Discovering conditional Functional Dependencies

Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong
ICDE 2009

Amira Ghenai

UNIVERSITY OF WATERLOO

Outline

- Introduction and Motivation
- Contributions of the paper
- Algorithms description
  - » CFDMiner
  - » CTANE
  - » FastCFD
- Experimental Evaluation
- Summary

Introduction & Motivation

- CFD (as previously discussed) are introduced for data cleaning purposes
- CFDs are more effective than FDs in detecting and repairing inconsistencies

» Unrealistic to rely on human experts to design CFDs via experiments

» *Automatically* discover CFDs

» The discovery problem is highly non-trivial

Example

cust: (country code (CC), area code (AC), phone number (PN)), name (NM), and address (street (STR), city (CT), zip code (ZIP)).

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202

FDs:

$$f_1 : [\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}$$
$$f_2 : [\mathrm{CC}, \mathrm{AC}, \mathrm{PN}] \rightarrow \mathrm{STR}.$$

CFDs:

Variable CFDs

$$\phi_0 : ([\mathrm{CC}, \mathrm{ZIP}] \rightarrow \mathrm{STR}, \boxed{(44, \_ \parallel \_)})$$

(CC,ZIP,STR)

Constant CFDs

$$\phi_1 : ([\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}, (01, 908 \parallel \mathrm{MH}))$$
$$\phi_2 : ([\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}, (44, 131 \parallel \mathrm{EDI}))$$
$$\phi_3 : ([\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}, (01, 212 \parallel \mathrm{NYC}))$$

Main Contributions

- Three algorithms for CFDs discovery:
  1. **CFDMiner**: for discovering constant CFDs only using depth-first search schema
  2. **CTANE**: extension of TANE (presented last week) that uses levelwise approach to discover FDs
  3. **FastCFD**: depth-first approach to discover general CFDs and it's an extension to FastFD.
- Experimental study on real life datasets

Problem Statement

- Minimal CFDs
  - » A minimal CFD is a non-trivial one i.e. *left-reduced*.
  - » A CFD φ=(X→A, t_p) is left-reduced if:
    - None of its LRS attributes can be removed (X)
    - None of the constants in the LHS can be upgraded to "_" i.e. make t_p "most general". (Applied in variable CFDs only)

Problem Statement

- Minimal CFDs Example

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202

$\varphi_3 = ([CC, AC] \rightarrow CT, (01,212 | NYC))$

- Only true for t_3
- Even if we remove CC from LRS, still holds

-> Non-minimal CFD

$\varphi_2 = ([CC, AC] \rightarrow (44,131 || EDI))$

- Constant CFD
- True for t_5 and t_6
- Can't remove CC or AC from LRS

-> Minimal CFD

Problem Statement

- Frequent CFDs
  - » Given CFD φ=(X→A, t_p) in r, there exist a *support* denoted by sup(φ,r) defined as a set of tuples that t[X]≤t_p[X]and t[A] ≤t_p[A].
  - » Example:

