УДК 681.3.053

Балашов К.Ю. **Сжатие информации: анализ методов и подходов**. – Минск, 2000. – 42 с (Препринт / Ин-т техн. Кибернетики НАН Беларуси; № 6)

В настоящее время сжатие информации выделилось как самостоятельная научная дисциплина. В то же время знания о сжатии различных типов данных разрознены и не систематизированы. Существуют отдельно алгоритмы сжатия графики, алгоритмы сжатия текста и т.д. В настоящей работе приводится анализ принципов, на которых основан любой алгоритм сжатия. Предпринимается попытка систематизировать алгоритмы сжатия, их составные части, а также подходы к сжатию информации. Предлагается формализованный метод для синтеза новых алгоритмов сжатия информации, позволяющий достигать максимальной степени сжатия**.**

Ил. 6, табл. 7, библ. 37 назв.

Рецензент
кандидат технических наук А.Я. Кулешов

ВВЕДЕНИЕ

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. Кроме того, им были проведены опыты по эмпирической оценке избыточности английского текста. Он предлагал людям угадывать следующую букву и оценивал вероятность правильного угадывания. 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]. Несмотря на то, что результаты исследований Шеннона были по-настоящему востребованы лишь десятилетия спустя, трудно переоценить их значение.

Первые алгоритмы сжатия были примитивными в связи с тем, что была примитивной вычислительная техника. С развитием мощностей компьютеров стали возможными все более мощные алгоритмы. Настоящим прорывом было изобретение Лемпелем и Зивом в 1977 г. словарных алгоритмов. До этого момента сжатие сводилось к примитивному кодированию символов. Словарные алгоритмы позволяли кодировать повторяющиеся строки символов, что позволило резко повысить степень сжатия. Важную роль сыграло изобретение примерно в это же время арифметического кодирования, позволившего воплотить в жизнь идею Шеннона об оптимальном кодировании. Следующим прорывом было изобретение в 1984 г. алгоритма PPM. Следует отметить, что это изобретение долго оставалось незамеченным. Дело в том, что алгоритм сложен и требует больших ресурсов, в первую очередь больших объемов памяти, что было серьезной проблемой в то время. Изобретенный в том же 1984 г. алгоритм LZW был чрезвычайно популярен благодаря своей простоте, хорошей рекламе и нетребо–вательности к ресурсам, несмотря на относительно низкую степень сжатия. На сегодняшний день алгоритм PPM является наилучшим алгоритмом для сжатия текстовой информации, а LZW давно уже не встраивается в новые приложения (однако широко используется в старых).

Будущее алгоритмов сжатия тесно связано с будущим компьютерных технологий. Современные алгоритмы уже вплотную приблизились к Шенноновской оценке 1.3 бита на символ, но ученые не видят причин, по которым компьютер не может предсказывать лучше, чем человек. Для достижения высоких степеней сжатия приходится использовать более сложные алгоритмы. Однако существовавшее одно время предубеждение, что сложные алгоритмы с более высокой

степенью сжатия всегда более медленны, несостоятельно. Так, существуют крайне быстрые реализации алгоритмов PPM для текстовой информации и SPIHT для графики, имеющие очень высокую степень сжатия.

Таким образом, будущее за новыми алгоритмами с высокими требованиями к ресурсам и все более и более высокой степенью сжатия.

Устаревают не только алгоритмы, но и типы информации, на которые они ориентированы. Так, на смену графике с малым числом цветов и неформатированному тексту пришли высококачественные изображения и электронные документы в различных форматах. Известные алгоритмы не всегда эффективны на новых типах данных. Это делает крайне актуальной проблему синтеза новых алгоритмов.

1. ОСНОВЫ ТЕОРИИ СЖАТИЯ ИНФОРМАЦИИ

1.1. Актуальность проблемы сжатия информации

Количество нужной человеку информации неуклонно растет. Объемы устройств для хранения данных и пропускная способность линий связи также растут. Однако количество информации растет быстрее. У этой проблемы есть три решения. Первое – ограничение количества информации. К сожалению, оно не всегда приемлемо. Например, для изображений это означает уменьшение разрешения, что приведет к потере мелких деталей и может сделать изображения вообще бесполезными (например, для медицинских или космических изображений). Второе – увеличение объема носителей информации и пропускной способности каналов связи. Это решение связано с материальными затратами, причем иногда весьма значительными. Третье решение – использование сжатия информации. Это решение позволяет в несколько раз сократить требования к объему устройств хранения данных и пропускной способности каналов связи без дополнительных издержек (за исключением издержек на реализацию алгоритмов сжатия). Условиями его применимости является избыточность информации и возможность установки специального программного обеспечения либо аппаратуры как вблизи источника, так и вблизи приемника информации. Как правило, оба эти условия удовлетворяются.

Именно благодаря необходимости использования сжатия информации методы сжатия достаточно широко распространены. Однако существуют две серьезные проблемы. Во-первых, широко используемые методы сжатия, как правило, устарели и не обеспечивают достаточной степени сжатия. В то же время они встроены в большое количество программных продуктов и библиотек и поэтому будут использоваться еще достаточно долгое время. Второй проблемой является частое применение методов сжатия, не соответствующих характеру данных. Например, для сжатия графики широко используется алгоритм LZW, ориентированный на сжатие одномерной информации, например текста. Решение этих проблем позволяет резко повысить эффективность применения алгоритмов сжатия.

Таким образом, разработка и внедрение новых алгоритмов сжатия, а также правильное использование существующих позволит значительно сократить издержки на аппаратное обеспечение вычислительных систем.

1.2. Энтропия и количество информации

Под энтропией в теории информации понимают меру неопределенности (например, меру неопределенности состояния некоторого объекта). Для того чтобы снять эту неопределенность, необходимо сообщить некоторое количество информации. При этом энтропия численно равна минимальному количеству информации, которую необходимо сообщить для полного снятия неопределенности. Энтропия также может быть использована в качестве оценки наилучшей возможной степени сжатия для некоторого потока событий.

Здесь и далее понятие события используется как наиболее общее понятие сущности, которую необходимо сжать. Так, при сжатии потока символов под событием может пониматься появление во входном потоке того или иного символа, при сжатии графики – пикселя того или иного цвета и т.д.

1.3. Комбинаторная, вероятностная и алгоритмическая оценка количества информации

Различные способы оценки количества информации представлены в [2]. Наиболее простым является комбинаторный подход. Согласно этому подходу, если переменная x может принадлежать к множеству из N элементов, то энтропия переменного

$$H(x) = \log_2 N.$$

Таким образом, для передачи состояния объекта достаточно $I=log_2N$ бит информации. Заметим, что количество информации может быть дробным. Разумеется, дробное количество информации невозможно сохранить на носителе или передать по каналам связи. В то же время, если необходимо передать либо сохранить большое количество блоков информации дробной длины, их всегда можно сгруппировать таким образом, чтобы полностью исключить потери (например, посредством арифметического кодирования).

Основным недостатком комбинаторного подхода является его ориентированность на системы с равновероятными состояниями. В реальном мире события, как правило, не равновероятны. Вероятностный подход к оценке количества информации, учитывающий этот фактор, является наиболее широко используемым на сегодняшний день. Пусть переменная x может принимать N значений x_i с вероятностью $p(x_i)$. Тогда энтропия

$$Hw(x) = -\sum_{i=1}^{N} p(x_i)\log_2(p(x_i))$$

Обозначим через $p(y|x)$ условную вероятность того, что наступит событие y если событие x уже наступило. В таком случае условная энтропия для переменной Y, которая может принимать M значений y_i с условными вероятностями $p(y_i|x)$ будет

$$Hw(y|x) = -\sum_{i=1}^{M} p(y_i|x)\log_2(p(y_i|x))$$

Приведенные формулы показывают, что вне зависимости от того, как были получены вероятности наступления следующих событий, для кодирования события с вероятностью p достаточно $-\log_2 p$ бит (в полном соответствии с теоремой Шеннона об оптимальном кодировании [3]).

Алгоритмический подход применим в тех случаях, когда данные обладают некоторыми закономерностями. Согласно этому подходу, если данные можно описать посредством некоторых формул либо порождающих алгоритмов, энтропия данных будет равна минимальному количеству информации, необходимой для передачи этих формул либо алгоритмов от источника информации к приемнику. Алгоритмический подход используется самостоятельно или совместно с вероятностным, например в некоторых алгоритмах сжатия графической информации.

1.4. Моделирование и кодирование

Энтропия набора данных, а значит и максимально возможная степень сжатия, зависит от модели. Чем адекватнее модель (чем качественнее мы можем предсказать распределение вероятности значений следующего элемента), тем ниже энтропия и тем лучше максимально достижимая степень сжатия. Таким образом, сжатие данных разбивается на две самостоятельные задачи - моделирование и кодирование [4, 5].

Моделирование обеспечивает предсказание вероятности наступления возможных событий, кодирование обеспечивает представление события в виде $-\log_2 p$ бит, где p - предсказанная вероятность наступления события. Задача моделирования, как правило, более сложная. Это обусловлено высокой сложностью современных моделей данных. В то же время кодирование не является серьезной проблемой. Существует большое количество стандартных кодеров, различающихся по степени сжатия и быстродействию. Как правило, в системах сжатия используемый кодер при необходимости может быть легко заменен другим.

2. МОДЕЛИРОВАНИЕ

2.1. Виды избыточности и подходы к их устранению. Свойства избыточности

Приведем основные типы избыточности, перечисленные Т. Вел-чем в работе [6].

