На правах рукописи

РАТУШНЯК Олег Александрович

Методы сжатия данных без потерь с помощью сортировки параллельных блоков

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей.

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Красноярск — 2002

Работа выполнена в Институте систем информатики им.А.П.Ершова Сибирского отделения Российской академии наук

Научный руководитель: доктор физико-математических наук, профессор А.Г.Марчук

Официальные оппоненты: доктор физико-математических наук, профессор Носков М.В.

кандидат физико-математических наук Мурзин Ф.А.

Ведущая организация: Институт автоматики и электрометрии СО РАН

Защита состоится 2 июля 2002 г. в 14 часов на заседании диссертационного совета Д 212.098.03 при Красноярском государственном техническом университете по адресу: 660074, г. Красноярск, ул. акад. Киренского, 26.

С диссертацией можно ознакомиться в читальном зале библиотеки Красноярского государственного технического университета.

Автореферат разослан "__" июня 2002 г.

Ученый секретарь
диссертационного совета
кандидат технических наук Е.А.Вейсов

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. В течение последних двадцати лет наблюдается стремительное увеличение количества используемых компьютеров, сопровождаемое неуклонным ростом их возможностей. Это привело к значительному расширению как круга пользователей ЭВМ, так и сферы применения компьютеров. Одновременное развитие телекоммуникационных систем, рост пропускной способности и общего количества линий связи, появление и развитие глобальных компьютерных сетей, в настоящее время доступных сотням миллионов пользователей, вызвали быстрое увеличение объемов хранимой и передаваемой информации.

В связи с продолжающимся ростом глобальных сетей и расширением спектра предоставляемых ими услуг при экспоненциальном росте числа пользователей сетей, а также успешно выполняемыми проектами по переводу в цифровой вид содержимого огромных хранилищ информации, в частности библиотек, выставочных залов и художественных галерей, следует ожидать, что объемы хранимой и передаваемой по системам связи информации будут продолжать увеличиваться с такой же или даже еще большей скоростью.

Таким образом, задача хранения и передачи текстовой, графической, звуковой и другой информации в наиболее компактном виде достаточно актуальна.

Цель работы. Цель данной работы заключается в разработке высокоэффективных методов и алгоритмов неискажающего сжатия данных (в первую очередь графических), которые одновременно с высокой скоростью обладают и высоким качеством сжатия.

Особое значение придавалось достижению сравнительно высокой средней скорости работы алгоритма сжатия на типичных изображениях, которая имеет наибольшее практическое значение, при приемлемой скорости сжатия в наихудшем случае.

Качеству сжатия также уделялось пристальное внимание. В некоторых случаях допускалось ухудшение качества на 0.5-1.5%, если это ускоряло алгоритм в 1.5-2 раза.

Задачи исследования. В данной работе рассматривались только методы сжатия, допускающие эффективную программную реализацию на последовательных ЭВМ, поскольку в настоящее время параллельные компьютеры

встречаются достаточно редко, а область применения аппаратных решений очень узкая.
Основные усилия были направлены на изучение, анализ и дальнейшее развитие основных методов, используемых для сжатия изображений: методов первичной обработки, контекстной обработки, методов обхода плоскости, методов сжатия одномерных данных без контекстной корреляции.
Особое внимание уделено методам сортировки, разработанным автором и описываемым впервые:

- сортировка параллельных блоков;
- фрагментирование.

Сортировка производится для уменьшения контекстной корреляции и в общем случае зависит только от размера данных, но не от их содержания.

Методы исследования. В процессе исследований были использованы основные положения теории информации, теории кодирования дискретных источников сообщений, комбинаторики, дискретной математики, теории чисел, методы динамического программирования, математического анализа и теории асимптотических функций.

Научная новизна результатов работы. Автором разработаны и исследованы методы решения описанных задач и алгоритмы их реализации.
Как правило, при разработке алгоритмов использовался тот факт, что длина сжимаемых данных естественным образом ограничена сверху: например, ввиду ограниченных объемов носителей информации длина кодируемой последовательности заведомо меньше $2^{64} \approx 1.6*10^{19}$.
Предложенные алгоритмы не обязательно являются асимптотически оптимальными, однако близки к таковым с практической точки зрения, обладая при этом рядом преимуществ.
На защиту выносятся следующие результаты.

1. Разработаны новая система классификации методов сжатия и карта групп методов сжатия, иллюстрирующая эту систему. Классификации и карта групп позволяют лучше понимать конкретные методы сжатия и приблизительные границы областей их применимости.
2. Разработан и исследован новый метод (обратимой) сортировки данных с целью уменьшения контекстной избыточности: сортировка параллельных блоков. Показано, что все известные методы сортировки, в том числе полная (BWT) и частичная (ST) яв-

ляются частными случаями сортировки параллельных блоков (PBS).