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202

$\varphi_1 = ([CC, AC]$

$\rightarrow CT, (01{,}908 || MH))$

→ **3-frequent**

f1: *[CC,AC]→ CT*

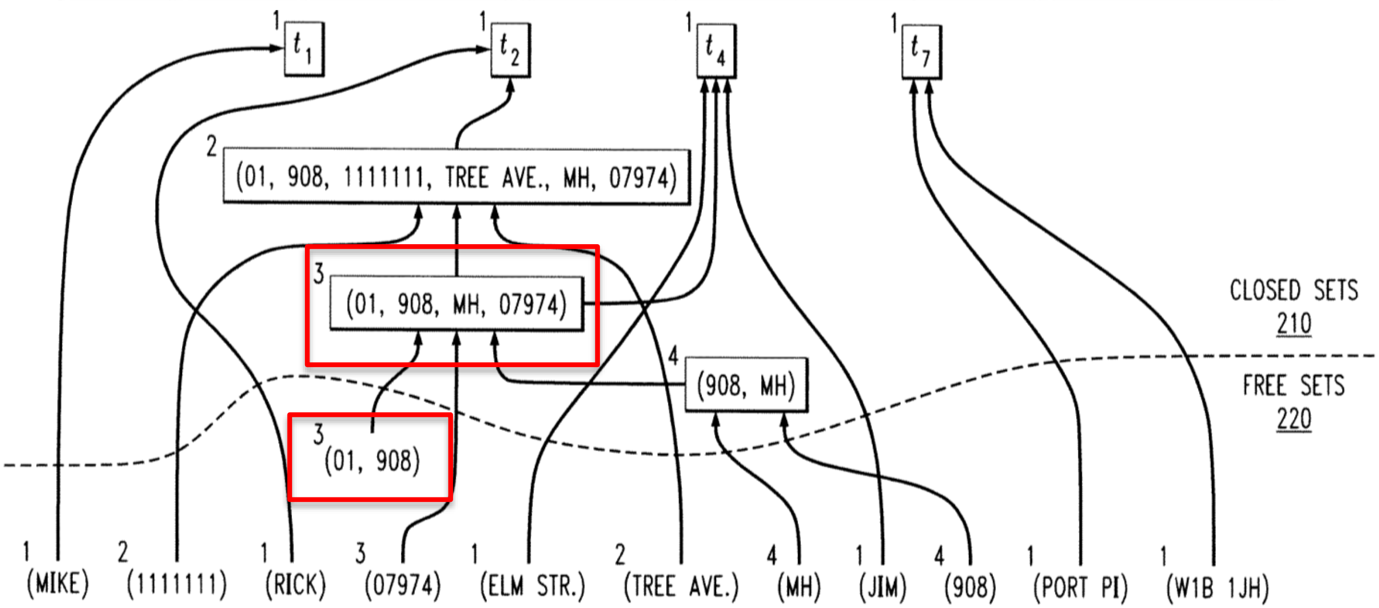
→ **8-frequent**

CFDMiner Algorithm

- Goal: Given an instance *r* of *R* and a support threshold *k*, the algorithm finds a canonical cover of *k*-frequent minimal constant CFDs of the form X→A, $(t_p || a)$
- The algorithm users the notion of **free** and **closed** item sets for a given item set pair (X,t_p):
  - » Closed set: can't be extended without decreasing support
  - » Free set: can't be generalized without increasing support

CFDMiner Algorithm

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202



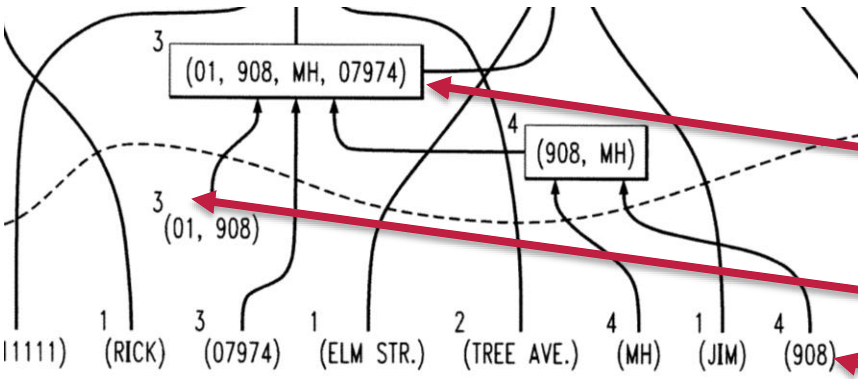
CFDMiner Algorithm

- The relation between free/closed item sets and left-reduced constant CFDs is:
  - » For an instance *r* in *R*, any k-frequent left-reduced constant CFD φ=(X→A, t_p||a) holds iff:

1. Item set (X,t_p) is a free k-frequent set and does not contain (A,a)
2. Item set clo(X,t_p) $\preceq$(A,a) (less general) and
3. (X,t_p) does not contain a smaller free set (Y, s_p) such that
   1. (X,t_p) $\preceq$ (Y,s_p) (i.e (Y,s_p) is *more general*) and
   2. Clo (Y,s_p) $\preceq$ (A,a)

CFDMiner Algorithm

- Example



Closed sets and free sets that contain (CT,MH);
i.e (A,a) = (CT,MH)

$\varphi_1 = ([CC, AC] \rightarrow CT, (01,908||MH))$

1. φ_1 is extracted from 3-constant CFD and matches the **free** item set ([CC,AC],(01,908)) ✓
2. φ_1 contains a **free** item set ([AC],908) which belongs to a **closed** set ([AC,CT],908,MH) which is **more general**

- **Clo (Y,s_p) $\preceq$ (A,a)** ✗

-> **not left-reduced**

CFDMiner Algorithm

1. Get top k-frequent closet item sets (X, t_p) and their corresponding free sets
2. Associate with every free item set (Y, s_p) the RHS (Y, s_p) = (X\Y, t_p[X\Y])
3. An ordered list L will be constructed to keep track of all k-frequent free item sets.
4. For each free item set (Y, s_p) in L:
   a. Replace RHS(Y, s_p) with RHS(Y, s_p)∩ RHS(Y', $s_p[Y']$) where Y'⊈Y.
   b. After checking all subsets, CFDMiner outputs K-frequent CFDs

