Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 23.Б08-мм

Реализация алгоритма поиска числовых зависимостей в “Desbordante”

СЕНИЧЕНКОВ Пётр Ильич

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
ассистент кафедры ИАС Чернышев Г. А.

Санкт-Петербург
2024

Оглавление

Введение

Профилирование даннных — это процесс, направленный на анализ и извлечение метаданных из наборов данных [13]. Оно позволяет не только обнаружить простые характеристики, такие как размер файла и наличие пропущенных значений, но и выявить более сложные закономерности и взаимосвязи между значениями. Такие закономерности называются *примитивами* [2]. Профилирование данных применяется для исследования данных (data exploration), очистки данных (data cleansing), интеграции данных (data integration) и оптимизации запросов в системах управления базами данных [13].

Одним из наиболее часто встречающихся примитивов является функциональная зависимость (Functional Dependencies, FD) [1]. Функциональная зависимость $X \to Y$ означает, что значения из Y однозначно определяются значениями из X. Такие зависимости могут быть неприменимы к реальным данным, которые часто содержат неточности и ошибки.

В таком случае можно естественным образом обобщить функциональные зависимости, введя числовые зависимости (Numerical Dependencies, ND) [7]. В отличие от функциональных зависимостей, числовая зависимость $X \xrightarrow{k} Y$ означает, что каждому значению из X соответствует не более, чем k значений из Y. Числовые зависимости позволяют обнаружить закономерности в тех случаях, когда недостаточно функциональных зависимостей — например, если данные содержат неточности вследствие погрешности измерений или ошибки ввода.

Одним из инструментов, позволяющих обнаруживать функциональные зависимости, является Desbordante — наукоёмкий профилировщик данных с открытым исходным кодом[1], разрабатываемый на языке программирования C++. Desbordante предлагает несколько алгоритмов поиска функциональных зависимостей, однако для числовых зависимостей представлен только алгоритм верификации. В данной работе будет рассмотрен алгоритм BBND [3] для поиска числовых зависимостей, и предложена его реализация в рамках Desbordante.

[1] https://github.com/Desbordante/desbordante-core/

1. Постановка задачи

Целью работы является реализация алгоритма BBND для поиска числовых зависимостей и внедрение его в Desbordante. Для достижения цели были поставлены следующие задачи:

1. изучить предметную область алгоритмов поиска числовых зависимостей;
2. реализовать следующие компоненты алгоритма BBND:
   (a) поиск числовых зависимостей между одиночными атрибутами;
   (b) вывод числовых зависимостей.
3. разработать тесты для получившейся реализации и проверить на корректность.

2. Обзор

Числовые зависимости были впервые введены в 1985 году в работе J. Grant и J. Minker [7]. Там же доказано, что, в отличие от FD [1], для ND не существует конечного полного непротиворечивого набора аксиом, вследствие чего в дальнейшем эта тема практически не изучалась. Как уже упоминалось, функциональные зависимости являются частным случаем ND. Рассмотрим также другие примитивы, являющиеся обобщением FD.

Пусть $R(U)$ — схема отношения (в терминах реляционной модели данных), где R — имя отношения, а $U = \{ A_i, \dots, A_n \}$ — множество атрибутов. Заглавными латинскими буквами из начала алфавита будем обозначать одиночные атрибуты, буквами из конца алфавита — множества атрибутов. Домен атрибута A обозначим $dom(A)$. Через $t[X]$ для данного множества атрибутов X обозначим множество значений кортежа t на X. Отношением r над схемой $R(U)$ будем называть конечный набор кортежей над $R(U)$.

Функциональные зависимости

Функциональная зависимость (Functional Dependency, FD) $X \to Y$ означает, что каждому значению в левой части соответствует ровно одно значение в правой части.

Определение 1 (FD)**.** *Говорят, что атрибут B над отношением r функционально зависит от атрибута A, если любые два кортежа, совпадающие по значениям в A совпадают и по значениям в B.*

FD были впервые предложены E. Codd [4] и нашли применение во многих сферах, связанных с базами данных. Например, с помощью FD можно определить, находится ли отношение во второй или третьей нормальных формах, или в нормальной форме Бойса—Кодда.

Многозначные зависимости

Многозначные зависимости (Multivalued Dependencies, MVD) [6] показывают, что значения в правой части зависят только от значений в левой части.

