EFFICIENT LOSSLESS COMPRESSION OF LARGE INFORMATION ARRAYS

M. P. Bakulina

Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2022-4-63-69
EDN: HIQWPK

The problem of efficient lossless compression of large information arrays is considered. The use of efficient coding for such data allows not only to reduce their physical size, but also to increase the speed of query execution speed. In this paper, a new coding large information data arrays is offered.It allows you to efficiently compress both numeric and string data. An experimental data confirm of compression increase and coding speed increase of the proposed method.

Key words: lossless coding, information array, compression ratio, coding time, method efficiency.

References

1. Bakulina M. P. Ispolzovanie zakona Tcipfa dlia szhatiia tekstov // Diskretnyi analiz i issledovanie operatcii, 2007. S 2. T. 14. N 2. S. 3–13.

2. Riabko B. Ia. Effektivnyi metod kodirovaniia istochnikov informatcii, ispolzuiushchii algoritm bystrogo umnozheniia // Problemy peredachi informatcii, 1995. T. 31. V 1. S. 3–12.

3. Li J., Rotem D., Wong H. A New Compression Method with Fast Searching on Large Databases // Proceedings of 13th International Conference on Very Large Data Bases, Brighton, 1987. P. 311–318.

4. Eggers S., Shoshani A. Efficient Access of Compressed Data Performance // Proc. VLDB, Montreal, 1980. P. 205.

5. Eggers S., Olken F., Shoshani A. A Compression Technique for Large Statistical databases // Proc. VLDB Conf, 1981. P. 114.

6. Li J., Rotem D., Wong H. A New Compression Method with Fast Searching on Large Databases // Proceedings of 13th International Conference on Very Large Data Bases, Brighton, 1987. P. 311–318.

7. Ziv J., Lempel A. Compression of individual sequences via variable-length coding // IEEE Trans. Inform. Theory, 1978. V. IT-24. N 5. P. 530–536.

8. Elias P. Interval and recency rank source encoding: two on-line adaptive variable-length schemes // IEEE Trans. Inform. Theory, 1987. V. 33. N 1. P. 3–10.

9. Bell T. C., Cleary J. H., Witten I. H. Text Compression. Prentice Hall. Englewood Cliffs, 1990.

10. Zipf G. K. Human behavior and the principle of least effort. Cambridge: Addison Wesley, 1949.

This work was carried out under state contract with ICMMG SB RAS (0251-2021-0005).

ЭФФЕКТИВНОЕ СЖАТИЕ БЕЗ ПОТЕРЬ БОЛЬШИХ МАССИВОВ ИНФОРМАЦИОННЫХ ДАННЫХ

М. П. Бакулина

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

УДК 519.722
DOI: 10.24412/2073-0667-2022-4-63-69
EDN: HIQWPK

Рассматривается задача эффективного сжатия без потерь больших информационных массивов. Использование эффективного кодирования для таких данных позволяет не только уменьшить их физический размер и объем занимаемой ими оперативной памяти, но и увеличить скорость выполнения запросов. В данной работе предлагается алгоритм кодирования, позволяющий эффективно сжимать встречающиеся в массиве как числовые, так и строковые данные. Проведен эксперимент, подтверждающий увеличение степени сжатия и скорости кодирования и декодирования больших информационных массивов при использовании предложенного метода по сравнению с ранее известными методами.

Ключевые слова: кодирование без потерь, информационный массив, коэффициент сжатия, время кодирования, эффективность метода.

Введение. Рассматривается задача эффективного сжатия больших информационных массивов. Использование кодирования без потерь таких данных приводит не только к уменьшению их физического размера, но и к увеличению скорости выполнения запросов, а также к уменьшению объема занимаемой оперативной памяти, поэтому данная задача не потеряла своей актуальности по сей день.

Как известно, наиболее распространенными типами данных, используемых в больших информационных массивах, являются строковые и числовые, причем кодирование этих двух типов имеет определенную специфику.

