Санкт-Петербургский государственный университет

Кафедра системного программирования

Расширение возможностей профилировщика данных Desbordante по работе с графовыми зависимостями

Черников Антон Александрович, группа 23.М04-мм

25 апреля 2025 г.

Научный руководитель: ассистент кафедры ИАС Г.А. Чернышев

Санкт-Петербург
2025

Введение: Desbordante

Desbordante — это инструмент для профилирования данных, разрабатываемый группой студентов под руководством Г. А. Чернышева.

- Способен обнаруживать закономерности в данных с использованием различных алгоритмов
- Весь проект имеет открытый исходный код
- Высокопроизводителен

Введение: функциональные зависимости

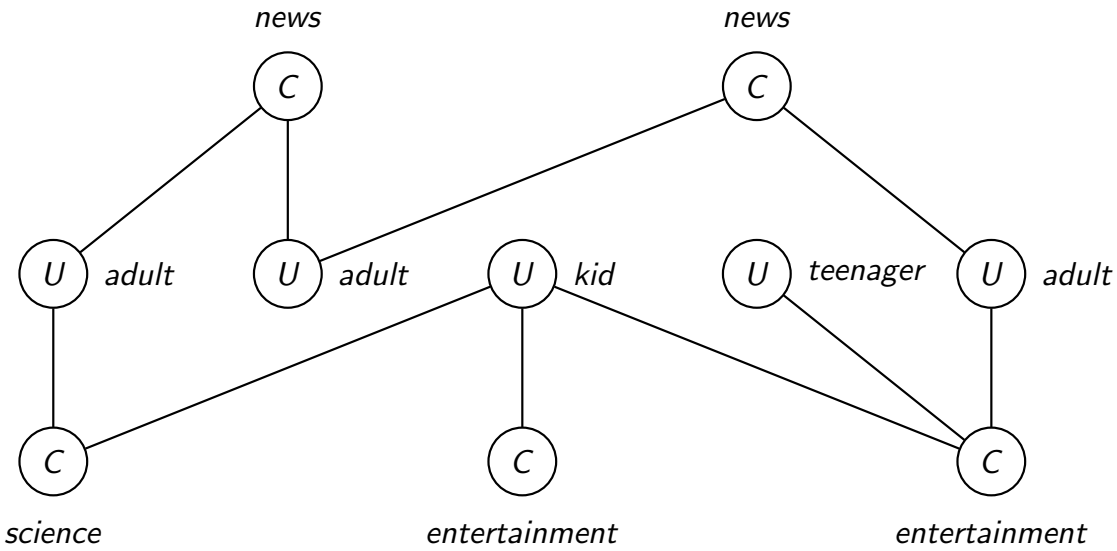
Определение: Отношение R удовлетворяет функциональной зависимости $X \to Y$ (где $X, Y \subset R$) тогда и только тогда, когда для любых кортежей $t_1,\ t_2 \in R$ выполняется: если $t_1[X] = t_2[X]$, то $t_1[Y] = t_2[Y]$.

Данные о студентах и их оценках

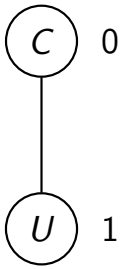
ID	Name	Course	Grade
1	Alice	Math	A
2	Bob	Math	B
3	Charlie	Science	A
1	Alice	Science	B
2	Bob	Science	A

Зависимость $\{ID \to Name\}$ выполняется, а зависимость $\{Name \to Course\}$ — нет.

Введение: графовые зависимости



Связь каналов C с пользователями U
Атрибуты у C — $\{topic\}$, у U — $\{age\_group\}$



$\{0.topic = news\} \rightarrow \{1.age\_group = adult\}$

Графовая зависимость[1][2]

[1] "Functional Dependencies for Graphs", Wenfei Fan, Yinghui Wu, Jingbo Xu, SIGMOD'16

[2] "Discovering Graph Differential Dependencies", Yidi Zhang, Selasi Kwashie, Michael Bewong, Junwei Hu, Arash Mahboubi, Xi Guo, Zaiwen Feng

Предыдущие результаты и позиционирование результатов этого семестра

Во втором семестре было сделано:

- Реализован алгоритм поиска **графовых функциональных зависимостей**
- Реализована возможность запускать его из Python-программ и через консоль

В третьем семестре будем заниматься:

