Санкт-Петербургский государственный университет

Группа 20Б.09-мм

***БАРУТКИН Илья Дмитриевич***

# Реализация алгоритма валидации вероятностных функциональных зависимостей в платформе Desbordante

Отчёт по производственной практике

Научный руководитель:
ассистент кафедры ИАС Г.А. Чернышев

Санкт-Петербург
2024

# Оглавление

# Введение

Функциональные зависимости [1] — полезная разновидность метаданных. Метаданными являются, среди прочего, и их обобщения: приближенные (AFD) [7], условные (CFD) [3] и вероятностные (pFD) [5] функциональные зависимости.

Отношение R удовлетворяет функциональной зависимости $X \to Y \iff \forall t_1, t_2 \in R \quad t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$, где X, Y это наборы столбцов. На практике же некоторые значения данных определяют другие значения не точно, а просто с высокой вероятностью. Такая взаимосвязь не может определяться точной функциональной зависимостью. По этой причине разработаны различные определения неточных функциональных зависимостей, такие как AFD [7], SFD [2], pFD [5], где значения одного атрибута хорошо прогнозируются значениями другого. Последний тип неточных функциональных зависимостей, pFD, находил применение в задачах поиска зависимостей среди множества отдельных таблиц в pay-as-you-go системах [5].

Desbordante [6] — это высокопроизводительный профилировщик данных, разработанный на языке C++. На момент начала написания данной работы в Desbordante реализованы основные алгоритмы для поиска точных и приближенных функциональных зависимостей. Новая функциональность поиска вероятностных зависимостей представлена в настоящей работе.

# 1. Постановка задачи

Целью работы является реализация алгоритма поиска вероятностных функциональных зависимостей в платформе Desbordante. Для её выполнения были поставлены следующие задачи:

- произвести обзор примитива вероятностных функциональных зависимостей
- на основе алгоритма Tane реализовать функциональность поиска pFD в платформе Desbordante с поддержкой метрик PerValue и PerTuple
- разработать автоматические тесты
- сравнить производительность со стандартным алгоритмом Tane

# 2. Предварительные сведения

## 2.1. Определение pFD

Вероятностная функциональная зависимость (pFD) [5] обозначается как $pfd : X \xrightarrow{p} A$, где p — вероятность удержания зависимости, которая определяется по следующим формулам. Пусть даны отношение r, набор атрибутов $X$ и атрибут $A \notin X$. Обозначим множество уникальных значений атрибутов из X как $D_X = \{t[X] \mid t \in r\}$, множество кортежей с данными значениями $X_1$ атрибутов из X как $V_{X_1} = \{t \in r \mid t[X] = X_1\}$, и еще одно множество кортежей $(V_Y, V_{X_1}) = \{t \in r \mid t[X] = X_1 \wedge t[Y] = \underset{Y_k \in r}{argmax}\{|V_{X_1} \cap V_{Y_k}|\}\}$.

Тогда вероятность зависимости для одного значения $X_1$ атрибута $X$ определяется как $P(X \to Y, V_{X_1}) = \frac{|V_Y, V_{X_1}|}{|V_{X_1}|}$.

Вероятность функциональной зависимости между атрибутами X и Y на отношении r определяется двумя формулами, называемыми метриками PerValue и PerTuple:

$$P_{PerValue}(X \to Y, r) = \frac{\sum_{V_X \in D_X} P(X \to Y, V_X)}{|D_X|},$$

$$P_{PerTuple}(X \to Y, r) = \sum_{V_X \in D_X} \frac{|V_X|}{|r|} P(X \to Y, V_X).$$

В алгоритме Tane изначально было предложено использование метрики $g_3$ [7] для поиска приближенных зависимостей (AFD), которая определяется как $g_3(X \to Y, r) = 1 - \frac{max\{|s||s \subseteq r, s \models X \to Y\}}{|r|}$. Нетрудно видеть, что $P_{PerTuple}(X \to Y, r) = 1 - g_3(X \to Y, r)$. В реализации алгоритма Tane в платформе Desbordante используется следующая формула ошибки $e(X \to Y, r) = \frac{|\{(t_1,t_2) \in r^2 | t_1[X] = t_2[X] \wedge t_1[Y] \neq t_2[Y]\}|}{|r|^2 - |r|}$.

