УДК 004.62

СПОСОБ СЖАТИЯ ПРОМЕЖУТОЧНЫХ ДАННЫХ ПРИ РАСПРЕДЕЛЕННОМ ПОИСКЕ АССОЦИАТИВНЫХ ПРАВИЛ

Е. О. Храмшина, аспирант, ассистент кафедры ВПМ РГРТУ, Рязань, Россия;
orcid.org/0000-0002-4490-8403, e-mail: evgenia.khramshina@gmail.com

Целью работы *является сокращение объема памяти, занимаемого трехмерным массивом для хранения и передачи при распределенном поиске ассоциативных правил. Сокращение объема достигается за счет разреженности массива и использования нефиксированного размера значений элементов массива. Элементы массива с ненулевыми значениями записываются в виде пары чисел: смещения относительно предыдущего элемента с ненулевым значением и собственно значения. Чтобы разделять числа, один из битов в байте используется как служебный для пометки конца значения. Проведены эксперименты и выявлено, что способ позволяет сократить размер файла в среднем на 74 % по сравнению с исходным массивом. Для проведения экспериментов разработано программное обеспечение на языке Java. Разработанный способ сжатия наряду с разработанным алгоритмом поиска ассоциативных правил 3D2ARM будет использован в разработке распределенного поиска ассоциативных правил.*

Ключевые слова: *интеллектуальная обработка данных, поиск ассоциативных правил, алгоритм 3D2ARM, структуры данных, передача данных, сжатие данных, метод подавления нулевой длины, нефиксированный размер значений массива.*

DOI: 10.21667/1995-4565-2021-78-55-62

Введение

Поиск ассоциативных правил является одним из актуальных направлений интеллектуальной обработки данных, которое используется в различных сферах деятельности человека – в медицине, торговле, интернет-трекинге [1]. Поиск ассоциативных правил помогает найти неочевидные зависимости между элементами. В настоящее время разработаны алгоритмы решения этой задачи, отличающиеся друг от друга временно́й сложностью, количеством проходов списка транзакций и используемыми структурами данных [2]. В общем случае, транзакция – это логически завершенная операция, при рассмотрении примера в статье используется список транзакций супермаркета, то есть одна транзакция – это покупка совершенная в определенный момент времени отдельным покупателем. Для удобства восприятия информации данные транзакций приведенны к виду кодов товаров. Алгоритмы поиска ассоциативных правил используются и как самостоятельная единица программы, и как часть больших информационных систем, в том числе распределенных систем хранения и анализа данных.

Распределенный поиск ассоциативных правил

Распределенная система – вычислительная система, в которой несколько автономных компьютеров, называемых узлами, взаимодействуют друг с другом [3]. Появление распределенных систем привело к появлению распределенных алгоритмов. Распределенный алгоритм – подтип параллельного алгоритма – отдельные части алгоритма запускаются одновременно на независимых процессорах [4, 5]. Распределенный алгоритм поиска ассоциативных правил включает поиск ассоциативных правил на каждом из второстепенных узлов и передачу промежуточных результатов на главный узел. На главном узле из результатов извлекаются ассоциативные правила. Распределенный алгоритм поиска ассоциативных правил основывается на алгоритме поиска ассоциативных правил 3D2ARM.

Алгоритм 3D2ARM

Алгоритм 3D2ARM подразумевает запись наборов из списка транзакций в промежуточное представление, извлечение наборов-кандидатов из промежуточного представления, подсчет частости наборов кандидатов в списке транзакций [6]. Преимуществом алгоритма является необходимость только одного прохода по списку транзакций для вероятностного результата и двух проходов – для точного результата.

В качестве промежуточного представления используется трехмерный массив. Наборы записываются в массив следующим образом. Сначала по транзакции генерируются все двухэлементные наборы. Затем для каждого сгенерированного набора увеличиваются на 1 значения элементов массива с индексами $[x, y, z]$, где x – первый элемент в наборе; $x < m$; y – второй элемент в наборе; $x < y \le m$; $z = x, x+1, \ldots, t_{max}$; m – количество уникальных элементов в транзакциях; t_{max} – максимальная длина транзакции. Обозначим x и y индексы элемента массива, а z – уровень элемента массива. Для сокращения памяти, применяется для хранения массива используется ступенчатый (зубчатый) массив.

