УДК 681.3.053

И.В. ТРУФАНОВ, НТУ "ХПИ" (г. Харьков)

СЖАТИЕ ИНФОРМАЦИИ АНАЛИТИЧЕСКИМ ПРЕДСТАВЛЕНИЕМ ФУНКЦИИ ИСТОЧНИКА ДАННЫХ

У статті розглянуте питання стиснення інформації з використанням позиційного підходу до розташування певних частин інформаційної послідовності й наступного аналітичного визначення функції, що описує дані позиції.

The article considers the question of compression of the information which uses the positional approach to layout of defined parts of an information sequence and consequent analytical definition of the function describing these positions.

Постановка проблемы. Количество нужной человеку информации неуклонно растет. Скорость ее роста превышает рост объема устройств для хранения данных и рост пропускной способности линий связи. Использование сжатия информации позволяет в несколько раз сократить требования к объему устройств хранения данных и пропускной способности каналов связи без дополнительных издержек. Наряду с этим широко используемые методы сжатия, как правило, устарели и не обеспечивают достаточной степени сжатия. Таким образом, разработка и внедрение новых алгоритмов сжатия позволит значительно сократить издержки на аппаратное обеспечение вычислительных систем.

Анализ литературы. Основоположником науки о сжатии информации принято считать Клода Шеннона. Его теорема об оптимальном кодировании показывает, к чему нужно стремиться при кодировании информации и на сколько та или иная информация при этом сожмется. На основе ряда опытов он пришел к выводу, что количество информации в английском тексте колеблется в пределах 0.6 – 1.3 бита на символ [1].

Первые алгоритмы сжатия были примитивными в связи с тем, что была примитивной вычислительная техника. Многие из них применяются и в настоящее время. К ним можно отнести сжатие кодами Шеннона-Фоно [2], Хаффмена [3], RLE-кодирование [4]. С развитием мощностей компьютеров стали возможными все более мощные алгоритмы. Настоящим прорывом было изобретение Лемпелем и Зивом в 1977 г. словарных алгоритмов [5]. Суть алгоритма состоит в том, что цепочки событий помещаются в словарь. Если в дальнейшем во входном потоке встречается цепочка событий, которая присутствует в словаре, то вместо этой цепочки в выходной поток помещается ссылка на элемент словаря. Словарные алгоритмы позволяли кодировать повторяющиеся строки символов, что дало возможность резко повысить степень сжатия.

Важную роль сыграло изобретение примерно в это же время арифметического кодирования [2], позволившего воплотить в жизнь идею Шеннона об оптимальном кодировании. Текст, сжатый арифметическим кодером, рассматривается как некоторая двоичная дробь из интервала [0, 1). Каждый символ исходного текста представляется отрезком на числовой оси с длиной, равной вероятности его появления, и началом, совпадающим с концом отрезка символа, предшествующего ему в алфавите. Удачной альтернативой арифметическому кодеру, широко используемой в настоящее время, является Range-кодер [6]. Основным его отличием от арифметического кодера является не побитное, а побайтное сравнение верхней и нижней границы, а также побайтное помещение закодированного результата в выходной поток. Такой подход позволяет заметно повысить быстродействие и качество кодера.

В 1984 г. Т. Велч предложил словарный алгоритм LZW, в котором все литеральные события являются элементами словаря [7]. Таким образом, в выходной поток помещаются только ссылки на элементы словаря, что упрощает алгоритм. Велч показал, что алгоритм может быть легко реализован аппаратно.

Следующим прорывом было изобретение в 1984 г. алгоритма предсказания по частичному совпадению (PPM) [8] – метода контекстно-ограниченного моделирования, который позволяет оценить вероятность символа в зависимости от предыдущих символов. На сегодняшний день алгоритм PPM является наилучшим алгоритмом для сжатия текстовой информации.

Последние работы в области сжатия информации ориентированы на представление сжимаемой информации не в виде потока бит (байт, слов и т.д.), а в виде дискретной или аналоговой функции. Одним из таких методов являются спектровыделяющие преобразования [9]. Метод применяется для сжатия последовательностей, содержащих пространственную избыточность – при сжатии звука, растровой графики, растровых трехмерных изображений, видео и т.д. Данные преобразования преобразуют поток событий к его спектру. После чего низкочастотные составляющие (содержащие основную информацию) кодируются, а низкочастотные (содержащие малозначимую информацию и шум) отбрасываются.

