УДК 521.1, 514.01

Р.В. СКУРАТОВСКИЙ

МЕТОД БЫСТРОГО ТАЙМЕРНОГО КОДИРОВАНИЯ ТЕКСТОВ

Ключевые слова: *блочное кодирование, архивация, алгоритм сжатия данных, таймерное кодирование.*

ПОСТАНОВКА ЗАДАЧИ

Возможности таймерного кодирования изучала целая плеяда ученых. Были выявлены возможности таймерного маскирования информаци [1], построения корпоративной системы защиты информации [2], шифрования [3, 6, 7]. Однако главным преимуществом таймерного кодирования является возможность кодирования и сжатие информации. Этим вопросом занимались В.Ф. Бардаченко и Е.О. Осадчий [1, 5].

Известно, что при архивации данных иногда необходимо применять алгоритмы поиска, поскольку при работе, в частности, такого архиватора как RAR, используется алгоритм поиска BWT, оценка сложности которого составляет $O(n\log_2 n)$. Важной характеристикой алгоритма архивации является коэффициент сжатия информации. Его целевая функция определяется формулой

$$F = k_1 x + k_2 y \to \min ,$$

где x — объем, y — время (аддитивна форма), $x \le Q_1$, $y \le t_1$ (Q_1 — константа, задающая максимально допустимый объем).

Цель настоящей статьи — достичь наибольшего сжатия информации с наименьшими затратами времени. Граничной величиной сжатия произвольного текстового файла будем считать 8 бит. На практике при таймерном кодировании имеем линейный рост объема выделенной памяти, который зависит от количества меток. Для дальнейшего изложения материала используются термины из работ [7–9]. Предложеный в статье метод сжатия можно применять после сжатия текста уже известным ранее архиватором, который использует методы символьного кодирования. Оценка его сложности для процесса сжатия текста с n символами составляет величину $3(\text{length}\,|A|)\cdot([n/2]+1)n$, которая в отличие от существующих методов таймерного кодирования намного меньше, поскольку последние имели экспоненциальную зависимость времени работы от n. Здесь A — алфавит всех используемых в тексте символов.

В процессе работы алгоритма должны быть соблюдены следующие ограничения. Объединенные блоки текста, соответствующие раскодированным меткам,

отвечают всему тексту, скорость разархивирования которого не менее чем s_0. При сравнении границ сжатия данного метода с другими методами следует учитывать, что при символьном кодировании с однозначным декодированием, т.е. в префиксном кодировании [1], существует код с наибольшей длиной символа $L(c, \xi)$: $H(\xi) \le L(c, \xi) \le H(\xi) + 1$, где $H(\xi)$ — энтропия источника, т.е. его нижняя грань, $H(\xi) = \sum_{i=1}^{n} p_i \log_2 (1/\, p_i)$, представляющая среднее ожидаемое количество информации, которую содержит один символ. Очевидно, что при увеличении длины текста T длина кода, в частности арифметического, будет линейно возрастать. Предложенный здесь метод при использовании одной метки для закодирования текста требует память, необходимую для сохранения метки и оптимального порядка символов для текстового генератора. Этот порядок единый для всего текста, поэтому не влияет на функцию роста количества операций. При этом длина кода возрастает не более чем линейно с увеличением длины текста. Однако когда используется система меток с длиной блока кодируемого текста 3 байта, то даже после архивирования методом символьного кодирования и достижения нижней границы сжатия по Шеннону можно достичь сжатия, как максимум, в 24 раза после применения таймерного кодирования. Напомним, что нижняя граница сжатия по Шеннону при 32-символьном алфавите — это 5 бит на символ (а с учетом неравномерного распределения букв и взаимозависимости в последовательности букв — это 2 бита). В настоящей статье впервые предложены новые методы таймерного кодирования, доказана их эффективность, разработано соответствующее математическое обеспечение для его оптимизации.