- Улучшением реализации алгоритма поиска **графовых функциональных зависимостей**
- Созданием обзора алгоритма поиска **графовых дифференциальных зависимостей**

Постановка задачи

Целью работы является расширение и улучшение инструментария Desbordante для работы с графовыми зависимостями.
Для достижения этой цели были поставлены следующие **задачи**:

- Произвести анализ и улучшение алгоритма поиска функциональных зависимостей в соответствии с предложениями рецензентов кода
- Создать скрипт-пример работы алгоритма поиска графовых зависимостей на языке программирования Python
- Выполнить обзор алгоритма поиска графовых дифференциальных зависимостей и описать его основные свойства

Улучшение алгоритма поиска графовых функциональных зависимостей

- Переработаны пространства имён: алгоритмы в *algos*, периферия в *gfd* и *model*
- Созданы директории для валидации и поиска графовых зависимостей
- Заменена двойная проверка наличия элемента на метод *try_emplace*
- Добавлены комментарии и описания алгоритмов
- Упрощён код: ссылки в *CmpCallback* заменены на лямбда-выражения
- Циклы заменены на алгоритмы (например, *std* :: *ranges* :: *min*)
- Рассмотрена возможность использования *std* :: *unordered_map*
- *emplace_back* используется для создания объектов в контейнере
- Извлечён общий функционал из методов *Validate* и *Support*
- Параметры алгоритма передаются через поля класса (*graph_*, *k_* и *sigma_*)

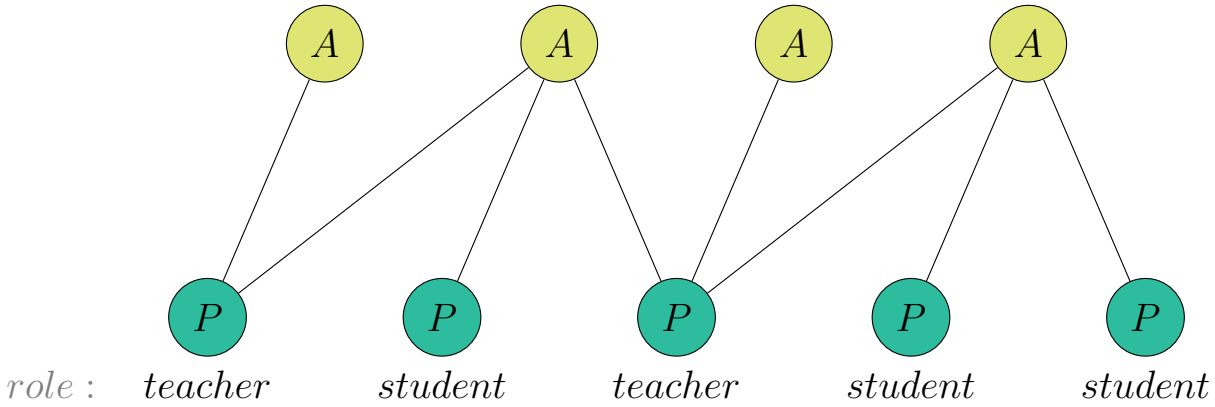
Тесты:

- Создан отдельный класс для конфигурации тестов
- Общая логика вынесена в отдельные функции
- Создан файл *all_gfd_paths.h* для путей к входным данным

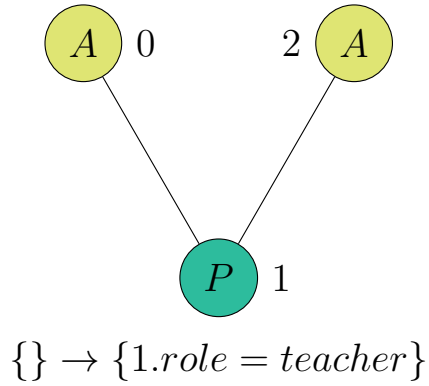
Пример работы алгоритма поиска графовых функциональных зависимостей

- В Desbordante нет технического писателя
- Отсутствует документация по пользованию алгоритмами
- Существует необходимость снабжать каждый алгоритм примером на Python