Перспективным направлением является использование для сжатия информации нейроалгоритмов. Первые удачные применения нейросетей при сжатии немецкого текста описаны в совместной работе ученых Швейцарии и Германии Юргена Шмидхубера и Стефана Хайла [10], которые обучили нейронную сеть с 25 нейронами в скрытом слое сжимать текст 20 статей (10000 – 20000 символов в каждой) с качеством сжатия выше, чем у описанных методов. Представленный Шмидхубером и Хайлом метод пока далек от практического применения, поскольку обучение сети заняло 3 дня и основной задачей по модификации метода, до настоящего времени, является

увеличение быстродействия до приемлемых величин. Обзор имеющихся источников показал, что в настоящее время практически все алгоритмы сжатия данных подходят к источнику информации как к источнику потока данных. Поток рассматривается без учета позиций сжимаемых данных, а в виде последовательности бит (байт, слов и т.д.), что, возможно, несколько ограничивает континуум алгоритмов, применение которых может привести к сжатию информации.

Цель статьи. Возможность применения алгоритмов из смежных областей информатики может привести к созданию алгоритма сжатия информации с более высоким качеством сжатия. Для реализации этой возможности информацию источника данных можно представить в виде дискретной функции. Статья содержит анализ данного подхода с целью определения применимости одного из алгоритмов прикладной теории цифровых автоматов для задачи сжатия данных.

Алгоритм сжатия. Предположим, что источник информации поместил во входной поток N бит двоичной информации. Алгоритм сжатия можно представить в следующем виде:

Шаг 1. Входной поток данных преобразуется к виду k-значной дискретной функции V, содержащей $l =]N / \log_2(k)[$ значений.

Шаг 2. Составляется таблица значений функции, содержащая l строк и k столбцов. Для каждого значения функции на пересечении соответствующей позиции и столбца заносится m-значный код присутствия. Все остальные позиции заполняются кодом, отличным от кода присутствия.

Шаг 3. Составляется аналитическое описание функций $f_i(j), i \subset R^k$, $j \subset [0, l-1]$ для каждого из k столбцов.

Шаг 4. Поскольку каждая из функций f_i описывает только позиции присутствующих значений, то набор описаний является избыточным, и одну функцию можно исключить, заполнив при восстановлении все позиции соответствующим значением. Поэтому функцию с максимальной длиной описания не учитываем, а сохраняем только значение, которое она кодирует – код заполнения (FC).

Шаг 5. Аналитическое описание функций f_i кодируется одним из известных методов [2, 3], полученные коды заносятся в выходной поток.

Алгоритм восстановления. Для восстановления сжатой информации необходимо определить величины k – количество значений в каждой позиции функции, l – количество позиций функции, m – количество значений в коде присутствия, код заполнения – FC. Величины k и m можно использовать статические, величина l определяется из входного потока по сохраненной

длине закодированной последовательности, *FC* также извлекается из входного потока.

Шаг 1. Определяется величина *l*.

Шаг 2. Составляется *k*-значный вектор $\bar{r}$ значений функции длиной *l*. Все элементы вектора заполняются кодом заполнения *FC*.

Шаг 3. Для всех $i \subset R^k$ и $j \subset [0, l-1]$ проводится декодирование аналитического представления функции f_i и вычисляется ее значение $f_i(j)$. Если значение $f_i(j)$ равно коду присутствия, то в j позиции вектора $\bar{r}$ устанавливается значение *i*.

Шаг 4. Вектор $\bar{r}$ преобразуется к двоичному виду и помещается в выходной поток.

Пример. Пусть источник информации поместил во входной поток последовательность из 128 бит информации. Поток сформируем случайным образом, например, символьную последовательность следующего вида: "2322233323332222". Количество значений функции и кода присутствия выберем двоичные, т.е. *k* = 2, *m* = 2.

