Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Чижов Антон Игоревич

Реализация алгоритма поиска условных функциональных зависимостей в платформе Desbordante

Отчёт по учебной практике

Научный руководитель:
ассистент кафедры ИАС Чернышев Г. А.

Санкт-Петербург
2022

Оглавление

Введение **3**

1. Постановка задачи **4**

2. Обзор предметной области **5**

2.1. Терминология 5

2.2. Проблема поиска УФЗ 7

2.3. Проверка валидности УФЗ 9

2.4. Алгоритм CTANE 12

3. Реализация алгоритма **14**

4. Эксперименты **16**

4.1. Описание экспериментов 16

4.2. Результаты 17

5. Заключение **20**

Список литературы **21**

Введение

Условные функциональные зависимости (УФЗ) обобщают традиционные функциональные зависимости (ФЗ). УФЗ более гибкие, поскольку могут содержать зависимости, которые выполняются только на подмножестве данных.

Как ФЗ, так и УФЗ являются метаданными (информацией о данных), которые могут использоваться для анализа данных (data mining [2]). Известно, что УФЗ являются более эффективными, чем ФЗ при поиске и устранении проблем с качеством данных [1, 7]. Например, для проблем, связанных с неточными или несовместимыми значениями и т. д.

Одним из самых известных алгоритмов для поисках УФЗ является алгоритм CTANE [5], основанный на концепции партиций. В данной статье представлен алгоритм для поиска точных УФЗ, но он может быть модифицирован для того, чтобы иметь возможность находить приблизительные УФЗ. Пример подобной модификации был представлен в [8].

На момент написания данной работы в Desbordante реализованы основные алгоритмы для поиска ФЗ, но отсутствуют для поиска УФЗ. В данной работе предоставлен отчёт о реализации алгоритма CTANE [4,8] в платформе Desbordante.

1 Постановка задачи

Целью данной практики является реализация алгоритма поиска условных функциональных зависимостей CTANE на языке программирования C++ в платформе Desbordante и исследование производительности. Для достижения этой цели были поставлены следующие задачи:

- Произвести обзор предметной области и технологий поиска условных функциональных зависимостей;
- Реализовать алгоритм CTANE на языке программирования C++;
- Сравнить производительность алгоритма для различных входных данных.

2 Обзор предметной области

2.1 Терминология

ID	Surname	Country	Workshop	Gender	On vacation
1	Moore	USA	StuntSet	m	true
2	Moore	USA	StuntSet	w	true
3	Lee	USA	StuntSet	m	false
4	Mitchell	France	StuntSet	m	false
5	Mitchell	Spain	MetaPlay	m	false
6	Mitchell	Spain	MetaPlay	w	true
7	Lee	Spain	MetaPlay	w	false
8	Moore	France	StuntSet	m	true
9	Moore	Spain	MetaPlay	m	false
10	Mitchell	France	MetaPlay	m	false
11	Moore	France	MetaPlay	w	false
12	Lee	France	StuntSet	w	false
13	Lee	USA	MetaPlay	m	false
14	Mitchell	France	StuntSet	w	true

Таблица 1: Отношение Employee

В данной главе приведены определения из работ [4, 8].

Пусть r — тело отношения и R — схема отношений, определённая над множеством атрибутов $\mathcal{A}$, где каждый атрибут $A \in \mathcal{A}$ имеет конечный набор различных значений $dom(A)$.

Определение. Пусть X и Y это наборы атрибутов из $\mathcal{A}$. Говорят, что существует *функциональная зависимость* f над R между X и Y, если и только если любые два кортежа, совпадающие на атрибутах X, совпадают на атрибутах Y.

Определение. *Условной функциональной зависимостью* φ над R называется пара $(X \to A, t_p)$, где (1) X это набор атрибутов из $\mathcal{A}$, и A это атрибут из $\mathcal{A}$; (2) $X \to A$ это стандартная ФЗ; (3) t_p это шаблонный кортеж (tuple pattern) с атрибутами в X и A, где для каждого

$B \in X \cup \{A\}$, $t_p[B]$ это либо константа ‘a’ из $dom(A)$, либо безымянная переменная ‘_’.

Для разделения атрибутов X и A в шаблонном кортеже в данной работе используется знак ‘||’.

Определение. УФЗ $\varphi = (X \to A, t_p)$, в которой $t_p[A] =$ ‘_’, называется *переменной* (variable), в противном случае — *постоянной* (constant).

Стандартные ФЗ являются частным случаем УФЗ, так как любая ФЗ $X \to A$ может быть представлена в виде УФЗ $(X \to A,\ t_p)$, где $t_p[B] =$ _ для каждого $B \in X \cup \{A\}$.

Примеры. Рассмотрим таблицу 1, в которой содержится информация о сотрудниках. Пусть r' — тело данного отношения.

