РАЗДЕЛ IV. МЕТОДЫ ОБРАБОТКИ СИГНАЛОВ И ДАННЫХ

УДК 519.722

ЭФФЕКТИВНОЕ СЖАТИЕ ДАННЫХ НА ОСНОВЕ ПРЕОБРАЗОВАНИЯ БАРРОУЗА-УИЛЕРА*

М.П. Бакулина

Рассматривается одна из важных задач теории информации – задача сжатия данных с сохранением возможности их однозначного восстановления (декодирования). Предлагается эффективный алгоритм сжатия, основанный на преобразовании Барроуза-Уилера и двухэтапном кодировании полученной последовательности. Приводятся экспериментальные результаты сжатия, подтверждающие эффективность предложенного метода

Ключевые слова: сжатие данных, кодирование длин серий, объем памяти, скорость кодирования

Введение

Сжатие данных является одной из важных задач теории информации. Это связано с тем, что задача компактного представления данных с сохранением возможности их однозначного восстановления (декодирования) возникает достаточно часто на практике. Так, предварительное сжатие позволяет существенно улучшить характеристики системы связи. Сжатие применяется при хранении информации, ее передаче со спутников и в других приложениях, причем актуальность проблемы сжатия возрастает в связи с увеличением объемов передаваемой и хранимой информации.

В настоящее время широко известны и находят практическое применение многие алгоритмы компактного представления данных, которые дают сжатие лучше, чем стандартные архиваторы. Один из таких подходов при создании алгоритмов сжатия основан на преобразовании Барроуза-Уилера или BWT-преобразовании [1]. Данное преобразование можно рассматривать как способ перестановки символов в блоке данных, позволяющий осуществить эффективное сжатие. BWT меняет порядок во входной строке таким образом, что повторяющиеся подстроки образуют на выходе последовательности одинаковых символов, редко перемежающиеся другими символами. Процедуру преобразования можно условно разделить на 4 этапа:

1. выделяется блок из входного потока исходных данных;
2. формируется матрица всех перестановок, полученных в результате циклического сдвига блока;
3. все перестановки сортируются в соответствии с лексикографическим порядком символов каждой перестановки;
4. на выход подается последний столбец матрицы и номер строки, соответствующей первоначальному блоку.

Так как в полученной после BWT-преобразования последовательности вероятности появления одинаковых символов выше, чем редких, то возникает задача эффективного кодирования длин серий. В известных алгоритмах сжатия на основе BWT для этого выполняется шаг по замене каждого символа расстоянием до его предыдущей встречи (например в алгоритме move to front, MTF [2]), а далее к полученному набору чисел применяется метод энтропийного сжатия, например, кодирование Хаффмана или арифметическое кодирование. На практике алгоритм сжатия вида BWT → MTF → Хаффман применён в архиваторе bzip2.

В данной работе предлагается новый алгоритм сжатия данных, основанный на преобразовании Барроуза-Уилера. В этом алгоритме к последовательности, полученной после BWT-преобразования, применяется эффективный двухэтапный метод кодирования длин серий. Приводится сравнение экспериментальных результатов предложенного метода и архиватора bzip2, подтверждающее эффективность построенного алгоритма.

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (грант № 11-07-00183а)

Эффективный алгоритм кодирования длин серий

Построим алгоритм эффективного кодирования длин серий, полученных после BWT-преобразования, то есть низкоэнтропийного бернуллиевского источника, порождающего буквы из алфавита $A=\{0,1\}$. Отметим, что предлагаемая конструкция легко обобщается и на случай произвольного алфавита $A=\{a_1\dots a_n\}, n>2$.

В предлагаемом алгоритме кодирование осуществляется в два этапа. На первом этапе порождаемое источником сообщение сжимается простым кодом, что позволяет ввиду малой энтропии источника существенно сократить длину входной последовательности, а на втором этапе полученная последовательность кодируется более сложным методом, позволяющим достигать наперед заданной избыточности. На втором этапе кодирования мы будем использовать адаптивный арифметический код из [3], однако отметим, что могут быть использованы и другие универсальные коды, применение которых дает тот же результат.

