

УДК 681.3.05

## Методы сжатия и компактного дискретного представления структурированных блоков информации

*Рассмотрена проблема избыточности при дискретном представлении блоков информации с внутренней структурой. Предложены новые специализированные методы сжатия указанных информационных блоков. Важными свойствами предлагаемых методов являются высокие параметры сжатия, симметричность прямого и обратного преобразований относительно трудоемкости, простые требования к вычислительным ресурсам, а также высокая скорость преобразований.*

А. А. Курочкин, А. П. Мартынов,  
Д. Б. Николаев, В. Н. Фомченко

В настоящее время обеспечение безопасности данных в сетях и многопользовательских системах принимает все большее значение. Для этой цели среди традиционных методов преобразования информации существуют два базовых типа: перестановка и подстановка, композиция которых позволяет синтезировать довольно хорошие преобразования. Метод перестановки "перемешивает" символы сообщения по определенному правилу. Метод подстановки замещает одни символы другими и сохраняет порядок их следования в сообщении. Методы подстановки в формализованном виде представляются таблицами подстановки, которые при дискретном представлении, как правило, обладают существенной избыточностью. Например, перестановка из 24 символов представляется в виде 192-битной строки (24 байта), в то время как теоретически для представления такой перестановки требуется лишь 80 бит, т. е. возникает более чем двукратная избыточность.

Избыточность информации снижает стойкость систем обеспечения безопасности данных, поэтому параллельно развивается наука о сжатии данных. Кроме того, сжатие данных используется с целью увеличения емкости различных устройств, предназначенных для хранения информации (так как потребность в размещении большего объема информации в меньшей области хранения по-прежнему сохраняет актуальность), а также для "расширения" пропускной способности каналов связи. Таким образом, сжатие данных позволяет:

- обеспечить должный уровень защиты данных (за счет повышения стойкости);
- увеличить емкость области хранения данных при сохранении объема;
- сократить время передачи данных (что особенно актуально для каналов связи с ограниченной пропускной способностью, например, телефонных линий).

Существуют два простых, но довольно эффективных метода сжатия текстовых данных. Первый метод сжимает данные из 8 бит в 7. Если восьмой бит всегда фиксирован и не используется, например, для проверки на четность, то простейшим способом уплотнить 8 символов до 7 является распределение семи значащих битов первого байта по семи незанятым позициям неис-



пользуемого восьмого бита всех остальных байтов от второго до восьмого. Для восстановления первого байта необходимо составить его, используя значения восьмого бита оставшихся семи байт. Данный метод сжатия уплотняет текстовые файлы на 12,5 %.

Второй метод сжимает текстовые данные из 4 бит в 3. Поскольку 6 бит позволяют закодировать значения от 0 до 63, то все строчные и прописные буквы, например, английского алфавита, содержащего 26 букв, а также наиболее распространенные знаки препинания могут быть представлены с помощью 6-битных слов. Для сжатия символов в 6-битные слова следует отказаться от кодовой таблицы ASCII и каждому символу присвоить новое 6-битное значение, которое представляет собой индекс в таблице символов. При сжатии данных сохраняются индексы, причем 4 значения индекса представляются 3-байтным пакетом. При распаковке индексы используются для получения стандартного ASCII-кода для каждого символа. Таким образом, достигается сжатие на 25 %.

Рассмотренные методы сжатия данных ориентированы преимущественно на текст и достаточно просты, но, несмотря на свою простоту, обеспечивают вполне удовлетворительное сжатие данных. Подобные методы сжатия, в которых для кодирования символов используется минимально необходимая длина кода символа, вполне применимы для так называемого "грубого" сжатия перестановок. Например, перестановка из 24 символов при использовании такого метода сжатия представляется в виде 120-битной строки (24 слова по 5 бит, так как с помощью 5-битных слов могут быть закодированы все 24 символа).

Основными классическими универсальными методами сжатия данных являются кодирование Хаффмана (Huffman coding) и методы Лемпела – Зива (Lempel-Ziv method).

Кодирование Хаффмана использует только частоту появления одинаковых байтов во входном блоке данных и базируется на чрезвычайно полезной во многих приложениях структуре данных бинарное дерево. Часто встречающимся символам входного потока данных ставится в соответствие цепочка битов меньшей длины, а встречающимся редко – цепочка большей длины.

Рассмотрим следующую задачу. Существует алфавит из  $n$  символов и состоящее из символов этого алфавита длинное сообщение. Необходимо закодировать сообщение в виде строки битов следующим образом: каждому символу алфавита присваивается определенная последовательность битов – код, а затем отдельные коды составляющих сообщение символов соединяются, и получается кодировка сообщения. Например, алфавит состоит из четырех символов – A, B, C и D. Если каждому символу назначен следующий 2-битный код: A – 00, B – 01, C – 10, D – 11, то сообщение ABACCDА кодируется 14-битной строкой 00010010101100.