Традиционные ФЗ, которые выполняются на r':

f_1: $\{Surname, Country, Gender\} \to On\ vacation$

f_2: $\{Surname, Workshop, Gender\} \to On\ vacation$

ФЗ f_1 означает, что работники из одной страны, одного пола и одной страны либо все находятся в отпуске, либо все работают. Вторая ФЗ f_2 аналогична первой зависимости, но в данном случае сотрудники должны находиться в одной компании, а не в одной стране.

Для данного отношения также могут быть найдены следующие УФЗ:

φ_1: $\big\{(Surname, Moore), (Country, France), (Gender, w)\big\}$
$\to (On\ vacation, false)$

φ_2: $\big\{(Country, \_), (Workshop, MetaPlay), (Gender, m)\big\}$
$\to (On\ vacation, \_)$

УФЗ φ_1 “уточняет” зависимость f_1. Данная зависимость выполняется только на части отношения Employee, а именно: все женщины-работники с Фамилией Moore из Франции не находятся в отпуске. Вторая УФЗ не “уточняет” никакой ФЗ и говорит о том, что мужчины-работники из компании MetaPlay, которые живут в одной стране имеют

одинаковый статус On vacation. То есть либо все находятся в отпуске, либо все работают. Данная зависимость не уточняет никакую существующую ФЗ.

В отличие от ФЗ, условная ФЗ может быть нарушена одним кортежём. Например, рассмотрим УФЗ $\varphi' = \big\{(ID, 1)\big\} \to (Surname, Lee)$. Действительно, кортеж t_1 с $ID = 1$ нарушает данную УФЗ: $t_1[ID] = 1$, но $t_1[Surname] \neq Lee$.

Семантика УФЗ $\varphi = (X \to A, t_p)$ над r определяется следующим образом: кортеж $t \in r$ *соответствует* шаблонному кортежу t_p на атрибутах X (обозначается как $t[X] \asymp t_p[X]$), если $\forall B \in X$, либо $t_p[B] =$ '_', либо $t[B] = t_p[B]$.

Замечание. Кортеж t *нарушает* переменную УФЗ $\varphi = (X \to A, t_p)$ если $t[X] \asymp t_p[X]$ и существует другой кортеж $t' \in r$ такой, что $t[X] = t'[X]$ и $t[A] \neq t'[A]$. Кортеж t *нарушает* постоянную УФЗ $\varphi = (X \to A, t_p)$ если $t[X] \asymp t_p[X]$ и $t[A] \neq t_p[A]$.

Определение. Пусть $\varphi = (X \to A, t_p)$. Тогда, по определению,
$VIO(\varphi, r)$ — набор кортежей, которые нарушают УФЗ φ.
Если $VIO(\varphi, r) = \varnothing$, то r удовлетворяет УФЗ φ (обозначается $r \models \varphi$).

Утверждение. Пусть $\varphi = (X \to A, t_p)$ — постоянная УФЗ, тогда существует эквивалентная эквивалентная УФЗ $\varphi' = (X' \to A, t'_p[X'] \parallel a)$, где X' состоит из всех атрибутов $B \in X$ таких, что $t_p[B]$ является константой. Таким образом, когда $t_p[A]$ — константа, то мы можем безопасно убрать всех атрибуты B в левой части УФЗ φ с $t_p[B] =$ '_'.

2.2 Проблема поиска УФЗ

Очевидно, что при поиске УФЗ возникает возникают тривиальные и избыточные УФЗ. Известно, что для алгоритмов поиска ФЗ было введено понятие канонического покрытия (canonical cover). В работе [4] представлено схожее понятие для алгоритмов поиска УФЗ.

Определение. УФЗ $\varphi = (X \to A, t_p)$ над R называется *тривиальной*, если $A \in X$. Если φ тривиальна, то либо $VIO(\varphi, r) = \varnothing$, либо

$VIO(\varphi, r) = r$.
Постоянная УФЗ $(X \to A, (t_p \parallel a))$ называется *приведённой слева* (left-reduced), если $\forall Y \subsetneq X$, $r \not\models (Y \to A, (t_p[Y] \parallel a))$.
Переменная УФЗ $(X \to A, (t_p \parallel \_))$ называется *приведённой слева*, если:

(1) $r \not\models (Y \to A, (t_p[Y] \parallel \_))\ \forall Y \subsetneq X$,

(2) $r \not\models (X \to A, (t'_p[Y] \parallel \_))\ \forall t'_p : t_p \ll t'_p$.

Пункт (2) означает, что ни одна из констант в $t_p[X]$ не может заменена на '_'. Другими словами, t_p является самым общим шаблоном.
УФЗ φ на r называется *минимальной*, если она нетривиальна, приведена слева и удовлетворяет высказыванию $r \models \varphi$.

Каноническое покрытие УФЗ над r может содержать много зависимостей, которые также являются избыточными. Например, УФЗ, которые выполняются только на небольшом количестве кортежей. Поэтому в статье [8] было введено понятие поддержки.

