МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ

ТЕХНОЛОГИЙ

Реферат

Алгоритм MAFIA (maximal frequent itemset algorithm)

Подготовила

Студентка 3 курса 6КБ группы

Шкурная М. Д.

Преподаватель

Дигрис А. В.

Минск, 2025

**Содержание**

Введение.............................................................................................................3

1. Дерево набора элементов-кандидатов.........................................................3

2. Алгоритмические описания..........................................................................3

2.1 Сокращение поискового пространства (Обрезка дерева)…………......4

2.2 Отсечение родительской эквивалентности (PEP)..................................4

2.3 FHUT..........................................................................................................5

2.4 HUTMFI.....................................................................................................5

2.5 Динамическое переупорядочение...........................................................5

3. Вертикальное растровое представление......................................................6

4. Описание решения.........................................................................................7

Заключение.........................................................................................................9

Список использованных источников.............................................................10

**Введение**

Ассоциативный анализ данных является мощным инструментом для выявления скрытых взаимосвязей и ассоциаций между элементами в больших объемах данных. Одним из ключевых алгоритмов ассоциативного анализа является алгоритм MAFIA (Maximal Frequent Itemset Algorithm).

MAFIA - это эффективный алгоритм, разработанный для обнаружения максимально частых наборов элементов в данных. Он позволяет идентифицировать наборы элементов, которые встречаются в данных с высокой частотой, превышающей заранее заданный порог поддержки. Однако, в отличие от других алгоритмов, MAFIA определяет максимально частые наборы элементов, которые не содержат других частых наборов. Это позволяет получить более компактные и информативные результаты, фокусируясь только на наиболее значимых шаблонах.

Алгоритм MAFIA находит широкое применение в различных областях, включая анализ покупательского поведения, биоинформатику, вебаналитику и другие. Он позволяет выявлять интересные ассоциации и закономерности в данных, что может привести к важным открытиям и принятию более обоснованных решений. Благодаря своей эффективности и способности обрабатывать большие объемы данных, алгоритм MAFIA становится все более популярным среди исследователей и практиков.

# **Дерево наборов элементов-кандидатов**

Процесс генерации наборов элементов-кандидатов выполняется с использованием поиска в глубину, и этот процесс можно представить в виде дерева наборов элементов-кандидатов. С каждым шагом вниз по дереву добавляется один элемент в набор элементов. По мере того, как наборы товаров становятся все больше и больше, процент клиентов, у которых есть этот набор товаров (или процент поддержки), будет становиться все меньше и меньше. В конце концов, значение поддержки упадет ниже минимальной поддержки, необходимой для того, чтобы набор элементов считался частым.

Глядя на лексикографическое дерево на рис. 1, мы можем провести линию, пересекающую все точки, в которых появление расширяемого набора элементов меняется от частого к редкому. Все наборы элементов, находящиеся над этой линией, называются максимально частыми наборами элементов. По принципу Априори никакие расширения набора элементов ниже этой линии не могут быть частыми, поскольку все они содержат внутри себя другие наборы элементов, которые оказались нечастыми.

**Алгоритмическое описание**

В простом алгоритме (см. рисунок 2) мы проходим лексикографическое дерево в порядке чистой глубины. На каждом узле n каждый элемент в хвосте узла генерируется и считается возможным 1расширением. Если поддержка {n head} U {1-расширение} меньше minSup, то мы можем остановиться по принципу Априори, поскольку любой набор элементов из этого возможного 1-расширения будет иметь редкое подмножество. Если ни одно из 1-расширений не приводит к частому набору элементов, узел является листом.

Когда мы доходим до листа на дереве, у нас есть кандидат на вступление в MFI (максимально частый набор элементов). Однако частое надмножество набора элементов, возможно, уже было обнаружено. Поэтому нам нужно проверить, является ли надмножество кандидата набором элементов, который уже находится в MFI. Если надмножество не существует, то мы добавляем набор элементов-кандидатов в MFI. Важно отметить, что при обходе в глубину нам никогда не нужно беспокоиться о удалении подмножеств из MFI. Это происходит потому, что наборы позиций, уже вставленные в MFI, будут лексикографически упорядочены ранее.

### ***2.1 Сокращение поискового пространства (Обрезка дерева)***

В некоторых случаях ветви дерева набора элементов-кандидатов могут быть «обрезаны», что приводит к меньшему количеству наборов элементов, которые необходимо проверить, и, следовательно, к более быстрому времени работы. В этом разделе объясняется, что делает каждый из этих шагов обрезки.