В данной работе предлагается новый метод сжатия больших информационных массивов, позволяющий эффективно кодировать как числовые, так и строковые данные. В этом методе предлагается сжимать встречающиеся в массиве числовые данные с помощью алгоритма, представляющего собой модификацию известного метода «битовой карты», а встречающиеся строковые данные сжимаются с помощью разработанного автором эффективного алгоритма сжатия текстовых данных [1]. Приводятся экспериментальные данные, подтверждающие эффективность предложенного метода.

1. Эффективные методы кодирования констант. Сначала рассмотрим задачу эффективного сжатия больших информационных массивов числовых данных. Основной выигрыш при кодировании таких данных может быть получен за счет сжатия ведущих нулей

Исследования выполнены в рамках государственного задания ИВМиМГ СО РАН (0251-2021-0005).

Статья по докладу на XVIII Международной Азиатской школе-семинаре «Проблемы оптимизации сложных систем», Киргизия, Иссык-Куль, 20.07.2022–30.07.2022.

и экономного кодирования часто повторяющихся значений, которыми обычно являются ноль и отсутствующие значения. Такие часто повторяющиеся значения будем называть константами.

В настоящее время существуют различные алгоритмы сжатия числовых данных. Наиболее известным алгоритмом сжатия числовых данных является метод RLE и его модификации. Это алгоритм сжатия данных, заменяющий повторяющиеся символы (серии) на один символ и число его повторов [3]. Серией при этом называется последовательность, состоящая из нескольких одинаковых символов. При кодировании (сжатии) строка одинаковых символов, составляющих серию, заменяется строкой, содержащей сам повторяющийся символ и количество его повторов. Главным недостатком группы алгоритмов RLE является неэффективность на неповторяющихся наборах символов. Использование специальных перестановок повышает эффективность алгоритма, но также сильно увеличивает время работы (особенно декодирования).

Устранение определенной константы может быть реализовано с помощью метода Bitmap [4]. В этом методе сохраняются только значения, отличные от констант. Местоположение констант определяется так: каждый разряд равен 1, если в соответствующей позиции находится обычное значение, и 0, если стоит константа. Например, если исходная строка $S = \{D_1, D_2, c, D_3, c, c, c, D_4\}$, то сжатая строка имеет вид $\tilde{S} = \{D_1, D_2, D_3, D_4\}$, а строка нулей и единиц $B = \{1,1,0,1,0,0,0,1\}$.

В работе [5] была предложена модификация метода Bitmap, обеспечивающая в среднем более быстрый поиск в сжатых данных. В этом методе строка нулей и единиц преобразована в специальный вектор F, в нечетных позициях которого записывается число несжатых значений с накоплением, а в четных — число пропущенных констант с накоплением. Например, для исходной строки $S = \{D_1, D_2, c, D_3, c, c, c, D_4\}$ специальный вектор $F = \{2,1,2+1,1+3,2+1+1\} = \{2,1,3,4,4\}$. Декодирование информации может быть выполнено с помощью поиска нулей и единиц в F.

В [6] описана еще одна модификация метода Bitmap, позволяющая сократить число обращений к внешней памяти при декодировании.

Рассмотрим теперь новый метод кодирования больших массивов числовых данных NEW_1, представляющий собой модификацию метода Bitmap, но позволяющий более эффективно сжимать этот тип данных.

2. Алгоритм кодирования числовых данных NEW_1. При сжатии будем использовать три вектора:

— входящий вектор IV, состоящий из нулей и единиц,

— определяющий вектор OV,

— дополнительный вектор DV.

В позиции входящего вектора записывается 0, если текущий символ равен константе, и 1, если наоборот. Кодирование такого вектора осуществляется следующим образом: вектор разбивается на блоки длины $l = 1/\sqrt{p}$, где $p = p(1)$. Если блок состоит из одних нулей, то код этого блока – 0. Иначе длина кодового слова равна $l + 1$: началом кодового слова является 1, за которым следует тот же самый блок длины l.