CTANE Algorithm

- Goal: Levelwise algorithm for discovering minimal k-frequent (variable and constant) CFDs. An extension of TANE algorithm.
- Briefly, the algorithm works as follows:
  1. Compute the RHS for minimal CFDs with their LHS in L_l(where $L_l$is the corresponding level in the lattice)
  2. For each (X, t_p)∈ L_l, we look for CFDs
  3. Prune L_l
  4. Generate next level L_{l+1}
- The following demonstrative example ...

CTANE Algorithm

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202

Assume a support threshold $k \geq 3$ for attributes [CC,AC,ZIP,STR]

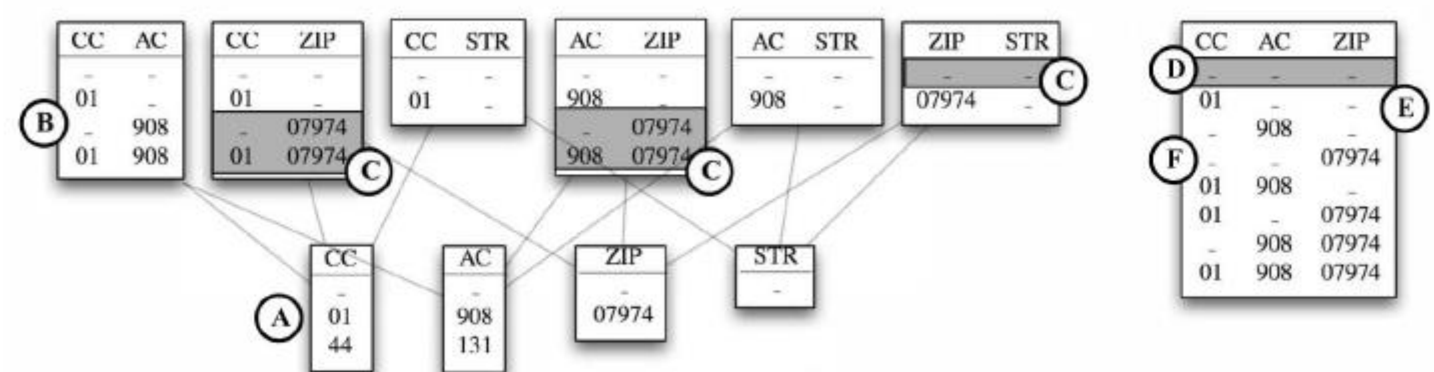


Figure showing two levels of the lattice and partial third level showing [CC,AC,ZIP] attributes

FastCFD Algorithm

- Goal: Find minimal *k*-frequent variable and constant CFDs in a *depth-first* search inspired by FastFD algorithm.
- Key idea: Minimal CFDs are minimal covers of difference sets
- Difference Sets:
  » $D(t_1, t_2; \mathrm{r})$={B∈attr(R)|$t_1[\mathrm{B}] \neq t_2[\mathrm{B}]$}

  $D(t_1, t_2; r_0)$={NM}

  (the set of attributes which are different in t_1 and t_2)

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974

» D_A^r is set {Y\{A}|Y∈ D_r, A ∈Y}

UNIVERSITY OF WATERLOO

FastCFD Algorithm

1. <u>FindCover</u> Algorithm:
   i. Extract the list of k-frequent free item sets in r (A)
   ii. For each item set, produces the minimal difference sets D_A^m (B)
   iii. Calls FindMin to find the minimal cover of D_A^m
2. <u>FindMin Algorithm:</u> (down-left of example)
   i. Orders attributes (alphabetically in example)
   ii. All subsets of attributes are enumerated in a *depth-first*, *left-to-right* fashion.

   Example: for sets{[PN],[AC,CT]}, we can have the possible subsets [AC,PN],[CT,PN]...

   iii. By getting possible subsets, the algorithm verifies if the CFD is minimal.