3. Разработан и исследован новый метод фрагментирования, позволяющий существенно (на 5-20%) улучшить степень сжатия неоднородных данных. Метод может использоваться не только для сжатия таких данных, но и для анализа их структуры.
4. Разработаны и исследованы новые варианты улучшения сжатия хорошо известных методов: разделения экспонент и мантисс (SEM), линейно-предсказывающего кодирования (LPC), субполосного кодирования (SC), методов обхода плоскости. Показано, как эти методы (а в перспективе и некоторые другие, в частности векторное квантование (VQ) и нумерующее кодирование (ENUC)) могут быть применены для сжатия изображений без потерь.
5. Разработано и исследовано новое семейство методов сжатия изображений без потерь с помощью сортировки параллельных блоков (ERI97). Совершенствование методов продолжается, но уже сейчас их практическая реализация превосходит все известные аналоги как по степени сжатия, так и по эфективности, на широком классе изображений.

Практическая ценность результатов. Все разработанные новые методы и новые варианты существующих методов позволяют существенно улучшить как степень сжатия данных, на которые они ориентированы, так и общую эффективность сжатия таких данных современными вычислительными машинами, в среднем на 5…20% по сравнению с лучшими из существующих в настоящее время аналогов, и на 10…90% по сравнению с современными промышленными стандартами (в первую очередь ZIP и PNG).

Реализация и внедрение результатов. Большинство алгоритмов и методов сжатия, описанных в данной работе, реализованы автором в компьютерной программе сжатия данных ERI (около 10 000 строк на языке Си).
Программа создавалась в основном с целью совершенствования алгоритмов сжатия изображений без потерь, поэтому именно в таком сжатии она уже сейчас является одной из лучших в своем классе, опережая по степени сжатия и общей эффективности сжатия лучшие из известных аналогов на 2…25% (на малошумных изображениях, достаточно больших, см. Приложения 1-4).
Совершенствование методов, алгоритмов и программы продолжается.

Апробация работы и публикации. Результаты работы были опубликованы автором в местной и центральной печати [1-6] (все публикации, кроме первой — без соавторов).
Работа была апробирована на семинарах Института систем информатики им. А. П. Ершова СО РАН и Красноярского государственного технического университета.
Некоторые результаты, в частности самые важные, были опубликованы (на английском языке) в Интернете, в группе новостей comp.compression, регулярно читаемой практически всеми специалистами в области сжатия данных. Выпуски сравнительных тестов, выполняемых автором, «Art Of Lossless Data Compression» публикуются в Интернете более четырех лет, с конца 1997-го года (http://go.to/artest , http://artst.narod.ru/) и являются лучшими по многим параметрам, в частности по числу и общему объему файлов, на которых производится тестирование программ сжатия, доступности и читабельности наборов команд, тестирующих компрессоры (файлы скриптов: *.BAT) , и другим параметрам.
Результаты работы используются в книге «Методы сжатия данных» (авторы – Д.Ватолин, А.Ратушняк, М.Смирнов, В.Юкин), публикуемой в Москве издательством «Диалог-МИФИ» в мае 2002-го года, электронный вариант книги можно увидеть в Интернете по адресу http://compression.graphicon.ru/ и http://compression.graphicon.ru/tmp/.

Структура работы. Диссертационная работа состоит из вводной части, пяти глав и четырёх приложений.
Первая часть (вводная) содержит все используемые в работе определения и аббревиатуры, а также систему классификации методов сжатия и карту групп методов сжатия, иллюстрирующую эту систему. По мнению автора и многих ведущих специалистов в области сжатия данных (в частности, Д.Ватолина, М.Смирнова и В.Юкина), классификации и карта групп позволяют лучше понимать конкретные методы сжатия и приблизительные границы областей их применимости, причем система автора полнее, чем все другие наборы классификаций. В частности, хорошо видно, почему методы, синтезирующие LZ+BWT или LZ+PPM, эффективны и находят применение, а синтезирующие PPM+BWT теоретически возможны, но на практике неэффективны.
Других систем классификаций и карт групп методов сжатия в настоящее время либо не существует, либо они автору не известны.

Главы 1-4 работы посвящены описанию новых разработанных автором методов, а также новых вариантов существующих методов.
В каждом случае изложение метода делается поступательно: от самого простого варианта к достаточно сложному.
Пятая глава работы содержит подробное описание нового разработанного автором семейства методов сжатия изображений без потерь ERI97, а также компьютерной программы ERI, реализующей некоторые варианты алгоритмов этих методов.
В приложениях даны результаты сравнения работы программы ERI с другими программами, сжимающими полноцветные изображения: лучшие из изветных программ, а также программы, реализующие промышленные стандарты (PNG, LOCO-I/JPEG-LS, baseline JPEG). Сравнение с алгоритмом JPEG, сжимающим изображения с потерями, приведено только для общего сведения.
Работа заканчивается списком использованной литературы.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении содержатся краткие сведения об используемых терминах, существующих методах сжатия, критериях их оценки и т. д. Вся терминология была разработана автором и в полном виде вошла в книгу «Методы сжатия данных». Терминов, необходимых для понимания классификации и карты групп, относительно немного.
Блок — конечная последовательность цифровой информации.
Поток — последовательность с неизвестными границами: данные поступают маленькими блоками, и нужно обрабатывать их сразу, не накапливая. Блок — последовательность с произвольным доступом, а поток — с последовательным.
R-битный элемент — совокупность R битов — имеет 2^R возможных значений-состояний. Большинство источников цифровой информации порождает элементы одного **размера** R. А в большинстве остальных случаев — элементы нескольких размеров: R_1, R_2, R_3... (например, 8, 16 и 32).
Конечная последовательность элементов называется **словом**, а количество элементов в слове — **длиной слова**.
Сжатием блока называется такое его описание, при котором создаваемый **сжатый** блок содержит меньше битов, чем исходный, но по нему возможно

