0.0101	0.4987$\pm$0.0091	0.2205$\pm$0.0003	0.1741$\pm$0.0038
a15e15	P	0.5634$\pm$0.0056	**0.5741**$\pm$0.0017	0.2067$\pm$0.0072	0.3822$\pm$0.0063	0.0743$\pm$0.0000	0.0661$\pm$0.0012
	R	**0.8496**$\pm$0.0050	0.5052$\pm$0.0031	0.3996$\pm$0.0202	0.6129$\pm$0.0030	0.9929$\pm$0.0002	0.6416$\pm$0.0190
	$F1$	**0.6771**$\pm$0.0059	0.5344$\pm$0.0010	0.2691$\pm$0.0102	0.4691$\pm$0.0058	0.1382$\pm$0.0000	0.1185$\pm$0.0034
a20e20	P	0.5422$\pm$0.0029	**0.7064**$\pm$0.0032	0.2992$\pm$0.0009	0.4033$\pm$0.0008	0.0530$\pm$0.0000	0.0592$\pm$0.0004
	R	0.8320$\pm$0.0067	0.5179$\pm$0.0048	0.5254$\pm$0.0019	0.6230$\pm$0.0007	**0.9920**$\pm$0.0001	0.6288$\pm$0.0085
	$F1$	**0.6559**$\pm$0.0038	0.5963$\pm$0.0040	0.3801($\pm$0.0009	0.4891$\pm$0.0006	0.1005$\pm$0.0000	0.1079$\pm$0.0011
a25e25	P	0.5851$\pm$0.0003	**0.6347**$\pm$0.0027	0.2827$\pm$0.0043	0.4107$\pm$0.0002	0.0404$\pm$0.0000	0.0406$\pm$0.0001
	R	0.8812$\pm$0.0005	0.4165$\pm$0.0087	0.5409$\pm$0.0164	0.6453$\pm$0.0007	**0.9815**$\pm$0.0002	0.5792$\pm$0.0028
	$F1$	**0.7029**$\pm$0.0001	0.4999$\pm$0.0063	0.3710$\pm$0.0074	0.5017$\pm$0.0003	0.0776$\pm$0.0000	0.0757$\pm$0.0002
a30e30	P	0.5473$\pm$0.0025	**0.7281**$\pm$0.0124	0.3272$\pm$0.0012	0.3878$\pm$0.0003	0.0330$\pm$0.0000	0.0359$\pm$0.0001
	R	0.8254$\pm$0.0035	0.4435$\pm$0.0018	0.5935$\pm$0.0059	0.6446$\pm$0.0015	**0.9778**$\pm$0.0004	0.6257$\pm$0.0095
	$F1$	**0.6577**$\pm$0.0027	0.5494$\pm$0.0033	0.4216$\pm$0.0023	0.4837$\pm$0.0003	0.0638$\pm$0.0000	0.0678$\pm$0.0004

Table 4: Comparison of effectiveness on small synthetic datasets. The best results are highlighted in **boldface**.

Dataset	Metric	BNAFD	FDX
a60e60	P	0.5456$\pm$0.0001	**0.7355**$\pm$0.0029
	R	**0.8248**$\pm$0.0031	0.4115$\pm$0.0022
	$F1$	**0.6562**$\pm$0.0005	0.5261$\pm$0.0021
a65e65	P	0.5764$\pm$0.0006	**0.6489**$\pm$0.0012
	R	**0.8823**$\pm$0.0004	0.3821$\pm$0.0011
	$F1$	**0.6967**$\pm$0.0003	0.4800$\pm$0.0009
a70e70	P	0.5428$\pm$0.0003	**0.6247**$\pm$0.0039
	R	**0.8420**$\pm$0.0029	0.3566$\pm$0.0023
	$F1$	**0.6596**$\pm$0.0006	0.4533$\pm$0.0029
a75e75	P	0.5854$\pm$0.0004	**0.6667**$\pm$0.0092
	R	**0.8491**$\pm$0.0013	0.3573$\pm$0.0023
	R	**0.6929**$\pm$0.0006	0.4651$\pm$0.0040
a80e80	P	0.5712$\pm$0.0002	**0.7443**$\pm$0.0155
	R	**0.8590**$\pm$0.0008	0.4011$\pm$0.0084
	$F1$	**0.6857**$\pm$0.0001	0.5205$\pm$0.0115

Table 5: Comparison of effectiveness on large synthetic datasets. The best results are highlighted in **boldface**.