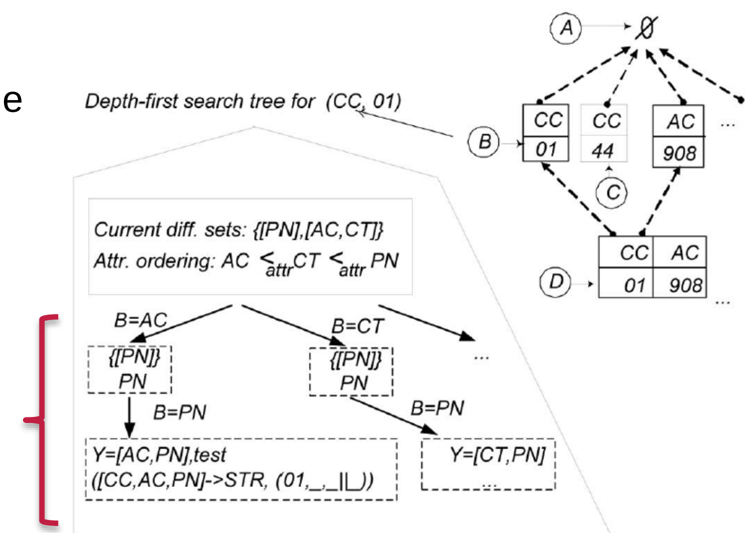
      For example:

      Tree for CC=01 and Y=[AC,PN] and we are looking for STR (input)

      $\varphi' =$ [CC,AC,PN]->STR,(01,-,-||-)

Free pattern $r_{CC=01}$AND $K\geq 2$ for [CC,AC,PN,CT,ZIP,STR]

	CC	AC	PN	NM	STR	CT	ZIP
t_1:	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2:	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3:	01	212	2222222	Joe	5th Ave	NYC	01202
t_4:	01	908	2222222	Jim	Elm Str.	MH	07974
t_5:	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6:	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7:	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8:	01	131	2222222	Sean	3rd Str.	UN	01202



FastCFD Algorithm

Input: $A \in \mathbf{attr}(R)$, $(X, t_p) \in \mathrm{Fr}_k(r)$, $Y \subseteq \mathbf{attr}(R) \setminus \{A\}$, $\mathcal{D}^m_A(r_{t_p})[Y]$, and $<_{\mathbf{attr}}$.

Output: Minimal CFDs $\varphi = ([X, Y] \to A, (t_p, \_, \ldots \_ \parallel t_a))$, where t_a is a constant or "_".

<u>**Base case**</u>:

1) If $\emptyset \in \mathcal{D}^m_A(r_{t_p})[Y]$, then return an empty set. By Lemma 4, $([X, Y], (t_p, \_, \ldots, \_))$ can never lead to a valid CFD.
2) If Y contains the last attributes in $\mathbf{attr}(R) \setminus \{A\}$ *w.r.t.* $<_{\mathbf{attr}}$, but $\mathcal{D}^m_A(r_{t_p})[Y] \neq \emptyset$, then return an empty set. By Lemma 4, $r \not\models ([X, Y] \to A, (t_p, \_, \ldots, \_ \parallel \_))$ because Y does not cover $\mathcal{D}^m_A(r_{t_p})$; moreover, since $([X, Y], (t_p, \_, \ldots, \_))$ cannot be further extended, this pattern does not lead to a valid CFD.
3) If $\mathcal{D}^m_A(r_{t_p})[Y] = \emptyset$, then Y is a cover of $\mathcal{D}^m_A(r_{t_p})$. There are two cases to consider corresponding to the conditions (a) and (b1-b2).
   a) If $\mathcal{D}^m_A(r_{t_p}) = \emptyset$, then by Lemma 4, there exists a constant t_a, $r \models (X \to A, (t_p \parallel t_a))$. In order to check for minimality, we need to verify whether there is no $X' \subset X$ of size $|X| - 1$ such that $r \models (X' \to A, (t_p[X'] \parallel t_a))$. If this holds, then output *constant* CFD $(X \to A, (t_p \parallel t_a))$.
   b) If $\mathcal{D}^m_A(r_{t_p}) \neq \emptyset$, then Lemma 4 implies that $r \models ([X, Y] \to A, (t_p, \_, \ldots, \_ \parallel \_))$. In order to check for minimality, we need to verify whether
      i) there is no $Y' \subset Y$ of size $|Y| - 1$ such that Y' covers $\mathcal{D}^m_A(r_{t_p[X]})$,
      ii) there is no $X' \subset X$ of size $|X| - 1$ such that $Y \cup (X \setminus X')$ covers $\mathcal{D}^m_A(r_{t_p[X']})$.

      If conditions 1) and 2) are both satisfied, then output *variable* CFD $([X, Y] \to A, (t_p, \_, \ldots, \_ \parallel \_))$.