Определение. Набором предметов (itemset) будем называть набор пар вида "атрибут-значение" вида (A, v), где $A \in \mathcal{A}$, и v это либо значение из $dom(A)$, либо '_'. Тогда предмет (A, v) с $v \in dom(A)$ поддерживается в кортеже t если $t[A] = v$. Кортеж t поддерживается набором предметов I в r если он поддерживается $\forall i \in I$. *Покрытием* набора предметов I в r $cov(I, r)$ является набор кортежей, которые в r поддерживают I. Поддержка I в r, которая обозначается как $supp(I, R)$, равна количеству кортежей, которые набор предметов I покрывает в D.

Замечание. В работе [4] приводится определение для поддержки УФЗ $\varphi = (X \to A, t_p \parallel a)$ в r. Очевидно, что для φ можно однозначно сопоставить набор предметов I_φ.
Например, для $\varphi = (\{B, C\} \to D, \{\_, c_1, d_1)\}$ $I_\varphi = \{(B, \_), (C, c_1), (D, d_1)\}$.
Поэтому будем полагать, что $supp(\varphi, r) := supp(I_\varphi, r)$.

Определение. Пусть $\varphi = (X \to A, t_p)$ это УФЗ над R. Говорят [4], что УФЗ φ является *k-частой* (k-frequent) в r, если $supp(\varphi, r) \geq k$.

Во многих случаях УФЗ не удерживаются точно. То есть зависимость может нарушаться на некотором наборе кортежей, которые согласуются с антецедентом (X), но не удовлетворяют консеквенту (Y). Например, это встречается, когда появляется задача очистки данных (предполагается, что доступ имеется только к грязным данным). В данном случае необходимо использовать приблизительные УФЗ.

Определение. Пусть $\varphi = (X \to A, t_p)$ это УФЗ над R.
Уверенность φ на r определяется [8] так: $conf(\varphi, r) = 1 - \frac{|r'|}{supp(\varphi,r)}$,
где $r' \subset r$ — минимальное подмножество r, на котором выполняется $r \backslash r' \models \varphi$.

УФЗ называется *точной*, если $conf(f, r) = 1$, и *приблизительной* в противном случае.

Таким образом, была поставлена следующая задача: необходимо найти все УФЗ φ над R такие, что $supp(\varphi, r) \geq \delta$ и $conf(\varphi, r) \geq 1 - \varepsilon$ (δ и ε это соответственно порог поддержки и ошибки).

2.3 Проверка валидности УФЗ

Алгоритм поиска УФЗ CTANE, также как и TANE [9], основан на концепции партиций эквивалентности (equivalence partitions). В статье [8] были введены следующие определения:

Определение. Пусть I — набор предметов. Кортежи $t, s \in r$ являются *эквивалентными для* I, если $\forall (B, v) \in I \ s[B] = t[B] \asymp v$. Класс эквивалентности для кортежа $s \in r$ обозначается как $[s]_I$ и состоит из всех кортежей $t \in r$, эквивалентных s. Таким образом, *партицией для набора предметов* I $\Pi(I)$ называется набор эквивалентных классов $[s]_I$ для $s \in r$.
Размер партиции $\Pi(I)$ обозначается как $|\Pi(I)|$ и равен количеству классов эквивалентности в $\Pi(I)$, а $||\Pi(I)||$ равняется количеству кортежей. Очевидно, что $support(I, r) = ||\Pi(I)||$.

Утверждение (Проверка валидности точных УФЗ).
Пусть I — набор предметов, $j \in I$, $\varphi = I \backslash \{j\} \to j$. Тогда:

(1) если $j = (C, c_j)$, где $c_j \in dom(C)$, то УФЗ φ удерживается тогда и только тогда, когда $||\Pi(I\backslash\{j\})|| = ||\Pi(I)||$;

(2) если $j = (C, \_)$, то УФЗ φ удерживается тогда и только тогда, когда $|\Pi(I\backslash\{j\})| = |\Pi(I)|$

Доказательство.

(1): Эквивалентное утверждение было приведено в в статье [3]. Действительно, если количество элементов в партиции $\Pi(I)$ осталось тем же, что и в $\Pi(I\backslash\{j\})$, то это означает, что $\forall t \in cov(I\backslash\{j\}, r)\ s[C] = c_j$. Другими словами, все кортежи, которые удовлетворяют антецеденту имеют одинаковое значение в консеквенте, равное c_j. Очевидно, что это условие является необходимым и достаточным.

(2): Данное утверждение имеет сходство с критерием валидности для точных ФЗ. УФЗ $\varphi = I\backslash\{(C, \_)\} \to (C, \_)$ можно представить как ФЗ, которая выполняется на части отношения r, а именно на $cov(I\backslash\{j\}, r)$. Тогда достаточно воспользоваться известным критерием для ФЗ. ■

Замечание. В некоторых работах [4, 8] приводится утверждение, в котором говорится, что для того, чтобы УФЗ $I\backslash\{j\} \to j$ удерживалась, необходимо и достаточно условия $|\Pi(I\backslash\{j\})| = |\Pi(I)|$. Это неверно. Для того, чтобы это показать достаточно рассмотреть контрпример.