Трехмерный массив наделяет алгоритм 3D2ARM свойством аддитивности промежуточных данных: сумма массивов, заполненных по частям списка транзакций, равна массиву, заполненному по всему списку транзакций. Наличие этого свойства позволяет использовать алгоритм 3D2ARM для разработки распределенного алгоритма поиска ассоциативных правил.

Цель работы

При распределенном поиске ассоциативных правил необходима передача трехмерного массива с второстепенных узлов на главный узел. Массив является разреженным, т.е. часть его элементов имеют нулевое значение. Следовательно, возможно сжатие массива перед его передачей на главный узел.

Целью статьи является представление разработанного формата сжатия трехмерного массива для повышения скорости его передачи за счет уменьшения объема, используемого в алгоритме поиска ассоциативных правил 3D2ARM и результатов такого сжатия

Краткий обзор методов сжатия

Алгоритмы сжатия данных делят на два типа: 1) сжатие данных с потерями (файлы видео, музыки, изображений); 2) сжатие данных без потерь (файлы изображений, документов) [7]. Для сжатия массива необходимо использовать сжатие без потерь, так как восстановленный массив должен быть равен исходному массиву.

Алгоритмы сжатия данных без потерь делятся на три группы: 1) преобразование потока; 2) статистические методы; 3) методы преобразования блока [8]. Будем использовать для сжатия способ, основанный на алгоритме первой группы, – преобразование потока.

Способ сжатия и формат файла для хранения и передачи промежуточных данных при поиске ассоциативных правил

Предлагается сжатие массива проводить в двух направлениях: I) сжатие за счет разреженности массива; II) использование нефиксированного размера для записи значений массива.

I. Для сжатия разреженных массивов используется метод подавления нулевой длины (*Zero Length Suppression*). Этот метод заключается в том, что массив представляется как последовательность пар $< s, v >$, где s – смещение элемента относительно предыдущего элемента с ненулевым значением; v – значение элемента массива. Например, если элемент с первым ненулевым значением 2 находится в ячейке с номером 5, а элемент со следующим ненулевым значением 6 хранится в ячейке номер 8, то будет получена последовательность чисел: 5 2 3 6. Таким образом, при разреженности массива можно сократить число хранимых элементов, и восстановление массива при этом не составит труда.

II. Значения s и v могут принимать различные значения, а следовательно, для их записи потребуется различное количество байт. Чтобы не фиксировать количество байт, выделяемых под эти значения, выделим под значение 7 бит байта. 8-й бит будет определять, является ли байт числа последним или нет. Если 8-й бит равен 1, то байт числа является последним. Таким образом, 7 бит каждого байта являются информационными, а 1 бит – служебным. Это позволяет разделять числа в потоке данных.

Кроме значений элементов массива, необходимо хранить значения размера массива: m и t_{max}.

Формат файла для хранения и передачи сжатого массива включает сигнатуру (последовательность символов, поясняющую программе, что это за данные и как их обрабатывать), размеры и значения элементов массива (таблица 1).

Таблица 1 – Формат файла для хранения и передачи сжатого массива

Table 1 – File format for storing and transferring a compressed array

Номер элемента	Размер, байт	Описание
1	1	Сигнатура
2	Нефиксированный	Значение m
3	Нефиксированный	Значение t_{max}
4	Нефиксированный	Пары значений s, v

В качестве сигнатуры выбрано значение $170_{10} = 10101010_2$.

Пример сжатия

Пусть имеется следующий список транзакций:

1: {1 2 3 4 5};
2: {1 3 6};
3: {6 7 8 9};
4: {1 4 5 6 8 9};
5: {1 2 3 4 5 6};
6: {1 4 6 8 9};
7: {2 6 9};
8: {2 3 4 5 8};
9: {4 5 7 8 9};
10: {1 2 4 5}.

Таким образом, $m = 9$, $t_{max} = 6$.

