УДК 621.391

Н. Н. ПОНОМАРЕНКО, Н. В. КОЖЕМЯКИНА, В. В. ЛУКИН

Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Украина

МЕТОД ЭНТРОПИЙНОГО РЕКУРСИВНОГО ГРУППОВОГО КОДИРОВАНИЯ

Рассмотрена задача кодирования информации с целью устранения ее статистической избыточности. Наиболее известными из методов, относящихся к этой области, являются арифметическое кодирование и кодирование Хаффмана. Недостатком этих и большинства других методов является недостаточная эффективность при кодировании символов сверхбольших алфавитов. В данной работе рассматривается метод энтропийного рекурсивного группового кодирования, который при высоком быстродействии и близкой к арифметическому кодированию эффективности лишен этого недостатка. Показано, что для больших алфавитов рекурсивное групповое кодирование обеспечивает выигрыш в степени сжатия по отношению к арифметическому кодированию до 40%.

Ключевые слова: *сжатие данных, сжатие изображений, арифметическое кодирование, кодирование Хаффмана, энтропийное кодирование.*

Введение

С каждым годом количество данных, передаваемых по каналам связи, увеличивается, причем опережающими темпами по отношению к увеличению пропускной способности линий связи. Это объясняется постоянным ростом трафика мультимедийных данных (изображений, видео, звука), внедрением новых стандартов, таких как 8к в телевидении [1], сопровождающихся увеличением объема передаваемых данных в десятки раз, увеличением трафика m2m (от устройств к устройствам). Поэтому исследования в области сжатия информации сохраняют высокую актуальность и повышенное внимание со стороны ученых всего мира.

Современные исследования в области сжатия данных продвигаются параллельно в нескольких направлениях. Во-первых, разрабатываются новые высокоуровневые методы сжатия изображений [2, 3], видео [4, 5] и звука [6]. Во- вторых, продолжают совершенствоваться универсальные высокоуровневые методы сжатия, такие как словарные методы сжатия [7], сжатие на основе предсказания по частичному совпадению [8]. В- третьих, продолжаются исследования в области низкоуровневых методов устранения статистической избыточности в данных, таких как арифметическое кодирование (АК) [9, 10] или кодирование Хаффмана (КХ) [11], которые входят в состав большинства высокоуровневых методов. Данная работа посвящена исследованиям именно в области подобных низкоуровневых методов сжатия, которые в последнее время становятся особенно актуальными. Дело в том, что задачи сжатия данных в телекоммуникационных системах, как правило, характеризуются огромным объемом обрабатываемых данных и их высокой статистической однородностью. При этом основным требованием, предъявляемым к методу сжатия, являются высокое быстродействие при эффективности кодирования, близком к арифметическому кодированию.

Одним из недостатков АК, которое, в отличие от более простого в реализации КХ, обеспечивает меньшую избыточность кода, являются большие вычислительные затраты, особенно при адаптивном моделировании [12]. Это стимулировало разработку различных быстрых вариантов АК, не использующих при кодировании операций умножения и деления [13, 14], а также быстрых алгоритмов для АК с адаптивным моделированием [15-18].

Вторым недостатком АК является его неэффективность при кодировании символов больших алфавитов, для которых номер символа может занимать четыре и больше байт. В этих ситуациях приходится использовать высокоуровневые методы сжатия, например, предсказание по частичному совпадению, которые, однако, не обеспечивают достаточно высокого быстродействия для кодирования больших потоков информации в реальном времени.

Сравнительно недавно оформилось новое направление в ускорении кодирования данных, связанное с разделением каждого символа на две части (префикс и суффикс), только одна из которых кодируется (например, с помощью АК), а вторая просто нумеруется. При этом кодировать приходится символы гораздо меньших по размеру алфавитов, чем исходный алфавит, за счет чего и происходит ускорение кодирования.

В рамках этого направления можно выделить два подхода. В [19] предлагаются K-flat коды, для которых длина всех префиксов одинакова. Префиксы символов просто нумеруются (эти номера сохраняются в сжатом файле) и в дальнейшем, например, могут использоваться для осуществления быстрого

поиска в сжатых данных. Суффиксы кодируются, например, с использованием АК. Экономия времени кодирования происходит в данном случае за счет отсутствия кодирования префиксов.