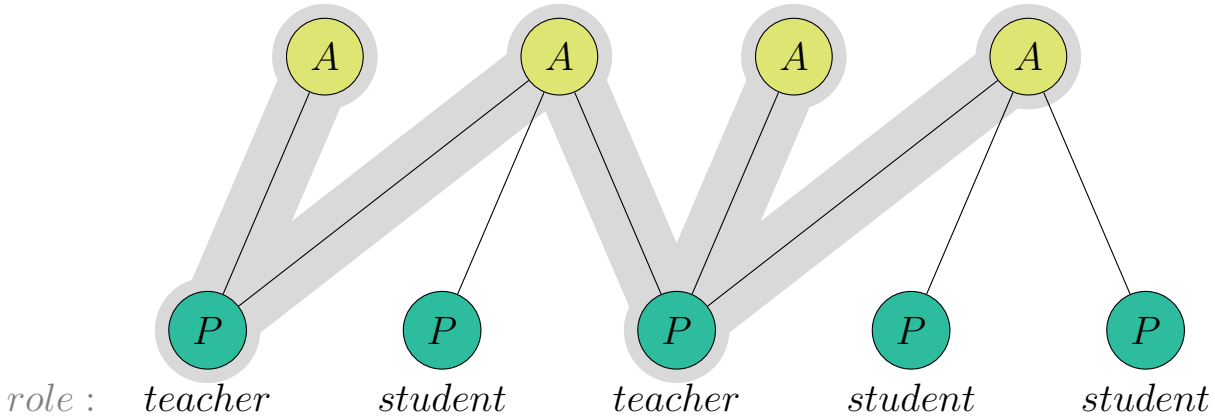
Пример работы алгоритма поиска графовых функциональных зависимостей



Пример графа



Графовая зависимость



Вложения

Пример: вывод в консоль

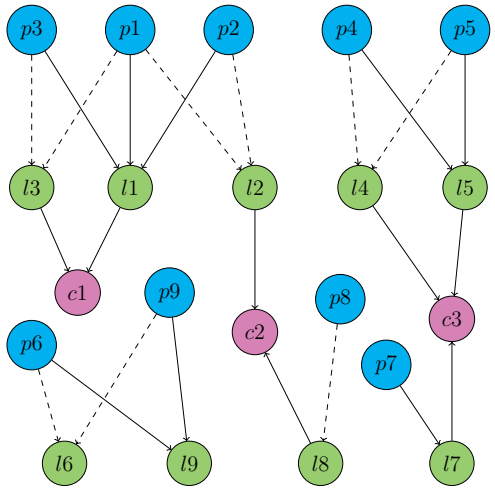
```
GFDs are functional dependencies that consist of a pattern - a graph that specifies the scope - and a rule. The nature of this object will become clearer through
 the example that follows.

Let's analyze GFD mining through an example. Look at the graph presented on the top left in the figure. It describes the connections between scientific articles
and their authors. The vertices of this graph have two labels: Article (A) and Person (P). Each vertex has its own set of attributes depending on the label.

Article:
- title denotes the title of the article.

Person:
- name denotes the name of a person,
- role can take one of two values: "teacher" or "student".

The discovery algorithm, in addition to the graph, takes two parameters as input:
- k: the maximum number of vertices in the pattern,
- sigma: the minimum frequency of GFD occurrences in the original graph.

Let's run the algorithm and look at the result. We will set k=3 and sigma=2.

Desbordante > Mined GFDs: 1

Let's print found dependency (in DOT language):

1.role=teacher
graph G {
0[label=article];
1[label=person];
2[label=article];
0--1 [label="*"];
1--2 [label="*"];
}

It may be difficult to interpret, so let's rewrite it to a more human-readable format. Note that the empty line immediately following the colon (":") indicates t
hat the left-hand side of the dependency has no conditions. Conversely, if the right-hand side of the dependency had no conditions, the second line would be empt
y.

     0    1    2
    (A)--(P)--(A)
{} --> {1.role=teacher}

The mined dependency can also be seen on the right in the figure.

The discovered dependency can be expressed as the following fact: If a person has two published articles, then they are a teacher.
```

Алгоритм поиска GDD: Псевдокод

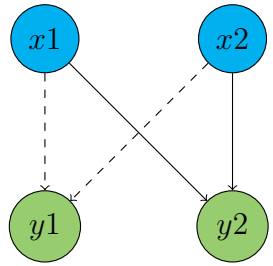
Алгоритм: GDDMiner
Input: Граф G, частотный порог τ, ограничения расстояний Δ (опционально)
Output: Минимальное покрытие Σ_c GDD