а) избыточность распределения событий (разные события имеют разные вероятности);

б) избыточность повторения событий (несколько одинаковых событий могут следовать друг за другом);

в) избыточность цепочек событий (цепочки событий могут повторяться);

г) позиционная избыточность – повышение вероятности появления определенных событий в некоторых позициях потока событий (например, в записях базы данных).

Приведенная классификация, сделанная в 1984 г., практически не устарела, однако нуждается в некоторых дополнениях.

Дополнение 1. Классификация ориентирована в первую очередь на символьные источники информации. Нетекстовым источникам данных характерны свои виды избыточности. Например, графическим данным характерна пространственная избыточность, характеризующаяся высокой вероятностью близости не только значений соседних пикселей, но и их пространственных производных. Будем называть пространственную избыточность избыточностью типа «д».

Дополнение 2. Избыточность цепочек событий может быть представлена альтернативным способом – как избыточность распределения событий после наступления некоторого количества непосредственно предшествующих событий. Примером является приведенная выше оценка вероятности появления символа "o" в контексте "to be or not t". Будем называть такую избыточность избыточночстью типа «в2». Практические реализации, основанные на таком подходе, обеспечивают заметно более высокую степень сжатия, чем просто устраняющие цепочечную избыточность.

Следует также отметить, что при устранении одних видов избыточности другие виды избыточности, присущие тому же потоку событий, могут исчезать (например, после устранения в потоке символов избыточности типа «а» устранить другие типы избыточности не представляется возможным) или сохраняться (например, при устранении избыточности типа «б» может сохраняться избыточность типа «а», хотя ее характеристики несколько меняются). Кроме того, в

выходном потоке при устранении определенных видов избыточности могут появляться новые виды избыточности. Так, при устранении избыточности типа «в» в выходной поток помещаются длины цепочек, обладающие избыточностью типа «а».

2.2. Типы моделей

Для того чтобы информация занимала меньше места, ее необходимо закодировать, т.е. представить в виде более компактных кодов. Схемы кодирования существуют только для случая, когда необходимо закодировать одно из возможного ряда событий и их вероятности известны. Иными словами, если необходимо устранить только избыточность типа «а», то достаточно просто оптимально закодировать события. Такой подход использовался в первых алгоритмах сжатия. Поскольку он не позволял устранить другие виды избыточности, степень сжатия была низкой. В современных алгоритмах сжатия для преобразования различных видов избыточности к вероятностям событий (или избыточности типа «а») используют модели входных данных. Для некоторых типов избыточности перед моделированием используют специальные преобразования, трансформирующие эти виды избыточности в другие.

Для каждого типа избыточности существуют свои модели. Для избыточности типа «а» используют статистическую модель нулевого порядка. Эта модель содержит вероятности наступления следующих событий вне зависимости от непосредственно предшествующих.

Для избыточности типа «б» используют модели RLE – Run Length Encoding [7]. Эти модели содержат вероятности появления цепочек одинаковых событий различной длины.

Для избыточности типа «в» используют различные словарные модели [4-10].

Для избыточности типа «в2» используют статистические модели высших порядков либо выполняют специальные группирующие преобразования [4, 5, 11-19].

Избыточность типа «г» в первоначально описанном виде, как правило, не встречается либо ее приводят к другим типам избыточности. Следует, однако, отметить, что избыточность, возникающую в результате спектровыделяющих преобразований и заключающуюся в зависимости распределения вероятностей от позиции в преобразованном изображении, также можно классифицировать как избыточность типа «г».

Для избыточности типа «д» либо используют модели на основе

пространственных предикторов и дополнительные модели ошибок, либо выполняют спектровыделяющие преобразования, которые не устраняют избыточности из данных, но группируют информацию таким образом, чтобы избыточность можно было легко устранить.

2.3. Словарные модели

Словарные алгоритмы предназначены для устранения избыточности цепочек событий. 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. Идея словарных алгоритмов была предложена Я. Зивом и А. Лемпелем в 1977 г. [8]. Идея была воплощена в алгоритме LZ77, ставшем основой целого семейства алгоритмов. Альтернативный подход, предложенный ими же в 1978 г., породил семейство алгоритмов LZ78.

Словарные алгоритмы обладают не самым высоким коэффициентом сжатия, но допускают очень быструю реализацию, особенно для распаковщика.

2.3.1. Алгоритмы LZ77, LZSS, LZH

Под словарем в алгоритмах семейства LZ77 понимается некоторое количество предыдущих событий. В качестве ссылок на элементы словаря используются расстояние до встретившейся ранее цепочки событий и длина этой цепочки [4-8].

Естественно, что не для всех цепочек событий входного потока можно найти соответствующие цепочки в словаре. Чтобы обеспечить различие между цепочками событий и литеральными событиями (событиями, не вошедшими в цепочки из словаря), алгоритм LZ77 просто чередует ссылки на элементы словаря с литеральными событиями.

В алгоритме LZSS ссылки на элементы словаря и литеральные события различаются флажком-битом.

Алгоритм LZH аналогичен LZSS, но использует для указателей кодирование Хаффмана.

В настоящее время алгоритмы семейства LZ77 широко используются в программах универсальных архиваторов. Это, как правило, модификации LZSS с использованием кодов Хаффмана (LHARC), Шеннона-Фоно (PKZIP) или арифметического кодирования (HA, метод 1).

2.3.2. Алгоритмы LZ78, LZW, LZC, LZFG, RFGD

В алгоритмах семейства LZ78 цепочки в словаре увеличиваются на один символ каждый раз, когда цепочка из словаря встречается во входном потоке. В алгоритме LZ78 так же, как и в LZ77, литеральные события и элементы словаря чередовались.

В 1984 г. Т. Велч предложил алгоритм LZW, в котором все литеральные события являются элементами словаря [6]. Таким образом, в выходной поток помещаются только ссылки на элементы словаря, что упрощает алгоритм. Велч показал, что алгоритм может быть легко реализован аппаратно. Программная реализация, как правило, выполняется на основе вектора двоичных деревьев. Корнями деревьев служат элементы входного алфавита.

LZC – модификация алгоритма LZW, направленная на повышение коэффициента сжатия. Во-первых, в нем используется переменная длина кодов. Например, при сжатии изображений с 4 битами на пиксель использование 5-битных кодов при пустом словаре дает значительную экономию по сравнению с 12-битными. По мере заполнения словаря длина кодов увеличивается. Во-вторых, отслеживается степень сжатия алгоритма. Если она падает ниже определенного предела, словарь полностью очищается. Этим достигается быстрая адаптация алгоритма к резкой перемене характера данных.

LZC используется в утилите COMPRESS операционной системы UNIX и графическом формате GIF.

Алгоритм LZFG является в некотором смысле является гибридом LZ77 и LZ78 [9]. Как и LZ77, он допускает кодирование цепочек произвольной длины (в алгоритмах LZ78 длина цепочек, как правило, ограничена количеством предыдущих появлений цепочки во входном потоке). С другой стороны, адресация элементов словаря аналогична технике LZ78 с той разницей, что вместо двоичного дерева используется так называемое дерево Patricia. Его отличие состоит в том, что если дерево не содержит разветвлений, то последовательность вершин объединяется в одну.

Алгоритм RFGD является модификацией LZFG с более высокой степенью сжатия [10]. Он помещает в выходной поток данные четырех типов: терминальные вершины дерева, нетерминальные вершины дерева, литералы, встречавшиеся ранее во входном потоке и литералы, ранее не встречавшиеся. Каждый из них кодируется по собственной схеме. Для определения того, какого именно типа данные помещаются в выходной поток, непосредственно перед ними в выходной поток

помещается специальный префикс. Алгоритм реализован аппаратно.

2.4. Статистические модели

Статистические модели содержат информацию о вероятностях наступления того или иного события. Таким образом, статистические модели могут являться составной частью других моделей, например словарных. Следует отметить, что приведенные выше кодеры ориен–тированы на работу именно со статистическими моделями. В то же время сложные статистические модели могут использоваться как самостоятельные для алгоритмов сжатия.

2.4.1. Контекстно-ограниченные модели (статистические модели высших порядков)

Под статистическими моделями высших порядков понимается сложная модель, в которой для каждой из встречавшихся ранее цепочек событий (или некоторого их подмножества) хранится распределение вероятности для следующего события [5]. Предшествующую цепочку событий и соответствующее ей распределение вероятностей принято называть контекстом. Если максимальная длина цепочки ограничена некоторым значением N, то говорят, что порядок модели равен N. Статистические модели высших порядков используют при сжатии текста (семейство алгоритмов PPM [11-14]) и графики.

Контекст как средство хранения вероятностей событий может быть реализован несколькими способами. Поскольку основное его назначение – обеспечение необходимыми данными арифметического или аналогичного кодера, то основным требованием к контексту является простота выполнения следующих операций:

а) определения начала и конца вероятностного диапазона для произвольного события (рис.1);

б) нахождения диапазона по значению, которое в этот диапазон попадает (для декодировщика);

в) расширения одного из диапазонов (для реализации адаптивной схемы);

г) маскирования, т.е. исключение некоторых событий из диапазона. Диапазоны других событий при этом должны быть пропорционально увеличены. Эта операция применяется только для сложных статистических моделей.

Классическая реализация арифметического кодера ориентирована на представление дробных чисел в виде пар чисел от 0 до 16383, которые являются числителем и знаменателем дроби.