Из определений следует, что метрика PerTuple принимает во внимание частоту значения левостороннего атрибута среди всех кортежей отношения, в то время как PerValue является средним арифметическим вероятностей удержания зависимости на каждом значении левостороннего атрибута. В таблице 1 показано на примере влияние данного отли-

Таблица 1: Рассмотрим зависимость $X \to Y$

| X | Y |
|---|---|
| null | 1 |
| null | 2 |
| null | 3 |
| null | 4 |
| null | 5 |
| ... | ... |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |

*Пусть* $\epsilon = 0.114$. *При* $|V_{null}| \to \infty$ *pFD с метрикой PerValue удерживается, но* $g_3(X \xrightarrow{\epsilon} Y, r') \to 1$, $e(X \xrightarrow{\epsilon} Y, r') \to 1$ *и AFD не удерживается.*

чия. Было показано, что на наборах с низким качеством данных, вызванным шумом, метрика PerTuple обычно превосходила PerValue [5]. С другой стороны, метрика PerValue подходит для ненормализованных данных, где высокая частота некоторого значения левостороннего атрибута вызвана тем, что кортежи с этим значением представляют собой иной тип сущности. В экспериментах с улучшенным алгоритмом Tane с уточнением зависимостей по правилу транзитивности метрика PerValue превосходила PerTuple [4].

## 2.2. Алгоритм Tane

Алгоритм Tane [7] — алгоритм поиска точных и приближенных минимальных функциональных зависимостей. Алгоритм представляет собой обход графа, называемого решеткой, чьи вершины соответствуют всем возможным наборам атрибутов, а ребра соединяют вершины вида X и XA, где X — набор вершин, а A — другой атрибут. Иначе говоря, каждое ребро означает функциональную зависимость $X \to A$. Алго-

ритм Tane последовательно проверяет существование функциональных зависимостей между соседними уровнями решетки, при возможности исключая часть вершин из рассмотрения.

В платформе Desbordante алгоритм Tane реализован на языке C++. Валидация функциональной зависимости происходит в методах CalculateFdError и CalculateZeroAryFdError, которые принимают на вход информацию о кластерах кортежей в виде двух или одного объекта класса PositionListIndex соответственно. В классе содержится массив кластеров кортежей _index для данного набора атрибутов. Создание структуры PLI для набора атрибутов XA происходит путем пересечения индексов PLI атрибутов X и A, что реализовано в методе PositionListIndex::Intersect. В данной реализации алгоритма Tane кластеры отсортированы по индексу первого кортежа в кластере.

В новом алгоритме сортировка кластеров получившегося индекса для атрибутов XA происходит таким образом, чтобы сохранялся такой относительный порядок строк, как в индексе для атрибутов X.

# 3. Реализация

Алгоритм поиска вероятностных функциональных зависимостей основан на алгоритме Tane. В проекте Desbordante реализация Tane предстлавена в виде одноименного класса с методами подсчета ошибки по формуле $e$. Для добавления нового алгоритма в Desbordante можно было, с одной стороны, обобщить алгоритмы, перенося общие методы, такие как ExecuteInternal, в новый надкласс CommonTane, сделав методы подсчета ошибки виртуальными. Однако вызовы виртуальных методов требуют больше операций и могут снизить производительность программы [8]. В целях сохранения производительности стандартного алгоритма Tane на данном этапе работы класс Tane был раскопирован в класс PFDTane. Для подсчета ошибки по формулам PerValue и PerTuple были добавлены новые невиртуальные методы.

В описываемой реализации, метод подсчета ошибки получает не все отношение, а массив кластеров, инкапсулированный в объекте класса PositionListIndex, где каждый кластер представляет собой массив с индексами строк, имеющих одинаковое значение всех атрибутов XA. Сортировка отношения R, обозначенная на первой строке, реализована как сортировка массива кластеров по атрибутам XA и использует массив, возвращаемый методом PositionListIndex::CalculateAndGetProbingTable.

**Input:** Отношение $R$, атрибуты X и A
**Output:** Значение метрики PerValue для зависимости $X \to A$

```
1  SORT(R, {X, A})
2  c ← t_1(X); |π(X)| ← 1; count(c) ← 0
3  c' ← t_1(X, A); count(c') ← 0; maxCount(c) ← 0
4  for t ∈ R do
5      if t(X) == c then
6          count(c) ← count(c) + 1
7          if t(X, A) == c' then
8              count(c') ← count(c') + 1
9          end
10         else
11             if maxCount(c) < count(c') then
12                 maxCount(c) ← count(c')
13             end
14             c' ← t(X, A); count(c') ← 0
15         end
16     end
17     else
18         sum ← sum + macCount(c)/count(c)
19         c ← t(X); |π(X)| + 1 count(c) ← 0 maxCount(c) ← 0
20     end
21 end
22 return sum/|π(X)|
```