Вторым подходом является формирование супербукв из символов с близкими вероятностями [20]. При этом с помощью, например, АК, кодируются номера супербукв, а символы, входящие в каждую супербукву, просто нумеруются, за счет чего и происходит ускорение кодирования. Этот подход может с успехом использоваться для блочного кодирования [21], а также хорошо сочетается как со статичным, так и с адаптивным моделированием [12].

В нашей работе [22] предлагается новый метод рекурсивного группового кодирования (РГК), в основе которого положена та же идея, что и в [20]. Однако в РГК используется только нумерация суффиксов. Номера же супербукв не кодируются, а попарно объединяются и к новому тексту рекурсивно применяется этот же метод кодирования. Таким образом, в РГК не требуется использования АК или какого либо другого метода кодирования. РГК является полностью самостоятельным методом, который может использоваться в качестве альтернативы АК или КХ. В [22] показано, что РГК за счет рекурсии эффективно кодирует символы очень больших алфавитов (каждый символ занимает 64 байта), в худшем случае обладая эффективностью устранения энтропии, близкой к АК. При этом РГК обладает значительно более высоким быстродействием и в основном цикле кодирования содержит лишь операции побитового сдвига и логического "или".

Недостатком РГК является то, что на данный момент для него разработан лишь статический вариант кодирования, что ограничивает области практического применения этого метода лишь сжатием статистически однородных данных, в которых отсутствуют словарные повторения. По этой же причине РГК является неэффективным при кодировании небольших файлов, так как необходимость хранения данных по супербуквам (списков символов, входящих в каждую супербукву) нивелирует выигрыш в сжатии. Кроме того, остается непроработанным ряд вопросов, таких как получение строгой оценки быстродействия метода, оптимизация метода формирования супербукв, разработка метода сжатия данных по супербуквам.

В данной работе продолжается исследование РГК. В частности, проводится сравнительный анализ различных методов формирования супербукв, анализируется вклад данных по супербуквам в размер сжатого файла. Осуществляется сравнительный анализ РГК с АК и КХ для расширенного набора тестовых файлов.

В подразделе 1 данной работы описывается принцип РГК. Подраздел 2 посвящен разработке адаптивного варианта группирования символов в супербуквы. И, наконец, в подразделе 3 осуществляется сравнительный анализ РГК с АК и КХ.

1. Принцип рекурсивного группового кодирования

Разобьем все символы алфавита входного текста на группы, которые назовем супербуквами (пока не будем конкретизировать, как это делается). Тогда каждый символ текста можно будет представить в виде двух частей: номера супербуквы, в которую он входит (это будет префикс) и порядкового номера этого символа в списке символов, входящих в эту супербукву (это будет суффикс). Если в супербукву входит только один символ, то суффикс ему не нужен.

Возьмем для примера такой текст: "AACBADBCAABCFABCDGAEAACB". Символ A в нем встречается 9 раз, символы B и C - по 5 раз, символ D - 2 раза, а символы E, F и G - по одному разу.

Пусть каким-либо образом все символы разбиты на три супербуквы (группы) S_0, S_1 и S_2. Супербуква S_0 состоит из одного символа A, супербуква S_1 состоит из двух символов B и C (имеющих одинаковую встречаемость в тексте), а супербуква S_2 состоит из четырех символов D, E, F и G (имеющих близкую встречаемость в тексте). Тогда, если представить каждый символ в виде {X,Y}, где X - номер супербуквы (префикс), а Y - порядковый номер символа в супербукве (суффикс), то символ A можно заменить на $\{S_0\}$, символы B и C, соответственно, на $\{S_1, 0_2\}$ и $\{S_1, 1_2\}$, а символы D, E, F и G, соответственно, на $\{S_2, 00_2\}$, $\{S_2, 01_2\}$, $(S_2, 10_2)$, $\{S_2, 11_2\}$.

Разделим исходный текст на два массива данных. В первом массиве будут префиксы, а во втором - суффиксы.

Массив префиксов будет выглядеть так: "$S_0S_0S_1S_1S_0S_2S_1S_1S_0S_0S_1S_1S_2S_0S_1S_1S_2S_2S_0S_2S_0S_0S_1S_1$".