ID	Surname	Country	Workshop	Gender	On vacation
1	Moore	USA	StuntSet	m	true
2	Moore	USA	StuntSet	w	true
3	Lee	USA	StuntSet	m	false
4	Mitchell	France	StuntSet	m	false
5	Mitchell	Spain	MetaPlay	m	false
6	Mitchell	Spain	MetaPlay	w	true
7	Lee	Spain	MetaPlay	w	false
8	Moore	France	StuntSet	m	true
9	Moore	Spain	MetaPlay	m	false
10	Mitchell	France	MetaPlay	m	false
11	Moore	France	MetaPlay	w	false
12	Lee	France	StuntSet	w	false
13	Lee	USA	MetaPlay	m	false
14	Mitchell	France	StuntSet	w	true

Таблица 2: Датасет “Employee”

Пример (контрпример)**.**

Рассмотрим датасет “Employee”, представленный в таблице 2 и УФЗ $\{(Surname, Mitchel), (Country, \_)\} \to (On\,vacation, false)$. Соответственно $I = \{(Surname, Mitchel), (Country, \_), (On\ vacation, false)\}$ и $j = (On\ vacation, false)$.

Очевидно, что зависимость нарушается для кортежа с номером 6, так как данный кортеж принадлежит покрытию набора предметов $I \backslash \{j\}$, но не удовлетворяет консеквенту. Строки таблицы, на которых зависимость выполняется выделена серым цветом, а строка, которая нарушает данную зависимость — тёмно-серым цветом.

Заметим, что $\Pi(I \backslash \{j\}) = \big\{\{4, 10, 14\}, \{5, 6\}\big\}$, $\Pi(I) = \big\{\{4, 10\}, \{5\}\big\}$ и размеры партиций совпадают, но УФЗ не удерживается. Получено противоречие.

2.4 Алгоритм CTANE

Алгоритм CTANE ищет минимальные УФЗ. Данный алгоритм является расширением алгоритма TANE для поиска ФЗ, который уже был реализован и подробно описан в работе [10]. В алгоритме CTANE граф состоит из элементов вида (X, t_p), где $X \in \mathcal{A}$ и t_p это кортеж шаблонов над X. В отличие от TANE, множества атрибутов у разных вершин могут совпадать. Каждая вершина (X, t_p) будет соединена с вершинами $(X \cup \{A\}, t'_p)$, где $t'_p[X] = t_p[X]$, а $t'[A] = c_a$, $A \notin X$.

Алгоритм CTANE, описанный в книге [6] отличается от алгоритма CTANE, представленного в работе [8]. В первой работе приведён алгоритм для поиска точных УФЗ, во второй же было использовано понятие уверенности, приблизительных УФЗ, использовался другой критерий для исключения лишних кандидатов. В данной работе реализован алгоритм из первой работы с использованием некоторых модификаций из второй работы.

Далее приведены реализации методов, которые были модифицированы:

(1) на этапе проверки валидности УФЗ используются разные подходы в зависимости от того, какие УФЗ необходимо найти (точные или приблизительные):

   (a) в случае, когда рассматриваются точные зависимости, используется утверждение из 2.3;

   (b) в случае, когда рассматриваются приблизительные зависимости, рассчитывается их уверенность с помощью алгоритма, представленного в статье [9].

(2) на этапе просмотра решётки пропускаются УФЗ $(X \to A, (t_p \parallel a)$, у которых:

   (c) $t_p[X]$ состоит из констант, а a — безымянная переменная. Такие УФЗ являются избыточными. Допустим, была найдена

УФЗ $\varphi = (\{(A, c_1)\} \to (B, \_)$. Очевидно, что данная зависимость менее информативна, чем УФЗ $\varphi' = (\{A, c_1\} \to (B, c_2))$.

(d) $t_p[X]$ состоит из безымянных переменных, а a — константа.

УФЗ такого вида можно не рассматривать, это следует из утверждения, описанного в 2.1.

Подробное описание алгоритма с примером его работы можно найти в книге [6].