**Algorithm 1:** Вычисление метрики PerValue [5]

# 4. Апробация

Для проверки корректности работы алгоритма были написаны автоматические тесты. В платформе Desbordante для тестирования используется библиотека googletest. Добавлен тест для проверки поиска функциональных зависимостей на тестовом датасете с метрикой PerValue. Для проверки корректности подсчета метрики PerTuple был добавлен тест, содержащий восемь проверок вычисления метрики для различных наборов атрибутов из датасета TestFD.csv (файл tests/test_pfdtane.cpp). Для того, чтобы убедиться, что новый алгоритм при нулевой ошибке возвращает точные ФЗ, реализация проверяется уже существующими тестами, общими для всех FD-алгоритмов (файл tests/test_fd_algorithm.cpp).

# 5. Эксперименты

По сравнению со стандартным алгоритмом Tane, в новом алгоритме изменена процедура валидация ошибки. Для того, чтобы выяснить влияние внесенных изменений на производительность, а также проверить как порог ошибки влияет на время работы, были произведены следующие эксперименты.

Было произведено сравнение производительности алгоритмов PFDTane с метрикой PerValue и Tane. Эксперименты проводились на системе Ubuntu 20.04, 64bit, Intel Xeon Processor (Cascadelake), 2.2GHz (8 cores), 8GiB ОЗУ. В таблице показаны результаты описанного эксперимента. Под размерностью набора данных имеется в виду совокупность двух показателей: количество атрибутов в таблице и количество кортежей соответственно. Далее приведено среднее время работы алгоритмов Tane и PFDTane в миллисекундах. Также было замерено время работы алгоритма при различных порогах ошибки.

Таблица 2: Время работы алгоритмов Tane и PFDTane

| Набор данных | Размерность | Tane, ms | PFDTane, ms |
|---|---|---|---|
| LegacyPayors.csv | 4 x 1465233 | 3440 | 4170 |
| neighbors100k | 7 x 100000 | 976 | 1632 |
| CIPublicHighway100k.csv | 18 x 100000 | 32408 | 83762 |
| EpicVitals.csv | 7 x 1246303 | 64707 | 159883 |

В результате эксперимента обнаружено, что новый алгоритм требует больше времени для поиска зависимостей. При увеличении числа атрибутов таблицы наблюдается все большее отличие по времени работы относительно стандартной реализацией Tane.

В следующем эксперименте постепенно увеличивался порог ошибки на датасетах с 18 атрибутами, 100000 строками и 7 атрибутами, 1246303 строками. Если исключить особый случай нулевого порога ошибки, то с увеличением порога время работы алгоритма уменьшалось. В алгоритме PFDTane при нахождении минимальной зависимости часть решетки отсекается, поэтому при нахождении большого количества зависимостей на первых уровнях решетки означает меньшее число валидаций.