Сформированный массив суффиксов будет выглядеть так: "10000101100100110110" и занимать 20 бит памяти.

Объединим соседние префиксы следующим образом. Пару S_0S_0 будем заменять символом A, пару S_0S_1 - символом B, S_0S_2 - C, S_1S_0 - D, S_1S_1 - E, S_1S_2 - F, S_2S_0 - G, S_2S_1 - H, S_2S_2 - I.

После объединения массив префиксов будет выглядеть так: "AECEAEGEICAE".

Проанализируем полученный результат. Во-первых, суффиксы всех символов выделены в отдельный массив, в котором практически нет стати-

стической избыточности, и который занимает 20 бит памяти. Этот массив уже дальше не нужно сжимать. Во-вторых, если посмотреть на итоговый массив префиксов, то это текст, в два раза более короткий, чем исходный, и с приблизительно таким же размером алфавита.

Идея РГК заключается в том, чтобы применять описанную на данном примере процедуру рекурсивно к итоговому массиву префиксов до тех пор, пока его длина не станет незначительной по сравнению с длиной исходного текста. При этом, чтобы процесс можно было обратить вспять (декодировать сжатые данные) необходимо, кроме массивов суффиксов, сохранять еще и информацию о составе супербукв. При длине исходного текста N_0 для кодирования текста потребуется не более $\log_2(N_0)$ шагов.

Чтобы размер алфавита после попарного объединения префиксов не увеличивался по отношению к исходному размеру алфавита, должно выполняться условие:

$$K \geq K_S^2, \qquad (1)$$

где K - размер исходного алфавита, K_S - число супербукв.

На рис. 1 приведена структурная схема РГК.

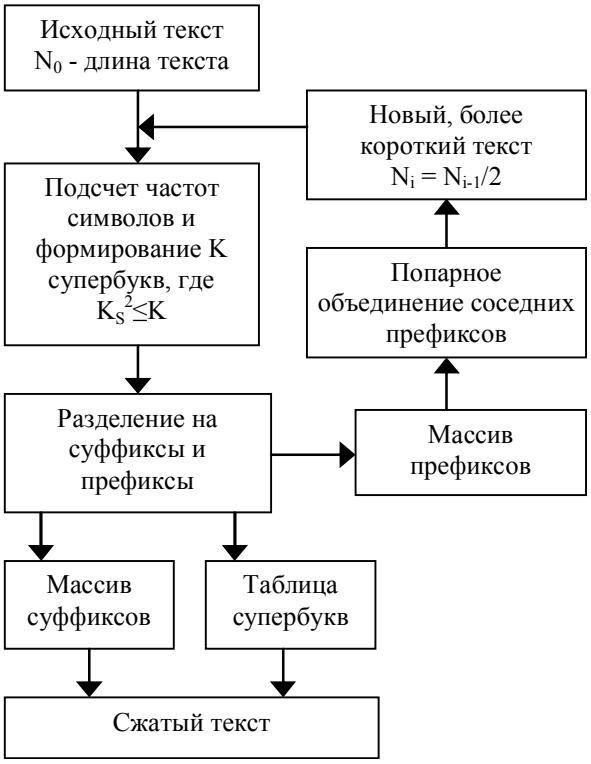

Рис. 1. Структурная схема РГК

Обратим внимание на тот факт, что для РГК при статическом моделировании не требуется запоминать таблицу частот, как для АК. Для однозначного декодирования достаточно знать (запомнить в выходном потоке данных) только количество и состав супербукв.

При разработке практической реализации РГК необходимо выбрать размер исходного алфавита K и метод объединения символов в супербуквы.

Алфавит с $K=2^{16}$ подходит только для адаптивного моделирования, так как перечень входящих в символы супербукв будет слишком большим для сохранения в закодированных данных.

Для алфавита $K=2^8$ реализация РКГ со статичным моделированием возможна. При этом из-за объединения символов алфавита в супербукву увеличивается длина кода для этих символов. Это увеличение в относительных единицах можно выразить как

$$\Delta = -p_S(-\log_2 p_S + \log_2 M) / \sum_{i=1}^{M}(p_i \log_2 p_i), \qquad (2)$$

где M - число символов, объединенных в супербукву, p_S - суммарная вероятность этих символов $\sum_{i=1}^{M} p_i$.