Шаг 1. Определяем величину $l = 128/\log_2(2) = 128$. Дискретная функция входного потока *V* = 00110010 00110011 00110010 00110010 00110010 00110011 00110011 00110011 00110010 00110011 00110011 00110011 00110010 00110010 00110010 00110010 $_{(2)}$.

Шаг 2. Составляем таблицу значений (табл. 1). В качестве кода присутствия выберем значение "1". Поскольку *k* = 2 и *m* = 2, то второй столбец является инверсией первого.

Таблица 1

Таблица значений функции

j	*V*
0	0
1	0
2	1
3	1
4	0
.	.
.	.
.	.
122	1
123	1
124	0
125	0
126	1
127	0

Шаг 3. Для составления аналитического описания функции используем метод минимизации логического описания функции – метод Квайна – Мак-Класки [11, с. 200 – 201].

Функции запишем в следующем виде:

$$f_0(j) = 0 \vee 1 \vee 4 \vee 5 \vee 7 \vee 8 \vee 9 \vee 12 \vee 13 \vee 16 \vee 17 \vee 20 \vee 21 \vee 23 \vee 24 \vee 25 \vee 28 \vee 29 \vee$$
$$\vee 31 \vee 32 \vee 33 \vee 36 \vee 37 \vee 39 \vee 40 \vee 41 \vee 44 \vee 45 \vee 48 \vee 49 \vee 52 \vee 53 \vee 56 \vee 57 \vee$$
$$\vee 60 \vee 61 \vee 64 \vee 65 \vee 68 \vee 69 \vee 71 \vee 72 \vee 73 \vee 76 \vee 77 \vee 80 \vee 81 \vee 84 \vee 85 \vee 88 \vee$$
$$\vee 89 \vee 92 \vee 93 \vee 96 \vee 97 \vee 100 \vee 101 \vee 103 \vee 104 \vee 105 \vee 108 \vee 109 \vee 111 \vee 112 \vee$$
$$\vee 113 \vee 116 \vee 117 \vee 119 \vee 120 \vee 121 \vee 124 \vee 125 \vee 127;$$

$$f_1(j) = 2 \vee 3 \vee 6 \vee 10 \vee 11 \vee 14 \vee 15 \vee 18 \vee 19 \vee 22 \vee 26 \vee 27 \vee 30 \vee 34 \vee 35 \vee 38 \vee 42 \vee$$
$$\vee 43 \vee 46 \vee 47 \vee 50 \vee 51 \vee 54 \vee 55 \vee 58 \vee 59 \vee 62 \vee 63 \vee 66 \vee 67 \vee 70 \vee 74 \vee 75 \vee$$
$$\vee 78 \vee 79 \vee 82 \vee 83 \vee 86 \vee 87 \vee 90 \vee 91 \vee 94 \vee 95 \vee 98 \vee 99 \vee 102 \vee 106 \vee 107 \vee$$
$$\vee 110 \vee 114 \vee 115 \vee 118 \vee 122 \vee 123 \vee 126.$$

Представим номер позиции j в бинарном виде с количеством разрядов $p = \log_2(l) = \log_2(128) = 7$ (табл.2).

Таблица 2

Преобразованная таблица значений функции f_0

l_7	l_6	l_5	l_4	l_3	l_2	l_1	f_0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	
1	1	1	1	1	1	1	1

Минимизируя логическое описание указанным методом, получим выражения для функций f_0 (1) и f_1(2).

$$f_0(l_1, l_2, l_3, l_4, l_5, l_6, l_7) = \bar{l}_7 \cdot \bar{l}_6 \cdot l_5 \cdot l_3 \cdot l_1 \vee \bar{l}_5 \cdot \bar{l}_4 \cdot l_3 \cdot l_1 \vee l_7 \cdot l_6 \cdot l_3 \cdot l_1 \vee \cdot \bar{l}_2; \quad (1)$$

$$f_1(l_1, l_2, l_3, l_4, l_5, l_6, l_7) = \bar{l}_6 \cdot \bar{l}_5 \cdot l_4 \cdot l_2 \vee l_7 \cdot l_6 \cdot l_4 \cdot l_2 \vee \bar{l}_7 \cdot l_6 \cdot l_5 \cdot l_2 \vee$$
$$\vee l_7 \cdot \bar{l}_6 \cdot l_5 \cdot l_2 \vee \bar{l}_3 \cdot l_2 \vee l_2 \cdot \bar{l}_1. \quad (2)$$

Шаг 4. Функция f_1 содержит максимальное аналитическое описание, поэтому код заполнения *FC* принимаем равным 1. В качестве результирующего описания выбираем функцию f_0.