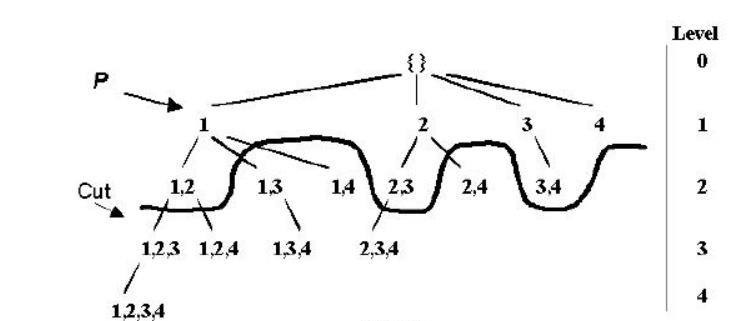


рис. 2

## ***2.2 Отсечение родительской эквивалентности (PEP)***

Один из методов сокращения включает в себя сравнение наборы транзакций каждой пары родитель/потомок. Пусть x — голова узла n, а y — элемент в хвост узла n. Если t(x)  t(y), то любая транзакция, содержащая x, также содержит y. Это гарантирует, что любое частое набор элементов z, содержащий x, но не y, имеет частый расширенный набор. Поскольку нам нужна только максимальная частота наборы элементов, нет необходимости считать наборы элементов, содержащие x. Следовательно, мы можем переместить элемент y из хвоста в голова. Для узла n x=x U y и элемент y удаляется. из н-х хвоста.

## ***2.3 FHUT***

Другой тип обрезки — обрезка надмножества. Узле n максимально возможная частота набор элементов, содержащийся в поддереве с корнем в n, является HUT n. (головное объединение хвоста), как заметил Баярдо. Если это HUT обнаруживается часто, нам никогда не придется исследовать любые подмножества HUT и, таким образом, можно исключить всю поддерево с корнем в узле n. Мы имеем в виду этот метод обрезка как обрезка FHUT (Frequent Head Union Tail). FHUT можно вычислить, исследуя крайний левый путь в поддереве с корнем в каждом узле. Фактически, поскольку алгоритм глубины уже исследует крайний левый путь, никаких дополнительных вычислений не требуется (см. рисунок 6). А Недостатком FHUT по сравнению с HUTMFI является что самый левый путь содержит наибольшее количество элементов любого пути в поддереве. Таким образом, хотя поддерево с корнем в этом узле обрезается, значительная часть дерева все еще остается генерируется и подсчитывается.

## ***2.4 HUTMFI***

Существует два метода определения того, является ли набор элементов x встречается часто: прямой подсчет поддержки x и проверяем, было ли уже расширенное множество x объявлено частым; FHUT использует первый метод. последний подход определяет, находится ли надмножество HUT в МФО. Если надмножество существует, то HUT должен быть частый и поддерево с корнем в узле, соответствующем до X можно обрезать. Мы называем этот тип обрезки надмножества HUTMFI. Обратите внимание, что HUTMFI не расширяет какие-либо детей, чтобы проверить успешность обрезки надмножества, в отличие от FHUT, где находится самая левая ветвь поддерева. исследовал. Поэтому в целом HUTMFI предпочтительнее, чем FHUT обрезка.

## ***2.5 Динамическое переупорядочение***

Преимущество динамического переупорядочения потомков каждый узел основан на поддержке, а не на следовании лексикографический порядок имеет значение. Однако динамичный переупорядочение требует подсчета поддержки всех расширения узла и, следовательно, не будет чистым обходом пространства в глубину.

Обратите внимание, что большинство элементов хвоста узла будут не быть частыми продлениями, а эти же нечастые предметы появляются во многих хвостах ниже. Алгоритм, который подстригает хвост только до частого выпрямления на более высоком уровне сэкономит много вычислений. Например, на рисунке 1 если набор элементов {1, 4} учитывается и оказывается нечастым, тогда элемент 4 можно обрезать из хвоста всех узлов в это поддерево, и наборы элементов {1, 2, 4}, {1, 3, 4} никогда не должны быть пересчитаны.

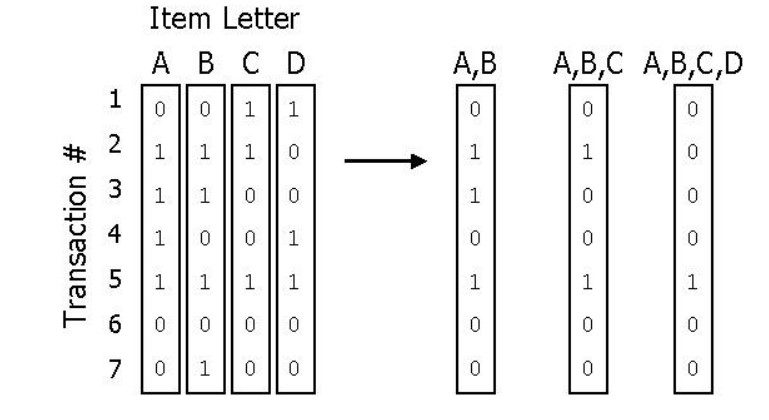
Особо следует отметить, что ПКП можно применять гораздо быстрее. с динамическим переупорядочением. Поскольку ПКП зависит от поддержка каждого ребенка относительно родителя, мы можем двигаться все элементы, за которые ПЭП держится от хвоста до головы сразу, быстро уменьшая размер хвоста. Порядок хвостовых элементов также важен. рассмотрение. Заказ хвостовых элементов (возможно детей) за счет увеличения поддержки сохранит пространство поиска как можно меньше.Эту эвристику впервые использовал Баярдо. Все компоненты алгоритма совместимы друг с другом.