В соответствии с алгоритмом 3D2ARM, подробное описание которого представлено в статье [6], сформирован массив промежуточных данных. Например, для первой транзакции будут сформированы пары {1, 2}, {1, 3}, {1, 4} {1, 5} {2, 3} {2, 4} {2,5} {3, 4} {3, 5} {4, 5}. В соответствии с этим в следующие ячейки будет помещена 1. Затем на каждом уровне будет добавлена 1 к текущему значению элемента в соответствии с этими комбинациями. Обозначим символом A массив транзакций, тогда для первой транзакции на первом уровне будут добавлены 1 в следующие ячейки массива: A[1, 2, 1] = 1, A[1, 3, 1] = 1, A[1, 4, 1] = 1, A[1, 5, 1] = 1, A[2, 3, 1] = 1, A[2, 4, 1] = 1, A[2, 5, 1] = 1, A[3, 4, 1] = 1, A[3, 5, 1] = 1, A[4, 5, 1] = 1. На следующих уровнях массив заполняется аналогичным образом. С увеличением уровня накладывается ограничение на используемые комбинации элементов – первый элемент комбинации должен быть больше заполняемого уровня.

После записи наборов из списка транзакций в массив его уровни будут иметь следующий вид (таблица 2 – таблица 6).

Таблица 2 – Элементы трехмерного массива первого уровня
Table 2 – First-level three-dimensional array elements

1 уровень	1	2	3	4	5	6	7	8
2	3							
3	3	3						
4	5	4	3					
5	4	4	3	6				
6	4	2	2	3	2			
7	0	0	0	1	1	1		
8	2	1	1	4	3	3	2	
9	2	1	0	3	2	4	2	4

Таблица 3 – Элементы трехмерного массива второго уровня
Table 3 – Second-level three-dimensional array elements

2 уровень	2	3	4	5	6	7	8
3	3						
4	4	3					
5	4	3	6				
6	2	2	3	2			
7	0	0	1	1	1		
8	1	1	4	3	3	2	
9	0	0	2	2	3	2	4

Таблица 4 –Элементы трехмерного массива третьего уровня
Table 4 – Third-level three-dimensional array elements

3 уровень	3	4	5	6	7	8
4	2					
5	3	4				
6	1	2	2			
7	0	0	0	0		
8	1	4	3	2	1	
9	0	0	1	2	1	4

Таблица 5 –Элементы трехмерного массива четвертого уровня
Table 5 – Fourth-level three-dimensional array elements

4 уровень	4	5	6	7	8
5	2				
6	1	1			
7	0	0	0		
8	2	2	1	0	
9	0	0	1	0	3

Таблица 6 – Элементы трехмерного массива пятого уровня
Table 6 – Fifth-level three-dimensional array elements

5 уровень	5	6	7	8
6	1			
7	0	0		
8	0	0	0	
9	0	0	0	1

Значения элементов массива будут записаны следующим образом (таблица 7). Для сокращения размера файла решено использовать нефиксированный размер элементов. Каждое записываемое число преобразуется в последовательность битов, каждый байт представляет собой 7 битов числа и последний бит, указывающий на окончание этого числа (1 – если биты

числа закончились и 0 – если нет). Для первых элементов степень сжатия будет небольшой. Однако к концу массива степень сжатия увеличивается. Так, например, для последнего элемента смещение будет равно 8.

Таблица 7 – Содержание сжатого файла

Table 7 – Compressed file content

Значение	Пояснение
170	Сигнатура
9	Значение m
6	Значение t_{max}
1	Смещение 1-го элемента
3	Значение 1-го элемента
1	Смещение 2-го элемента
3	Значение 2-го элемента
…	…
8	Смещение последнего элемента
1	Значение последнего элемента

Определение необходимого количества экспериментов

Вычислим необходимое количество экспериментов сжатия массивов по формуле, выведенной из неравенства Чебышева [9]:

$$n = \frac{C}{(1-\gamma)\varepsilon^2},$$

где C – верхняя граница дисперсии, $DX \le C$.