Шаг 5. Каждый элемент функции описания может принимать значения l_b, $\bar{l}_b$, "отсутствует". Выберем для кодирования описания простой двоичный код. Количество всех комбинаций в p = 7 разрядах равно 3^7 и составляет 2187 вариантов. Для кодирования в двоичном виде номера каждого набора понадобится $\log_2(2187) \approx 12$ бит. Код для функции f_0 займет 12∗4=48 бит.

Сжатие закончено. Размер выходных данных составляет 48 бит закодированного аналитического описания и один бит кода заполнения, итого 49 бит. Исходная последовательность сжата в 2,6 раза. В табл. 3 приведены

данные по степени сжатия сгенерированной последовательности разными методами, из которой следует, что на предложенной последовательности существующие алгоритмы показали степень сжатия в несколько раз хуже.

Таблица 3

Сравнительная степень сжатия последовательности

Метод	Степень сжатия
Предлагаемый метод	2,60
Арифметическое кодирование	1,14
RLE–кодирование	1,06
Предсказание по частичному совпадению (PPM)	0,94
Модификация LZW (gzip)	0,43

Физический смысл. В рассмотренном примере аналитическая функция представляет собой логическое описание кластеров значений кода присутствия в 7-мерном двоичном пространстве номеров позиций. Входные последовательности небольшой длины могут сжиматься с увеличением размера выходной последовательности, вследствие низкой вероятности образования кластеров большого размера. В больших последовательностях вероятность образования и количество кластеров должно быть выше, соответственно и степень (качество) сжатия будет выше.

Выводы. Несмотря на показанное в тестовом примере высокое качество сжатия, до практического внедрения алгоритма необходимо провести серии опытов по: 1) тестированию алгоритма на наборах различной длины; 2) аналитическому описанию *k*-значными функциями; 3) составлению аналитического описания другими методами; 4) применению более эффективного кодирования аналитического представления функций.

Список литературы: **1.** *Shannon C.E*. Prediction and entropy of printed English // Bell System Technical Journal, 1951. – P. 50 – 64. **2.** *Мастрюков Д.* Арифметическое кодирование. Алгоритмы сжатия информации. Часть 2. Арифметическое кодирование // Монитор. – 1994. – № 1. – С. 20 – 23. **3.** *Hirschberg D., Lelewer D.* Data compression. Computing Surveys. – 1987. – 19. – № 3. – P. 261 – 297. **4.** *Романов В.Ю.* Популярные форматы файлов для хранения графических изображений на IBM PC. – М.: Унитех, 1992. – 156 с. **5.** *Ватолин Д., Ратушняк А., Смирнов М., Юкин В*. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: Диалог-МИФИ, 2002. – 384 с. **6.** *Martin G. N*. Range encoding: an algorithm for removing redundancy from a digitized message // Video & Data Recording Conference. – Southampton. – 1979. **7**. *Сэломон Д.* Сжатие данных, изображений и звука. – М: Техносфера, 2004. – 368 с. **8.** *Sadakane K., Okazaki T., Imai H.* Implementing the Context Tree Weighting Method for Text Compression. // Proceedings of IEEE Data Compression Conference. – 2000. – Snowbird. – Utah. – P.123 – 132. **9.** *Said A., Pearlman W.A.* A new, fast, and efficient image codec based on set partitioning in hierarchical trees // IEEE Trans. On Circuits and Systems for Video Technology. – 1996. – P.234 – 250. **10.** *Schmidhuber J., Heil S.*. Sequential neural text compression // IEEE Transactions on Neural Networks. – Vol. 7 (1). – 1996. – P.142 – 146. **11.** Прикладная теория цифровых автоматов / *Самофалов К.Г., Романкевич А.М., Валуйский В.Н., Каневский Ю.С., Пиневич М.М*. – К.: Вища Школа, 1987. – 375 с.

Поступила в редакцию 30.04.2005