3 Реализация алгоритма

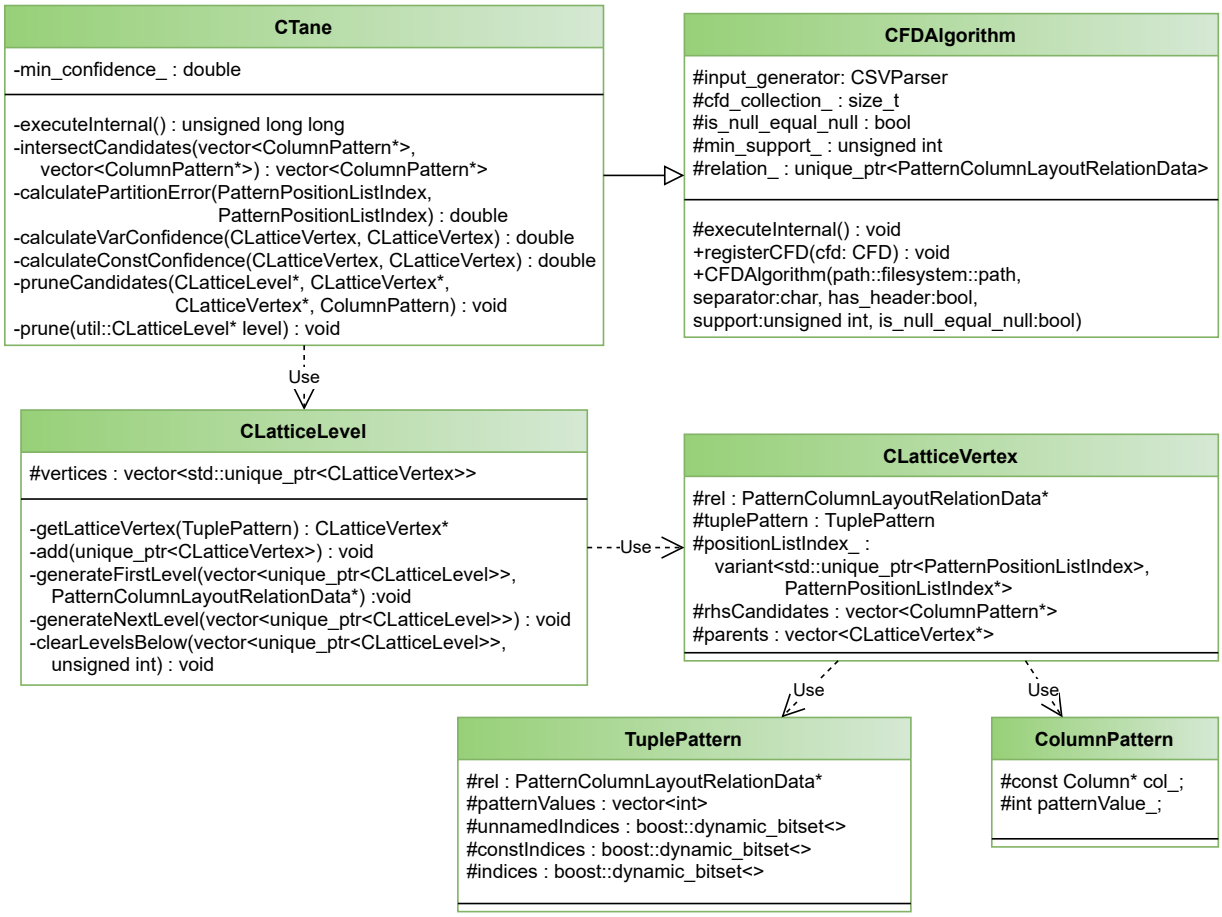

Рис. 1: Иерархия классов.

Иерархия классов алгоритмов представлена на Рис. 1. Некоторые детали реализации опущены. Классы, представленные на диаграмме, были реализованы с использованием уже существующих классов, использовавшихся для алгоритма TANE (за исключением классов `TuplePattern` и `ColumnPattern`).

Алгоритм начинает выполнение с обработки входной таблицы, парсинг которой выполняет уже реализованный в Desbordante класс `CSVParser`. Класс `CFDAlgorithm` хранит информацию об УФЗ, которые были найдены алгоритмом и некоторые конфигурационные данные. В классе `CTANE` содержится основная логика алгоритма.

В классе `CFDAlgorithm` определён единственный абстрактный метод `executeInternal()`, в котором должна быть реализована логика алго-

ритма. Этот метод вызывается внутри метода `execute()`. Вычисление партиций вынесено в данный метод, вся информация о которых содержится в поле `relation_` класса `CFDAlgorithm`.

В основном методы алгоритма повторяют псевдокод, который приводится в книге [6] и не представляют никакого интереса. Отдельно стоит отметить методы для вычисления уверенности УФЗ `calculateConstConfidence`, `calculateVarConfidence`. Первый метод вызывается для постоянных УФЗ, второй — для переменных. Алгоритм для вычисления взяты из открытого исходного кода[1], который был представлен в статье [8] и основан на эффективном алгоритме для вычисления уверенности из статьи о TANE [9].

[1]https://codeocean.com/capsule/6146641/tree/v1

4 Эксперименты

4.1 Описание экспериментов

Name	$\|r\|$	$\mathcal{A}$
Balance-Scale	625	5
Breast-Cancer	699	11
Abalone	4177	9

Таблица 3: Наборы данных из репозитория UCI

Алгоритм позволяет варьировать следующие параметры: минимальная поддержка, минимальная уверенность, максимальная размерность. Максимальная размерность для УФЗ — максимальное количество атрибутов в зависимости. Далее для его обозначения используется $|R|$.

Эксперименты разбиты на две части: первая часть состоит из тестирования алгоритма для поиска точных зависимостей, а вторая — для приблизительных.