Символы могут быть объединены в супербукву в том случае, если Δ не превышает некоторого допустимого порога T_Δ. Полный алгоритм объединения символов в супербуквы [20] будет выглядеть так:

1. Задается T_Δ, например, $T_\Delta = 0{,}01$.
2. Символы алфавита сортируются по возрастанию их вероятностей.
3. Перебираются в порядке убывания все возможные M со степенями 2 (для алфавита с $K = 2^8$ эти M равны 256, 128, 64, 32, 16, 8, 4, 2, 1). Для каждого M проверяется условие $\Delta \leq T_\Delta$ (берутся первые M символов из отсортированной последовательности). Как только условие выполняется, символы объединяются в супербукву и исключаются из дальнейшего формирования супербукв.
4. Если еще остались символы, не вошедшие в супербуквы, то для них повторяется шаг 3 алгоритма.

Время, затрачиваемое на формирование супербукв по описанному алгоритму, не зависит от N_0 (от N_0 зависит только время подсчета частот символов) и в большинстве практических ситуаций не вносит существенного вклада в общее время кодирования. Авторам не встречалась практическая ситуация, когда при $T_\Delta = 0{,}02$ число супербукв K_S было бы большим, чем 16. Однако формального доказательства, что K_S для $K=2^8$ не превысит 16, пока не известно. Поэтому, если в результате работы описанного алгоритма, это произойдет, то следует увели-

чить T_Δ и повторно осуществить формирование супербукв.

При кодировании текста, после того, как сформирован набор супербукв, необходимо разделить исходный текст на массивы префиксов и суффиксов. Благодаря тому, что используется статическое моделирование для каждого символа однозначно известны номер супербуквы, номер суффикса и длина суффикса в этой супербукве. При этом разрядность номеров супербукв для $K=2^8$ никогда не будет больше 4 бит, а разрядность номеров суффиксов никогда не будет больше 8 бит. Поэтому разумным представляется предварительное создание небольших таблиц, в которых каждому символу соответствовали бы номер супербуквы, а также номер и длина суффикса. Тогда при кодировании символа все действия сведутся к простой выборке из таблицы двух чисел и помещению их в массивы префиксов и суффиксов.

Далее необходимо попарно объединить все префиксы в массиве префиксов. Эта операция сводится к простому сдвигу первого из объединяемых префиксов на 4 бита и логическому «или» со вторым префиксом $Pr = (P1 \text{ shl } 4) \vee P2$, где P1 - первый префикс, P2 - второй префикс, а Pr - результат их объединения.

Как видим, все действия в основном цикле кодирования сводятся к выборке из таблицы, а также операции сдвига и логического "или". Алгоритм является рекурсивным, но благодаря тому, что на каждом шагу число кодируемых символов уменьшается в 2 раза, общее число кодируемых символов не превышает $2N_0$. На каждый кодируемый символ исходного текста в среднем придется не более двух выборок из таблицы, а также не больше, чем по одной операции сдвига и логического "или". На каждый символ текста также придется по два сложения для вычисления вероятностей символов. Таким образом, время W_e, затрачиваемое для РГК на кодирование текста, может быть оценено с помощью следующего выражения:

$$W_e = N_0(2W_a + 2W_t + W_s + W_o), \qquad (3)$$

где W_a - время выполнения операции сложения (увеличения на единицу), W_t - время выполнения операции извлечения из таблицы, W_s - время выполнения операции логического сдвига, W_o - время выполнения операции логического «или».

При декодировании все повторяется в обратном порядке. Вначале объединенные пары префиксов разделяются: $P1 = Pr \text{ shr } 4$ и $P2 = Pr \wedge 240$. Затем для каждой супербуквы по запомненной информации о ее составе строится табличка соответствия номеров суффиксов символам исходного текста. Декодирование может осуществляться несколько быстрее, чем кодирование, так как при декодировании не нужно вычислять вероятности символов. Приведем выражение для времени W_d, затрачиваемом для РГК на кодирование текста:

$$W_d = N_0(2W_t + W_s + W_o). \qquad (4)$$

В выражении (4) по сравнению с (3) отсутствует слагаемое $2W_a$, благодаря чему декодирование может осуществляться примерно на 30% быстрее, чем кодирование (в зависимости от особенностей выполнения команд в том или ином процессоре).