Dataset	Noise rate	BNAFD	FDX
a60e60	0	**0.6562**	0.5261
	0.05	**0.6410**	0.2013
	0.01	**0.6967**	0.4565
a70e70	0	**0.6596**	0.4533
	0.05	**0.6546**	0.0995
	0.01	**0.6932**	0.3002
a80e80	0	**0.6857**	0.5205
	0.05	**0.6675**	0.0966
	0.01	**0.4649**	0.3503
a65e65	0.01	**0.6613**	0.5261
	0	**0.6967**	0.4800
	0.05	**0.6877**	0.0668
a75e75	0.01	**0.6606**	0.4122
	0	**0.6929**	0.4651
	0.05	**0.6828**	0.0327
Alarm	0.01	**0.6791**	0.3397
	0	**0.5541**	0.4201
	0.05	**0.3834**	0.0727

Table 6: Comparison of $F1$ score on large noisy datasets. The best results are highlighted in **boldface**.

strate that BNAFD achieves superior robustness and scalability under different noise rates. For more details please refer to A.3 in Appendix.

5 CONCLUSION

We propose a probabilistic semantics guided AFD discovery method, BNAFD, by incorporating CI tests in terms of BN and branch-and-bound pruning. Experimental results demonstrate that high-quality AFDs could be discovered efficiently. Moreover, our method is robust to the noise in data and can guarantee the high precision of discovered FDs, providing a novel idea for the classical problem of FD discovery. By incorporating BN as the preliminary framework, our method is theoretical reasonable.

Since real-world data frequently suffers from missing values, we will investigate mining AFDs from incomplete datasets. Furthermore, applying BNAFD to anomaly detection is another promising direction for practical deployment.

Acknowledgments

This paper was supported by the Joint Key Project of National Natural Science Foundation of China (U23A20298), Key Project of Fundamental Research of Yunnan Province (202401AS070138), and Program of Yunnan Key Laboratory of Intelligent Systems and Computing (202405AV340009). For any correspondence, please refer to Kun Yue.

References

Sami Abu-El-Haija, Joshua V. Dillon, Bahare Fatemi, Kyriakos Axiotis, Neslihan Bulut, Johannes Gasteiger, Bryan Perozzi, and MohammadHossein Bateni. SubMix: Learning to mix graph sampling heuristics. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence* (UAI), pages 1–10, 2023.

Kevin Bello, Bryon Aragam, and Pradeep Ravikumar. Dagma: Learning dags via m-matrices and a log-determinant acyclicity characterization. In *Proceedings of the 35th Advances in Neural Information Processing Systems* (NIPS), volume 35, pages 8226–8239, 2022.

Louenas Bounia and Frédéric Koriche. Approximating probabilistic explanations via supermodular minimization. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence* (UAI), pages 216–225, 2023.

Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. FakeTables: Using GANs to generate functional dependency preserving tables with Bounded Real Data. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (IJCAI), pages 2074–2080, 2019.

James Cussens, David Haws, and Milan Studenỳ. Polyhedral aspects of score equivalence in bayesian network structure learning. *Mathematical Programming*, 164:285–324, 2017.

Marius Eich, Pit Fender, and Guido Moerkotte. Faster plan generation through consideration of functional dependencies and keys. *Proceedings of the VLDB Endowment*, 9 (10):756–767, 2016.

Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1):17–60, 1960.

Peter A Flach and Iztok Savnik. Database dependency discovery: a machine learning approach. *AI Communications*, 12(3):139–160, 1999.

Limor Gultchin, Virginia Aglietti, Alexis Bellot, and Silvia Chiappa. Functional causal Bayesian optimization. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence* (UAI), pages 756–765, 2023.

Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.

Sijia Jiang, Zijing Tan, Jiawei Wang, et al. Guided conditional functional dependency discovery. *Information Systems*, 114:102158, 2023.

Sebastian Kruse and Felix Naumann. Efficient discovery of approximate dependencies. *Proceedings of the VLDB Endowment*, 11(7):759–772, 2018.

Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Efficient discovery of functional dependencies and armstrong relations. In *Proceedings of the 7th International Conference on Extending Database Technology* (EDBT), pages 350–364, 2000.

Panagiotis Mandros, Mario Boley, and Jilles Vreeken. Discovering reliable approximate functional dependencies. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD), pages 355–363, 2017.

Noel Novelli and Rosine Cicchetti. Fun: An efficient algorithm for mining functional and embedded dependencies. In *Proceedings of the International Conference on Database Theory* (ICDT), pages 189–203, 2001.

Carlos Núñez-Molina, Masataro Asai, Pablo Mesejo, and Juan Fernández-Olivares. On using admissible bounds for learning forward search heuristics. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence* (IJCAI), pages 6761–6769, 2024.

Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data* (SIGMOD), pages 821–833, 2016.

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

Frédéric Pennerath, Panagiotis Mandros, and Jilles Vreeken. Discovering approximate functional dependencies using smoothed mutual information. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (SIGKDD), pages 1254–1264, 2020.

Shawn Skyler, Shahaf S. Shperberg, Dor Atzmon, Ariel Felner, Oren Salzman, Shao-Hung Chan, Han Zhang, Sven Koenig, William Yeoh, and Carlos Hernández Ulloa. Theoretical study on multi-objective heuristic search. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence* (IJCAI), pages 7021–7028, 2024.

Ziheng Wei and Sebastian Link. Discovery and ranking of functional dependencies. In *Proceedings of the IEEE 35th International Conference on Data Engineering* (ICDE), pages 1526–1537, 2019.

Ziheng Wei and Sebastian Link. Embedded functional dependencies and data-completeness tailored database design. *ACM Transactions on Database Systems*, 46(2): 1–46, 2021.

Catharine Wyss, Chris Giannella, and Edward Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. In *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery* (DWKD), pages 101–110, 2001.

Hong Yao, Howard J Hamilton, and Cory J Butz. Fd_mine: discovering functional dependencies in a database using equivalences. In *Proceedings of the IEEE International Conference on Data Mining* (ICDE), pages 729–732, 2002.

Yue Yu, Tian Gao, Naiyu Yin, and Qiang Ji. Dags with no curl: An efficient dag structure learning approach. In *Proceedings of the 38th International Conference on Machine Learning* (ICML), pages 12156–12166, 2021.

Yunjia Zhang, Zhihan Guo, and Theodoros Rekatsinas. A statistical perspective on discovering functional dependencies in noisy data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (SIGMOD), pages 861–876, 2020.

Probabilistic Semantics Guided Discovery of Approximate Functional Dependencies (Appendix)

Liang Duan[1,2] **Xinran Wu**[1,2] **Xinhui Li**[1,2] **Lixing Yu**[1,2] **Kun Yue**[1,2]

[1]Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, China
[2]School of Information Science and Engineering, Yunnan University, Kunming, China

A EXPERIMENTAL DETAILS

A.1 EXP-1: EFFECTIVENESS AND EFFICIENCY EVALUATION ON PUBLIC DATASETS

We evaluate the effectiveness and efficiency of our BNAFD by comparing it with other methods. For a fair comparison, each method receives identical inputs, and the running time is limited to 30,000 seconds. Table 3 summarizes the precision, recall, $F1$ score, and the number of AFDs discovered by each method, and Figure 3 shows the corresponding running time. The results tell us that:

- BNAFD achieves the highest $F1$ score and outperforms other methods on all datasets.
- FDX exhibits high precisions but low recalls, and performs well on the two datasets, Cancer and Earthquake, where the left-hand sides of the ground-truth AFDs contain the fewest attributes. This is consistent with our analysis as it tends to find simple AFDs. However, the $F1$ score of FDX is no higher than our method.
- Compared with RFI, BNAFD presents the equivalent recall while improving the precision by 0.1234 on average. Compared with SMI, the average recall of BNAFD decreases somewhat by 0.1116, whereas the average precision increases by 0.1967. The results demonstrate that it is reasonable to limit the search space to MBs and use CI tests to remove non-minimal dependencies.
- The number of AFDs found by BNAFD, FDX, RFI and SMI is at most equal to the number of the anticipated attributes. However, PYRO and TANE find all AFDs that satisfy the given error threshold, which leads to a large number of spurious AFDs.
- Only FDX and BNAFD can finish the tests on all datasets. Furthermore, BNAFD is always faster than RFI and SMI even when the structure learning incurs additional cost.