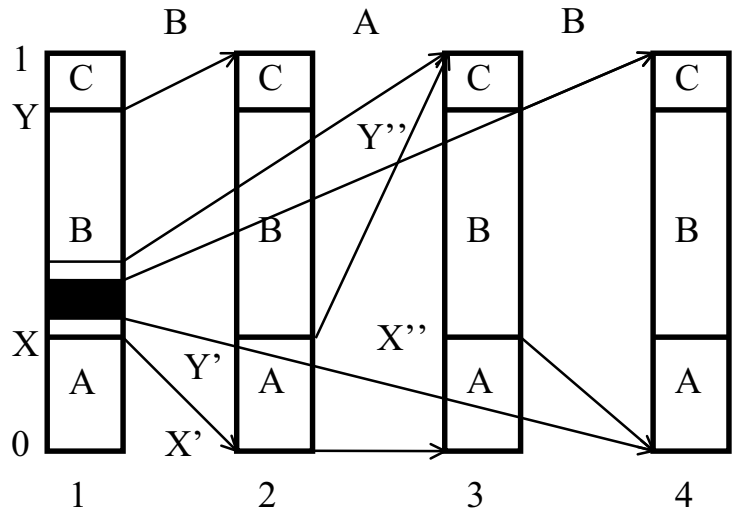

Рис. 1. Схема работы алгоритма арифметического кодирования

Существуют следующие способы реализации контекстов.

1. Каждая вероятность хранится как целое число. Отдельно хранится суммарная вероятность. Для выполнения операций а), б) необходимо просуммировать в среднем половину вероятностей элементов контекста. Операция в) выполняется простым увеличением вероятности одного требуемого элемента. Легко реализуется операция г). Этот тип представления контекста применяется там, где необходимо маскирование, т.е. в сложных статистических моделях.

2. Вероятности хранятся в виде нарастающих сумм. Для выполнения операции а) достаточно обращения по индексу. Для выполнения операции б) достаточно двоичного поиска. Для выполнения операции в) необходимо увеличение в среднем половины вероятностей элементов контекста. Операция г) реализуема, но крайне неэффективно. Это наиболее часто применяемый тип представления контекста.

3. Аналогичен «2», но наиболее вероятные события перемещаются ближе к концу диапазона. Таки образом, при выполнении операции в) в среднем увеличивается не половина элементов, а меньшее количество. Алгоритм хорошо работает на потоках событий, в которых вероятности сильно различаются. Если же события примерно равновероятны, то затраты на перемещение диапазонов оказываются больше, чем полученный выигрыш. Этот тип представления контекста применяется, например, в некоторых реализациях LZSS алгоритма с арифметическим кодированием.

4. Предлагается быстрая реализация контекста в виде двоичного дерева. Типичный размер входного алфавита для контекста равен 256

(как для текста, так и для полутоновой графики либо графики с индексированными цветами). В качестве оптимальной конфигурации двоичного дерева предлагается дерево с 4 ветвями из каждой вершины и глубиной 4. Для выполнения операций а), б) и в) достаточно в среднем 2*4=8 действий. Этот тип представления позволяет эффективно реализовать операцию г) посредством вычитания двоичных деревьев, но только если маскирующий и маскируемый контексты имеют одинаковые распределения вероятностей. К сожалению, на практике распределения вероятностей этих контекстов всегда различны, так что единственным типом реализации контекстов, поддерживающим маскирование является контекст типа «1».

Экспериментальные данные сравнения быстродействия контекстов различных типов приведены в табл. 1.

Таблица 1

Сравнительная характеристика быстродействия различных типов реализации контекстов

Операция	Время выполнения, с		
	Тип «1»	Тип «2»	Тип «4»
Кодирование	5.397	9.534	2.423
Декодирование	10.676	10.675	3.195

Как видно из табл. 1, использование контекстов, реализованных на двоичных деревьях, позволяет повысить быстродействие алгоритмов сжатия в несколько раз. В качестве данных использовался «1995 CIA World Factbook» из стандартного набора тестовых файлов ACT (act.by.net), в качестве модели - статистическая модель нулевого порядка, а в качестве кодера – Range кодер [20].

2.4.2. Модели состояний

Вероятностные модели с конечным числом состояний основываются на конечных автоматах. Они имеют множество состояний *S(i)* и вероятностей перехода *P(i,j)* модели из состояния *i* в состояние *j*. При этом каждый переход обозначается уникальным символом. Таким образом, через последовательность таких символов любой поток входящих событий задает уникальный путь в модели (если он существует). Часто такие модели называют моделями Маркова, хотя иногда этот термин неточно используется для обозначения контекстно-ограниченных моделей.

Модели с конечным числом состояний способны имитировать контекстно-ограниченные модели. Например, модель нулевой степени

простого английского текста имеет одно состояние с 27 переходами обратно к этому состоянию: 26 для букв и 1 для пробела. Модель первой степени имеет 27 состояний, каждое с 27 переходами. Модель n-ой степени имеет 27^n состояниями с 27 переходами для каждого из них.

Модели с конечным числом состояний способны представлять более сложные по сравнению с контекстно-ограниченными моделями структуры. В качестве примера на рис. 2 показана модель состояний для строки, в которой событие «a» всегда встречается дважды подряд. Такая строка не может быть представлена в контекстуальной модели, поскольку для оценки вероятности события, следующего за последовательностью событий «a», должны быть рассмотрены произвольно большие контексты.

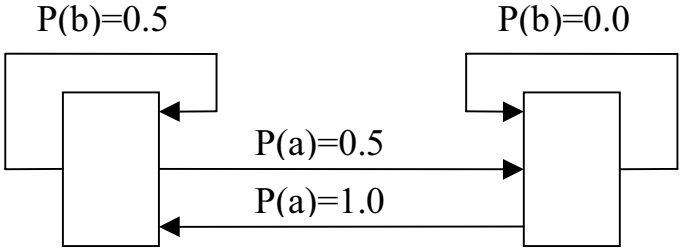

Рис. 2. Модель с ограниченным числом состояний для пар событий «a»

Единственный из приводимых в литературе метод моделирования с конечным числом состояний, работающий достаточно быстро, чтобы его можно было применять на практике, называется динамическим сжатием Маркова (ДМС) [15]. ДМС адаптивно работает, начиная с простой начальной модели, и добавляет по мере необходимости новые состояния. Основная идея ДМС состоит в поддержании счетчиков частот для каждого перехода в текущей модели с конечным числом состояний и "клонировании" состояния, когда соответствующий переход становится достаточно популярным.

К сожалению, оказывается, что выбор эвристики и начальной модели обеспечивает создаваемой модели контекстно-ограниченный характер, из-за чего возможности модели с конечным числом состояний не используются в полную силу. Главное преимущество ДМС над другими статистическими моделями состоит в предложении концептуально иного подхода, дающего ему возможность при соответствующей реализации работать быстрее.

2.5. Алгоритмы на основе преобразований

Цель использования преобразований в сжатии данных – преобразование потока входных событий к виду, позволяющему использовать более простые и эффективные модели. Фактически, они преобразовывают одни виды избыточности в другие, более просто моделируемые.

2.5.1. Группирующие преобразования. Преобразование MFT. Преобразование BWT

Преобразование MTF, называемое также преобразованием стопки книг, используется для преобразования избыточности повторения событий к избыточности распределения вероятностей событий. Событиям присваиваются индексы. Встретившееся событие получает индекс «1», остальные события сдвигаются. Таким образом, результатом преобразования потока событий будет поток с ярко выраженной избыточностью распределения:

«22223333333444333333444443»

«20003000000400100000100001».

Преобразование MTF практически не используется самостоятельно, но широко применяется как составная часть алгоритмов. Например, оно используется для преобразования выходного потока преобразования BWT.

Преобразование BWT (Burrows-Wheeler Transform) применяется для преобразования цепочечной избыточности в избыточность повторения событий [16-18].

Сначала входной поток событий циклически сдвигается на одну позицию и записывается под исходным входным потоком столько раз, сколько событий во входном потоке. Полученная квадратная матрица сортируется по строкам слева направо. Доказано, что для восстановления исходного потока событий достаточно последнего столбца матрицы (так называемого префиксного столбца) и номера строки исходного потока событий после сортировки. Как видно из табл. 2, префиксный столбец обладает большой избыточностью повторения событий и локальной избыточностью распределения вероятностей [18].

Существуют быстрые и эффективные реализации данного преобразования, не требующие полного построения квадратной матрицы в памяти (что привело бы к недопустимо высоким требованиям к используемой памяти).

Таблица 2

Пример результата преобразования BWT
для английского текста

Префикс	Отсортированные строки
T	hat acts like this:<13><10><1
T	hat buffer to the constructor
t	hat corrupted the heap, or wo
W	hat goes up must come down<13
t	hat happens, it isn't likely
w	hat if you want to dynamicall
t	hat indicates an error.<13><1
t	hat it removes arguments from
t	hat looks like this:<13><10><
t	hat looks something like this
t	hat looks something like this
T	hat once I detect the mangled

Выходной поток преобразования BWT, как правило, либо кодируется по методу RLE, либо дополнительно преобразуется по методу MTF и далее кодируется либо префиксным, либо арифметическим кодером.

Преобразование BWT используется для сжатия текстовой и графической информации.

2.5.2. Спектровыделяющие преобразования

Целью применения спектровыделяющих преобразований является преобразование пространственной избыточности к позиционной избыточности и избыточности распределения вероятностей. Они, как правило, используются в алгоритмах с потерями качества, но есть и специализированные преобразования без потерь и соответствующие алгоритмы сжатия преобразованных данных.

Спектровыделяющие преобразования применяются там, где есть пространственная избыточность – при сжатии звука, растровой графики, растровых трехмерных изображений, видео и т.д. Они преобразуют поток событий к его спектру [21, 22]. После чего низкочастотные составляющие (содержащие основную информацию) кодируются, а низкочастотные (содержащие малозначимую информацию и шум) отбрасываются.