Опишем предлагаемый метод кодирования более подробно. Пусть кодируется сообщение $x_1x_2\dots,\ x_i\in A,\ A=\{0,1\},\ i=1,2,\dots.$ Рассмотрим первый этап кодирования. Для этого введем в рассмотрение сначала скользящее окно длины ω, представляющее собой слово $x_{t-\omega}x_{t-\omega+1}\dots x_{t-1}$. Это окно используется для оценки неизвестных вероятностей. Зафиксируем избыточность $r>0$ и определим $\omega=\lceil 2/r\rceil$. Пусть $n(1,t)$ - число вхождений 1 в окно $x_{t-\omega}x_{t-\omega+1}\dots x_{t-1}$. Обозначим через $\widetilde{p}(1,t)$ оценку вероятности появления 1 в этом окне, через l_t длину блока, определяемую на основе статистики данного окна. Определим

$$\widetilde{p}(1,t)=\frac{n(1,t)+1}{\omega+2},$$

$$l_t=\left\lceil\frac{1}{\sqrt{\widetilde{p}(1,t)}}\right\rceil=\left\lceil\sqrt{\frac{\omega+2}{n(1,t)+1}}\right\rceil, \qquad (1)$$

$$\widetilde{q}(t)=\widetilde{p}(0,t)=1-\widetilde{p}(1,t)$$

При этом максимальная длина блока равна

$$l_{\max}=\left\lceil\sqrt{\omega+2}\right\rceil \qquad (2)$$

Кодирование каждого очередного блока осуществляется на основе статистики, содержащейся в предыдущем окне. Если блок состоит из одних нулей, то его кодом является 0, иначе длина кодового слова равна l_t+1: началом его является 1, за которой следует тот же блок длины l_t. Затем сдвигаем окно на l_t символов вправо и на основе статистики полученного окна аналогичным образом кодируем следующий блок.

Пусть $y_1y_2\dots y_s$ - последовательность, полученная после первого этапа кодирования, $y_i\in\{0,1\}$. Рассмотрим второй этап кодирования, осуществляемый адаптивным арифметическим кодом из [3]. Представим данную последовательность в виде

$$\underline{0}\dots\underline{0}\underline{1}\underbrace{y_1\dots y_{l_j}}_{l_j}\underline{0}\dots\underline{0}\underline{1}\underbrace{y_1\dots y_{l_j}}_{l_j}\dots$$

В этой последовательности выделены блоки длины l_j, следующие после появления 1, и “особые” символы $\underline{0},\underline{1}$, не входящие в блоки. Кодирование различных символов y_i осуществляется адаптивным арифметическим кодом из [3] с помощью различных кодеров, “настроенных” на различные вероятности появления нулей и единиц, и может быть описано следующим образом.

“Особые” символы $\underline{0},\underline{1}$ кодируются с помощью кодера K_0 с вероятностями $\widetilde{q}(i)^{l_i}$ и $1-\widetilde{q}(i)^{l_i}$ для 0 и 1 соответственно, где $\widetilde{q}(i)$ и l_i определяются по формуле (1) на основе статистики окна $x_{i-\omega}\dots x_{i-1}$.

Рассмотрим кодирование символов, расположенных внутри блока $y_1\dots y_{l_j}$ длины l_j. Пусть $y_1\dots y_{n-1}=\underbrace{0\dots0}_{n-1}$ $(n=1,2,\dots l_j)$. Тогда символ y_n, находящийся в n-ой позиции после $(n-1)$ нулей, кодируется с помощью кодера K_n адаптивным арифметическим кодом из [3].

Наконец, символы в блоке $y_1\dots y_{l_j}$, следующие после появления в этом блоке 1, кодируются с помощью кодера $\widetilde{K}$ с исходными

вероятностями $\tilde{q}(j)$ и $\tilde{p}(1,j)$ для 0 и 1 соответственно, где $\tilde{q}(j)$ и $\tilde{p}(1,j)$ определяются на основе статистики окна $x_{j-\omega}\dots x_{j-1}$ (расположенного перед блоком исходной последовательности, соответствующим блоку $y_1\dots y_{l_j}$).