ОСНОВНЫЕ МАТЕМАТИЧЕСКИЕ ВЕЛИЧИНЫ И СВОЙСТВА ИССЛЕДУЕМОГО МЕТОДА

Символы текста имеют относительные частоты f_s, $s \in A$, где A — алфавит символов из T, которые упорядочиваем по убыванию частот. Отметим, что генератор вырабатывает символы в заданном убывающем порядке последовательно по разрядам. Время и результат работы однозначно определяются таймерной меткой.

Определение 1. Статистически ориентированный генератор текстов *Gs* представляет алгоритм, который реализует необходимый ряд распределения частот $f_i > f_j$ символов $n_i < n_j$, где n_i, n_j — номера *i*-го и *j*-го символов в полученном порядке $\wp$.

Определение 2. Генератор текстов с алфавитным упорядочением обозначим *Gp*. Он последовательно реализует тексты, генерируя символы в алфавитном порядке, т.е. не учитывает частоты появления этих символов в определенном виде текстов.

Множество меток образует 1-сдвиг (сдвиг на один символ) B [9], а на множестве блоков кодируемого (сжимаемого) текста опредлен m-сдвиг A. Отображение φ: $A \to B$ является блочным кодированием. Текстовый генератор генерирует последовательно все тексты длиной k, рассматривая каждый из них в отдельном окне (блоке длиной k). Если после генерации одного из них на управляющий вход не поступил сигнал прерывания, то генератор начинает генерировать все тексты длиной $k+1$ и т.д. По окончании этого процесса получаем однозначно определенный символом из Y блок текста длиной m, где Y — алфавит пространства сдвига B. Таким образом, имеем биективное отображение φ^{-1}: $B \to A$. Поэтому для сокращения перебора текстов при разархивации данных в код сжатого текста C целесообразно включать также и длину исходного текста.

Определение 3. Таймерная метка соответствует необходимому состоянию текстового генератора. В общем случае таймерный генератор — это устройство, которое реализует отсчет (инкрементацию) в выбранной системе счисления с постоянной частотой и дискретизацией. В качестве системы счисления может быть

выбрана и временная система отсчета. Таймерные метки принимают значения из дискретного множества с одинаковыми расстояниями между соседними точками. Эти расстояния обратно пропорционально зависят от скорости работы *Gs*.

Определение 4. Номером символа $s \in A$ при заданном порядоке $\wp$ является его порядковый номер $n_s \in \mathbb{N}$ в этом упорядочении, которое осуществляется для статистического генератора *Gs* в соответствии с вероятностями, характерными для символов выбранного текста. Заметим, что весом символа будет именно его номер n_i в частотном упорядочении таком, что если $f_i > f_j$, то $n_j > n_i$.

Определение 5. Весом слова W или текста $T = a_1 a_2 a_3 \ldots a_n$ назовем величину

$$Wh(T) = \sum_{i=1}^{n} n_i |A|^{n-i},$$

где n_i — номер символа, который находится на i-м месте текста в заданном для генератора упорядочении, n — длина текста T, $|A| = N$ — мощность множества символов.

Текстовый генератор, поставив неправильно первую букву, вынужден после нее последовательно перебирать на всех позициях все буквы из A — множества символов, пока не дойдет до величины времени, равной таймерной метке для T. Например, если закодировано число $T = 1567$, а генератор поставил первую цифру 2 (что отвечает упорядочению 2,3,4,5,6,7,8,9,0,1), то он будет перебирать все цифры, увеличив сначала символ во втором разряде, затем все комбинации в следующих разрядах и так далее вплоть до набора $a_1 a_N a_N a_N$. Затем он возвратится к первой позиции, где заменит цифру 2 на следующую по порядку цифру 3 (которая также ошибочна) и снова начнет генерировать символы в меньших разрядах. Перебор продолжается до тех пор, пока генератор не поставит в первом разряде цифру 1. Таким образом, в каждом разряде от второго до четвертого было использовано девять или десять цифр, а в первом разряде были перебраны все десять цифр. Поэтому для данного T верхняя граница перебранных комбинаций при неправильно выбранном символе a_1 равна $n_1 \cdot 10^3$, а при неправильно выбраном символе a_2 этой оценкой будет $n_2 \cdot 10^2$ комбинаций и т.д. Тогда вес текста составляет $Wh(1567) = n_1 \cdot 10^3 + n_2 \cdot 10^2 + n_3 \cdot 10 + n_4$, где $n_1 = 10$, $n_2 = 4$, $n_3 = 5$, $n_4 = 6$, а при оптимальном упорядочении $n_1 = 1$, $n_2 = 2$, $n_3 = 3$, $n_4 = 4$.