<u>**Recursive case**</u>:

4) For each attribute B that appears after Y *w.r.t.* $<_{\mathbf{attr}}$, we do the following:
   a) Let $Y' = Y \cup \{B\}$ and $\mathcal{D}^m_A(r_{t_p})[Y']$ be the difference sets of $\mathcal{D}^m_A(r_{t_p})[Y]$ not covered by B.
   b) Call $\mathrm{FindMin}(A, (X, t_p), Y', \mathcal{D}^m_A(r_{t_p})[Y'], <_{\mathbf{attr}})$ recursively following the depth-first strategy.

Frk: list of free sets, D: minimal difference set, A: attributes in R

Conditions of checking whether a CFD is valid or no or whether it is minimal or not

FastCFD Algorithm

- Differences compared to FastFD:
  - » More complicated (constants, unnamed variables)
  - » K-frequent CFDs instead of 1-frequent FD
    - Needs efficient way of computing sets
  - » **NaiveFast Algorithm:** Stripped partition-passed, Naïve and fast approach
  - » **FastCFD Algorithm**:
    - Considering the 2-frequent closed item sets only in r which will be computed by CFDMiner algorithm.
    - Difference set can be computed more efficiently
  - » Reorder attributes such that ones that cover most of the sets are treated first to improve efficiency.

Experimental Evaluation

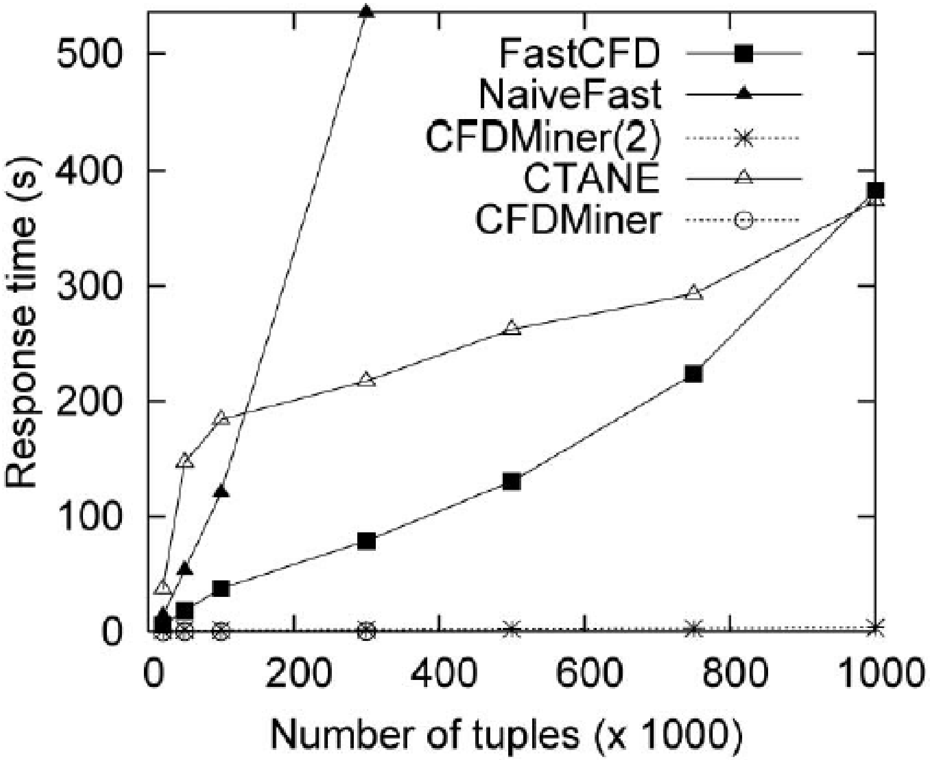
- Two Real-life datasets and a synthetic dataset:

Dataset	Arity	Size (# of tuples)
Wisconsin breast cancer (WBC)	11	699
Chess	7	28,056
Tax	14	20,000

- The experiments studied the effect of:
  » The support threshold *k*
  » The number of tuples DBSIZE
  » The number of columns (Arity)
  » The *correlation factor* (average range of distinct values in an attribute domain)