Эта формула гарантирует, что при числе испытаний n с вероятностью, не меньшей γ, можно быть уверенным в том, что абсолютная погрешность приближенного равенства $MX \approx \overline{X}$ не превысит ε.

Вычислим количество для различных значений погрешности ε при $C = 0,002$, $\gamma = 0,9$ (таблица 8).

Примем погрешность ε, равную 0,03, достаточной для эксперимента. Следовательно, для получения достоверного результата необходимо провести 23 эксперимента.

Таблица 8 – Количество экспериментов в зависимости от погрешности

Table 8 – Number of experiments depending on inaccuracies

Погрешность ε	Количество экспериментов
0,1	2
0,05	8
0,03	23
0,02	50
0,01	200
0,005	800
0,001	20000

Сравнение размеров файлов хранения и передачи несжатого и сжатого массивов

Алгоритмы сжатия массива и восстановления массива реализованы на языке программирования Java.

Были проведены эксперименты использования предлагаемого сжатия и восстановления массива. Списки транзакций были сгенерированы с помощью библиотеки Arules [10] языка программирования R [11]. Они были проанализированы с заданной минимальной поддержкой, равной 0,0001. Фрагменты исходного массива содержат последовательности нулей (рисунок 1), а сжатый массив уже нет (рисунок 2).

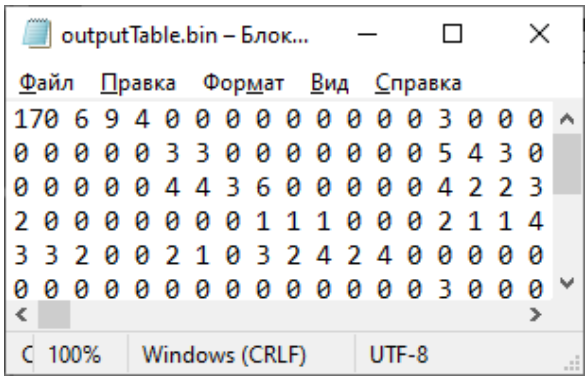

Рисунок 1 – Пример фрагмента исходного массива
Figure 1 – An example of original array fragment

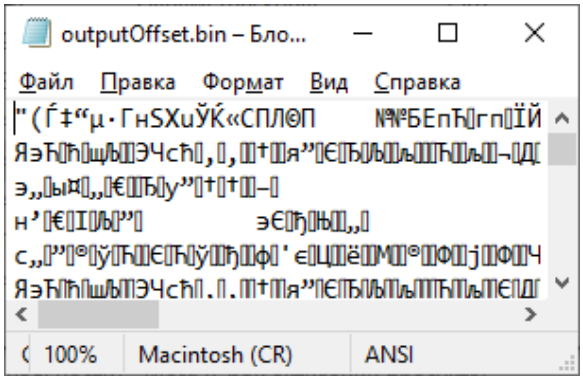

Рисунок 2 – Пример фрагмента сжатого массива
Figure 2 – An example of compressed array fragment

В результате были получены файлы исходного и сжатого массивов следующих размеров (таблица 9, рисунок 3). С увеличением количества транзакций коэффициент сжатия уменьшается.

Таблица 9 – Размеры файлов с исходными и сжатыми массивами и коэффициент сжатия
Table 9 – File sizes with original and compressed arrays and compression ratio

Количество транзакций	Размер файла с исходным массивом, кб	Размер файла с массивом после сжатия, кб	Коэффициент сжатия, %
1000	16,3	3,32	80 %
5000	18,2	4,04	78 %
10000	20,0	4,76	76 %
50000	21,1	4,94	77 %
100000	22,3	5,13	77 %
500000	24,6	7,13	71 %
1000000	25,7	7,39	71 %
5000000	27,1	8,03	70 %
10000000	28,3	8,82	69 %

Алгоритм сжатия массива рассматривает содержимое файла как последовательность целых значений. Алгоритм включает следующие шаги:

1) преобразовать последовательность целых значений в последовательность пар значений $< s, v >$;

2) при получении очередной пары значений $< s, v >$ преобразовать их в значения с нефиксированным размером.