2. Выбор порога относительного увеличения длины кода

Описанный в предыдущем подразделе алгоритм формирования супербукв требует предварительного выбора порога T_Δ. Выбор $T_\Delta=0{,}01$ может обеспечить меньшую избыточность кода, однако на практике встречаются ситуации, для которых при таком значении T_Δ не выполняется условие (1). Выбор $T_\Delta=0{,}02$ практически всегда обеспечивает выполнение условия (1) однако приводит к большей избыточности кода.

Отметим, что при декодировании не важно, каким именно способом происходило разбиение символов на группы (на супербуквы), так как состав супербукв сохраняется в сжатых данных. Поэтому использование того или иного алгоритма выбора T_Δ при кодировании текста не приведет к изменению процесса декодирования, однако может влиять на степень избыточности кода.

Оценим величину увеличения длины кода в результате РГК в зависимости от T_Δ. Будем вычислять верхнюю границу E_{max} этой величины как:

$$E_{max} = E_{tre} + E_{tab}, \qquad (5)$$

где E_{tre} - верхняя граница увеличения длины кода вследствие объединения символов в супербуквы, E_{tab} - верхняя граница увеличения длины кода вследствие необходимости сохранения в сжатых данных состава супербукв.

Отметим, что в выражении (2) возрастание длины кода происходит вследствие нумерации входящих в супербукву символов (слагаемое $p_S\log_2 M$) из-за возможной неодинаковости частот этих символов. Однако слагаемое $p_S\log_2 p_S$, определяющее длину кода для супербукв, также может возрасти при последующих итерациях из-за рекурсивного

применения РГК к тексту, полученному объединением в пары этих супербукв. Поэтому реальное увеличение длины кода будет больше, чем в T_Δ раз, но меньше, чем в $2T_\Delta$, и выражение для E_{tre} можно записать как

$$E_{tre} = (2T_\Delta - 1)N_o \sum_{j=1}^{K} p_j \log_2 p_j , \qquad (6)$$

где p_j - вероятность встречаемости в исходном тексте символа алфавита с индексом j.

E_{tab} можно записать как

$$E_{tab} = 2180 \log_2 N_0, \qquad (7)$$

где $\log_2 N_0$ - верхняя граница количества итераций. Множитель 2180 получен из выражения 4 + 8(16 +256), где 8 - количество бит в байте, 4 - число бит для кодирования количества супербукв (максимум - 16), 16 - объем памяти для хранения значений количества символов в каждой супербукве, 256 - объем памяти для хранения номера для каждого символа алфавита.

При небольшом значении N_0 в сумме (5) может превалировать слагаемое E_{tab}, а при большом значении - E_{tre}.

Далее, чтобы оценить возможный эффект от уменьшения Etre, рассмотрим две стратегии выбора T_Δ: с фиксированным значением (1,01) и с адаптивным выбором.

Адаптивный выбор T_Δ будет осуществляться следующим образом. На каждой итерации T_Δ инициализируется значением 1,001. Далее, если при попытке формирования супербукв условие (1) не выполняется, то T_Δ увеличивается на 0,001, и попытка сформировать супербуквы повторяется снова.

3. Анализ эффективности рекурсивного группового кодирования

Для проведения сбалансированного анализа эффективности рассматриваемого метода сжатия, был сформирован тестовый набор данных. Во-первых, в него были включены все файлы из calgary corpus test files [23] и все файлы из canterbury corpus test files [24].

Далее к тестовому набору были добавлены три файла, представляющие собой округленные значения случайных чисел с нормальным законом распределения, математическим ожиданием 128 и дисперсией σ^2 соответственно 25, 100 и 400 (файлы odn25, odn100 и odn400).