Например, пусть кодируется вектор 0010000000001100000000000. Тогда $p = 1/8$, $l = 3$. Разбиваем данный вектор на блоки 001-000-000-000-110-000-000-000. Тогда закодированная последовательность имеет вид 1001 0 0 0 1110 0 0 0.

Отметим, что блоки не надо хранить в памяти кодера и декодера; достаточно хранить лишь счетчики нулей объема $O(log?(1/p))$. Полученная последовательность кодируется

еще одним специальным кодом так, чтобы достигалась высокая скорость кодирования и декодирования.

В определяющем векторе OV перечисляются все значения, которые отличны от констант. В ячейке дополнительного вектора $DV(i)$ хранится положение (номер) последнего элемента, который отличен от константы, для блока с номером $i-1$. Первый элемент дополнительного вектора равен нулю. Если в предыдущем блоке встречаются только константы, то в дополнительный вектор записывается значение его предыдущего элемента.

Например, если размер блока равен 6, а константой является «0», то для строки $S = \{1,0,0,3,0,00,0,0,0,0,00,0,6,0,00,15,0,0,0,00,11,0,0,23,0\}$ получаем следующие векторы: $IV = \{1,0,0,1,0,00,0,0,0,0,00,0,1,0,0,00,1,0,0,0,00,1,0,0,1,0\}$, $OV = \{1,3,6,15,11,23\}$ и $DV = \{0,2,2,3,4\}$.

Отметим, что при кодировании и декодировании дополнительного вектора определяется, какой блок надо декодировать, и процессу декодирования подвергается только этот блок.

3. Алгоритм кодирования строковых данных NEW_2. Так как строковые данные представляют из себя буквенные символы, разделенные пробелом, то совокупность подряд идущих символов до пробела будем называть словом. Таким образом на входе мы имеем текстовые данные в виде слов, разделенных пробелами.

Наиболее известная схема сжатия текстовых данных, основанная на сведении к минимуму избыточности, — алгоритм Лемпеля-Зива, именуемый LZ-алгоритмом. В этом алгоритме кодируемое слово разделяется на подслова, кодами которых являются пары чисел. При этом на каждом шаге выбирается наиболее длинное начало остатка, которое совпадает с некоторым уже выделенным подсловом, и к нему добавляется еще один строковый символ. Отметим, что среди всех LZ-алгоритмов и их модификаций мы остановимся на алгоритме LZ78 [7] как наиболее эффективном по степени сжатия. Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря. Как только эта строка перестает соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре и символа, нарушившего совпадение. Затем в словарь добавляется введенная подстрока. Таким образом, алгоритм LZ78 можно рассматривать как кодирование с динамичным словарем. Словарь можно произвольным образом нумеровать и удалять из него слова, все продолжения которых уже имеются в словаре.

Среди методов эффективного кодирования строковых данных представляет также интерес MTF-схема [2], названная автором как «метод стопки книг». Модификация этого метода, названная как «кодирование по степени новизны», встречается также в [8]. В MTF-алгоритме поток входных строковых данных рассматривается как чередующаяся последовательность слов и символов, не являющихся словами. Каждое слово в тексте после его очередного появления удаляется с текущей позиции и перемещается вперед. В результате слова, которые в тексте появляются часто, оказываются впереди слов, встречающихся редко. При этом словам, расположенным впереди, присваиваются более короткие коды. Если слово или символ появляются впервые, то кодируются побуквенно, иначе кодирование осуществляется с учетом частоты появления слова в тексте. Посылаемое дополнительное кодовое слово (флаг) позволяет определить, является ли приходящее слово уже встречавшимся или новым.

Отметим, что наиболее распространенные и эффективные методы кодирования строковых данных, не использующие словарь, приведены в [5]. Среди них можно выделить

алгоритм PPM. В этом методе в процессе кодирования вместо безусловных вероятностей букв оцениваются их условные вероятности при известном «контексте», т. е. при известных предшествующих буквах. Таким образом, алгоритм PPM — это метод контекстно-ограниченного моделирования, позволяющий оценить вероятность символа в зависимости от предыдущих символов.