Исходя из этих шагов, предполагающих перебор всех элементов массива, временна́я сложность этого алгоритма будет иметь порядок $O(m^2)$.

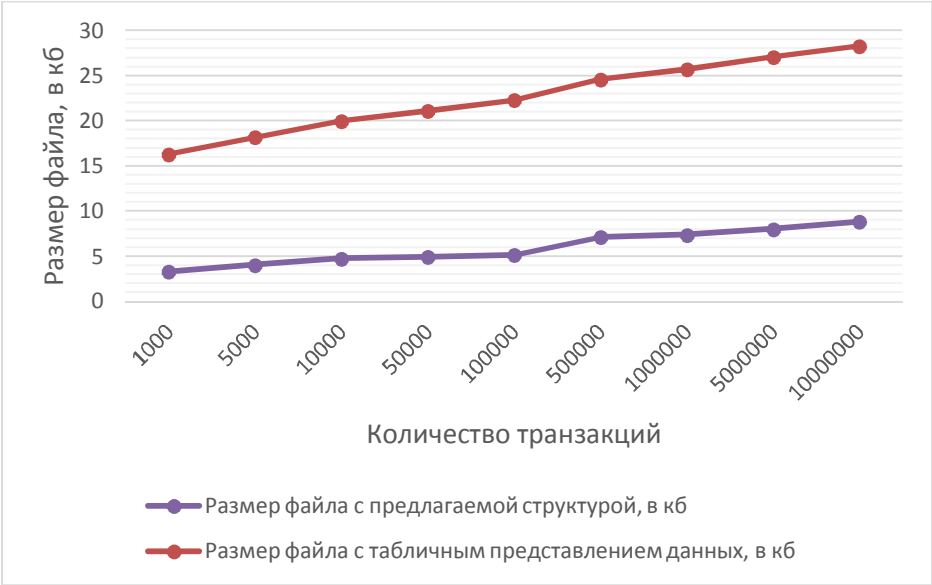

Рисунок 3 – Размеры файлов с исходными и сжатыми массивами
Figure 3 – File sizes with original and compressed arrays

Замер времени выполнения записи тестовых данных в файлы показал, что среднее время записи файла с исходным массивом в среднем равняется 2,1 мс, а среднее время записи файла со сжатием массива – 23,6 мс. Это позволяет сделать вывод, что при обычном формате записи времени тратится меньше, поскольку не выполняется предобработка данных.

Заключение

В статье изложены следующие результаты.

1. Предложен способ сжатия массива с промежуточными результатами поиска ассоциативных правил. Сжатие выполняется за счет исключения последовательности элементов с нулевыми значениями, а также использования нефиксированного количества байт под значения элементов и размеры массива.

2. Показано, что предложенный способ обеспечивает коэффициент сжатия в среднем 74 %. Этот результат получен с помощью разработанного программного обеспечения на языке Java.

Таким образом, предложенный способ сжатия сокращает требуемый объем памяти для хранения или требуемое время передачи промежуточных результатов распределенного поиска ассоциативных правил.

В дальнейшем планируется использование предложенного способа сжатия при разработке алгоритма распределенного поиска ассоциативных правил.

Библиографический список

1. **Фрэнкс Б.** Революция в аналитике: как в эпоху BigData улучшить ваш бизнес с помощью операционной аналитики. М.: Альпина Паблишер, 2017.

2. **Prutzkow A.** Algorithms and Data Structures for Association Rule Mining and its Complexity Analysis. In the European Proceedings of Social & Behavioural Sciences (EPSBS 2018), 2018, pp. 558-568. DOI: 10.15405/epsbs.2018.11.02.62.

3. **Косяков М. С.** Введение в распределенные вычисления: учеб. пособие. СПб.: НИУ ИТМО, 2014.

4. **Романов А. А.** Распределенные вычисления и приложения. Ульяновск: УлГТУ, 2018.

5. **Тель Ж.** Введение в распределенные алгоритмы. М.: МЦНМО, 2009.

6. **Khramshina E. O., Prutzkow A. V.** Association Rules Mining with Three-Dimensional Data Structure. In International Journal of Open Information Technologies, 2020, vol. 8, no. 8, pp. 8-12.