Определение 2. *Многозначная зависимость над схемой отношения* $R(U)$ *имеет вид* $X \twoheadrightarrow Y$*, где* $X \cup Y \cup Z = U$*. Говорят, что отношение* r *над* $R(U)$ *удовлетворяет MVD, если значения в* Y *зависят только от значений в* X *и не зависят от значений в* Z*, то есть для любых двух кортежей* $t_1, t_2 \in r$ *существуют кортежи* $t_3, t_4 \in r$ *такие, что*

$$\begin{cases} t_1[X] = t_2[X] = t_3[X] = t_4[X] \\ t_3[Y] = t_1[Y] \\ t_3[Z] = t_2[Z] \\ t_4[Y] = t_2[Y] \\ t_4[Z] = t_1[Z] \end{cases}$$

Любая FD вида $X \to Y$ также является MVD $X \twoheadrightarrow Y$, в которой множество значений Y, соответствующих заданному значению X всегда имеет единичную мощность. MVD применяются, чтобы определить, находится ли отношение в четвёртой нормальной форме. Также MVD могут применяться при подготовке данных для машинного обучения [11].

Приближённые функциональные зависимости

Приближённые функциональные зависимости (Approximate Functional Dependencies, AFD) допускают некоторое отклонение от точных FD. AFD обозначаются $X \to_{\varepsilon} Y$, где ε — максимальное допустимое отклонение от FD $X \to Y$. Для вычисления отклонения используются различные метрики [9], например:

- g_1 (показывает количество пар строк с нарушениями):

$$g_1(X \to Y, r) = \frac{|\,\{\ (t_1, t_2) \mid t_1, t_2 \in r,\ t_1[X] = t_2[X],\ t_1[Y] \neq t_2[Y]\ \}\,|}{|r|^2};$$

- g_2 (показывает количество строк, нарушающих FD):

$$g_2(X \to Y, r) = \\ = \frac{|\,\{\, t_1 \mid t_1 \in r,\ \exists\, t_2 \in r : t_1[X] = t_2[X],\ t_1[Y] \neq t_2[Y] \,\}\,|}{|r|};$$

- g_3 (показывает, сколько строк нужно удалить, чтобы выполнялась точная FD $X \to Y$):

$$g_3(X \to Y, r) = \frac{|r| - \max\{|s| \mid s \subset r,\ s \vDash X \to Y\}}{|r|},$$

  где s — множество кортежей из r, для которого выполнена FD $X \to Y$. Последнее условие обозначается как $s \vDash X \to Y$.

Любая FD $X \to Y$ является также AFD $X \to_0 Y$ (то есть, $\varepsilon = 0$).

AFD могут применяться для тех же задач, что и точные FD, в тех случаях, когда данные содержат ошибки, неточности, или пропущенные значения.

Вес (k) числовой зависимости зависит от количества различных "нарушений" для каждого значения в левой части, в то время как погрешность (ε) приближённой функциональной зависимости так или иначе (в зависимости от метрики) зависит от общего количества таких "нарушений". ND тоже можно использовать как замену FD для "грязных" данных в случаях, когда нужно рассматривать "неточность" для каждого отдельного значения, например, если данные содержат погрешность измерений (см. таблицу 1).

Условные функциональные зависимости

Условные функциональные зависимости (Conditional Functional Dependencies, CFD) [5] используются при очистке данных (data cleansing) для ограничения FD, которая выполнялась бы на всём отношении, на отдельные кортежи.

Определение 3. *Условная функциональная зависимость φ над схемой отношения $R(U)$ — это пара $X \to Y$, t_p, где*

Таблица 1: Сравнение ND и AFD. Рассмотрим столбцы Название и Продолжительность (далее — Н, П). ND: Н $\xrightarrow{k}$ П выполняется при $k = 2$, а AFD: Н $\to_\varepsilon$ П выполняется только при довольно большом ε. Например, g_1(Н $\to$ П) $= 1$, g_2(Н $\to$ П) $= \frac{1}{2}$. Источник таблицы — работа N. Koudas и др. [10].

Источник	Название	Продолжительность
movies.aol.com	Aliens	110
finnguide.fi	Aliens	112
amazon.com	Clockwork Orange	137
movie-vault.com	A Beautiful Mind	144
walmart.com	A Beautiful Mind	145
tesco.com	Clockwork Orange	131

1. *$X, Y \subset U$ — множества атрибутов;*

2. *$X \to Y$ — это классическая FD. Говорят, что она вложена (embedded) в φ;*

3. *t_p — шаблонный кортеж (pattern tuple) с атрибутами из X и Y. Для каждого $B \in X \cup Y$, $t_p[B]$ — это или константа 'a' из dom(B), или безымянная переменная '_'.*

В таком случае говорят, что $X \to Y$ условно выполняется (conditionally holds) над множеством кортежей, заданном t_p.