Более подробно спектровыделяющие преобразования и алгоритмы сжатия на их основе будут рассмотрены в разделе, посвященном сжатию

изображений.

3. КОДИРОВАНИЕ

Кодирование, используемое в алгоритмах сжатия, преобразует поток событий с известным распределением вероятностей в двоичный код. Поскольку назначением кодирования является уменьшение количества передаваемой информации вплоть до величины ее энтропии, кодирование также называют энтропийным.

3.1. Префиксное и арифметическое кодирование

Под префиксным кодированием понимается кодирование событий посредством кодов, состоящих из целого числа бит. Под префиксностью подразумевается, что никакой из кодов при просмотре слева направо не является частью другого, более длинного кода [7]. Недостатком таких кодов является целочисленное представление величины $-log_2\ p$. Например, наименьший код, который может быть произведен префиксными кодами, имеет 1 бит в длину, хотя часто желательно использовать меньший. Символ "o" в контексте "to be or not t" можно закодировать в 0.014 бита. Префиксные коды превышают необходимый выход в 71 раз, делая точное предсказание бесполезным. С другой стороны, достоинством этих кодов является их низкая вычислительная сложность и, как следствие, высокое быстродействие. Альтернативой префиксному кодированию является арифметическое. Оно обладает оптимальностью, но уступает префиксному по быстродействию [5].

3.2. Префиксные коды: бинарный, унарный Голомба, Райс, Шеннона-Фоно, Хаффмана

Простейшим префиксным кодом является бинарный код. Он может использоваться для кодирования равновероятных событий. Вторым примитивным префиксным кодом является унарный код. Он кодирует номер события числом двоичных единиц, расположенных перед разделяющим нулем.

Распределение вероятностей событий редко в точности соответствует этим кодам. Для остальных распределений может быть использован код Голомба. Идея этого кода состоит в кодировании значения N/M унарным кодом, а значения N mod M – бинарным. N – порядковый номер события в списке, отсортированном по убыванию вероятности, M – некоторая константа. Как правило, на практике применяется подмножество кодов Голомба, именуемое Райс кодом. В

нем M всегда равно степени 2 (табл. 3) [7, 19].

Таблица 3

Райс-код

Событие	M=1	M=2	M=4	M=8
1	0	0-0	0-00	0-000
2	10	0-1	0-01	0-001
3	110	10-0	0-10	0-010
4	1110	10-1	0-11	0-011
5	11110	110-0	10-00	0-100
6	111110	110-1	10-01	0-101
7	1111110	1110-0	10-10	0-110
8	11111110	1110-1	10-11	0-111
9	111111110	11110-0	110-00	10-000
10	1111111110	11110-1	110-01	10-001

Достоинство этих кодов – простота и легкая вычислимость. Недостаток – неоптимальность, что приводит к потерям в степени сжатия.

Код Шеннона-Фоно является простым префиксным кодом, и в то же время очень близок к оптимальному [5, 7]. Техника получения кодов такова: возможные события выписываются в порядке неубывания их вероятностей, после чего список делится на две группы с примерно равными суммарными вероятностями. Первой группе назначается первый бит «0», второй – «1». Разбиение рекурсивно повторяется для каждой из групп, пока каждому из событий не будет присвоен код. Код Хаффмана оптимален, но немного более вычислительно сложен [5, 7]. Для получения кодов Хаффмана события также выписываются в порядке неубывания их вероятностей, после чего берутся два наименее вероятные события и удаляются из списка. На их место помещается событие с суммарной вероятностью. К коду менее вероятного события слева добавляется бит «0», к коду более вероятного – «1». Эта операция выполняется последовательно, пока коды не будут полностью сформированы. Если наименее вероятное событие уже было обработано ранее, т.е. представляет собой группу событий, то бит добавляется слева к коду каждого из событий в группе (табл. 4) [7].

3.3. Арифметическое, квазиарифметическое и Range-кодирование

В отличие от префиксного кодирования, арифметическое кодирование позволяет кодировать события с вероятностью *p*

количеством бит, сколь угодно близким к величине $-log_2 p$ [4, 5].

Таблица 4

Сравнение кодов Шеннона-Фоно и Хаффмана

Событие	Вероятность	Код Шеннона-Фоно	Код Хаффмана
1	0.35	00	1
2	0.17	01	011
3	0.17	10	010
4	0.16	110	001
5	0.15	111	000
Средняя длина кода	-	2.31	2.30

Пусть необходимо закодировать последовательность событий, причем в каждый момент может наступить событие A, B либо C. Отложим вероятности событий A, B и C на отрезке [0,1).

Работа алгоритма наглядно показана на рис. 1, где на 4 диаграммах схематически отображены первые 3 шага работы алгоритма для входного алфавита, состоящего из событий «A», «B» и «C» при поступлении на вход алгоритма потока событий «B», «A», «B». Первая диаграмма показывает деление интервала (0,1) на начальные интервалы между символами входного алфавита до начала работы алгоритма. После поступления на вход алгоритма события «B» рабочий интервал (X,Y) аналогичным образом делится между символами входного алфавита (диаграмма 2) и т.д. Диаграмма 4 на рис. 1 соответствует заштрихованному участку диаграммы 1.

По мере кодирования событий из входного потока рабочий интервал (X,Y), (X’,Y’), (X’’,Y’’), ... сокращается. Насколько он сократится при кодировании одного события, зависит от ширины начального интервала, соответствующего событию (вероятности события).

Теоретически выходными данными алгоритма может служить любое число, входящее в рабочий интервал после обработки последнего события входного потока. На практике же при каждом сокращении интервала в выходной поток помещаются все совпадающие старшие биты верхней и нижней границы рабочего интервала. Очевидно, что при поступлении события с большей вероятностью появления и с более широким начальным интервалом рабочий интервал сократится меньше и

меньше битов попадет в выходной поток. Таким образом, события с большей вероятностью появления кодируются меньшим числом бит.

Недостатком арифметического кодера является его высокая вычислительная сложность. Для каждого бита выходного потока необходимо выполнять ряд операций сравнения и сдвига. Кроме того, для каждого кодируемого события необходимо выполнить ряд операций умножения и деления.

С целью повышения быстродействия был предложен ряд более быстрых алгоритмов. В качестве наиболее быстрой альтернативы арифметическому кодированию может быть использовано квазиарифметическое [19]. В нем операции умножения и деления заменены операциями выбора значения из таблицы. Алгоритм организован в виде конечного цифрового автомата. При поступлении на вход кодируемого события алгоритм в зависимости от текущего состояния при необходимости помещает в выходной поток некоторые биты и переходит в некоторое новое состояние. Таблицы переходов и выходов составлены таким образом, чтобы эмулировать работу арифметического кодера. К недостаткам квазиарифметического кодера можно отнести низкую точность (либо чрезмерно большой объем таблиц), а также то, что он реализован только для случая с двумя событиями.

Удачной альтернативой арифметическому кодеру, широко используемой в настоящее время, является Range-кодер [20]. Основным его отличием от арифметического кодера является не побитное, а побайтное сравнение верхней и нижней границы и побайтное помещение закодированного результата в выходной поток. Такой подход позволяет заметно повысить быстродействие кодера. При этом точность (качество кодирования) у отдельных реализаций Range кодеров оказывается даже выше, чем у классического арифметического кодера (при использовании 32-битной целочисленной арифметики). Сравнительная характеристика экспериментальных данных скорости работы кодеров приведена в табл. 5. В качестве данных использовался «1995 CIA World Factbook» из стандартного набора тестовых файлов ACT (act.by.net), в качестве модели использована статистическая модель нулевого порядка, реализованная на контексте типа «с» (см. п. 2.4.1). Как видно из таблицы, Range кодер работает с той же скоростью, что и сохранение без кодирования. Иными словами, он достаточно быстр для большинства приложений.

Сравнение табл. 5 и 1 наводит на мысль, что вопреки распространенному предубеждению алгоритмы с арифметическими кодерами

медленнее алгоритмов с префиксными в первую очередь из-за того, что в них по-разному представляются контексты. Так, для кодера Хаффмана контекст обычно хранится в виде двоичного дерева, чем и обусловлено его высокое быстродействие. Дальнейшим направлением ускорения алгоритмов сжатия должно быть ускорение контекста и модели в целом, а не кодера.

Таблица 5

Сравнительная характеристика скорости кодирования и декодирования арифметического и Range кодеров

Операция	Время выполнения, с		
	Арифметический кодер	Range кодер	Сохранение без кодирования
Кодирование	3.625	2.423	2.513
Декодирование	4.317	3.195	3.005

4. СЖАТИЕ ТЕКСТОВОЙ ИНФОРМАЦИИ

Текстовая информация характеризуется наличием избыточности распределения символов и избыточностью повторения одних и тех же цепочек символов.

4.1. Алгоритмы первого поколения

К алгоритмам первого поколения можно отнести LZ77- и LZ78-подобные алгоритмы, которые легли в основу большого количества практических реализаций программ-упаковщиков. Наиболее популярными были сочетания LZSS плюс префиксное либо арифметическое кодирование. Большинство широко известных программ-упаковщиков текстовой информации, таких как ARJ, LHA, PKZIP, GZIP, RAR, были созданы на их основе.

4.2. Алгоритмы второго поколения

К алгоритмам второго поколения можно отнести алгоритмы семейства PPM, алгоритмы на основе преобразования BWT и алгоритмы на основе ассоциативного кодирования, которое является одной из разновидностей контекстного моделирования. Основная их отличительная особенность - высокая степень сжатия.

Сравнительная характеристика лучших реализаций алгоритмов для сжатия текста (по данным интернет-сайта act.by.net за март 2000 г.) приведена в табл 6.