Тестирование алгоритма для поиска точных УФЗ проводилось на трёх реальных наборов данных из репозитория UCI[2], описанных в таблице 3. В таблице содержится информация о количестве кортежей в датасете и количестве атрибутов.

Для каждого набора данных было сделано три запуска, после чего было посчитано среднее время, прошедшее с окончания считывания программой таблицы до завершения работы алгоритма. Desbordante компилировался с флагом `-O3`.

Во второй части экспериментов, которая нацелена на исследование работы алгоритма для поиска приблизительных УФЗ, используется схожий алгоритм, что и в первой части.

Эксперименты проводились на системе Ubuntu 20.04.3 LTS, 5.11.0-43-generic, AMD Ryzen 7 4800H CPU @ 2.90GHz × 8, 16 GiB RAM, SSD A-Data S11 Pro AGAMMIXS11P-512GT-C 512GB.

[2]http://archive.ics.uci.edu/ml/

4.2 Результаты

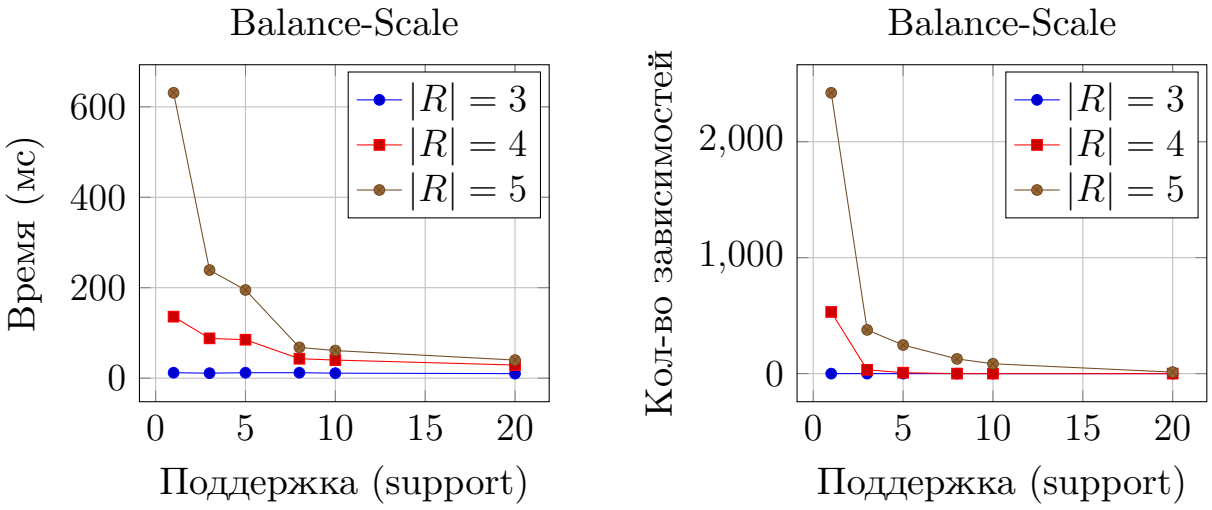

Рис. 2: Эксперименты на датасете Balance-Scale (точные УФЗ).

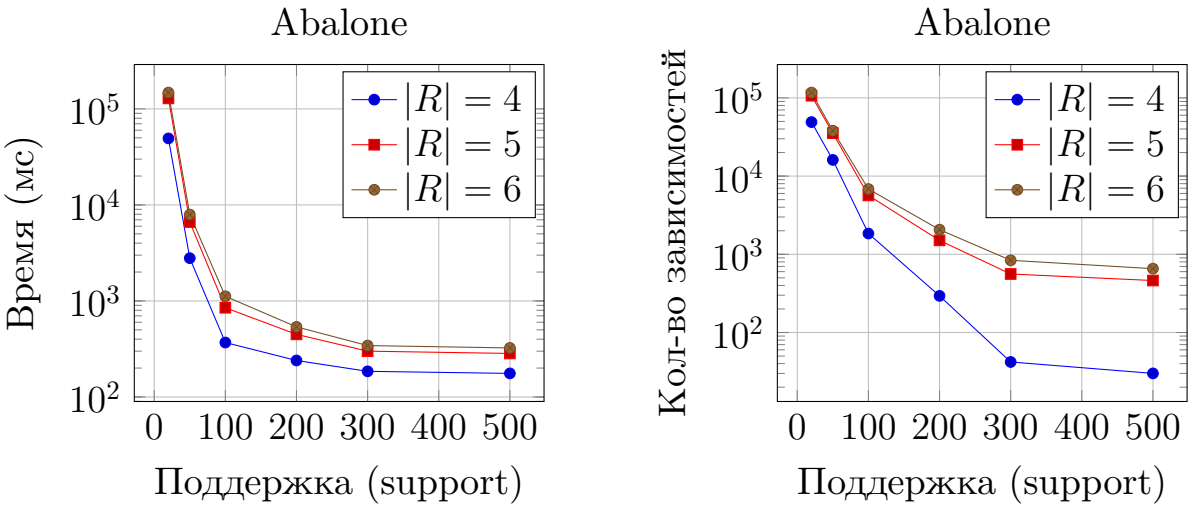

Рис. 3: Эксперименты на датасете Abalone (точные УФЗ).