Следует отметить, что в рассмотренном методе кодирования каждой длине блока l_j соответствует свой собственный набор кодеров и декодеров $K_0, K_1 \dots K_{l_j}, \tilde{K}$. Оценим общее число кодеров. Оно равно

$$\sum_{l=1}^{l_{\max}}(l+1)=\frac{l_{\max}^2}{2}+\frac{3}{2}\cdot l_{\max},$$

где $l_{\max}$ определяется формулой (2).

Отметим, что в адаптивном арифметическом коде со скользящим окном объем памяти кодера и декодера определяется необходимостью хранить окно длины ω, где $\omega < C/r$, а C - константа. Если избыточность $r \to 0$, то для метода адаптивного арифметического кодирования, основанного на скользящем окне, объем памяти кодера и декодера V и среднее время кодирования и декодирования T удовлетворяют соотношениям

$$V = O(1/r), \qquad (3)$$

$$T = O(\log(1/r)\log\log(1/r)\log\log\log(1/r))$$

Экспериментальные результаты

Формула(3) дает общее представление об эффективности предложенного метода. Однако более важным с точки зрения практического применения является вопрос о том, как предлагаемый метод работает на реальных данных. Для экспериментальной проверки эффективности рассмотренного алгоритма (назовем его NEW) и известного архиватора bzip2, применяющего BWT-преобразование, использовались несколько текстовых файлов разных размеров. Результаты работы обоих алгоритмов сравнивались по двум характеристикам: степени сжатия текстов k (в битах) и времени t (в секундах), требующегося кодеру для их упаковки. Под степенью сжатия в данном случае понимаем количество бит, которым представляется в сжатом файле один байт (8 бит) исходного (несжатого) текста. Иначе говоря, если L - размер исходного файла, L_1 - размер сжатого файла, то степень сжатия – это $8\cdot(L_1/L)$. Конфигурация тестового компьютера IBM PC следующая: процессор CPU Intel Pentium 4, тактовая частота 2,53 ГГц, объем оперативной памяти 512 Мбайт, операционная система Windows XP. В таблице 1 показаны результаты коэффициентов сжатия для нового алгоритма NEW и архиватора bzip2.

Таблица 1 - Результаты коэффициента сжатия алгоритмов NEW и bzip2.

№	Файл	Размер, Байт	k_{NEW}	k_{BZIP2}
1	article	67739	2,59	2,71
2	book	349270	2,89	3,12
3	journal	307930	2,35	2,58
4	document	962186	2,66	2,83
5	works	5410342	2,30	2,54

В приведенной ниже таблице 2 показаны результаты времени кодирования для обоих методов сжатия

Таблица 2 - Результаты скорости сжатия алгоритмов NEW и bzip2.

№	Файл	Размер, байт	t_{NEW}	t_{BZIP2}
1	article	67739	0,45	0,78
2	book	349270	2,74	4,12
3	journal	307930	2,15	3,96
4	document	962186	15,06	18,41
5	works	5410342	51,32	66,28

Из таблиц видно, что алгоритм NEW имеет лучший коэффициент сжатия и лучшую скорость кодирования, чем bzip2, что доказывает эффективность предложенного метода.

СПИСОК ЛИТЕРАТУРЫ

1. Burrows, M. A block sorting lossless data compression algorithm. / M. Burrows, D. Wheeler, - Technical Report 124, Digital Equipment Corporation, 1994. - 18 page.
2. Ryabko, B. Data compression by means of a «book stack» // B. Ryabko. - Problems of Information Transmission. - 1980, v. 16: (4). - P. 265-269.
3. Ryabko, B. Homophonic coding with logarithmic memory size // B. Ryabko, Fionov A. - Algorithms and Computation Berlin: Springer. - 1997. - P. 253-262.

Научный сотрудник, к.ф.-м.н. ***Бакулина М.П.*** *тел. 8-961-215-79-36, marina@rav.sscc.ru - Институт Вычислительной Математики и Математической Геофизики СО РАН*

УДК 621.391