При $t_p = (\_, \_, \ldots, \_)$ получаем частный случай CFD $X \to Y, t_p$ — FD $X \to Y$.

Для выявления несоответствий в данных (capturing data incosistencies) применяется задача поиска нарушений (violation detection problem): требуется найти кортежи, которые не удовлетворяют данным инстансам примитивов, т. е. нарушения ограничений. CFD хорошо подходит в качестве примитива в этой задаче для улучшения согласованности данных (improving data consistency) [8]. Было предложено несколько эффективных подходов к решению задачи поиска нарушений для CFD [11, стр. 27–30]. Также CFD могут применяться для интеграции данных, обмена данными (data exchange) и очистки данных.

3. Алгоритм поиска числовых зависимостей

3.1. Об алгоритме BBND

В оригинальной статье [3] описана процедура вывода одной числовой зависимости из множества известных. Однако, на практике чаще возникает задача поиска всех возможных ND на наборе данных, без известных ND. В ходе данной работы алгоритм BBND был адаптирован для этой задачи.

3.2. Основные понятия алгоритма BBND

Введём основные понятия, связанные с областью числовых зависимостей и алгоритмами их поиска, взятые из статьи P. Ciaccia и др. [3].

Определение 4 (ND)**.** *Пусть имеется схема $R(U)$, и конечное целое число $k \geq 1$. Будем говорить, что отношение r над схемой $R(U)$ удовлетворяет числовой зависимости $\delta : X \xrightarrow{k} Y$ для некоторых $X, Y \subset U$, если для любых $k+1$ кортежей $t_1, \ldots, t_{k+1}$ из r таких, что $t_1[X] = \cdots = t_{k+1}[X]$, существуют хотя бы два кортежа t_i и t_j $(i \neq j)$ таких, что $t_i[Y] = t_j[Y]$. В таком случае будем говорить, что δ является числовой зависимостью из X в Y с весом k, который будем также обозначать как* $\mathrm{w}(\delta)$.

Для ND $X \xrightarrow{k} Y$ будем X и Y называть левой и правой частями (LHS, RHS) соответственно. Количество атрибутов в левой и правой частях будем называть размерностью LHS и RHS соответственно.

Определение 5 (ND-граф)**.** *Пусть Δ — набор ND над схемой отношения $R(U)$. ND-графом $G_\Delta(\mathcal{V}, \mathcal{E})$, порождённым (induced by) Δ, называется ориентированный граф с вершинами $\mathcal{V} \subseteq 2^U$, рёбрами $\mathcal{E} = \mathcal{E}^f \sqcup \mathcal{E}^d$, и функцией маркировки рёбер (arc labeling function) $\omega : \mathcal{E} \to \mathbb{N}$ (весом) такой, что:*

1. *Для каждой ND $\delta : X \xrightarrow{k} Y \in \Delta$ в $\mathcal{V}$ есть вершины X и Y, а в $\mathcal{E}^f$ — сплошное ребро (full arc) $\langle X, Y \rangle$ (направленное из X в Y) такое, что $\omega(\langle X, Y \rangle) = k$. Таким образом, $\omega(\langle X, Y \rangle) = \mathrm{w}(\delta)$.*

2. *Для каждой составной (compound) вершины $X \in \mathcal{V}$, $X = A_1, \ldots, A_r$, $r > 1$ в $\mathcal{V}$ также есть r простых (simple) вершин $A_1, \ldots, A_r$, а в $\mathcal{E}^d$ — r пунктирных (dotted) рёбер $\langle X, A_1 \rangle, \ldots, \langle X, X_r \rangle$. При этом, $\omega(\langle X, A_i \rangle) = 1$.*

3. *Если в $\mathcal{V}$ есть пустой набор атрибутов $\perp$, то для каждой простой вершины $A_i \in \mathcal{V}$ в $\mathcal{E}^d$ есть пунктирное ребро $\langle A_i, \perp \rangle$. При этом, $\omega(\langle A_i, \perp \rangle) = 1$.*

В случае, когда $\Delta = \emptyset$, будем считать, что, для любого $X \subseteq U$, граф, включающий вершину X и дополненный по правилу 2 (если X состоит более чем из одного атрибута), также является ND-графом. Будем говорить, что такой граф порождён X.

Если $\Gamma \subseteq \Delta$, то G_Γ будем называть *ND-подграфом* G_Δ.