На Рис. 2 приведены результаты экспериментов для поиска точных УФЗ на датасете Balance-Scale. Время работы CTANE увеличивается экспоненциально для поддержки ≤ 20, когда $|R|$ увеличивается с 4 до 5. Для $|R| = 3$ кортежей найдено не было, При увеличении $|R|$ количество найденных зависимостей для маленьких значений поддержки стремительно растёт. При увеличении поддержки для каждого $|R|$ количество зависимостей стремится к нулю.

Аналогичные рассуждения справедливы для датасета Abalone и Breast-Cancer, результаты экспериментов для которых показаны на Рис. 3 и

Рис. 4.

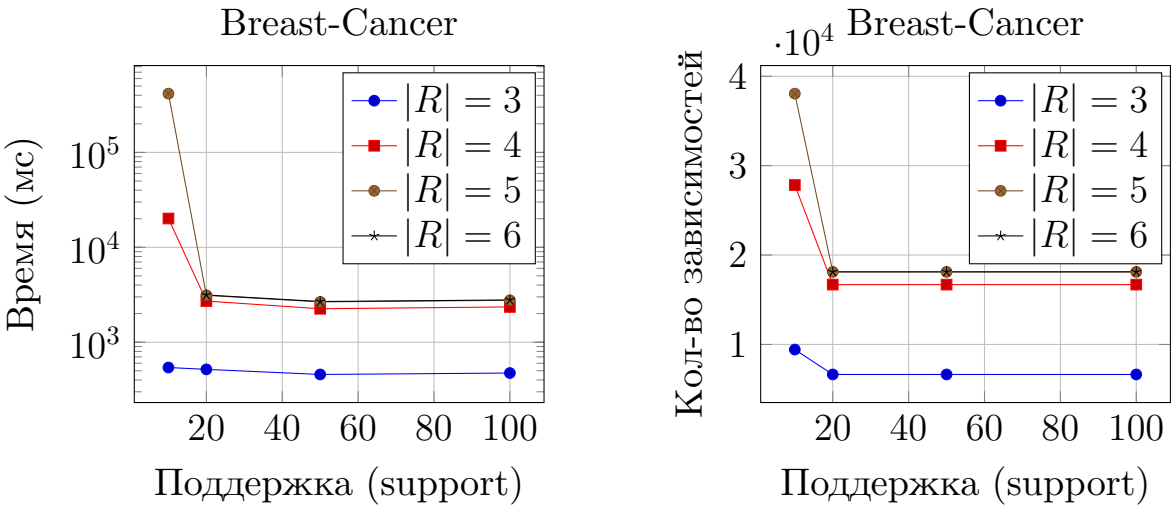

Рис. 4: Эксперименты на датасете Breast-Cancer (точные УФЗ).