Рассмотрим теперь новый алгоритм кодирования строковых данных (назовем его NEW_2), основанный на модификации метода кодирования, представленного автором в [1]. В этом методе предлагается кодировать поступающий на входе текст по частотному словарю. В этом словаре слова располагаются в порядке убывания частот и пронумеровываются от 1 до N, где N — объем словаря. При этом словарь может быть составлен как до начала кодирования (двухпроходная схема), так и составляться по мере прочтения строковых данных. Будем использовать однопроходную схему кодирования, в котором словарь наполняется по мере прохождения текста.

Сначала рассмотрим случай, когда на начальном шаге объем словаря известен. Пусть D — общее число слов в словаре. Кодирование слов будем осуществлять следующим образом. Составим частотный словарь для текста, прочитанного до определенного слова (это слово предстоит закодировать). Если оно не встречается в словаре, то добавляем его в словарь и применяем побуквенное кодирование. Если же слово уже встречалось, то кодируем его в соответствии с законом Ципфа [10], при этом длина кода равна $l_i = \lceil -log\frac{c(D)}{i} \rceil$, где $c(D) \approx 1/\ln D$, а i — номер слова в словаре. После прочтения очередного слова в тексте и занесения его в словарь все слова в текущем словаре переупорядочиваем заново и располагаем их в порядке убывания частот. Перед кодом каждого нового кода будем вставлять специальное кодовое слово (флаг), длина которого равна $\lceil -log\left(1 - \sum\limits_{i=1}^{j} \frac{c(D)}{i}\right) \rceil$, где j — число слов в текущем словаре. При декодировании сообщения на приемном конце получатель определяет флаг и идущую за ним кодовую последовательность.

Теперь рассмотрим случай, когда на начальном этапе объем словаря неизвестен. Алгоритм кодирования строится также, как и при известном объеме, но величина $c(D)$ оценивается по текущему словарю. Это означает, что если D_j — число слов в словаре, составленном для текста, прочитанного до j-го слова, то полагаем $c(D_j) \approx 1/\ln D_j$.

Отметим, что если частотный словарь до начала кодирования текстовых данных уже составлен, то можно применить двухпроходную схему кодирования, которая позволяет получить лучшую степень сжатия, чем однопроходная схема (см. [5]). Под степенью сжатия текста здесь понимается отношение объема текстовых данных, полученного в результате кодирования, к его первоначальному объему. Тогда если D — объем словаря, то каждому слову с номером i будем сопоставлять кодовое слово длины $\lceil -log\frac{c(D)}{i} \rceil$, где $c(D) \approx \frac{1}{\ln D}$.

4. Результаты экспериментального сравнения. Для подтверждения эффективности предложенного метода кодирования числовых и строковых данных было проведено экспериментальное сравнение результатов его работы с результатами работы других известных алгоритмов. Для этого поступающий на входе большой информационный массив сначала распределял входные данные на числовые и строковые. Числовые данные кодировались методом NEW_1, а строковые данные – алгоритмом NEW_2.

Для эксперимента использовались два больших информационные массива — Table 1 и Table 2:

— Table 1 — это искусственно сгенерированная таблица, содержащая 10964 записи, в которых преобладают в основном числовые данные;

Таблица 1

Результаты сравнения работы алгоритмов RLE, Bitmap, NEW

	$k(RLE)$	$k(Bitmap)$	$k(NEW)$	$t(RLE)$	$t(Bitmap)$	$t(NEW)$
Table 1 (581 байт)	43 %	37 %	**24 %**	3.2 с	4.8 с	**2.8 с**

Таблица 2

Результаты сравнения работы алгоритмов LZ, MTF, NEW

	$k(LZ)$	$k(MTF)$	$k(NEW)$	$t(LZ)$	$t(MTF)$	$t(NEW)$
Table 2 (1724 байт)	39 %	37 %	**34 %**	4.1 с	5.2 с	**3.9 с**

— Table 2 — это библиография научных публикаций, содержащая имена авторов, наименование издательства и т. д. Содержит 80690 записей, в которых преобладают строковые данные.