При необходимости будем записывать $\langle X, Y \rangle_k$, чтобы подчеркнуть, что $\omega(\langle X, Y \rangle) = k$.

Множество всех атрибутов, которые встречаются в вершинах графа G_Γ, будем обозначать $Attr(G_\Gamma)$.

Определение 6 (ND-путь из X)**.** *Пусть даны ND-граф $G_\Delta = (\mathcal{V}, \mathcal{E})$ и вершина $X \in \mathcal{V}$. ND-путём из X называется любой ND-подграф G_Δ, который может быть построен по следующим правилам:*

1. *ND-подграф, порождённый X является ND-путём из X.*

2. *Если G_Π — ND-путь из X, порождённый набором ND $\Pi \subset \Delta$, и $\langle W, Z \rangle_k \in \mathcal{E}^f$, где $W \subseteq Attr(G_\Pi)$, а $Z \nsubseteq Attr(G_\Pi)$, то ND-путь, порождённый $\Pi \cup \left\{ W \xrightarrow{k} Z \right\}$ также является ND-путём из X.*

3. *Никакой другой ND-подграф G_Δ не является ND-путём из X.*

ND-путь из X будем обозначать G^X_Π.

Весом $\omega(G^X_\Pi)$ ND-пути G^X_Π будем называть произведение весов всех его сплошных рёбер. Вес $G^X_\emptyset$ — ND-пути из X, не содержащего ни одного сплошного ребра, будем считать равным единице.

Определение 7 (ND-путь из X в Y)**.** *Пусть даны ND-путь G_{Π}^{X} из X и набор атрибутов Y. Если $Y \subseteq Attr(G_{\Pi}^{X})$, то G_{Π}^{X} также называется ND-путём из X в Y, или Y достижим из X (в G_{Π}^{X}).*

Определение 8 (минимальный ND-путь)**.** *ND-путь G_{Π}^{X} из X в Y называется минимальным для Y (Y-minimal) тогда и только тогда, когда не существует другого ND-пути $G_{\Pi'}^{X}$ из X в Y такого, что* $\Pi' \subsetneq \Pi$.

Определение 9 ("жадное доминирование")**.** *Пусть даны G_{Π}^{X}, G_{Γ}^{X} — два ND-пути из X. Говорят, что G_{Π}^{X} "жадно доминирует" (eagerly dominates) над G_{Γ}^{X}, если $Attr(G_{\Pi}^{X}) \supseteq Attr(G_{\Gamma}^{X})$ и $\omega(G_{\Pi}^{X}) \leq \omega(G_{\Gamma}^{X})$.*

3.3. Принцип работы алгоритма

Алгоритм BBND использует ND-граф и ND-пути для представления известных на данный момент ND и тех, которые могут быть выведены из них. Чтобы уменьшить пространство поиска (search space) используется метод ветвей и границ.

Вывод ND

Для вывода ND используется алгоритм, представленный на листинге 1. На вход алгоритму подаются ND-граф, содержащий уже известные ND, и два множества атрибутов X и Y. Сначала `RemoveUselessNDs` удаляет из ND-графа те сплошные рёбра, которые не лежат на пути из X в Y. Например, для набора рёбер $\{ \langle X, Y \rangle, \langle Y, Z \rangle \}$ будет удалено ребро $\langle Y, Z \rangle$. Далее каждый "активный" ND-путь (ND-путь помечается "активным", если его можно продолжить) расширяется добавлением одного сплошного ребра при помощи `SmartExtensions`. Если после этого ND-путь G_{Π_i} содержит Y, то существует ND $X \xrightarrow{\omega(G_{\Pi_i}^{X})} Y$. Иначе $G_{\Pi_i}^{X}$ должен быть продолжен и снова становится "активным", если выполняются следующие условия:

- $\omega(G_{\Pi_i}^{X}) < k_{RED}^{\perp}$, где $k_{RED}^{\perp}$ — наименьший вес ND $X \xrightarrow{k_{RED}^{\perp}} Y$, найденный на данный момент;

- нет другого “активного” пути, который “доминирует” (dominates) над $G^X_{\Pi_i}$;

- G_{Π_i} является минимальным для Y.

Вывод ND продолжается, пока есть “активные” пути. Наименьший найденный вес $k^{\perp}_{RED}$ принимается за вес ND $X \xrightarrow{k^{\perp}_{RED}} Y$ и эта ND записывается в ND-граф.

Листинг 1. Алгоритм BBND. Листинг взят из оригинальной статьи P. Ciaccia и др. [3].