1: $\Sigma := \emptyset, \Sigma_c := \emptyset$
   /* получение частых неизбыточных паттернов */
2: $Q \leftarrow rGrami(G, \tau)$
   /* поиск GDD на основе полученных паттернов */
3: **for** $Q_i[\overline{z_i}] \in Q$ **do**
4: $\quad H(Q_i[\overline{z_i}], G) \leftarrow hMatches(Q_i[\overline{z_i}], G)$
5: $\quad \Sigma_i \leftarrow redGDDs(H, \Delta)$
6: $\quad \Sigma \leftarrow \Sigma \cup \Sigma_i$
   /* отбрасывание избыточных GDD */
7: $\Sigma_c \leftarrow cover(\Sigma)$
8: **return** Σ_c

Алгоритм поиска GDD: Таблица вложений



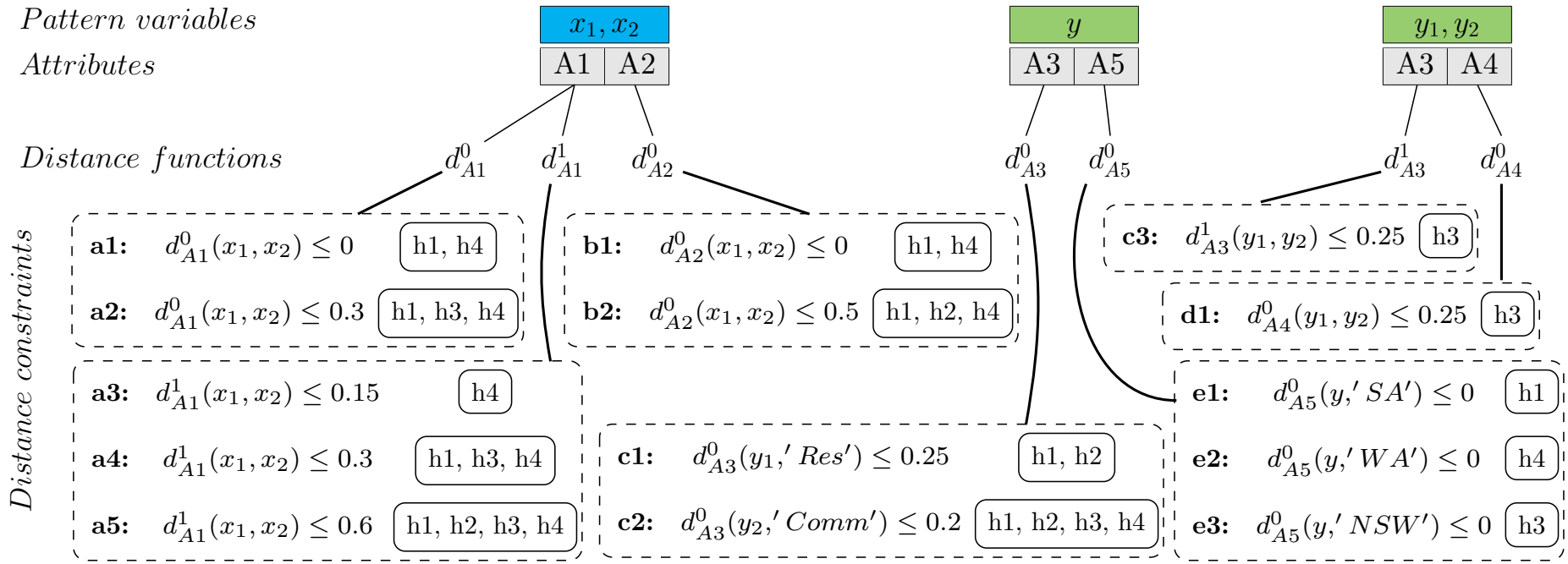
Граф



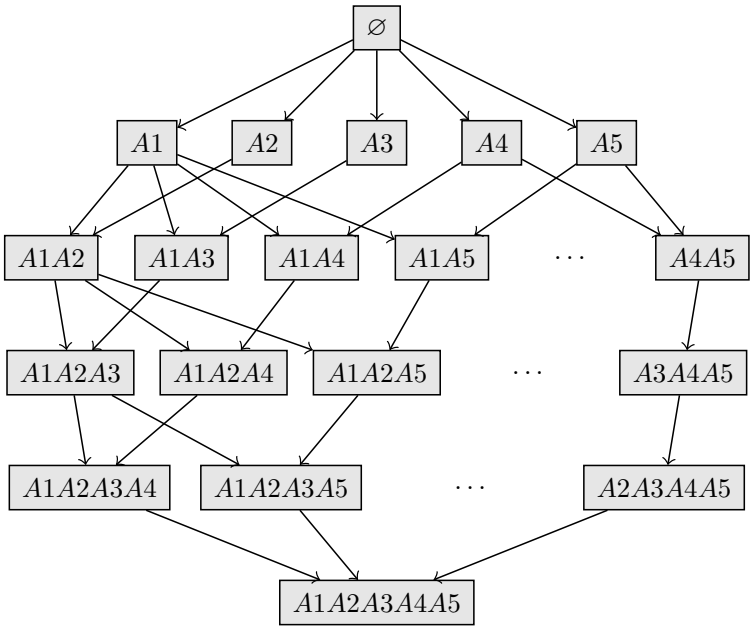
Паттерн

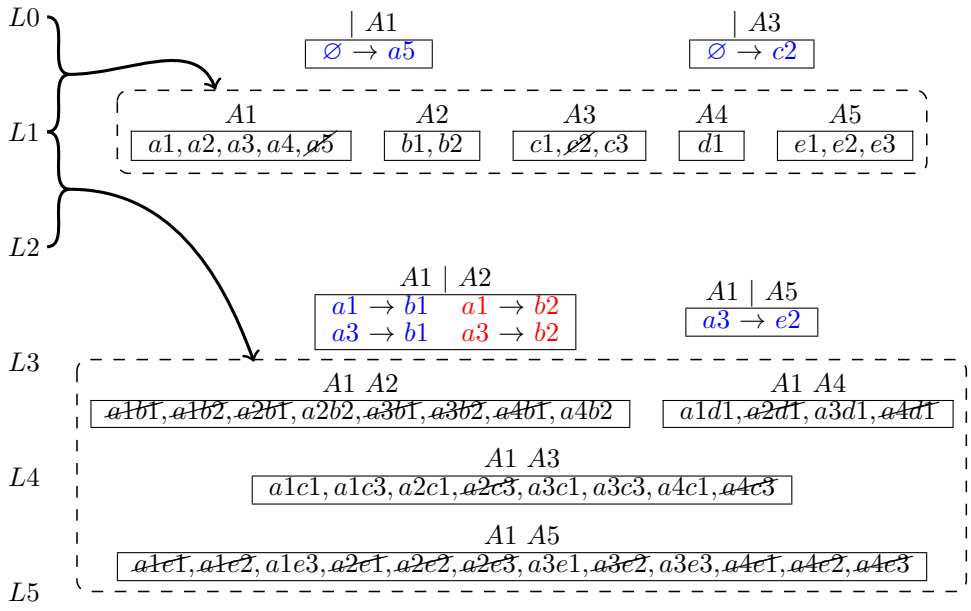
	x_1 : *person*			y_1 : *location*				x_2 : *person*			y_2 : *location*			
	id	Name (A1)	DOB (A2)	id	Type (A3)	Street name (A4)	State (A5)	id	Name (A1)	DOB (A2)	id	Type (A3)	Street name (A4)	State (A5)