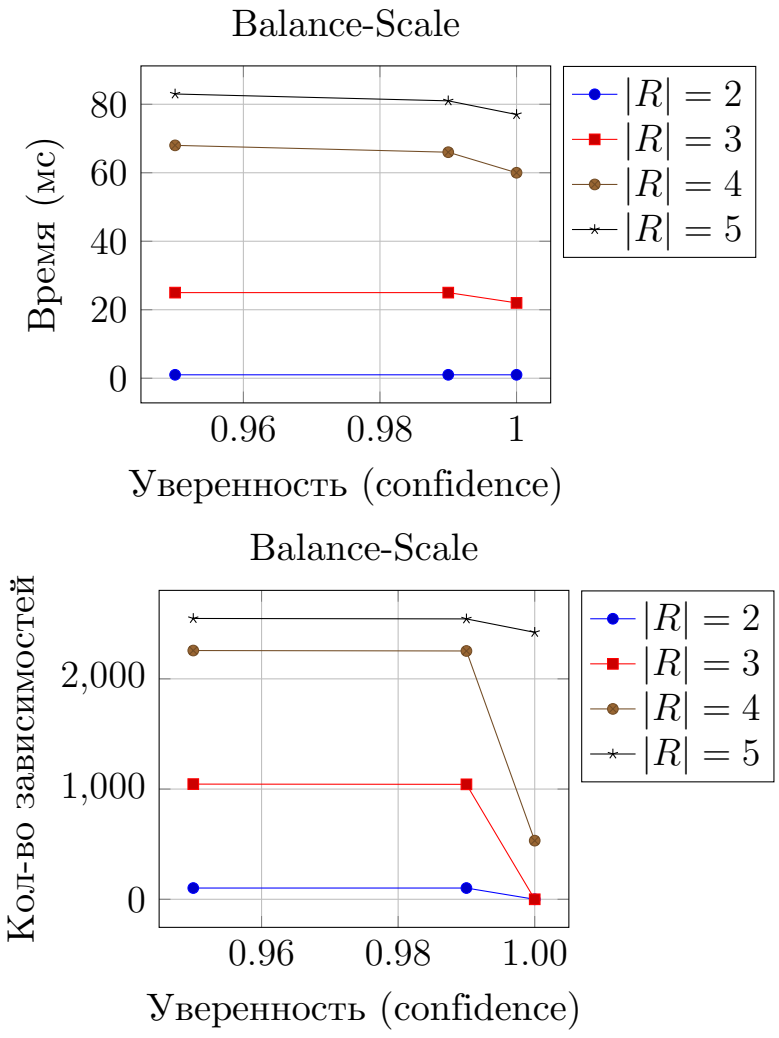

Рис. 5: Эксперименты на датасете Balance-Scale (приблизительные УФЗ).

На рис 5 представлены результаты экспериментов для второй части экспериментов на датасете Balance-Scale. Порог ошибки оказывает незначительное влияние на время выполнения алгоритма. Это верно, так как: поскольку на основе уверенности не происходит отсечения, все эти УФЗ проверяются независимо от порога ошибки, и единственная разница заключается в том, добавляются ли они к результату. Аналогичные результаты будут и на других датасетах.

5 Заключение

В рамках данной работы был реализован алгоритм поиска условных функциональных зависимостей CTANE в платформе Desbordante и исследована его производительность. В ходе данной работы были решены следующие задачи:

- Произведён обзор предметной области и технологий поиска условных функциональных зависимостей;
- Реализован алгоритм CTANE (исходный код доступен на Github[3]);
- Проведены эксперименты для замера производительности реализованного алгоритма на разных входных данных.

[3]github.com/vs9h/Desbordante/tree/ctane (дата обращения: 27.12.2021)

Список литературы

[1] Fan Wenfei, Geerts Floris, Jia Xibei, and Kementsietsidis Anastasios. Conditional functional dependencies for capturing data inconsistencies // ACM Trans. Database Syst. — 2008. — Vol. 33, no. 2. — Access mode: http://doi.acm.org/10.1145/1366102.1366103.

[2] Abedjan Ziawasch, Golab Lukasz, Naumann Felix, and Papenbrock Thorsten. Data Profiling. — First ed. — Morgan & Claypool Publishers, 2018. — Nov. — Vol. 10 of Synthesis Lectures on Data Management.

[3] Diallo Thierno, Novelli Noel, and Petit Jean-Marc. Discovering (frequent) constant conditional functional dependencies // IJDMMM. — 2012. — Vol. 4, no. 3. — P. 205–223. — Access mode: http://dx.doi.org/10.1504/IJDMMM.2012.048104.

[4] Fan Wenfei, Geerts Floris, Lakshmanan Laks V. S., and Xiong Ming. Discovering Conditional Functional Dependencies // Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China. — IEEE. — 2009. — P. 1231–1234. — Access mode: http://dx.doi.org/10.1109/ICDE.2009.208.

[5] Fan Wenfei, Geerts Floris, Li Jianzhong, and Xiong Ming. Discovering Conditional Functional Dependencies // IEEE Trans. Knowl. Data Eng. — 2011. — Vol. 23, no. 5. — P. 683–698. — Access mode: http://dx.doi.org/10.1109/TKDE.2010.154.

[6] Fan Wenfei and Geerts Floris. Foundations of Data Quality Management. Synthesis Lectures on Data Management. — Morgan & Claypool Publishers, 2012. — Access mode: http://dx.doi.org/10.2200/S00439ED1V01Y201207DTM030.

[7] Cong Gao, Fan Wenfei, Geerts Floris, Jia Xibei, and Ma Shuai.

Improving Data Quality: Consistency and Accuracy // Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007 / ed. by Koch Christoph, Gehrke Johannes, Garofalakis Minos N., Srivastava Divesh, Aberer Karl, Deshpande Anand, Florescu Daniela, Chan Chee Yong, Ganti Venkatesh, Kanne Carl-Christian, Klas Wolfgang, and Neuhold Erich J. — ACM. — 2007. — P. 315–326. — Access mode: http://www.vldb.org/conf/2007/papers/research/p315-cong.pdf.

[8] Rammelaere Joeri and Geerts Floris. Revisiting Conditional Functional Dependency Discovery: Splitting the "C" from the "FD". // Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part II / ed. by Berlingerio Michele, Bonchi Francesco, Gärtner Thomas, Hurley Neil, and Ifrim Georgiana. — Springer. — 2018. — sep. — Vol. 11052 of Lecture Notes in Computer Science. — P. 552–568. — Access mode: https://doi.org/10.1007/978-3-030-10928-8_33.

[9] Huhtala Ykä, Kärkkäinen Juha, Porkka Pasi and Toivonen Hannu. Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies // Computer Journal. — 1999. — Vol. 42, no. 2. — P. 100–111.

[10] Максим Струтовский. Реализация алгоритмов поиска функциональных зависимостей на языке программирования C++. — 2020.