Оценочная величина перебора при неправильно поставленном символе a_1 составляет $n_1 \cdot |A|^{n-1}$. В этом случае величина $|A|^{n-1}$ является весом позиции первого символа или первого разряда. В общем случае величина $|A|^{n-i}$ равна весу i-го символа.

Таким образом, $Wh(T)$ — это число перебранных гиперслов при выбранном упорядочении букв, после которого следует искомый текст T.

Утверждение 1. Зная веса символов, можно быстро вычислить вес всего текста.

Действительно, поскольку текст известен, то имеем символьное упорядочение $\wp$ с весами n_i символов из T. Поэтому можно применить формулу

$$Wh(T) = \sum_{i=1}^{n} n_i |A|^{n-i}.$$

Исследуем некоторые свойства таймерных меток текста. Пусть текст генерируется с одинаковой скоростью и фиксированным порядком.

Утверждение 2. Между множеством двоичных записей чисел и множеством таймерных меток этих чисел существует биективное отображение при фиксированном $\wp$, причем для построения отображения не нужно последовательного генерирования всех предшествующих текстов данному тексту.

Действительно, таймерную метку данного двоичного числа a, а значит и соответствующего ему текста, можно получить из формулы

$$t = \frac{N_2(a)}{V}.$$

Здесь $N_2(a)$ — двоичное число, которое отвечает коду данного текста (весу текста при данном порядке генерирования символов), V — скорость (частота) работы генератора. При этом $|N_2(a)| = Wh_p(T)$, где $Wh_p(T)$ — сложность порождения (веса) текста T при определенном линейном упорядочении. Для статистически ориентированного генератора с символьным упорядочением $\wp$ величина таймерной метки определяется формулой

$$t(\mu_0) = \frac{Wh_s(T)}{V}.$$

Зная вес текста, можно разделить его на k частей. Для каждой части вычислить вес и величину таймерной метки $t(\mu_i), 1 \le i \le k$. Таким образом, получена система таймерных меток $y = t(\mu_\Sigma)$ путем вычисления суммы ряда, а не генераторного перебора множества гиперслов в соответствии с полученным лексикографическим упорядочением согласно $\wp$.

Последовательность величин $|A|^n$ вычисляется рекурсивно и имеет временную сложность $\text{length}(|A|^2)\cdot([n/2]+1) = 2(\text{length}|A|)\cdot([n/2]+1) = 2(\log_2 A)([n/2]+1)$ согласно методу бинарного возведения в степень «четырех русских» [8], а длины символов из A ограничены одним байтом. Поэтому умножая $n = |T|$ на номер n_i, получаем точное общее время, при этом оценка временной сложности составляет $3(\text{length}|A|)\cdot([n/2]+1)n$. Таким образом, имеем время процесса архивации — таймерную метку генератора без запуска последовательного перебора текстов для получения метки.

Объем сжатой информации вычисляется, исходя из суммы объемов меток, которые соответствуют каждому блоку текста, представленному двоичным числом $N_2(a)$. Каждому блоку соответствует таймерная метка t. Ее объем равен $Q(t)$. Тогда объем после сжатия всего текста будет $Q(T) = \sum_{i=1}^{k} Q(t_i)$, где t_i — i-я метка из алфавита Y, k — количество блоков.