Таблица 6

Сравнительная характеристика некоторых реализаций алгоритмов сжатия текстовой информации

Реализация	Пара-метры	Алго-ритм	Степень сжатия, бит/символ	Скорость упаковки, с	Скорость распаковки, с
RK 1.01b1	-mx2 1	PPM	1.4258	159.37	162.34
ACB 2.00c	(u)f	АК	1.5621	332.40	333.37
PPMD vE	-o5 -m28	PPM	1.6455	11.93	13.48
SZIP 1.11b	-b25o0	BWT	1.6476	42.22	10.57
LZAP 0.20.0	(none)	LZP	1.8795	116.03	123.74
PKZIP 2.04g	-ex	LZ+ SF	2.5646	19.79	10.35

Как видно из таблицы, существуют реализации алгоритма PPM, превосходящие лучшие реализации других алгоритмов по степени сжатия и не уступающие им по скорости. Алгоритмы первого поколения (такие, как PKZIP) значительно отстают по степени сжатия от современных алгоритмов.

4.3. Алгоритм PPM

Алгоритм PPM (prediction by partial matching) - это метод контекстно-ограниченного моделирования, позволяющий оценить вероятность символа в зависимости от предыдущих символов [4, 11-14]. Строку символов, непосредственно предшествующую текущему символу, будем называть контекстом. Модели, в которых для оценки вероятности используются контексты длиной не более чем N, принято называть моделями порядка N.

Вероятность символа может быть оценена в контекстах разных порядков. Например, символ "o" в контексте "to be or not t" может быть оценен в контексте первого порядка «t», в контексте второго порядка «_t», в контексте третьего порядка «t_t» и т.д. Он также может быть оценен в контексте нулевого порядка, где вероятности символов не зависят от контекста, и в контексте минус первого порядка, где все символы равновероятны. Контекст минус первого порядка используется для того, чтобы исключить ситуацию, когда символ будет иметь нулевую вероятность и не сможет быть закодирован. Это может случиться, если вероятность символа не будет оценена ни в одном из контекстов (что возможно, если символ в них ранее не встречался).

Существуют два основных подхода к вычислению распределения

вероятностей следующего символа на основе вероятностей символов в контекстах. Первый подход называется «полное перемешивание». Он предполагает назначение весов контекстам разных порядков и получение суммарных вероятностей сложением вероятностей символов в контекстах, умноженных на веса этих контекстов. Применение такого подхода ограничено двумя факторами. Во-первых, не существует быстрой реализации данного алгоритма. Во-вторых, не разработан эффективный алгоритм вычисления весов контекстов. Примитивные же подходы не обеспечивают достаточно высокой точности оценки и, как следствие, степени сжатия.

Второй подход называется «методом исключений». При этом подходе сначала делается попытка оценить символ в контексте самого высокого порядка. Если символ кодируется, алгоритм переходит к кодированию следующего символа. В противном случае кодируется «уход» и предпринимается попытка закодировать символ в контексте меньшего порядка. И так далее, пока символ не будет закодирован.

4.3.1. Оценки вероятности ухода

Одной из проблем реализации алгоритма PPM на основе метода исключений является определение вероятности ухода. Существует несколько подходов к вычислению этой вероятности. Формулы для основных подходов приведены в табл.7. В ней C – количество символов в контексте, Q – количество просмотров контекста, Qi – количество таких разных символов в контексте, что они встречались ровно i раз [4].

Таблица 7

Методы оценки вероятности ухода для алгоритма PPM

Метод оценки вероятности ухода	Вероятность ухода
A	1/(C+1)
B	(Q-Q1)/C
C	Q/(C+Q)
D	Q/(2*C)
P	$Q1/C – Q2/C^2 – Q3/C^3$ - …
X	Q1/C
XC	Q1/C при 0 < Q1 < C, иначе по методу C

4.3.2. Реализации PPM на основе хэша и на основе двоичного дерева

Для нахождения контекста по символам, которые он представляет, могут использоваться два подхода. Первый подход – построить дерево.

Пусть контекст «t_t» содержит для символа «o» указатель на контекст «t_to». В этом случае после кодирования символа «o» проблема поиска следующих контекстов решается автоматически. Второй подход состоит в организации хэша, который преобразует строку, соответствующую контексту, в индекс этого контекста в таблице контекстов. Первый подход требует больше памяти, но позволяет достичь лучшего быстродействия. На нем основана реализация PPMD Дмитрия Шкарина, обеспечивающая рекордно высокое для алгоритма PPM быстродействие. На хэше основаны PPMDummy Максима Смирнова, HA (автор - Harry Hirvola) и PPMZ (автор - Charles Bloom).

4.3.3. Модели ограниченного и неограниченного порядка

Для текстовых данных использование контекста неограниченного порядка не дает преимущества относительно моделей порядка 5-6 [4, 13]. В то же время исключение контекстов некоторых порядков позволяет добиться некоторого повышения степени сжатия. Это вызвано тем, что соседние контексты высоких порядков (начиная с четвертого) имеют примерно одинаковое распределение и данные в них фактически дублируются. В то же время вероятность ухода приходится кодировать. В связи с этим в PPMDummy при использовании контекстов порядка 5 и выше контекст четвертого порядка не используется. В реализации RKUC используются контексты порядков 16-12-8-5-3-2-1-0-(-1).

4.3.4. Recence scaling, detemenistic scaling, LOE, SEE, биграммы

Существует ряд усовершенствований, позволяющих улучшить качество предсказания вероятностей и повысить степень сжатия алгоритма PPM. Под recence scaling понимается увеличение ожидаемой вероятности встретившегося недавно символа. Вероятность последнего встретившегося в контексте символа умножают на некоторый множитель, который сильно зависит от избыточности сжимаемых данных. Применение recency scaling позволяет улучшить сжатие в среднем на 0.5% [12].

Контекст, содержащий ровно один символ, называют deterministic контекстом. Метод C переоценивает вероятность ухода для deterministic контекстов. В связи с этим вероятность ухода для таких контекстов искусственно понижают [12, 13].

Local Order Estimation (LOE) и Secondary Escape Estimation (SEE) представляют собой попытки улучшить предсказание распределения вероятностей на основе некоторой информации о параметрах текущих контекстов различных порядков. LOE предполагает локальное

ограничение максимального порядка модели, а SEE – локальную коррекцию вероятности ухода. Существует множество эвристических подходов для реализации этих усовершенствований, некоторые из которых обеспечивают заметное улучшение степени сжатия [12].

Интерпретация часто используемых в языке биграмм (th, qu и т.д.), а также комбинаций из большего числа символов как дополнительных символов алфавита также позволяет несколько повысить степень сжатия [11].

5. СЖАТИЕ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Данная глава посвящена сжатию растровой графической информации. Растровая графическая информация характеризуется в первую очередь пространственной избыточностью. Тип избыточности векторной графической информации зависит от слишком большого числа факторов (в первую очередь от типа информации и ее представления) и не обладает достаточной универсальностью для создания общих формальных подходов к ее устранению. В то же время в главе 6 приводится универсальный формализованный подход к сжатию любых типов данных, который может быть использован и для векторной графики любого типа.

5.1. Виды изображений: бинарные, полутоновые, цветные, с индексными цветами

Бинарным изображением называется изображение с двумя цветами, как правило черным и белым (1 бит на пиксель). Примером бинарного изображения является факсимильный документ. Полутоновым изображением называется монохромное изображение, состоящее из градаций какого-либо цвета, например градаций серого (как правило, 8 бит на пиксель). Каждый пиксель цветного изображения представлен величиной красной, синей и зеленой составляющих его цвета (RGB, 24 бита на пиксель). Изображение с индексными цветами состоит из палитры (обычно 16-256 фиксированных цветов, каждый из которых задан в формате RGB, 24 бита на пиксель) и изображения, каждый пиксель которого задан индексом цвета в палитре (4 бита для 16-цветной палитры, 8 бит для 256-цветной) [23, 24].

Для полутоновых и цветных изображений соседние пиксели, как правило, имеют либо одинаковые, либо близкие цвета. Для изображений с индексными цветами соседние пиксели проверяют только на равенство, но не на близость значений.

Кроме того, для цветных изображений характерна сильная

корреляция между пространственными производными разных цветовых составляющих [21].

5.2. Форматы графических файлов – PCX, GIF, TIFF, JPG

Графический формат PCX поддерживает кодирование RLE [24]. Он рассчитан на изображения с индексными цветами. RLE-кодирование реализовано следующим образом. Кодируется длина цепочки повторяющихся пикселей. Затем кодируется цвет пикселя. Подразумевается, что цепочка пикселей одинаковой длины закончилась и далее следуют пиксели с другими цветами. Поэтому далее кодируется длина цепочки пикселей с разными цветами и сама цепочка, после чего все повторяется сначала. Алгоритм имеет достаточно низкую степень сжатия.

Графический формат GIF рассматривает изображение как одномерную последовательность пикселей [24]. Пиксели кодируются по строкам с использованием алгоритма LZC (модификация алгоритма LZW). Формат рассчитан на изображения с индексными цветами и обладает более высокой степенью сжатия, чем PCX. Существуют две модификации этого формата – GIF87a и GIF89a. Они отличаются количеством поддерживаемых типов блоков расширений. GIF98a поддерживает большее количество расширений, в том числе анимацию.

Формат TIFF поддерживает большое количество алгоритмов сжатия для различных типов изображений. Например, для бинарных изображений лучшую степень сжатия обеспечивает алгоритм CCITT Group 4 Fax. Однако могут быть использованы и другие алгоритмы, например LZW или CCITT Group 3 Fax. Для сжатия цветных изображений и изображений с индексными цветами используется алгоритм LZW.