Для минимизации длины закодированного сообщения необходимо, используя частоты появления каждого символа в сообщении, построить дерево Хаффмана. Находим два символа, появляющиеся наименее часто: B и D. Будем различать их по последнему биту кодов: B – 0, D – 1. Соединим эти символы в псевдосимвол BD, появление которого означает, что это либо символ B, либо символ D. Частота появления этого нового символа равна сумме частот двух составляющих символов, т. е. 2. Теперь у нас есть три символа: A (частота 3), C (частота 2) и BD (частота 2). Снова находим два символа с наименьшей частотой: C и BD. Будем различать их по последнему биту кодов: C – 0, BD – 1. Затем два символа объединяются в псевдосимвол CBD с частотой 4. Таким образом, осталось только два символа: A и CBD. Будем различать их по последнему биту кодов: A – 0, CBD – 1. Символы A и CBD объединяются в псевдосимвол ACBD.

Псевдосимвол ACBD содержит весь алфавит, ему присваивается в качестве кода пустая строка битов нулевой длины. Двум символам, составляющим ACBD (A и CBD), присваиваются соответственно коды 0 и 1. Если встречается 0, то закодирован символ A, если 1 – то это C, B или D. Аналогично двум символам, составляющим CBD (C и BD), назначаются соответственно коды 10 и 11. Первый бит указывает, что символ входит в группу CBD, а второй позволяет отличить C и BD. Символам, составляющим BD (B и D), назначаются соответственно коды 110 и 111. В результате этого процесса получается код переменной длины, причем префиксный, так как код лю-

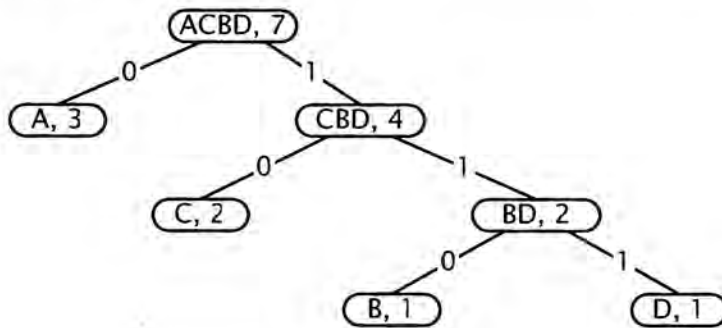


## СРЕДСТВА ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННЫХ ДЕЙСТВИЙ

бого символа не совпадает с началом кода другого символа, и символам, которые появляются в сообщении часто, присваиваются более короткие коды, чем тем, которые встречаются редко.

На практике для однотипных входных данных коды можно создавать не на основе частоты вхождения символов в отдельно взятом сообщении, а на основе их частоты во всем множестве таких сообщений и для кодирования сообщений использовать один и тот же код. Например, если сообщения состоят из английских слов, могут использоваться известные данные о частоте появления символов английского алфавита.

Операция объединения двух символов в один предполагает использование структуры бинарного дерева. Каждый лист представляет символ исходного алфавита. Каждый узел, не являющийся листом, представляет соединение символов из листьев – потомков данного узла. На рисунке приведено соответствующее бинарное дерево, называемое деревом Хаффмана по имени изобретателя этого метода кодирования.



Дерево Хаффмана

После построения дерева Хаффмана, код любого символа алфавита может быть определен просмотром дерева снизу вверх, начиная с листа, представляющего этот символ. Начальное значение кода – пустая строка. Каждый раз при подъеме по левой ветви к коду слева присписывается 0, при подъеме по правой ветви – 1.

Следует отметить, что дерево Хаффмана строго бинарное, т. е. каждый узел дерева, не являющийся листом, имеет непустые правые и левые поддеревья. Следовательно, если в алфавите  $n$  символов, то дерево Хаффмана, у которого  $n$  листьев, может быть представлено массивом узлов размером  $2n - 1$ . При использовании этого кода сообщение ABACCCDA кодируется как 0110010101110, что требует 13 бит.

Исходное сообщение при наличии кодировки сообщения и дерева Хаффмана может быть восстановлено следующим способом. Начиная с корня дерева, каждый раз, когда встречается 0, двигаемся по левой ветви, 1 – по правой ветви. Процесс повторяется, пока не дойдем до листа. Новый символ исходного сообщения есть символ, соответствующий этому листу.