Напомним, что вероятность появления T — это кумулятивная вероятность $P(T) = p_1 p_2 \dots p_n$, где p_i — вероятность появления i-й буквы в тексте из данной области знаний. Непосредственно сами тексты упорядочиваются лексикографически. В этом случае среднестатистический вес текста равен математическому ожиданию от всех весов слов

$$M(Wh(W)) = \sum_{i=1}^{l} P(W_i) W_i,$$

где l — число слов в тексте.

Докажем эффективность статистического метода генерирования для множества гиперслов. Гиперслово w при лексикографическом порядке, который использует адаптивный статистический генератор *Gs*, имеет меньший номер, чем при использовании лексикографического упорядочения перебирающего генератора *Gp*. Для префикса гиперслова вводятся отдельная система частот букв и соответствующая система меток, а при неправильном выборе начальной буквы, например из i-й позиции, происходит генерирование всех последующих $|A|^{n-i}$ комбинаций, что характерно при работе *Gp*.

В процессе формирования текста генератор получает его двоичную метку для каждого варианта текста с целью сравнения сгенерированного числа, кото-

рое, естественно, отвечает определенному тексту, с меткой оригинального текста (обозначим ее μ_0). Метка μ_0, вычисленная при работе статистического генератора, соответсвует меньшему двоичному числу, чем метка μ_1 для генератора, действующего прямым перебором. Это обусловлено тем, что буквы, которые имеют большую вероятность появления, кодируются в среднем меньшими двоичными числами в таблице символьного порядка, т.е. имеют меньшие двоичные значения n_i в формуле $Wh(T) = \sum_{i=1}^{n} n_i |A|^{n-i}$, чем номера этих же букв при работе переборного генератора. Поэтому символы слова и соответствующие им тексты, которые более характерны для данной отрасли знаний, также получат меньшие номера и меньшие веса $Wh_S(T)$ при созданном статистическом упорядочении. Соответственно в единичной системе счисления [6] им соответсвуют меньшие таймерные метки:

$$t(\mu_0) = \frac{Wh(T)}{V}.$$

Для переборного генератора текста вес совпадает с двоичным числом $N_2(a)$, так как $t(\mu_1) = \frac{N_2(a)}{V}$. Однако $N_2(a) > Wh_S(T)$, поскольку для большинства значений i для n_i и m_i имеем $m_i > n_i$, где m_i — номер буквы, которая находится в i-м месте текста в порядке, используемом Gp, отсюда $t(\mu_0) < t(\mu_1)$.

Для архивации текста, что равносильно созданию его метки или системы меток, достаточно вычислить $t(\mu_0) = \frac{Wh_S(T)}{V}$, а для генератора Gp — определить $t(\mu_1) = \frac{N_2(a)}{V}$. Для этого нужно сделать считывание всех символов из T, где $|T| = N$, и выполнить $N-1$ умножений на вес разряда, т.е. на $|A|^k$, $0 \le k \le N-1$. Время вычисления произведения чисел равно произведению длин этих чисел. Последовательность величин $|A|^n$ вычисляется рекурсивно (сначала вычисляется A, а затем его кратные произведения и произведение с A) и имеет временную сложность $\text{length}(|A|^2) \cdot ([n/2]+1) = (\text{length}|A|)^2 \cdot ([n/2]+1)$, при этом длины символов из A ограничены одним байтом. Поэтому современный компьютер такое вычисление-определение $t(\mu_0)$ выполнит за доли секунды, поскольку его сложность равна $O(n^2)$. В отличие от экспоненциальной зависимости при ранее известных методах архивации, использующих посимвольное генерирование текста генератором Gp, которое предусматривало перебор большого количества вариантов и зависело от частот символьного и таймерного генераторов, рассмотренный здесь метод имеет принципиально меньшую временную оценку. Новая оценка сложности работы алгоритма принадлежит классу сложности методов классического символьного кодирования, например с использованием метода LamplZiv (оценка его сложности равна$n \log n$) и вспомогательного метода Burrowse-Wheller transformation (где используются циклические сдвиги всех подстрок и подслова, полученные в конце строки после сдвига), которые используют архиватор RAR. Как известно, для текста с равномерным распределением вероятностей символов (предположим, что их 32, тогда $H(\xi) = \sum_{i=1}^{32} p_i \log_2 p_i^{-1} = \sum_{i=1}^{32} 32^{-1} \log_2 2^5 = 5$) на один символ текста необходимо, как минимум, 5 бит.