h1	p1	William Johnson	1970.7.31	l3	Res	46 Adrian Ave	SA	p3	Bill Johnson	7/31/70	l1	Comm	17 Tea Tree Gully	SA
h2	p1	William Johnson	1970.7.31	l2	Res	28 Main Rd	NSW	p2	Mary Smith	Jun. 1990	l1	Comm	17 Tea Tree Gully	SA
h3	p4	Paul H. Colbert		l4	Com	93 High St	NSW	p5	Colber P. H.	25/12/89	l5	Com	93 High Street	NSW
h4	p6	Tom Engel	1 Jun. 1956	l6		99 Mawson blvd	WA	p9	Engel, Thomas	06/01/65	l9	Com	199 Main Road	WA

Алгоритм поиска GDD: Ограничения расстояния



Алгоритм поиска GDD: Решётка атрибутов





Результаты работы

- Произведены анализ и улучшение алгоритма поиска функциональных зависимостей в соответствии с предложениями рецензентов кода
- Создан скрипт-пример работы алгоритма поиска графовых зависимостей на языке программирования Python
- Выполнен обзор алгоритма поиска графовых дифференциальных зависимостей и описаны его основные свойства

Исходный код доступен на GitHub[1].

[1] https://github.com/Desbordante/desbordante-core/pull/465