Средняя степень сжатия при использовании кодирования Хаффмана равна 1,5 (сжатие на 33,3%), кроме того, это один из немногих методов, которые не увеличивают размер исходных данных в худшем случае. За счет применения этого метода для сжатия очень длинных сообщений, которые содержат встречающиеся чрезвычайно редко символы, достигается существенная экономия: степень сжатия может достигать 8. Сказанное справедливо в большей части для текстовых сообщений и сообщений с неравномерной плотностью распределения символов. При сжатии перестановок существенных результатов метод не дает. Например, перестановку из 24 символов с использованием кодирования Хаффмана можно представить в виде 112-битной строки.



Методы Лемпела – Зива (LZ77, LZ78) относятся к методам словарного сжатия. Входную последовательность символов можно рассматривать как последовательность строк, содержащих произвольное количество символов. Идея словарных методов состоит в замене строк символов на индексы строк некоторого словаря. Строки, образующие словарь, называются фразами. Словарь – это набор таких фраз, которые, как мы полагаем, будут встречаться в обрабатываемой последовательности. В методах Лемпела – Зива словарь формируется на основании уже обработанной части входного потока, т. е. адаптивно. Принципиальным отличием методов является способ формирования фраз.

Метод LZ77 является методом со скользящим словарем или скользящим окном. В качестве словаря используется блок уже закодированной последовательности. Как правило, по мере выполнения обработки положение этого блока относительно начала последовательности постоянно меняется, словарь "скользит" по входному потоку данных. В схеме LZ77 указатели позволяют делать ссылки на любую фразу в окне установленного размера, которое предшествует текущей фразе. Если соответствие найдено, текущая фраза заменяется указателем на своего предшествующего двойника. Указатель включает в себя смещение в окне и длину фразы. Скользящее окно имеет длину  $N$  и состоит из двух частей:

- последовательности длины  $w$  уже закодированных символов, которая является словарем;
- буфера предварительного просмотра длины  $n$  (обычно  $n$  на порядок меньше  $w$ ).

Если к текущему моменту времени закодировано  $t$  символов  $S_1, S_2, \dots, S_t$ , то словарем являются  $w$  предшествующих символов  $S_{t-(w-1)}, S_{t-(w-2)}, \dots, S_t$ , а в буфере находятся ожидающие кодирования символы  $S_{t+1}, S_{t+2}, \dots, S_{t+n}$ . Необходимо найти самое длинное совпадение между строкой буфера, начинающейся с символа  $S_{t+1}$ , и всеми фразами словаря, которые могут начинаться с любого символа словаря и лежат в окне. Полученная в результате поиска фраза кодируется с помощью двух чисел: смещения от начала буфера и длины совпадения. Дополнительно в выходной поток записывается непосредственно следующий за совпавшей строкой буфера символ. Затем окно смещается на длину совпавшей фразы плюс один символ и выполняется новый цикл кодирования.

Декодирование данных осуществляется путем простой замены кода на блок символов, состоящий из фразы словаря и явно передаваемого символа. Декодер выполняет те же действия по изменению окна, что и кодер. Фраза словаря элементарно определяется по смещению и длине, поэтому важным свойством LZ77 является очень быстрая работа декодера, а поскольку при сжатии много времени тратится на поиск фраз, метод характеризуется сильной несимметричностью по времени – кодирование значительно медленнее декодирования. Способ LZ77 позволяет сжимать только сравнительно длинные последовательности.

Важной отличительной особенностью метода LZ78 является использование вместо скользящего окна грамматического разбора предшествующего текста с разложением его на фразы, причем в словарь записываются не все встречающиеся при кодировании строки, а лишь перспективные с точки зрения вероятности последующего использования.

На каждом шаге в словарь вставляется новая фраза, которая представляет собой конкатенацию одной из фраз словаря, имеющей самое длинное совпадение со строкой буфера, и символ, следующий за строкой буфера, для которой найдена совпадающая фраза. В отличие от семейства LZ77 в словаре не может быть одинаковых фраз. Указатели могут в дальнейшем ссылаться на эти фразы, однако они не могут ссылаться на составные части фраз. Кодер порождает только последовательность кодов фраз. Каждый код состоит из индекса фразы и символа.

Скорость раскодирования для метода LZ78 потенциально всегда меньше скорости метода со скользящим окном LZ77, так как в LZ77 затраты на поддержания словаря в правильном состоянии минимальны. С другой стороны, для LZ78 существуют эффективные реализации процедур поиска и добавления фраз в словарь, что обеспечивает значительное преимущество над методом LZ77 в скорости сжатия. Соотношение скоростей кодирования и декодирования обычно равно 3:2.