И, наконец, в тестовый набор были включены квантованные коэффициенты дискретного косинусного преобразования блоков 8x8 пикселей изображений. Этот класс тестовых данных соответствует задаче сжатия изображений и алфавиту с $K=2^{512}$ (длина каждого символа равна 64 байтам). При этом использовались два стандартных тестовых изображения (Barbara и Lena) и два шага квантования (использовалось равномерное квантование): 10 и 50. Таким образом, в тестовый набор было добавлено четыре изображения: barb10, barb50, len10 и len50.

В табл. 1-4 приведены результаты верификации РГК для этого тестового набора файлов (количество бит в сжатых данных на символ исходного текста). Для сравнения приведено значение, предельно достижимое в соответствии с теоремой Шеннона для однобайтного алфавита.

Таблица 1

Результаты сжатия РГК для квантованных коэффициентов ДКП блоков изображений

Файл	Шеннон	РГК, T_Δ =1,01	РГК, адапт. T_Δ	Число итер.	Доля E_{tab}, %
barb10	1,983	1,677	1,677	11	3,4
barb50	0,734	0,556	0,555	11	7,3
len10	1,546	1,312	7,291	11	3,5
len50	0,504	0,336	0,343	11	12,2

Таблица 2

Результаты сжатия РГК для Гауссова шума

Файл	Шеннон	РГК, T_Δ =1,01	РГК, адапт. T_Δ	Число итер.	Доля E_{tab}, %
odn25	4,371	4,447	4,439	11	1,4
odn100	5,370	5,448	5,445	11	1,1
odn400	6,368	6,447	6,444	11	0,9

Таблица 3

Результаты сжатия РГК для Calgary corpus test files

Файл	Шеннон	РГК, T_Δ =1,01	РГК, адапт. T_Δ	Число итер.	Доля E_{tab}, %
bib	5,229	5,195	5,128	10	2,2
book1	4,520	4,469	4,469	12	0,6
book2	4,790	4,762	4,790	12	0,6
geo	5,634	4,706	4,571	10	3,0
news	5,195	5,195	5,128	11	0,8
obj1	5,926	5,882	5,970	7	7,9
obj2	6,250	6,015	5,797	11	1,2
paper1	4,969	5,063	5,063	9	3,8
paper2	4,598	4,651	4,651	9	3,4
pic	1,210	0,964	0,955	12	3,9
progc	5,195	5,333	5,333	8	3,9
progl	4,762	4,819	4,790	9	3,3
progp	4,878	4,878	4,908	8	4,1
trans	5,517	5,556	5,479	9	2,2

Таблица 4

Результаты сжатия для Canterbury corpus test files

Файл	Шеннон	РГК, T_Δ =1,01	РГК, адапт. T_Δ	Число итер.	Доля E_{tab}, %
alice29.txt	4,571	4,598	4,571	10	1,9
asyoulik.txt	4,819	4,819	4,790	10	1,9
cp.html	5,229	5,442	5,517	7	6,4
fields.c	5,000	5,333	5,442	6	11,4
grammar.lsp	4,624	5,479	5,755	5	26,7
kennedy.xls	3,571	3,150	3,213	13	0,7
lcet10.txt	4,678	4,598	4,571	12	1,0
plrabn12.txt	4,520	4,469	4,469	12	0,9
ptt5	1,210	0,965	0,955	12	3,9
sum	5,333	5,229	5,333	8	6,3
xargs.1	4,908	5,674	5,926	5	21,3

По данным, приведенным в таблицах, можно сделать несколько выводов.

Рекурсивность РГК в большинстве случаев, за исключением сжатия Гауссова шума, позволяет достичь большей степени сжатия, чем арифметическое кодирование для 256-символьного алфавита. В ряде случаев выигрыш в степени сжатия превышает 30%.

Так как выражение (6) показывает, что увеличение кода вследствие введения допустимого порога T_Δ не может превышать $2\,T_\Delta$ раза, то и выигрыш от использования адаптивного выбора T_Δ не может превышать этого числа. Это подтверждают и данные таблиц. РГК с адаптивным выбором T_Δ в большинстве случаев обеспечивает более высокую степень сжатия, но не более чем на 1%.