Учитывая взаимозависимость символов, т.е. условные вероятности, а также неравномерное распределение вероятности символов, получаем, что энтропия источника (нижний предел сжатия) значительно уменьшается (2 бита на символ из 32-буквенного алфавита по исследованиям Шеннона).

Заметим, что если применить сначала классическое символьное кодирование, а затем таймерное кодирование, имеющее расстояние 3 байта между метками, то получим повторное сжатие в 24 раза. Действительно, поскольку в трех байтах содержится 24 бита, то теперь вместо них сохраняем всего $Q(a)$ бит на метку, что имеет меньший объем [7].

Утверждение 3. Если согласно лексикографическому упорядочению выполняется неравенсво $w_1 < w_2$, то $Wh(w_1) < Wh(w_2)$ и $t(w_1) < t(w_2)$.

Доказательство. Утверждение следует из того, что $\exists i$: $n_i < m_i$, $n_j = m_j$, $j < i$, поэтому $Wh(w_1) < Wh(w_2)$. Поскольку символы с бóльшим весом генерируются после символов с меньшим весом согласно статистическому порядку символов, то $t(w_1) < t(w_2)$.

ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТА СЖАТИЯ

Определение 6. Длина потокового кода на выходе L_{out} — это сумма длин кодов слов (или блоков слов) в новом алфавите Y. Средняя длина кода вычисляется так: $L_{\text{out}} = \sum_{i=1}^{k} l(C(w_i))p(w_i)$, где k — число блоков, на которые разбит текст; каждому из них соответствует метка — символ из Y.

Определение 7. Длина исходного потокового кода — это сумма длин слов $L_{in} = \sum_{i=1}^{k} l(w_i)$, средняя длина слова составляет $L_{in} = \sum_{i=1}^{k} l(w_i)p(w_i)$, где $l(w_i)$ — длина i-го блока текста, $p(w_i)$ — вероятность его появления.

Формула коэффициента сжатия имеет вид

$$K = \frac{\sum_{i=1}^{k} l(C(w_i))\,p_i}{\sum_{i=1}^{k} l(w_i)\,p_i} = \frac{k \cdot p_i}{k \cdot m \cdot p_i} = \frac{1}{m},$$

поскольку длина метки равна 1 байту (при наилучшем упорядочении символов). Более детально о длине метки, представленной в другой кодировке, изложено в [1, 4, 7]. Отсюда следует, что коэффициент сжатия при таком коде обратно пропорционален длине блока w_i, который отображается в символ из нового алфавита Y. Таким образом, при m-сдвиге, т.е. кодировании блоками длиной m, коэффициент сжатия равен $\frac{1}{m}$, а это меньше, чем у алгоритма кодирования LZW, имеющего K между границами $O\left(\frac{\ln n}{n}\right)$ и $O(1)$, где n — длина текста, значительно превышающая m — длину блока. Величину m можно увеличить до n, получив $K = 1/n < \ln n / n$. Значение m зависит от быстродействия текстового генератора.