Input: G_Δ, X, Y
Output: $k^{\perp}_{RED}(X, Y)$

```
k⊥_RED(X,Y) ← ∞
ActiveNDPaths ← { G^X_∅ }
if Y ⊈ Attr(G_Δ) then return k⊥_RED(X,Y)
RemoveUselessNDs(G_Δ, X, Y)
while ActiveNDPaths ≠ ∅ do
    G^X_Π ← ActiveNDPaths.Pop()
    foreach G^X_Πi ∈ SmartExtensions(G^X_Π) do
        if Y ⊆ Attr(G^X_Πi) then                      // Found a solution…
            if ω(G^X_Πi) < k⊥_RED(X,Y) then  // …better than the current one
                K⊥_RED ← ω(G^X_Πi)
            end
        else if ω(G^X_Πi) < k⊥_RED(X,Y) ∧ ¬IsDominated(G^X_Πi, ActiveNDPaths) ∧
         IsMinimal(G^X_Πi) then
            ActiveNDPaths.Push(G^X_Πi)
        end
    end
end
return k⊥_RED(X,Y)
```

3.4. Адаптация алгоритма BBND для поиска числовых зависимостей в таблице

В ходе адаптации алгоритма BBND, предложенного в оригинальной статье, для задачи поиска ND:

1. была разработана процедура построения начального графа;

2. использован “обход решётки” для вывода всех требуемых ND;

3. применено “отсечение” ND по размерности и весу.

Идеи данных модификаций были предложены автором и вдохновлялись алгоритмом Tane [12].

Построение стартового графа

Алгоритм BBND работает с заранее заданным множеством ND, однако на практике чаще требуется искать ND на наборе данных, для которого нет известных ND. Для этого была разработана процедура построения стартового ND-графа.

Стартовый ND-граф состоит из всех ND вида $A \xrightarrow{k} B$, где A и B — одиночные атрибуты. Для его построения используются разбиения (partitions) [12]. Множество классов эквивалентности по отношению $t \sim u \iff t[A] = u[A]\ \forall A \in X$ для некоторого X называется разбиением r относительно X и обозначается π_X. В работе Y. Huhtala и др. доказано, что FD $X \to A$ выполняется тогда и только тогда, когда $|\pi_X| = |\pi_{X \cup \{ A \}}|$. Этот результат можно обобщить для ND.

Обозначим $deps_W(\overline{z}) = |\,\{\, \overline{w} \mid \overline{z} \cap \overline{w} \neq \emptyset,\ \overline{w} \in W \,\}\,|$ — количество классов из W, пересекающихся с $\overline{z}$ для данного класса $\overline{z}$. Если $k = \max\limits_{\overline{x} \in \pi_X} \{\, deps_{\pi_Y}(\overline{x}) \,\}$, то выполняется ND $X \xrightarrow{k} Y$.

При этом, k не изменится, если вместо π_X использовать $\hat{\pi}_X$ — т. н. сокращённое разбиение (stripped partition) [12]. Сокращённое разбиение $\hat{\pi}_X$ определяется как разбиение π_X, из которой удалены все одноэлементные множества.

Вывод ND

Для вывода ND была применена техника, известная как обход решётки (lattice traversal), предложенная Y. Huhtala и др. [12]. На каждом шаге выводятся все ND с фиксированными размерностями LHS и RHS. Далее, мощность LHS или RHS увеличивается на единицу и процедура повторяется.

Для функциональных зависимостей существует понятие минимальной FD: FD $X \to Y$ называется минимальной, если FD $Z \to Y$ не выполняется для любого $Z \subsetneq X$. Если выполняется FD $Z \to Y$, то выполняется и FD $X \to Y$ для любого $X \supset Z$, поэтому при обходе решётки для функциональных зависимостей рассматривают только минимальные FD.

В случае ND аналогом минимальности является доминирование (опр. 9). Доказано, что в ND-граф достаточно добавлять только те ND, над которыми не доминирует ни одна ND, уже присутствующая в графе [3]. Однако, ND с меньшим весом может встретиться на более поздних уровнях решётки, поэтому при выводе ND необходимо обходить всю решётку.

“Отсечение” по размерности и весу

На таблице всегда существует ND между любыми двумя наборами атрибутов. Таким образом, для таблицы, содержащей N атрибутов, всегда существует ровно 2^{N+1} ND (с учётом ND вида $X \xrightarrow{k} X$). При этом, ND достаточно большой размерности обычно не представляют интереса. Поэтому были введены два параметра, ограничивающих максимальную размерность левой и правой частей выводимых ND.