Формат JPEG предназначен для сжатия цветных изображений. Он может сжимать как с потерей качества, так и без потерь [21]. Сначала изображение разделяется на яркостную и две цветовые составляющие. Яркостная составляющая более важна при восприятии изображения человеческим глазом, поэтому ее следует сжимать с лучшим качеством. Затем эти составляющие разбиваются на квадраты размером 8x8 пикселей, и для каждого из квадратов выполняется дискретное косинус-преобразование Фурье. После этого производится квантизация результатов и энтропийное кодирование. В результате преобразования наиболее важные (и имеющие большие значения) низкочастотные коэффициенты располагаются в левых верхних углах квадратов. Наименее важные – в правых нижних. Поэтому коэффициенты берутся в зигзагообразном порядке, начиная из левого верхнего угла и заканчивая

правым нижним. Для кодирования коэффициентов используется кодирование Хаффмана с фиксированными таблицами либо арифметическое кодирование.

Существует огромное количество других графических форматов, однако большинство из них не лучше, чем GIF и JPEG [24]. В то же время GIF и JPEG являются самыми распространенными на сегодняшний день графическими форматами в первую очередь потому, что это единственные форматы, поддерживаемые всеми Интернет-баузерами. Вся графическая информация в сети Интернет представлена именно в этих форматах.

В то же время эти форматы являются существенно устаревшими. Существуют подходы, позволяющие сжимать изображения значительно лучше и быстрее, но их внедрение тормозится инертностью процессов обновления информационных технологий и крупных компаний, от которых это обновление в сильной степени зависит. Например, компания Микрософт основывает свои новейшие продукты (Windows 2000) на устаревших алгоритмах (JPEG и PNG).

5.3. Сжатие полутоновых изображений

Наибольшее внимание в настоящее время уделяется именно сжатию полутоновых изображений. Результаты решения этой задачи легко могут быть использованы для сжатия цветных изображений – яркостная и цветовые составляющие могут рассматриваться как полутоновые изображения и сжиматься независимо.

5.3.1. Алгоритмы на основе последовательного сканирования изображения. Алгоритмы FELICS, MLP, LOCO-I, CALIC

Алгоритмы на основе последовательного сканирования обрабатывают изображения слева направо, сверху вниз. Они предсказывают значение следующего пикселя на основе значений уже закодированных соседних (рис. 3). В работе алгоритмов этой группы можно выделить следующие фазы: предсказание, моделирование ошибки и кодирование [25-31]. Предсказание определяет ожидаемое значение следующего пикселя. В качестве предсказывающего элемента могут быть использованы различные эвристические модели, многоконтекстные модели, различные предикторы (двоичные [31], L [30] и др.) и т.д. Простые алгоритмы, такие как FELICS [25] и LOCO-I [27], используют для предсказания значения двух соседних пикселей. Более серьезные алгоритмы используют 10-15 пикселей.

Моделирование ошибок оценивает вероятность того, что ошибка

предсказания будет иметь ту или иную величину. Функция плотности распределения вероятности ошибок близка к нормальному распределению для высоких и средних дисперсий и к распределению Лапласа для низких [31]. Алгоритм CALIC применяет результаты моделирования ошибок для дополнительной коррекции результатов предсказания [28, 29].

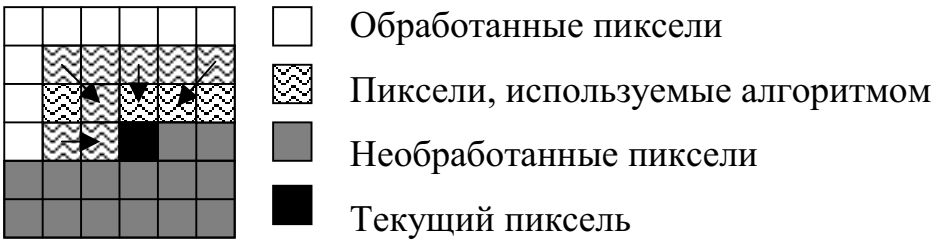

Рис. 3. Предсказание следующего пикселя по 12 известным соседним

В качестве кодера используют различные энтропийные кодеры, например арифметический.

Оригинальный подход используется в алгоритме MLP [26]. При первом проходе обрабатывается не каждый пиксель, а каждый N-й (например, каждый второй). При следующем проходе сжимаются пиксели на пересечении диагональных линий, проведенных через обработанные пиксели, при следующем – на пересечении вертикальных и горизонтальных и т.д. В результате достигается возможность прогрессивной передачи сжатого изображения, а также некоторое повышение степени сжатия, так как теперь для предсказания можно использовать не только левых и верхних соседей, но также правых и нижних, обработанных на предыдущих проходах.

Алгоритмы на основе последовательного сканирования ориентированы на сжатие изображений без потерь. Лучшие из них обеспечивают очень высокую степень сжатия (выше, чем лучшие реализации алгоритмов на основе преобразований, например SPIHT). Это скорее всего вызвано тем, что алгоритмам на основе преобразований без потерь качества просто не уделялось достаточно внимания.

5.3.2. Алгоритмы на основе преобразований

Алгоритмы на основе преобразований предназначены в первую очередь для сжатия изображений с потерями качества. Схема алгоритма такова – преобразование, квантизация, моделирование преобразованных коэффициентов, кодирование. Существуют два подхода к выполнению преобразований. Первый подход заключается в выполнении

преобразования, результатом которого является спектр изображения. Высокая вычислительная сложность таких преобразований не позволяет применять их для фрагментов изображения большого размера. Поэтому при этом подходе изображения разбиваются на фрагменты размером 8x8 пикселей, и преобразование выполняется для каждого из этих фрагментов. Типичным алгоритмом этого типа является алгоритм JPEG, описанный выше [20]. Другой подход заключается в многократном выполнении преобразования, отделяющего самую высокочастотную составляющую изображения [30].

5.3.3. Преобразование Фурье, S+P, вавелеты, лифтинг-преобразования

Используемые преобразования фактически одномерны, они используются последовательно для всех строк, затем для всех столбцов изображения. Входными данными преобразования является одномерный вектор, результатом – два одномерных вектора в два раза меньшей длины. В одном из них находится самая высокочастотная составляющая, в другом – остальная часть сигнала.

В результате одного вертикального и горизонтального преобразования образуются четыре квадрата (рис. 4). Левый верхний содержит низкочастотную составляющую, три других – высокочастотные. Затем левый верхний квадрат преобразуется еще раз и так далее, пока его размер не уменьшится до 4x4 – 16x16 пикселей. Дальнейшие преобразования не улучшают качество работы алгоритма.

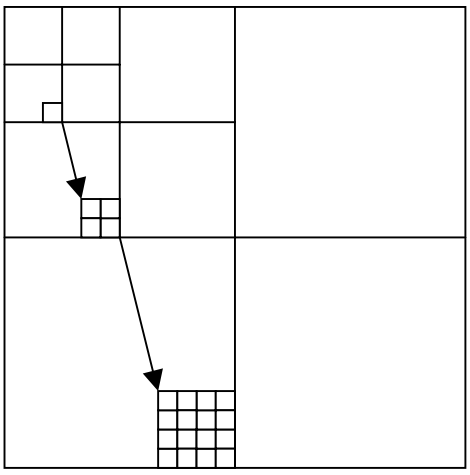

Рис. 4. Матрица коффициентов преобразованного изображения

Наиболее простым и быстрым является S-преобразование. В качестве высокочастотной составляющей используется разность четных и нечетных пикселей. В качестве низкочастотной – половина их суммы.

В результате этого преобразования в высокочастотную составляющую все же попадает некоторая часть низкочастотного сигнала. Для ее фильтрации используется P-преобразование. Оно по трем или более значениям низкочастотной составляющей вычисляет по эмпирическим формулам корректирующее значение для высокочастотной составляющей. S+P-преобразование является целочисленным и полностью обратимым. Его применяют для алгоритмов сжатия без потери качества и быстрых алгоритмов. Преобразование хорошо работает на искусственных изображениях, но хуже чем, например, вавелет-преобразования на изображениях с плавными переходами цветов.

Вавелет-преобразования делают то же, что и S+P, но для вычисления низкочастотной и высокочастотной составляющей используется сложение умноженных на вавелет-коэффициенты значений некоторого количества соседних пикселей. Некоторые вавелет-преобразования также являются обратимыми, но требуют больше двоичных разрядов для представления преобразованных значений.

Наиболее перспективными на сегодняшний день представляются лифтинг-преобразования, называемые также вавелетами второго поколения [33-35].

Лифтинг преобразования на самом деле не какой-то новый вид преобразований, а просто новый подход к описанию преобразований. Многие преобразования могут быть представлены в лифтинг-схеме. Основная идея лифтинг-подхода – при каждой операции должна изменяться только одна составляющая. При таком подходе построение обратного преобразования не составляет труда. Достаточно сделать зеркальное отображение прямого преобразования, добавить инвертирование сигнала там, где его нет и убрать там, где оно есть. Лифтинг подход позволяет использовать даже функции, не имеющие обратных.

Пример сравнения классического и лифтинг-подхода для представления S-преобразования приведен на рис. 5 и 6. При лифтинг-подходе обратное преобразование строится элементарно просто, для классического подхода даже существование обратного преобразования не очевидно.

5.3.4. Сжатие преобразованных изображений. Алгоритм SPIHT

Результатом преобразования является матрица коэффициентов величиной от -2^{15} до 2^{15}. В ней присутствуют следующие виды избыточности:

1. Значения в левом верхнем углу, как правило, больше по модулю, чем значения в других частях матрицы.