Таким образом, в настоящей статье предложен метод построения статистически ориентированного текстового генератора, быстродействие которого выше, что позволяет уменьшить K. Для достижения максимального уменьшения объема всего текста после его сжатия можно использовать только одну метку, объем которой представлен серией нулей, еще одним битом или меткой в другой кодировке в конце серии символов, имеющей нужную длину, и соответсвующей величине $t(\mu_0)$. Серию нулей s можно условно разбить на блоки b так, чтобы величина $\text{length}(b)$ была одного порядка с этой серией, и сохранить их количество и длину таких блоков. При этом существует остаток от деления s на $\text{length}(b)$. Дли-

ну блока length(b) целесообразно выбрать, равную числу 2 в некоторой степени. Если у пользователя, который будет осуществлять разархивирование, есть статистический порядок символов для данного типа текстов, то объем архивированной информации будет равен объему записи одной метки. При этом время разворачивания текста незначительно превышает время разворачивания текста при множественной системе меток, но его можно уменьшить наращиванием частоты таймерного счетчика.

Основной результат при использовании предложенного метода состоит в том, что время сжатия текста не зависит от частот символьного генератора и таймерного счетчика и не определяется в процессе статистически-ориентированого перебора или простого перебора. Оно вычисляется как сумма степенного ряда с весовыми коэффициентами соответствующего символьного упорядочения $\wp$ и обладает лишь квадратичной зависимостью сложности от длины текста n, а именно $3(\text{length}|A|)\cdot([n/2]+1)n$, что в отличие от экспоненциальной зависимости при методе статистического перебора является принципиальным уменьшением сложности процесса архивирования. На практике для современного компьютера с ресурсами ноутбука удалось программно реализовать таймерные счетчики с частотой, приблизительно равной половине тактовой частоты процессора. Этот скоростной метод можно успешно применять для сжатия SMS сообщений. При этом нет необходимости встраивать таймерный счетчик в мобильный телефон, поскольку можно воспользоваться системой спутниковой связи JPRS для соединения с эталонными атомными часами и синхронизатором, который можно реализовать на базе телефонной станции. А поскольку атомные часы обладают большей величиной дискретизации времени, чем таймерный счетчик, который можно реализовать на базе ноутбука, то можно использовать генератор текстов с большей частотой, что также увеличивает скорость работы.

СПИСОК ЛИТЕРАТУРЫ

1. Бардаченко В.Ф., Джунин Е.Н. Анализ характеристик таймерных маскраторов информации // Управляющие системы и машины. — 1994. — № 3. — С. 16–22.
2. Бардаченко В.Ф., Кишинский С.И. Построение корпоративной системы фондовой биржи с использованием таймерных технологий обработки и защиты информации // Там же. — 2000. — № 5. — С. 66–73.
3. Бардаченко В.Ф., Колесицкий О.К. Анализ современных средств аутентификации для систем защиты информации // Там же. — 2004. — № 3. — С. 81–92.
4. Осадчий Є.О. Підхід до покращення таймерних методів захисту інформації // Вісн. Терноп. академії народ. господарства. — 1999. — № 1. — С. 30–35.
5. Пат. 2128878 Российской Федерации, МКИ 6 H 03 K 23/00. *N*-разрядный счетчик / Е.А. Осадчий, А.Е. Осадчий, Опубл. 10.04.99; Бюл. № 10. — 16 с.
6. А.с. 1275471 СССР, МКИ G06F 15/38, 9/44. Устройство для преобразования кодов с одного языка на другой / В.И. Корнейчук, А.П. Марковский, Е.А. Осадчий, В.С. Бабак. — Опубл. 07.12.86, Бюл. № 45. — 8 с.
7. Осадчий Є.О., Осадчий О.Є., Ткаченко В.В., Курченко В.Д. Таймерне кодування в прогресивних інформаційних технологіях // Тез. допов. 12 Міжнар. конф. з автоматизації керування «Автоматика 2005» НТУУ «КПІ». — 2005. — **3**. — С. 55–75.
8. Koblitz N. Algebraic aspects of cryptography / Algori thms and Computation in Mathematics. — Berlin: Springer-Verlag, 2004. — **3**. — 207 p.
9. Lind D., Marcus B. An introduction to symbolic dynamics and coding. — Cambridge: Cambridge University Press, 1995. — 490 p.

Поступила 05.02.2012