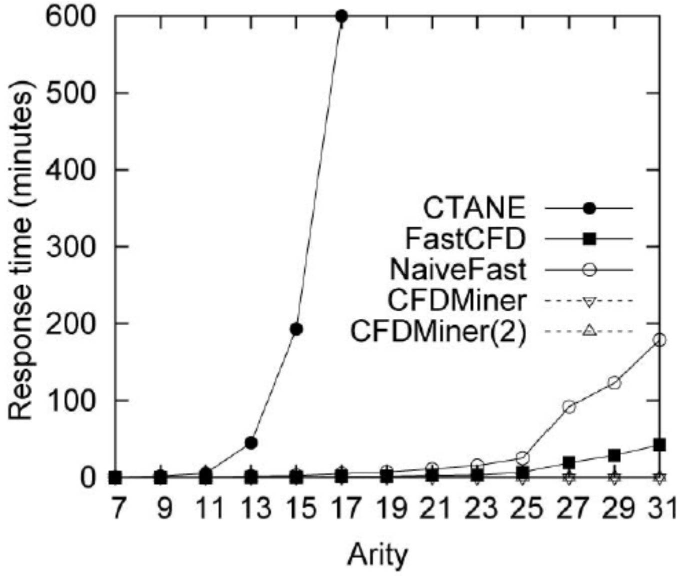
Experimental Evaluation

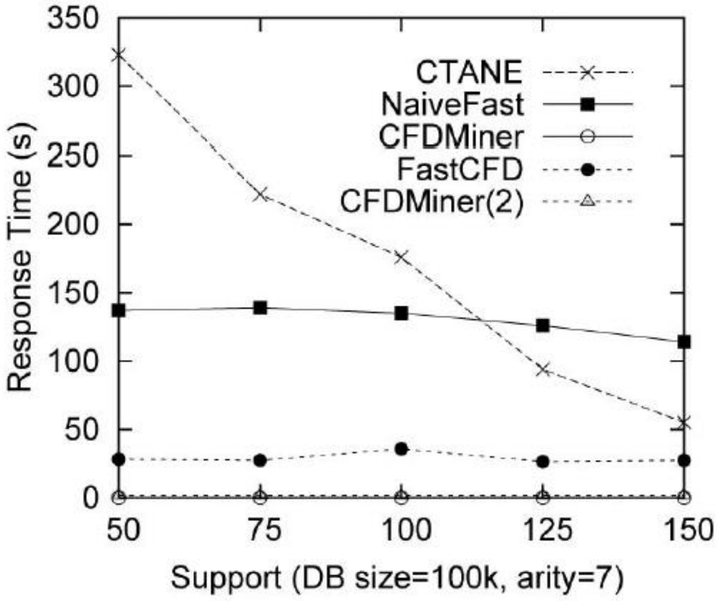
- Scalability wrt DBSIZE



Experimental Evaluation

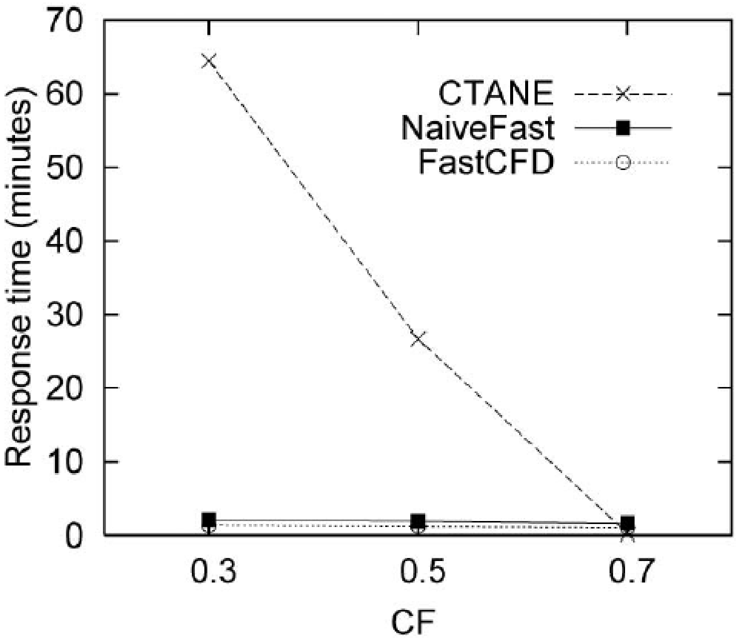
- Scalability wrt Arity and *k*





Experimental Evaluation

- Scalability wrt CF



- The results were on synthetic dataset
- Similar results were achieved on real datasets

Summary

- CFDMiner is efficient in discovering constant CFDs.
- CTANE works well with databases where arity is small and support threshold is large.
- NaiveFast and FastCFD are very efficient when arity of relation is very large.
- FastCFD is more efficient than the NaiveFast implementation especially when the arity is large.