Как это предполагалось в подразделе 2, E_{tab} превалирует для небольших изображений. Например, для obj1 (21504 байта) E_{tab} равняется 7,9%, для grammar.lsp (3721 байт) - 26,7%. Из-за этого (необходимости сохранять таблицы в) РГК для такой реализации (статическое кодирование) может на небольших файлах уступать по степени сжатия арифметическому кодированию. Поэтому является актуальной разработка методов кодирования таблиц символов супербукв и вариантов РГК с динамическим кодированием.

Заключение

В работе рассмотрено РГК, которое за счет своей рекурсивности способно эффективно кодировать символы больших алфавитов. Предложен адаптивный вариант выбора порога T_Δ, получены оценки скорости кодирования и декодирования, верхняя оценка увеличения длины кода вследствие РГК. Проанализирована эффективность РГК для четырех наборов данных, намечены пути дальнейших исследований.

Литература

1. Naito, S. Development of real-time encoder for 8K ultra-high definition television [Text] / S. Naito, S. Matsumoto // Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). – December 6-8, 2010. – P.1-4.

2. ADCTC: A new high quality DCT based coder for lossy image compression [Electronic resource] / N. Ponomarenko, V. Lukin, K. Egiazarian, J. Astola. – 80 Min / 700 MB. CD ROM Proceedings of LNLA. - Switzerland, August - 2008, 6 p. – 1 electronic optical disc (CD-ROM).

3. Xiong, Z. Wavelet image coding using trellis coded space-frequency quantization [Text] / Z. Xiong, X. Wu // in IEEE Signal Processing Letters. – 1999. – Vol. 6. - issue 7. – P. 158-161.

4. The encoding efficiency of H.265/HEVC and H.264/AV [Text] / H. Koumaras, H. Kourti, M. Martakos, D. Benchmarking // Proceedings of Future Network & Mobile Summit (FutureNetw), 4-6 July 2012. - P. 1-7.

5. Bazhyna, A. V. Efficient bit-planes based method for compression of 3D-DCT coefficients [Text] / A. V. Bazhyna, K. O. Egiazarian, N. N. Ponomarenko // Proceedings of Picture Coding Symposium, Lisboa, Portugal, 7-9 November, 2007. – P. 4.

6. Nowak, N. Methods of sound data compression [Text] / N. Nowak, W. Zabierowski // Comparison of different standards, Proceedings of International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 23-25 February 2011. – P. 431-434.

7. Методы сжатия данных [Текст] : учебно-справочное издание / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : Диалог-Мифи, 2002. – 383 с.

8. Cleary, J. Data compression using adaptive coding and partial string matching [Text] / J. Cleary, I. Witten // IEEE Transactions on Communications. – April, 1984. – Vol. COM-32. – P. 396-402.

9. Rissanen, J. Generalized kraft inequality and arithmetic coding [Text] / J. Rissanen // IBM J. Res. Develop. – May, 1976. – Vol. 20. – P. 198-203.

10. Rissanen, J. Arithmetic coding [Text] / J. Rissanen, G. Langdon // IBM J. Res. Develop. – March, 1979. – Vol. 23, N 2. – P. 149-162.

11. Huffman, D. A. A method for the construction of minimum-redundancy codes [Text] / D. A. Huffman // Proc. Inst. Radio Eng. – September, 1952. – Vol. 40, N 9. – P. 1098-1101.

12. Bell, T. Modeling for text compression [Text] / T. Bell, I. H. Witten, J. G. Cleary // ACM Computing Surveys. – December, 1989. – Vol. 21(4). – P. 557-591.

13. Rissanen, J. A multiplication-free multialphabet arithmetic code [Text] / J. Rissanen, K. M. Mohiuddin // IEEE Transactions on Communications. – February, 1989. – Vol. 37, № 2. – P. 93 – 98.

14. Chevion, D. High efficiency, multiplication free approximation of arithmetic coding [Text] / D. Chevion, E. D. Karnin, E. Walach // Data Compression Confer-

ence. – April 8-11, 1991. – P. 43-52.

15. Jones, D. W. Application of splay trees to data compression [Text] / D. W. Jones // Communications of the ACM. – 1988. – № 31(8). – P. 996-1007.