Аналогично, был введён параметр, ограничивающий максимальный вес выводимых ND, так как чем больше вес ND, тем меньше информации о данных она даёт.

Алгоритм BBND использует метод ветвей и границ, поэтому ограничения по весу и размерности ND сильно уменьшают пространство поиска, в результате чего время работы алгоритма уменьшается.

По правилам вывода REDS [3] из ND $X \xrightarrow{k} Y$ можно вывести только ND $X' \xrightarrow{k'} Y'$, где $|X'| \geq |X|,\ |Y'| \geq |Y|,\ k' \geq k$, следовательно ограничения по весу и размерности не приводят к потере “значимых” ND.

ActiveNDPaths

Максимом Емельяновым [14] была разработана структура для хранения “активных” ND-путей (ActiveNDPaths в листинге 1). Она представляет собой множество ND-путей, упорядоченных по $P = |Attr(G_{\Pi}^{X}) \cap Y|$, где G_{Π}^{X} — ND-путь, Y — множество “целевых” атрибутов (требуется вывести ND $X \to Y$). Таким образом, P показывает “близость” G_{Π}^{X} к Y (стратегия best-first из оригинальной статьи). Более подробно ActiveNDPaths описан в работе [14].

3.5. Реализация

Иерархия классов показана на диаграмме (рис. 1). Зелёным цветом обозначены классы и методы класса Bbnd, реализованные в ходе данной работы, красным — классы и методы класса Bbnd, реализованные Максимом Емельяновым [14], синим — все остальные классы.

В Desbordante каждый алгоритм представлен отдельным классом, который наследуется от абстрактного класса Algorithm. Для алгоритма BBND был реализован класс Bbnd. Также были реализованы NDGraph, NDPath — классы для ND-графа и ND-пути соответственно. Класс очереди ActiveNdPaths был реализован Максимом Емельяновым. Класс, представляющий один инстанс числовой зависимости — ND — уже был реализован в Desbordante и использовался для валидации числовых зависимостей. Также была реализована функция `BuildInitialGraph` для построения стартового ND-графа.