7. **Стивенс Р.** Алгоритмы. Теория и практическое применение. М.: Эксмо, 2019.

8. **Gupta A., Bansal A., Khanduja V.** Modern Lossless Compression Techniques: Review, Comparison and Analysis. In Proc. of 2nd IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2017.

9. **Калинина В. Н., Панкин В. Ф.** Математическая статистика. М.: Высш. шк., 1998. 336 с.

10. **Hahsler M., Grün B., Hornik K.** Arules – A Computational Environment for Mining Association Rules and Frequent Item Sets. Journal of Statistical Software, 14:15, 2005.

11. The R Project for Statistical Computing. [Электронный ресурс]. URL: https://www.r-project.org/about.html (дата обращения: 05.09.2021).

UDC 004.62

THE METHOD TO COMPRESS IN INTERMEDIATE DATA IN DISTRIBUTED MINING OF ASSOCIATION RULES

E. O. Khramshina, postgraduate student, lecturer assistant of the Department of Computational and Applied Mathematics, RSREU, Ryazan, Russia;
orcid 0000-0002-4490-8403, e-mail: evgenia.khramshina@gmail.com

The aim of this work *is to decrease memory volume for storage and transfer taken by three-dimensional array in distributed mining of association rules. Volume decrease is achieved due to sparse array and variable size of array element values. Array elements with non-zero values are written as a number pair: the offset from the previous element with non-zero value and the value itself. To separate numbers, one of the bits in a byte is used as a service one, to point the end of the value. The experiments have shown that this method allows file size 74% less on average in comparison with the original array. Software in Java programming language has been developed for these experiments. The compression method together with 3D2ARM association rules algorithm can be used to develop distributed mining of association rules.*

Key words*: data mining, association rules mining, 3D2ARM algorithm, data structures, data transfer, data compression, zero-length suppression method, variable size of array values.*

DOI: 10.21667/1995-4565-2021-78-55-62

References

1. **Frenks B.** *Revolyuciya v analitike: Kak v epohu BigData uluchshit' vash biznes s pomoshch'yu operacionnoj analitiki.* Moscow: Alpina Pablisher, 2017. (in Russian).

2. **Prutzkow A.** Algorithms and Data Structures for Association Rule Mining and its Complexity Analysis. *In the European Proceedings of Social & Behavioural Sciences* (*EPSBS 2018*), 2018, pp. 558-568. DOI: 10.15405/epsbs.2018.11.02.62.

3. **Kosyakov M. S.** *Vvedenie v raspredelennye vychisleniya. Uchebnoe posobie*. Spb.: NIU ITMO, 2014. (in Russian).

4. **Romanov A. A.** *Raspredelennye vychisleniya i prilozheniya*. Ulyanovsk: UlGTU, 2018. (in Russian).

5. **Tel Zh.** *Vvedenie v raspredelennye algoritmy*. M.: MCNMO, 2009. (in Russian).

6. **Khramshina, E. O., Prutzkow, A. V.** Association Rules Mining with Three-Dimensional Data Structure. *In International Journal of Open Information Technologies*, 2020, vol. 8, no. 8, pp. 8-12.

7. **Stivens R.** *Algoritmy. Teoriya i prakticheskoe primenenie*. M.: Eksmo, 2019. (in Russian).

8. **Gupta A., Bansal A., Khanduja V.** Modern Lossless Compression Techniques: Review, Comparison and Analysis. *In Proc. of 2nd IEEE International Conference on Electrical, Computer and Communication Technologies* (*ICECCT*), 2017.

9. **Stivens R.** *Algoritmy. Teoriya i prakticheskoe primenenie*. M.: Eksmo, 2019. (in Russian).

10. **Hahsler M., Grün B., Hornik K.** *Arules – A Computational Environment for Mining Association Rules and Frequent Item Sets. Journal of Statistical Software*, 14:15, 2005.

11. The R Project for Statistical Computing. [Electronic resource]. URL: https://www.r-project.org/about.html (date of use: 05.09.2021).