2. Если некоторый участок изображения имеет большую низкочастотную составляющую, он, как правило, также имеет и большие низкочастотные составляющие.

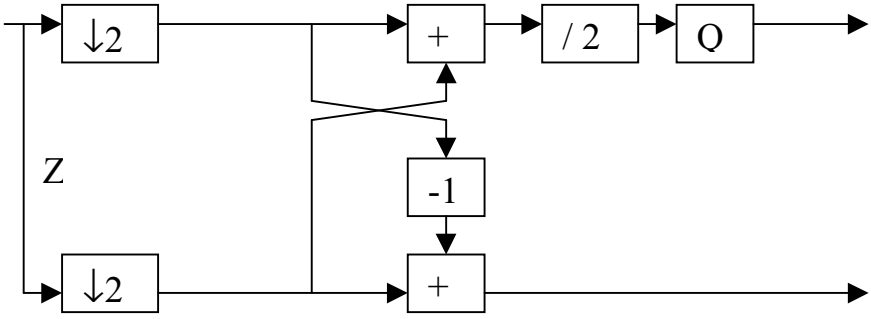

Рис. 5. S-преобразование

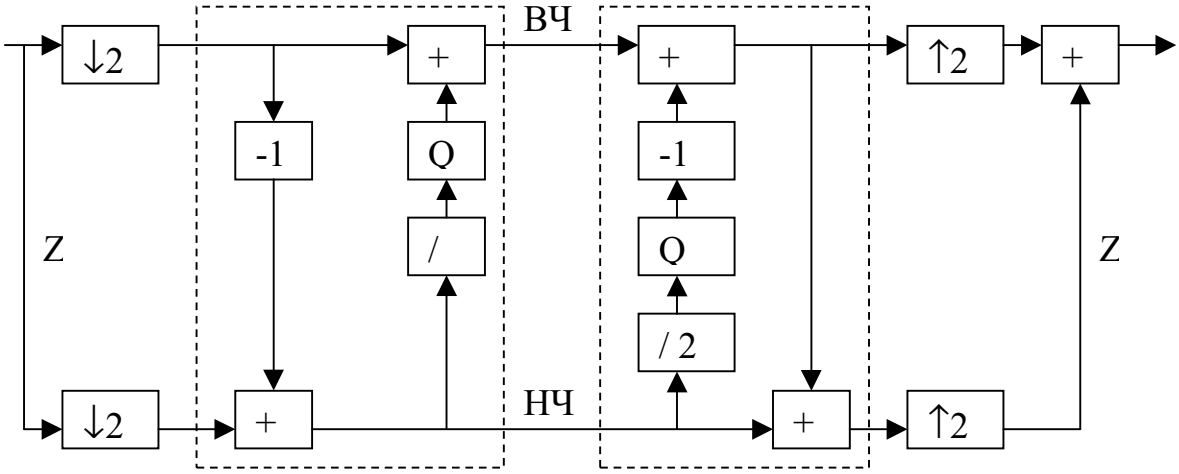

Рис. 6. Представление прямого и обратного S преобразования в лифтинг схеме

Следует отметить, что знаки коэффициентов матрицы распределены почти случайным образом и практически безызбыточны.

Поскольку подход ориентирован на сжатие с потерями, задача алгоритма сжатия – получение максимального соотношения сигнал/шум в восстановленном изображении при заданном коэффициенте сжатия. Для большого числа преобразований доказано, что при наличии некоторой среднеквадратичной ошибки в преобразованной матрице точно такая же ошибка будет и в восстановленном изображении. Поэтому задача алгоритма сжатия – закодировать как можно точнее в первую очередь коэффициенты, имеющие большое абсолютное значение.

Простейший алгоритм сжатия EZW (Embedded Zerotree Wavelet)

предполагает простое отбрасывание коэффициентов со значениями, меньшими некоторой величины. Не намного более вычислительно сложный алгоритм SPIHT (Set Partitioning in Hierarchical Trees) дает гораздо лучшие результаты [22].

Каждому участку изображения соответствует некоторое количество коэффициентов в матрице. Эти коэффициенты можно представить в виде дерева, где корнем является самая низкочастотная составляющая, относящаяся к этому участку, а ветвями – низкочастотные составляющие (см. рис. 4). Алгоритм оперирует следующими понятиями: список значащих пикселей (СЗП) – список пикселей, значения которых больше текущего порогового значения; список незначащих пикселей (СНП) – список пикселей, значения которых меньше порога; список незначащих деревьев типа А и Б (СДА, СДБ) – список деревьев (СД), в которых все значения меньше порога, включая и не включая четыре пикселя верхнего уровня соответственно. Под пикселями понимаются элементы матрицы. Под порогом понимается число, равное степени двойки. Оно уменьшается в два раза между проходами. Алгоритм построен таким образом, что матрица обрабатывается бит за битом от старших к младшим. Таким образом, наиболее значащая информация кодируется в первую очередь.

Перед началом работы алгоритм вычисляет для каждой вершины дерева максимальное абсолютное значение дочерних вершин. Все деревья верхнего уровня заносятся в СД, все корни этих деревьев – в СНП. Далее в цикле повторяются фаза сортировки и фаза уточнения, после чего пороговое значение уменьшается в два раза. Работа алгоритма может быть прекращена в любой момент. Чем дольше будет работать алгоритм и чем больше данных попадет в выходной поток, тем лучше будет качество восстановленного изображения.

Фаза сортировки.

Для каждого элемента СНП кодируется текущий бит. Если он равен «1», то кодируется знак элемента, а элемент переносится в СЗП.

Для каждого элемента СД типа СДА

- Кодируется текущий бит. Если бит равен «1», то элемент разбивается на 4 поддерева. При этом для каждого поддерева кодируется текущий бит его корневого элемента. Если он равен единице, то кодируется знак элемента, а элемент переносится в СЗП. В противном случае элемент помещается в СНП.
- Если текущий бит максимального значения поддерева, не включая значение корневого элемента, равен «1», то пометить поддерево как СДБ и поместить в конец списка СД для обработке в

этой же фазе, в противном случае просто удалить поддерево из списка СД.

Для каждого элемента СД типа СДБ если текущий бит максимального значения дерева, не включая значение корневого элемента, равен «1», то дерево разбивается на 4 поддерева типа А, которые заносятся в конец списка СД для обработки в этой же фазе. В противном случае элемент удаляется из списка СД.

Фаза уточнения

Для каждого элемента, за исключением помещенных в текущей фазе сортировки, закодировать текущий бит.

Реализация, предложенная авторами алгоритма, содержит два способа кодирования – двоичное и арифметическое. Двоичное просто выводит значения текущего бита. Для поддержки арифметического используется достаточно сложная модель избыточности кодируемых данных, реализованная на основе системы контекстов.

Восстановленное изображение при больших коэффициентах сжатия бывает искажено. Для алгоритмов сжатия типа JPEG это искажения совмещения квадратов 8x8. Для алгоритмов на основе вавелет-подобных преобразований это искажения на границе областей с сильно отличающейся яркостью. Существует специальная техника фильтрации, которая позволяет существенно улучшить качество восстановленных изображений.

6. ТЕХНИКА ПОСТРОЕНИЯ НОВЫХ АЛГОРИТМОВ СЖАТИЯ

Для сжатия данных известных типов имеет смысл использовать один из существующих алгоритмов сжатия. Однако если требуется сжимать большое количество данных какого-нибудь определенного типа (например, данные на некотором формальном языке, документы некоторого типа и т.д.) либо необходимо сжать данные, полученные в результате некоторого преобразования, целесообразно синтезировать новый алгоритм сжатия, что позволит достичь более высокой степени сжатия.

Решение проблемы синтеза подобных алгоритмов для некоторых частных случаев приводится в [36] (для HTML документов) и [37] (для генеалогических баз данных). Настоящая глава посвящена разработке универсального подхода к синтезу алгоритмов для любых типов данных.

6.1. Анализ требований к новому алгоритму

Построение алгоритма сжатия информации - это всегда поиск

компромисса между требованиями к степени сжатия, быстродействием, сложностью алгоритма и количеством требуемой для работы алгоритма памяти. Перед построением алгоритма следует ответить на следующие вопросы:

- Сколько памяти будет доступно алгоритму?
- Должен ли алгоритм быть максимально быстрым?
- Должен ли алгоритм быть простым?
- Должен ли алгоритм иметь максимально возможную степень сжатия?
- Должен ли алгоритм быть однопроходным (потоковым)?

Если главным критерием является быстродействие, имеет смысл использовать префиксное кодирование. Если нет, то лучше использовать Range кодирование. Если размер свободной памяти менее 100 кбайт (например, алгоритм будет использоваться во встраиваемых системах) и данные имеют сложную модель (например, нужно сжимать текстовую информацию), то статистические модели не подходят и следует использовать словарные. Если памяти достаточно (6-8 мбайт и более), то следует использовать статистические модели. Алгоритмы на основе BWT-преобразования не могут быть эффективно реализованы как потоковые (например, для использования в модемах).

6.2. Оценка избыточности

Далее следует оценить виды избыточности, присутствующие в данных, и определить, какие из них будут устранены и насколько. Если требуется достичь максимальной степени сжатия, следует максимально устранить избыточность всех видов. Если основным требованием является простота алгоритма, следует постараться устранить только наиболее значимые виды избыточности.

Сжимаемые данные могут иметь сложную структуру. Например, они могут иметь блоки, содержащие данные с разными характеристиками. Например, документ Microsoft Word может содержать текст, изображения в двоичном виде, а также разную служебную информацию. Гипертекстовый документ или программа на алгоритмическом языке может содержать ключевые слова, текст и служебные символы. Если простота алгоритма сжатия не является основным требованием, для достижения максимальной степени сжатия крайне желательно устранять разные виды избыточности в данных разных типов по отдельности.