# **Вертикальное растровое представление**

MAFIA эффективно хранит транзакционную базу данных в виде серии вертикальных битовых карт (рис. 2), где каждая битовая карта представляет набор элементов в базе данных, а бит в каждой битовой карте показывает, имеет ли данный клиент соответствующий набор элементов.

Первоначально каждое растровое изображение соответствует набору из 1 элемента или одному элементу. Наборы элементов, частота которых проверяется в базе данных, рекурсивно становятся все длиннее и длиннее, а вертикальное растровое представление прекрасно работает в сочетании с этим расширением набора элементов. Например, растровое изображение для набора элементов (a, b) можно создать, просто выполнив операцию «И» для всех битов в растровых изображениях для (a) и (b). Затем, чтобы подсчитать количество клиентов, у которых есть (a, b), все, что нужно сделать, это подсчитать количество единиц в битовой карте (a, b), равное количеству клиентов, у которых есть (a, b). Очевидно, что растровая структура идеальна как для генерации набора элементов-кандидатов, так и для подсчета поддержки.

рис. 2



# 

# **Описание решения**

Для реализации алгоритма MAFIA в Python был использован следующий подход:

1. Загрузка и предварительная обработка данных:

* 1. Чтение данных из файла в формате CSV
  2. Преобразование данных в формат, подходящий для построения FP-дерева

2. Построение FP-дерева:

* 1. Реализация класса FPTree для построения и хранения FPдерева
  2. Заполнение дерева на основе загруженных данных

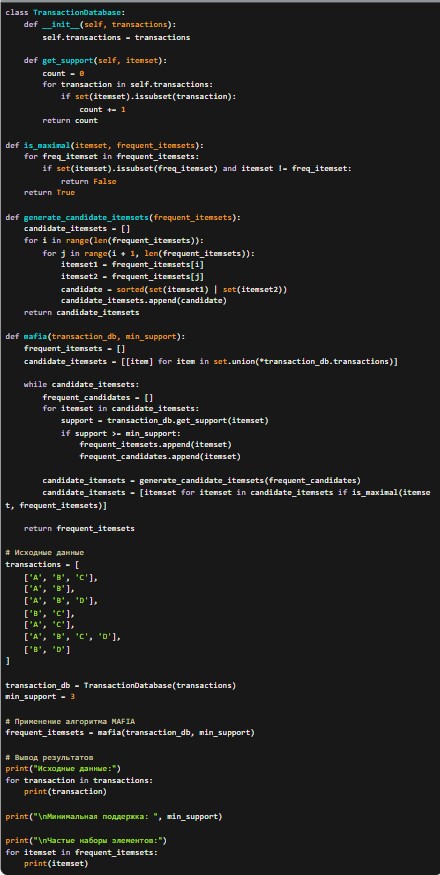
3. Поиск максимальных частых наборов:

* 1. Реализация функции find\_maximal\_frequent\_itemsets, выполняющей обратный обход FP-дерева
  2. Применение эвристики для отсечения неподходящих наборов на основе свойства максимальности

4. Вывод результатов:

* 1. Отображение найденных максимальных частых наборов

4.2 Оценка качества результатов, например, по количеству найденных наборов, времени работы алгоритма и т. д.



В этой версии примера добавлены исходные данные transactions, которые представляют собой список транзакций. Минимальная поддержка min\_support установлена равной 3. Результат выполнения программы будет выглядеть следующим образом:



В результате выполнения алгоритма MAFIA найдены все максимальные частые наборы элементов, удовлетворяющие минимальной поддержке 3. В данном случае это наборы ['A', 'B'], ['A', 'C'] и ['B', 'C'].

# **Вывод**

Алгоритм MAFIA в контексте анализа больших объемов данных и выявления часто встречающихся наборов элементов. Этот алгоритм обладает значительными преимуществами, такими как эффективная обработка больших наборов данных, способность работать с различными типами данных и высокая точность выявления максимально частых наборов элементов. Он является важным инструментом в таких областях, как анализ данных, ассоциативное извлечение информации и обнаружение ассоциативных правил. Однако, несмотря на его эффективность, алгоритм MAFIA также имеет недостатки, включая ограничения на размер данных и потребление вычислительных ресурсов. Тем не менее, с учетом быстрого развития вычислительных технологий и методов оптимизации, алгоритм MAFIA остается актуальным и востребованным инструментом для анализа данных и извлечения ценной информации из больших объемов данных.

**Список используемой литературы**

1.Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

2.Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. Neural networks, 2(1), 53-58.

3.Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008).

Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning (pp.

1096-1103).

4.Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In Advances in neural information processing systems (pp. 153-160).

5.Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes.

arXiv preprint arXiv:1312.6114.