A.2 EXP-2: EFFECTIVENESS EVALUATION ON SYNTHETIC DATASETS

In this set of tests, we evaluate the effectiveness of our BNAFD by comparing with other methods. We divide the synthetic datasets into two categories: small and large datasets. For the large datasets, we only present the results of BNAFD and FDX, since the running time of the other methods exceed 30,000 seconds. For each configuration, we conduct the tests for 5 times and report the average result and variance. Table 4 and Table 5 summarize and compare the precision, recall, $F1$ score. The results tell us that:

- BNAFD achieves the highest $F1$ score and outperforms other methods on all datasets. The findings demonstrate that BNAFD is not overfitting to the BN datasets, and it can also obtian high-quality AFDs on randomly generated synthetic datasets.
- The performance of all methods is similar to that on public BN datasets, with FDX having higher precision but lower recall, RFI and SMI having higher recall but lower precision, PYRO and TANE obtaining a large number of spurious dependencies, resulting in low precision.

- BNAFD exhibits attribute scalability due to the search space reduction.

A.3 EXP-3: ROBUSTNESS EVALUATION ON NOISY DATASETS

In this set of tests, we evaluate the robustness of our BNAFD by comparing with other methods under different noise rates. To simulate noisy datasets, we randomly alter each value in dataset to another value within its domain at a probability corresponding to the noise rate. For each synthetic dataset, we conduct the tests for 5 times and report the average result. For public datasets, we choose Cancer and Alarm to represent small datasets and large datasets, respectively. Table A.1 and Table 6 summarize the $F1$ score of each method. The results tell us that:

Dataset	Noise rate	BNAFD	FDX	RFI	SMI	PYRO	TANE
a10e10	0	**0.7026**	0.5635	0.3944	0.4987	0.2205	0.1741
	0.01	**0.7026**	0.4737	0.3001	0.4987	0.2195	0.2111
	0.05	**0.6796**	0.4548	0.2216	0.4647	0.2176	0.1535
a15e15	0	**0.6771**	0.5344	0.2691	0.4691	0.1382	0.1185
	0.01	**0.6771**	0.5344	0.2282	0.4691	0.1375	0.1494
	0.05	**0.6611**	0.4923	0.2063	0.4604	0.1416	0.1217
a20e20	0	**0.6559**	0.5963	0.3801	0.4891	0.1005	0.1079
	0.01	**0.6612**	0.5269	0.2810	0.4891	0.1006	0.1109
	0.05	**0.6240**	0.4817	0.2188	0.4728	0.1054	0.0958
a25e25	0	**0.7029**	0.4999	0.3710	0.5017	0.0776	0.0757
	0.01	**0.6954**	0.4999	0.2980	0.5017	0.0785	0.0847
	0.05	**0.6904**	0.4949	0.2454	0.5017	0.0838	0.1019
a30e30	0	**0.6577**	0.5494	0.4216	0.4837	0.0638	0.0678
	0.01	**0.6639**	0.5528	0.2653	0.4840	0.0643	0.0719
	0.05	**0.6504**	0.5429	0.2352	0.4810	0.0678	0.0764
Cancer	0	**0.6667**	0.5714	0.5714	0.3333	0.2222	0.0000
	0.01	**0.6667**	0.5714	0.3562	0.3333	0.4000	0.0000
	0.05	**0.5914**	0.0000	0.3226	0.3333	0.0000	0.0000

Table A.1: Comparison of $F1$ score on small noisy datasets. The best results are highlighted in **boldface**.