Так как первая таблица Table 1 содержала в основном числовые данные, то эффективность работы предложенного метода (назовем его NEW) сравнивалась с эффективностью известных алгоритмов для кодирования числовых данных RLE и Bitmap. Сравнение проводилось по двум параметрам: коэффициенту сжатия k (в %) и времени кодирования и декодирования t (в секундах). Результаты сравнения приведены в табл. 1.

Из таблицы видно, что предложенный алгоритм NEW почти в 2 раза выигрывает у RLE по степени сжатия при почти такой же скорости сжатия. У метода Bitmap предложенный метод NEW на 6–7 % в среднем выигрывает по степени сжатия и почти в 2 раза по скорости сжатия.

Эффективность работы алгоритма NEW при кодировании строковых данных была проверена на таблице Table 2. Сравнение проводилось с двумя известными алгоритмами для кодирования текстовых данных LZ и MTF. Результаты сравнения приведены в табл. 2.

Из таблицы видно, что предложенный алгоритм NEW показывает лучшую степень сжатия и лучшее время кодирования по сравнению с алгоритмами LZ и MTF.

Заключение. Проведенное экспериментальное сравнение методов подтверждает эффективность предложенного алгоритма NEW. Данный метод может быть использован как для сжатия больших информационных массивов числовых данных (например, данных, полученных в результате физического эксперимента), так и для сжатия массивов, содержащих большое количество строковых данных (например, библиотечные базы данных), что позволит уменьшить объем занимаемой памяти и повысить скорость кодирования.

Список литературы

1. Бакулина М. П. Использование закона Ципфа для сжатия текстов // Дискретный анализ и исследование операций, 2007. Серия 2. том 14. № 2. С. 3–13.

2. Рябко Б. Я. Эффективный метод кодирования источников информации, использующий алгоритм быстрого умножения // Проблемы передачи информации, 1995. Т. 31. выпуск 1. С. 3–12.

3. Li J., Rotem D., Wong H. A New Compression Method with Fast Searching on Large Databases // Proceedings of 13th International Conference on Very Large Data Bases, Brighton, 1987. P. 311–318.

4. Eggers S., Shoshani A. Efficient Access of Compressed Data Performance // Proc. VLDB, Montreal, 1980. P. 205.

5. Eggers S., Olken F., Shoshani A. A Compression Technique for Large Statistical databases // Proc. VLDB Conf, 1981. P. 114.

6. Li J., Rotem D., Wong H. A New Compression Method with Fast Searching on Large Databases // Proceedings of 13th International Conference on Very Large Data Bases, Brighton, 1987. P. 311–318.

7. Ziv J., Lempel A. Compression of individual sequences via variable-length coding // IEEE Trans. Inform. Theory, 1978. V. IT-24. N 5. P. 530–536.

8. Elias P. Interval and recency rank source encoding: two on-line adaptive variable-length schemes // IEEE Trans. Inform. Theory, 1987. V. 33. N 1. P. 3–10.

9. Bell T. C., Cleary J. H., Witten I. H. Text Compression. Prentice Hall. Englewood Cliffs, 1990.

10. Zipf G. K. Human behavior and the principle of least effort. Cambridge: Addison Wesley, 1949.

Бакулина Марина Павловна — канд. физ.-мат. наук, науч. сотрудник Института вычислительной математики и математической геофизики (ИВМиМГ СО РАН). Научные интересы: теория кодирования, сжатие данных, моделирование информационных систем, e-mail: `marina@rav.sscc.ru`.

Bakulina Marina Pavlovna is a Researcher of Institute of Computational Mathematics and Mathematical Geophysics (ICMMG SB RAS). PhD in Physics and Mathematics. Research interests: coding theory, data compression, information systems modeling.

Дата поступления — 17.08.2022