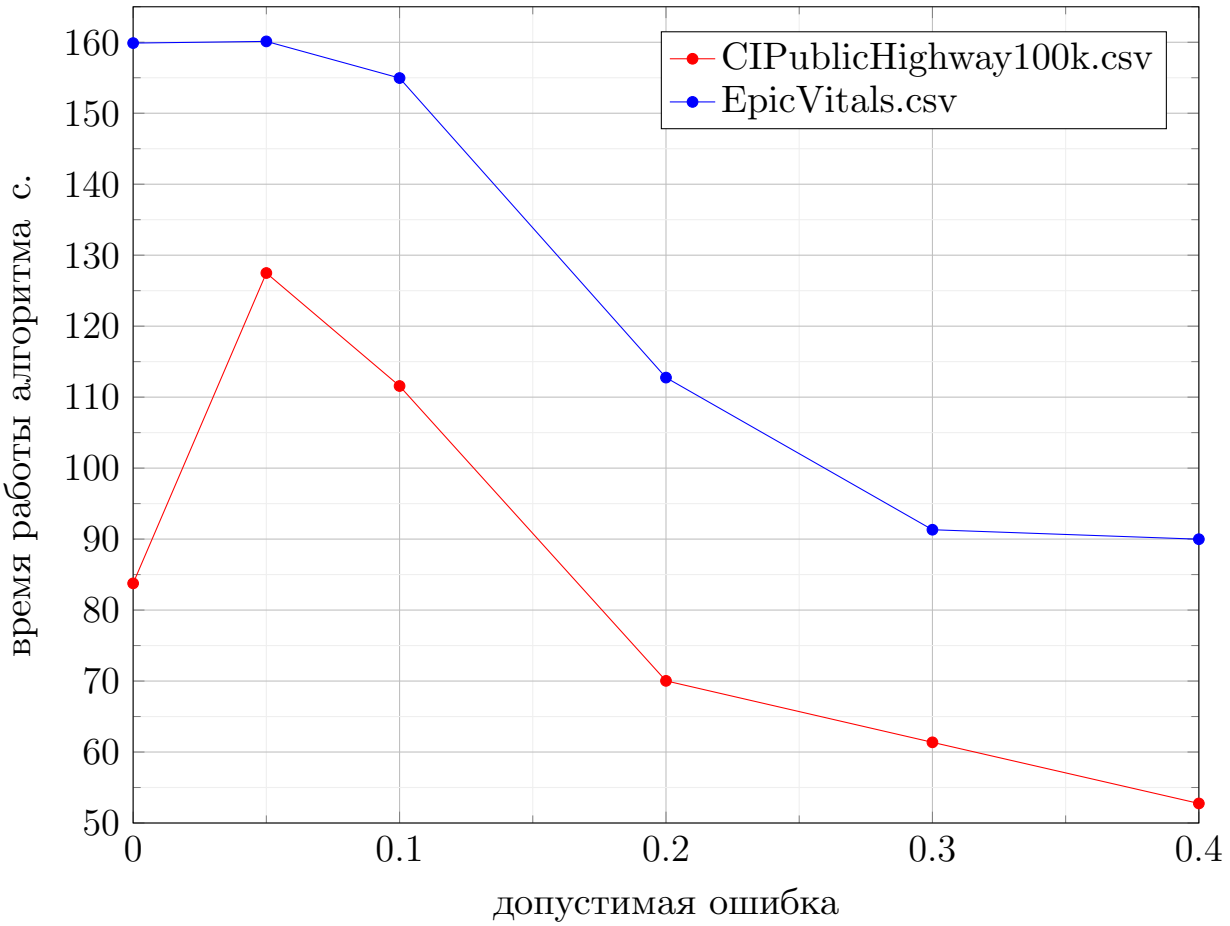


Рис. 1: время работы алгоритма с различным порогом ошибки

# Заключение

В ходе данной работы были выполнены следующие задачи:

- произведен обзор примитива вероятностных функциональных зависимостей
- на основе алгоритма Tane реализована функциональность поиска pFD в платформе Desbordante с поддержкой метрик PerValue и PerTuple
- разработаны автоматические тесты
- произведено сравнения времени работы алгоритма со стандартной реализацией Tane

Ссылка на GitHub-репозиторий https://github.com/iliya-b/Desbordante/tree/pfdtane-nongeneralized (имя пользователя — iliya-b).

# Список литературы


[1] Abedjan Ziawasch, Golab Lukasz, Naumann Felix. Profiling Relational Data: A Survey // The VLDB Journal. — 2015. — aug. — Vol. 24, no. 4. — P. 557–581. — URL: https://doi.org/10.1007/s00778-015-0389-y.

[2] CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. / Ihab Ilyas, Volker Markl, Peter Haas et al. — 2004. — 06. — P. 647–658.

[3] Discovering Conditional Functional Dependencies / Wenfei Fan, Floris Geerts, Laks V. S. Lakshmanan, Ming Xiong // 2009 IEEE 25th International Conference on Data Engineering. — 2009. — P. 1231–1234.

[4] Discovering Functional Dependencies in Pay-As-You-Go Data Integration Systems : Rep. : UCB/EECS-2009-119 ; Executor: Daisy Zhe Wang, Michael Franklin, Luna Dong et al. : 2009. — URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-119.pdf.

[5] Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems / Daisy Zhe Wang, Xin Luna Dong, Anish Das Sarma et al. // 12th International Workshop on the Web and Databases, WebDB 2009, Providence, Rhode Island, USA, June 28, 2009. — 2009. — URL: http://webdb09.cse.buffalo.edu/papers/Paper18/webdb09.pdf.

[6] M. Strutovskiy N. Bobrov K. Smirnov, Chernishev G. Desbordante: a Framework for Exploring Limits of Dependency Discovery Algorithms // 2021 29th Conference of Open Innovations Association (FRUCT). — 2021. — P. 344–354.

[7] Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies / Yká Huhtala, Juha Kärkkäinen, Pasi Porkka,

Hannu Toivonen // The Computer Journal. — 1999. — Vol. 42, no. 2. — P. 100–111.

[8] Technical Report on C++ Performance : Rep. : ISO/IEC TR 18015:2006(E) ; Executor: 21 Working Group WG : 2006. — URL: https://www.open-std.org/jtc1/sc22/wg21/docs/TR18015.pdf. — accessed: 2024-01-09.