Методы Лемпела – Зива ориентированы на сжатие качественных данных, причем хорошая эффективность применения достигается в случае, если статистические характеристики обрабатываемых данных соответствуют модели источника с памятью, при этом вероятности порождения элементов определяются состоянием источника, а вероятности перехода из одного состояния в другое зависят от совокупности предыдущих и последующих состояний.

Интересное свойство методов Лемпела – Зива заключается в том, что если исходные данные порождены источником с определенными характеристиками, то коэффициент сжатия по мере кодирования приближается к минимально достижимому. Иначе говоря, количество битов, затрачиваемых на кодирование каждого символа, в среднем равно энтропии источника. Но, к сожалению, сходимость медленная, причем у метода LZ77 скорость приближения к энтропии источника меньше, чем у LZ78.

Очевидно, что для достижения хорошего (близкого к теоретически возможному) сжатия перестановок, необходимо применять способы, использующие характерные для перестановок свойства. В качестве одного из вариантов такого сжатия можно предложить метод пропорционального динамического кодирования (ПДК). Идея метода ПДК заключается в том, чтобы в зависимости от позиции кодируемого символа динамически изменять длину его кода, причем пропорционально мощности множества допустимых символов. Метод базируется на следующем свойстве перестановок: уже закодированные символы в последующем исключаются из множества допустимых символов еще незакодированных. Таким образом, алфавит и его размер в процессе кодирования динамически изменяются.

Рассмотрим, например, перестановку из 24 символов. Размер алфавита первого символа составляет 24 символа, второго – 23, ..., двадцать третьего – 2, двадцать четвертого – один соответственно, для кодирования символов с первого по восьмой требуется 5-битный код, с девятого по шестнадцатый – 4-битный код, с семнадцатого по двадцатый – 3-битный код, двадцать первого и двадцать второго – 2-битный код, двадцать третьего – 1-битный код, а двадцать четвертый кодировать вообще не требуется. Таким образом, с использованием метода ПДК перестановка из 24 символов представляется в виде 89-битной строки (8 слов по 5 бит, 8 слов по 4 бита, 4 слова по 3 бита, 2 слова по 2 бита и 1-битное слово), длина которой приближается к теоретически возможной.

Для достижения минимального теоретически возможного размера представления перестановки предлагается применять метод мультипликативно-аддитивного кодирования (МАК), использующий основное свойство перестановок: факториальную зависимость. В методе МАК перестановка изначально подвергается обработке с использованием метода ПДК, а затем представляется факториальным полиномом, члены которого являются произведением полученных после обработки ПДК символов на соответствующие размерам алфавитов символов факториалы. По существу, в методе МАК предлагается для кодирования перестановок использовать факториальную систему счисления.

Метод МАК позволяет достичь максимального теоретически возможного сжатия перестановок, так как число значений полинома, принимающего значения от 0 до своего максимального значения, равно факториалу числа символов в перестановке. Таким образом, перестановка из 24 символов с использованием метода МАК представляется в виде 80-битной строки, что соответствует минимальной теоретически необходимой для представления такой перестановки длине строки, т. е. исключается избыточность.

Важными свойствами методов ПДК и МАК являются хорошее сжатие перестановок ввиду того, что методы специально разрабатывались для сжатия такого типа данных, относительно высокая скорость кодирования и декодирования, связанная с невысокой сложностью этих процедур, а также скромные требования к вычислительным ресурсам, в частности к объему памяти для реализации соответствующих преобразований. Кроме того, указанные методы симметричны относительно трудоемкости и соответственно скорости прямого и обратного преобразований. Практиче-

ская программная реализация методов ПДК и МАК подтвердила осуществимость и практическую ценность предлагаемых методов.

### Список литературы

1. Шилдт Г. Теория и практика C++: Пер. с англ. СПб.: BHV – Санкт-Петербург, 1996.
2. Лэнгсам Й., Огенстайн М., Тененбаум А. Структуры данных для персональных ЭВМ: Пер. с англ. М.: Мир, 1989.

### **Analysis of the Problem of Compression and Compact Discrete Expression of Rearrangements**

A. A. Kurochkin, A. P. Martynov, D. B. Nikolayev, V. N. Fomchenko

*The problem of redundancy accompanying the discrete expression of data blocks featuring the inherent structure is considered. The new special methods of compression of the above data blocks are offered. The important properties of the methods set forth include high compression parameters, symmetry of the direct and reverse transformation with regard to labor content, unsophisticated requirements to the computing power, as well as the high speed of transformation.*