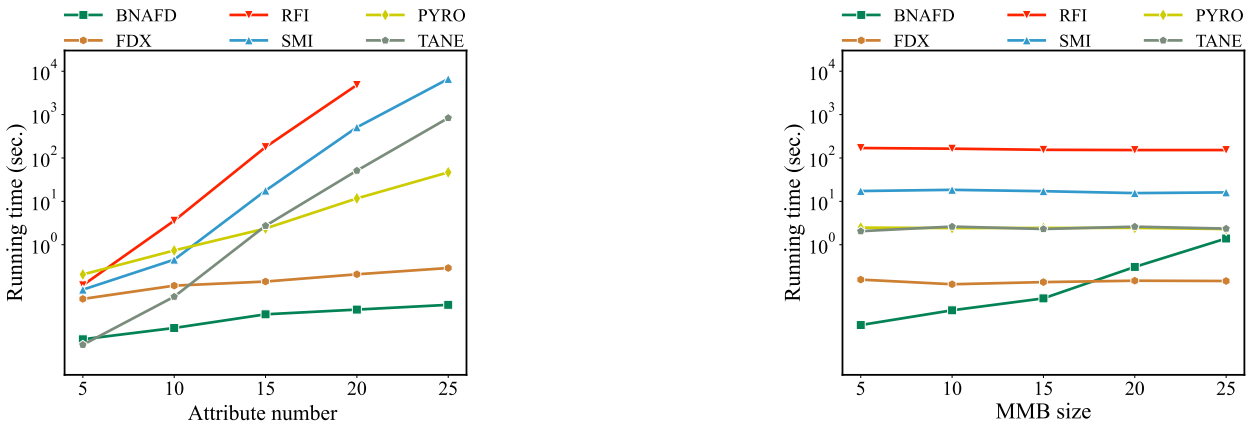
- BNAFD achieves the highest $F1$ score and outperforms other methods under different noise rates.
- For small datasets, all the methods are robust to noise on synthetic datasets. Since Cancer has smaller value domains and is more probably affected by noise, only BNAFD, RFI and SMI maintain the robustness on Cancer.
- For large datasets, only BNAFD exhibits good robustness and attribute scalability.

A.4 EXP-4: IMPACTS OF PARAMETERS ON SYNTHETIC DATASETS

In this set of tests, we evaluate the efficiency of our BNAFD by comparing with other methods under different parameter settings. We vary the attribute number and MMB size independently to evaluate how each parameter affects the running time, which is limited to 30,000 seconds. By varying the parameters of BN structures in BNAFD, the structures of the synthetic datasets are utilized as inputs. Figure A.1 shows the running time of all methods. The results tell us that:

- For a fixed MMB size, our BNAFD exhibits the slowest increase of running time as the number of attributes increases, following a linear growth curve. In contrast, the running time of RFI, SMI, TANE, and PYRO increases exponentially with the number of attributes.
- When the number of attributes is fixed but the MMB size grows, the running time of the comparison methods remains basically constant, while our method grows exponentially. However, BNAFD is consistently faster than RFI and SMI.

These findings show that our BNAFD significantly reduces the time complexity from exponential in the number of attributes to exponential in the MMB size.



(a) Fixing the size of MMB to 5

(b) Fixing the number of attributes to 15

Figure A.1: Impacts of parameters on efficiency.

Dataset	Structure Pruning Branch-and-bound Algorithm Count Generation				
	×	✓	×	×	✓
	×	×	✓	×	✓
	×	×	×	✓	✓
Earthquake	0.2272	0.1104	0.1245	0.0375	**0.0357**
Cancer	0.2164	0.1272	0.1357	0.0394	**0.0377**
Asia	0.7504	0.3805	0.3424	0.0427	**0.0389**
Insurance	257.8796	135.9594	217.7140	1.5280	**0.7881**
Water	31.1385	14.7965	24.4691	0.2540	**0.1705**
Alarm	67.3109	33.3371	48.9376	0.4399	**0.2689**
a30e30	8441.6901	3123.9453	8152.4624	28.8785	**8.6985**

Table A.2: Ablation experiments. The best results are highlighted in **boldface**.

A.5 EXP-5: ABLATION EXPERIMENTS

In this set of tests, we valuate whether our proposed algorithms, structure pruning, branch-and-bound algorithm and count generation, can improve the efficiency. Five distinct methods are explored: employing no algorithms, using each algorithm individually, and combining all three algorithms. These tests are conducted on the public datasets and a synthetic dataset. The running time for structure learning is not recorded, since it is consistent across all the methods. Table A.2 summarizes the running time of all methods. The results tell us that:

- The method combining all three algorithms exhibits the shortest running time on all datasets.
- The methods using any one of these algorithms are faster than the method employing no algorithms.

These indicate that the three proposed algorithms can improve the efficiency of our AFD discovery method.