16. Ryabko, B. Y. A fast sequential code [Text] / B. Y. Ryabko // Dokl. Akad. Nauk SSSR 306:3(1989). – P. 548-552 (Russian); translation in: Soviet Math. Dokl., 39:3(1989). – P. 533-537.

17. Schindler, M. A Fast renormalisation for arithmetic coding [Text] / M. Schindler // Proceedings of Data Compression Conference. – March 30 - April 1, 1998. – P. 572.

18. Ryabko, B. Fast adaptive arithmetic code for large alphabet sources with asymmetrical distributions [Text] / B. Ryabko, J. Rissanen // IEEE Communications Letters. – January, 2003. – Vol. 7, № 1. – P. 33–35.

19. Liddell, M. Hybrid Prefix Code for Practical Use [Text] / M. Liddell, A. Moffat // Procedeengs of Data Compression Conference. – 2003. – P. 392-401.

20. Ryabko, B. Fast codes for large alphabets [Text] / B. Ryabko, J. Astola, K. Egiazarian // Communications in information and systems. – October, 2003. – Vol. 3, N 2. – P. 65-78.

21. The fast algorithm for the block codes and its application to image compression [Text] / B. Ryabko, G. Mrchokov, K. Egiazarian, J. Astola // Proceedings of International Conference on Image Processing. – September 14-17, 2003. – Vol. 2. – P. 205-207.

22. Fast recursive coding based on grouping of Symbols [Text] / N. Ponomarenko, V. Lukin, K. Egiazarian, J. Astola // Telecommunications and Radio Engineering. – 2009. – Vol. 68, N 20. – P. 1857-1863.

23. Archive Comparison Test. Calgary corpus test files [Electronic resource]. – Access mode: http://compression.ca/act/act-files.html. – 25.05.2014.

24. Canterbury corpus test files [Electronic resource]. – Access mode: http://corpus.canterbury.ac.nz. – 25.05.2014.

Поступила в редакцию 20.05.2014, рассмотрена на редколлегии 11.06.2014

Рецензент: д-р техн. наук, проф., проф. каф. «Проектирования радиоэлектронных систем летательных аппаратов» Э. Н. Хомяков, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Харьков.

МЕТОД ЕНТРОПІЙНОГО РЕКУРСИВНОГО ГРУПОВОГО КОДУВАННЯ

М. М. Пономаренко, Н. В. Кожемякіна, В. В. Лукін

Розглянуто задачу кодування інформації з метою усунення її статистичної надмірності. Найбільш відомими з методів, що відносяться до цієї області, є арифметичне кодування і кодування Хафмана. Недоліком цих і більшості інших методів є недостатня ефективність при кодуванні символів надвеликих алфавітів. У даній роботі розглядається метод ентропійного рекурсивного групового кодування, який при високій швидкодії і близькій до арифметичного кодування ефективності позбавлений цього недоліку. Показано, що для великих алфавітів рекурсивне групове кодування забезпечує виграш в ступені стиснення по відношенню до арифметичного кодування до 40%.

Ключові слова: стиснення даних, стиснення зображень, арифметичне кодування, кодування Хафмана, ентропійне кодування.

METHOD OF ENTROPY RECURSIVE GROUP CODING

N. N. Ponomarenko, N. V. Kozhemiakina, V. V. Lukin

The problem of encoding information in order to reduce its statistical redundancy is considered. The best known of the methods related to this area are arithmetic coding and Huffman coding. The disadvantage of these and most other methods is the lack of efficiency by encoding the characters of large alphabets. In this paper the method of entropy recursive group coding, which at high speed and close to the arithmetic coding efficiency does not have this drawback is described. It is shown that for large alphabets the method provides compression ratios less than for arithmetic coding up to 40%.

Key words*:* data compression, image compression, arithmetical coding, Huffman coding, entropy coding.

Пономаренко Николай Николаевич - д-р техн. наук, доцент, доцент каф. приема, передачи и обработки сигналов, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: nikolay@ponomarenko.info.

Кожемякина Надежда Владимировна – аспирант каф. приема, передачи и обработки сигналов, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: nadejda_kozickaya@mail.ru.

Лукин Владимир Васильевич – д-р техн. наук, профессор, профессор каф. приема, передачи и обработки сигналов, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: lukin@ai.kharkov.com.