6.3. Построение модели

Для устранения избыточности необходимо построить модель данных. Надо быть готовым к тому, что для данных сложной структуры модель будет сложной. После разбора файла данных на блоки в соответствии с его структурой (блоками могут выступать некоторые фрагменты файлов либо ключевые слова и символы, либо управляющие коды – в зависимости от типа и структуры сжимаемого файла). Далее строят модель вероятности следования одних типов блоков за другими, а также модели для внутренней структуры недетерминированных блоков (например, блоков текста).

6.4. Предсказание

Для правильного предсказания вероятностей необходимо иметь некоторую начальную модель, а также динамически изменять (адаптировать) вероятности по ходу работы алгоритма для подстройки их под сжимаемые данные. Начальная модель не так важна, как может показаться на первый взгляд. Однако для достижения максимальной степени сжатия алгоритм тренируют на реальных данных и полученные в результате его работы вероятности используют в качестве начальной модели. Гораздо более важна адаптация алгоритма. Для каждого типа данных существует определенный оптимальный уровень адаптивности (скорости изменения вероятностей под воздействием тех или иных данных). Использование большего или меньшего уровня адаптивности резко понижает адекватность модели, точность предсказания и степень сжатия.

6.5. Кодирование

Как было сказано выше, префиксное кодирование имеет смысл, только если необходимо достичь максимально высокого быстродействия. В большинстве же случаев рекомендуется использовать Range-кодирование [20].

6.6. Быстродействие

Результаты экспериментальных исследований показали, что алгоритмы, построенные на статистических моделях, имеют быстродействие на современных персональных компьютерах более 200 кбайт в секунду, что вполне достаточно для большинства приложений.

6.7. Результаты

Для проверки того, правильно ли был построен алгоритм, необходимо сравнить результаты его работы с существующими аналогами. Для универсальных алгоритмов степень сжатия должна быть не хуже, чем у существующих аналогов. Алгоритмы для нестандартных данных (например, специальный алгоритм для сжатия исходных кодов на некотором языке или документов некоторого типа) должны превосходить по степени сжатия лучшие существующие аналоги не менее чем на 10-20%.

6.8. Примеры

С использованием данного подхода были разработаны алгоритмы сжатия HTML документов CHTML2, а также алгоритм сжатия смешанной информации для мобильных телефонов NeoPoint Neo2. Разработанные алгоритмы обеспечивают степень сжатия на 10-20% выше, чем все существующие аналоги.

ЗАКЛЮЧЕНИЕ

В данной работе приведена общая теория сжатия информации, описание классов алгоритмов, подробное описание наиболее эффективных из известных на сегодняшний день алгоритмов для сжатия текстовой и графической информации.

Работа содержит анализ и систематизацию алгоритмов сжатия и их составных частей.

Приводится и обосновывается новый взгляд на преобразования и моделирование как на средства приведения одних видов избыточности к другим, что позволяет правильно выбирать модели и преобразования для различных типов данных и правильно их использовать.

Предлагается хорошо зарекомендовавший себя на практике универсальный формализованный подход для построения новых алгоритмов сжатия информации. На основе использования этого подхода разработан и исследован ряд эффективных алгоритмов сжатия информации.

Материалы препринта могут быть полезны аспирантам, научным сотрудникам и специалистам в области сжатия информации, а также в смежных областях.

ЛИТЕРАТУРА

1. Shannon C.E.. Prediction and entropy of printed English. Bell System Technical Journal, 1951, pp.50-64.

2. Колмогоров А.Н. Три подхода к определению понятия "количество информации"// Проблемы передачи информации. - 1965 - №1. - С.3-11.

3. Shannon C.E. A mathematical theory of communication. Bell Syst.Tech.J.27, Jul. 1948, pp.398-403.

4. Bell T., Witten I.H., Cleary J.G. Modeling For Text Compression. ACM Computing Surveys. vol.21, No.4, Dec. 1989, pp.557-591.

5. Nelson M. The data compression book. Henry Holt & Co Inc, 1991.

6. Terry A. Welch. A Technique for High-Peformance Data Compression, Computer, 17, № 6, 1984, pp. 8-19.

7. Hirschberg D., Lelewer D. Data compression, Computing Surveys, 19,3, 1987, pp.261-297.

8. Ziv J., Lempel A., “Compression of individual sequences via Variable-Rate Coding”, IEEE Trans. Information Theory, vol. IT-24 No. 5, Sept. 1978, pp. 5306.

9. Fiala E.R. and Greene D.H. Data compression with finite windows. Commun.ACM, 32,4, Apr. 1989, pp.490-505.

10. De la Cruz et al. A high performance block compression algorithm for small systems. Software and hardware implementations, DCC’95, Utah, USA, 1995.

11. Teahan W.J., Cleary J.G. The entropy of English using PPM-based models, Proceeding DDC'96, IEEE Society Press, 1996, pp.53-62.

12. Teahan W.J. Probability estimation for PPM, In proc. of the N.Z. comp.sci.research Students' Conf., Univ. of Waikato, Hamilton, Mew Zenland, 1995.

13. Cleary J.G., Teahan W.J., Witten I.H. Unbounded length contexts for PPM, In J.A. Storer and M. Cohn, editors, Proceedings DDC'95. IEEE Computer Sosiety Pres, 1995.

14. Teahan W.J. and Cleary J.G. Adaptive models of English text, Tech. Report, Dept. of Computer Science, University of Waikato, 1998.

15. Cormack G.V., Horspool R.N. Data compression using dynamic Markov modelling. Comput. J. 30,6 Dec. 1987, pp.541-550.

16. Burrows M., Wheeler D.J., A Block-sorting Lossless Data Compression Algorithm, Digital Systems Research Center Research Report 124, May 1994.

17. Fenwick P. Block Sorting Text Compression, Proceedings of the

19th Australasian Computer Science Conference, Melbourne, Australia, Jan 31 - Feb 2, 1996.

18. Nelson M. Data Compression with the Burrows-Wheeler Transform. Dr. Dobb's Journal, September, 1996.

19. Howard P.G., Vitter J.S. Design and Analysis of Fast Text Compression Based on Quasi-Arithmetic Coding, 1993 DCC, Snowbird, Utah, Mar-1993, pp.98-107.

20. Martin G. N. Range encoding: an algorithm for removing redundancy from a digitised message. Video & Data Recording Conference, Southampton July 1979.

21. Georgey K. Wallace The JPEG still picture compression standard, IEEE Transaction on Consumer Electronics, December, 1991.

22. 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, June 1996, pp.234-250.

23. Абламейко С.В., Лагуновский Д.М. Обработка изображений: технология, методы, применение. - Минск: Ин-т техн. кибернетики НАН Беларуси, 1999.

24 Романов В.Ю. Популярные форматы файлов для хранения графических изображений на IBM PC.- М.:Унитех, 1992.

25. Howard P. G., Vitter J. S, Fast and efficient lossless image compression, Proc. of Data Compression Conference, 1993, pp.351-360.

26. Howard P. G., Vitter J. S., Fast Progressive Lossless Image Compression, Image and Video Compression Conference, Symposium on Electronic Imaging: Science & Technology SPIE-2186, San Jose, California, Feb. 9-10, 1994, pp.98-109.

27. Weinberger M. J., Seroussi G., Shapiro G., LOCO-I: A low complexity, context-based, lossless image compression algorithm, in Proc. 1996 DCC, (Snowbird, Utah, USA), Mar. 1996, pp. 140-149.

28. Xiaolin Wu. Lossless Compression of Continuous-tone Images via Context Selection, Quantization, and Modeling, IEEE Trans. on Image Processing, vol. 6, no. 5, May 1997, pp. 656-664.

29. Xiaolin Wu. An Algorithmic Study on Lossless Image Compression. Data Compression Conference 1996, pp.150-159

30. Tabus I, Rissanen J, Astola J. Adaptive L-predictors based on finite state machine context selection. In Proc. ICIP'97 International Conference on Image Processing, Santa Barbara, California, Oct. 1997, pp 401-404.

31. Tabus I., Astola J. Adaptive Boolean predictive modelling with application to lossless image coding. In SPIE - Statistical and Stochastic Methods for Image Processing II, San Diego, California, Jul. 1997,

pp.234-245.

32. Howard P.G., Vitter J.S. Error Modeling for Hierarchical Lossless Image Compression, Proceedings of the 1992 IEEE Data Compression Conference (DCC '92), Snowbird, UT, March 1992, pp.269-278.

33. Claypoole R.L., Davis G.M. Nonlinear Wavelet Transforms for Image Coding via Lifting. IEEE Trans. On Image Processing, Aug. 1999.

34. Ingrid Daubechies, Wim Sweldens. Factoring Wavelet Transforms into Lifting Steps J. Fourier Anal. Appl., vol. 4, Nr. 3, 1998, pp.247-269.

35. Wim Sweldens The Lifting Scheme: A new philosophy in biorthogonal wavelet constructions. Wavelet Applications in Signal and Image Processing III, Proc. SPIE 2569, 1995, pp. 68-79.

36. Процкевич А.А., Яковишин В.С. Форма хранения и передачи кодированных знаний.// Комплексная защита информации - Вып. 2 – Минск: Ин-т техн. кибернетики НАН Беларуси, 1999. - С.95-103.

37. Nevill-Manning C.G., Witten I.H. Compression and explanation using hierarchical grammars. Computer Journal, 40(2/3) 1997, pp.103-116.

ОГЛАВЛЕНИЕ