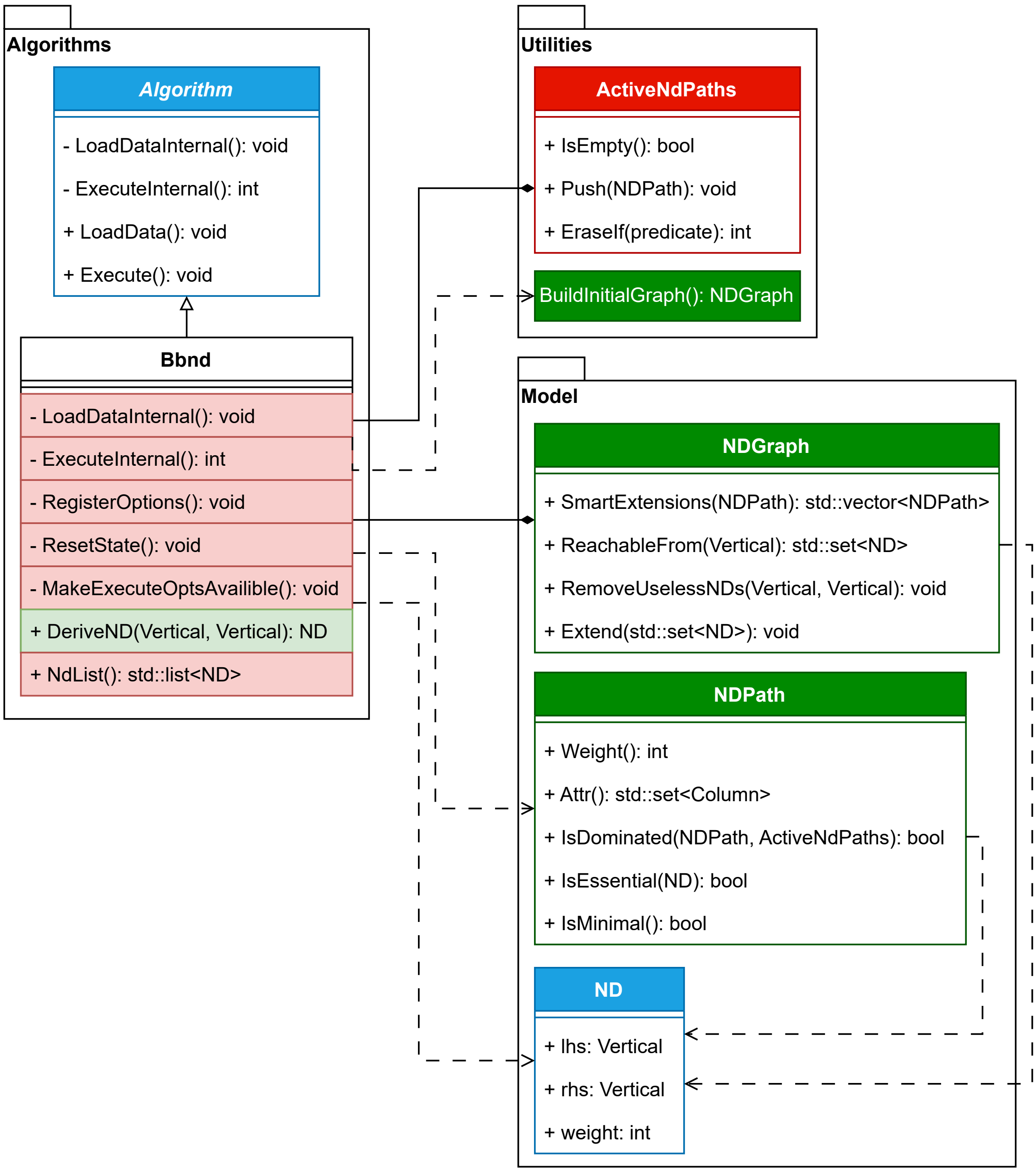

Рис. 1: Иерархия классов алгоритма BBND.

4. Апробация

4.1. Проверка корректности

Для проверки корректности были разработаны следующие тесты:

- тест для проверки корректности построения стартового графа;

- два теста для проверки корректности вывода ND:
  - вывод ND с $|LHS| = 2$ и $|RHS| = 2$;
  - вывод ND с $|LHS| = 3$ и $|RHS| = 3$.
- два теста для проверки корректности работы всего алгоритма BBND:
  - поиск ND с $|LHS| = 2$ и $|RHS| = 2$;
  - поиск ND с $|LHS| = 3$ и $|RHS| = 3$.
- тест для проверки корректности ограничения по весу ND.

Все тесты проводились на данных, представленных в таблице 2. Тестовые данные также доступны в репозитории Desbordante на GitHub[2].

Таблица 2: Тестовые данные для проверки корректности алгоритма BBND.

Col0	Col1	Col2	Col3	Col4	Col5	Col6
1	a	x	1.233	-	11	aa
1	a	x	0	8	22	
1	a	xy	0	8	33	
1	b	y	hjkl	444	44	aa
1	b	y	hjkl	444	44	bb
1	b	xy	hjkl	444	55	aa
1	c	z	0	9	66	-
1	c	z	0	9	66	-
1	c	z	999	-	77	bb
1	d	k	hjkl	555	88	aa
1	d	k	hjkl	555	88	aa
1	d	abc	hjkl	555	99	

[2] https://github.com/Desbordante/desbordante-core/blob/main/test_input_data/TestND.csv

Некорректный вес ND

В силу неполноты правил вывода ND при помощи алгоритма BBND не всегда возможно вывести ND с корректным весом. Рассмотрим, например, $X = \{ Col1, Col2 \}$, $Y = \{ Col4, Col6 \}$ на таблице 2. Каждому значению из X соответствует не более двух различных значений из Y (например, значению $\{ c, z \}$ соответствуют $\{ 9, - \}$ и $\{ -, bb \}$), поэтому, по определению ND, выполняется ND $X \xrightarrow{2} Y$. Однако, используя алгоритм BBND, можно получить только ND $X \xrightarrow{4} Y$, например, из ND $\{ Col1 \} \xrightarrow{2} \{ Col4 \}$, $\{ Col1 \} \xrightarrow{2} \{ Col4 \}$, $\{ Col2 \} \xrightarrow{2} \{ Col4 \}$, $\{ Col2 \} \xrightarrow{2} \{ Col6 \}$. Соответствующая часть ND-графа изображена на рис. 2.

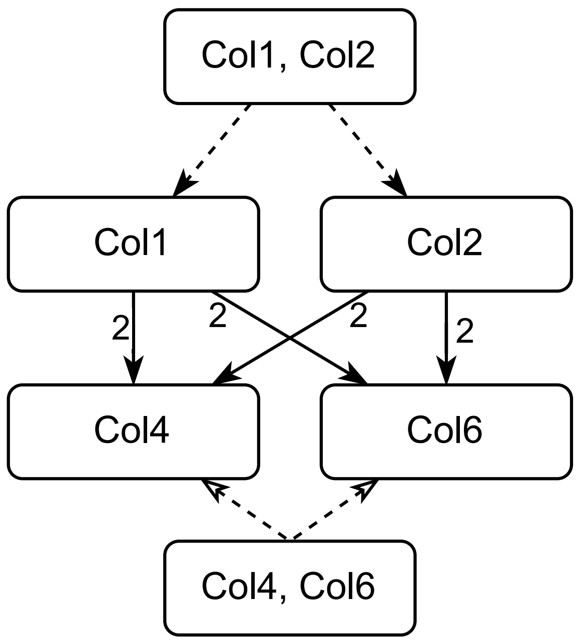

Рис. 2: Часть ND-графа, по которому выводится ND с некорректным весом

4.2. Эксперименты

Максимом Емельяновым [14] были проведены эксперименты, по результатам которых было принято решение не включать данную реализацию алгоритма BBND в основной репозиторий Desbordante.

5. Заключение

В ходе данной работы были выполнены следующие задачи:

- исследованы числовые зависимости и показаны отличия от других примитивов, обобщающих функциональные зависимости;
- реализованы следующие компоненты алгоритма BBND:
  - классы NDGraph и NDPath, представляющие ND-граф и ND-путь соответственно;
  - функция `BuildInitialGraph` для поиска ND между одиночными атрибутами;
  - метод `DeriveND` класса Bbnd для вывода ND.
- разработаны тесты на корректность реализации.

Код разработанного алгоритма доступен на GitHub[3] (имя пользователя — p-senichenkov).

[3] https://github.com/Sneper-Breeze/Desbordante/tree/NDs

Список литературы

[1] Armstrong William Ward. Dependency structures of data base relationships // IFIP congress / Geneva, Switzerland. — Vol. 74. — 1974. — P. 580–583.

[2] Chernishev George et al. Desbordante: from benchmarking suite to high-performance science-intensive data profiler. — 2023. — URL: https://arxiv.org/abs/2301.05965v1.

[3] Ciaccia Paolo, Golfarelli Matteo, Rizzi Stefano. Efficient derivation of numerical dependencies // Information Systems. — 2013. — Vol. 38, no. 3. — P. 410–429. — URL: https://www.sciencedirect.com/science/article/pii/S0306437912001044.

[4] Codd Edgar F. Further normalization of the data base relational model // Data base systems. — 1972. — Vol. 6. — P. 33–64.

[5] Conditional Functional Dependencies for Data Cleaning / Philip Bohannon, Wenfei Fan, Floris Geerts et al. // 2007 IEEE 23rd International Conference on Data Engineering. — 2007. — P. 746–755.

[6] Fagin Ronald. Multivalued dependencies and a new normal form for relational databases // ACM Trans. Database Syst. — 1977. — Sep. — Vol. 2, no. 3. — P. 262–278. — URL: https://doi.org/10.1145/320557.320571.

[7] Grant John, Minker Jack. Inferences for numerical dependencies // Theoretical Computer Science. — 1985. — Vol. 41. — P. 271–287. — URL: https://www.sciencedirect.com/science/article/pii/0304397585900751.

[8] Improving data quality: consistency and accuracy / Gao Cong, Wenfei Fan, Floris Geerts et al. // Proceedings of the 33rd International Conference on Very Large Data Bases. — VLDB '07. — VLDB Endowment, 2007. — P. 315–326.

[9] Kivinen Jyrki, Mannila Heikki. Approximate dependency inference from relations // Database Theory — ICDT '92 / Ed. by Joachim Biskup, Richard Hull. — Berlin, Heidelberg : Springer Berlin Heidelberg, 1992. — P. 86–98.

[10] Metric Functional Dependencies / Nick Koudas, Avishek Saha, Srivastava Divesh, Suresh Venkatasubramanian // 2009 IEEE 25th International Conference on Data Engineering. — 2009. — P. 1275–1278.

[11] Song Shaoxu, Chen Lei. Integrity Constraints on Rich Data Types. — Springer, 2023. — Mar. — ISBN: 978-3-031-27176-2.

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

[13] Ziawasc Abedjan, Lukasz Golab, Felix Naumann. Profiling Relational Data: A Survey // The VLDB Journal. — 2015. — Aug. — Vol. 24, no. 4. — P. 557–581.

[14] Емельянов Максим. Внедрение алгоритма BBND в проект с открытым исходным кодом "Desbordante". — 2024. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/BBNDbenchmark-EmelyanovMaksim-2024autumn.pdf.