однозначное восстановление каждого бита исходного блока. Обратный процесс, восстановление по описанию, называется **разжатием**.

Источник данных порождает поток либо содержит блок данных. Вероятности порождения элементов определяются состоянием источника. У источника данных **без памяти** состояние одно, у источника **с памятью** — множество состояний, и вероятности перехода из одного состояния в другое зависят от совокупности предыдущих и последующих (еще не реализованных, в случае потока) состояний.

Можно говорить, что источник без памяти порождает *"элементы"*, а источник данных с памятью — *"слова"*, поскольку во втором случае

- учет значений соседних элементов (**контекста**) улучшает сжатие, то есть имеет смысл трактовать данные как слова;
- поток данных выглядит как поток слов.

В первом же случае имеем дело с перестановкой элементов, и рассматривать данные как слова нет смысла.

Кавычки показывают, что это условные названия способов интерпретации входных данных: *"слова", "элементы", "биты"*.

По традиции бинарный источник без памяти называют обычно «**источник Бернулли**», а важнейшим частным случаем источника данных с памятью является «**источник Маркова**» (N-го порядка): состояние на i-ом шаге зависит от состояний на N предыдущих шагах: i-1, i-2,…, i-N.

Третья важнейшая применяемая при сжатии данных математическая модель — «**аналоговый сигнал**»:

- данные считаются количественными;
- источник данных считается источником Маркова 1-го порядка.

Если использовать модель «аналоговый сигнал» с $N > 1$, то при малых N эффективность сжатия неизменна или незначительно лучше, но метод существенно сложнее, а при дальнейшем увеличении N эффективность резко уменьшается.

Эффективность сжатия учитывает не только **степень сжатия** (отношение длины несжатых данных к длине соответствующих им сжатых данных), но и скорости сжатия и разжатия. Часто пользуются обратной к степени сжатия величиной — **коэффициентом сжатия**, определяемым как отношение длины сжатых данных к длине соответствующих им несжатых.

Еще две важных характеристики алгоритма сжатия — объемы памяти, необходимые для сжатия и для разжатия (для хранения данных, создаваемых и/или используемых алгоритмом).

Карта групп методов сжатия

	Статистические		Трансформирующие	
	Поточные	Блочные	Поточные	Блочные
для *"слов"*, модель «Источник с памятью»	CM, DMC, все PPM	CMBZ, pre-conditioned PPMZ	Все LZ, в т.ч. LZH и LZW	ST, в т.ч. BWT
для *"элементов"*, модели «Источник без памяти» или «Аналоговый сигнал»	адаптивный HUFF	статический HUFF	SEM, VQ, MTF, DC, SC, DWT	DCT, FT, Фрактальные методы
для *"элементов"* или *"битов"*	адаптивный ARIC	статический ARIC	RLE, LPC, в т.ч. Дельта	PBS, ENUC

Каждая группа (ветвь, семейство) содержит множество методов. Исключением является блочно-ориентированный CM — это относительно мало исследованная область. Автору не известны другие практические реализации, кроме компрессоров CM Булата Зиганшина и "pre-conditioned PPMZ" Чарльза Блума.

Статистические методы оперируют величинами вероятностей элементов напрямую (или величинами относительных частот, что по сути то же самое), а трансформирующие используют статистические свойства данных опосредованно. Есть и методы смешанного типа, но их меньше.

Все поточные методы применимы и к блокам, но обратное неверно. Блочные методы неприменимы к потокам, поскольку не могут начать выполнение, пока не задана длина блока, заполненного данными, подлежащими сжатию.

В первой строке «карты групп» — методы для источников с памятью, порождаемые ими данные выгодно трактовать как слова. Однако методы для потоков *"слов"* оперируют, как правило, элементами заданного размера, а не словами, поскольку разбиение потока элементов на слова заранее в общем случае неизвестно.

Во второй строке — методы для источников без памяти и аналоговых сигналов. Эти данные при сжатии невыгодно рассматривать как слова.

Не все методы для потоков R-битных *"элементов"* применимы к *"битам"* (только те, которые в третьей строке «карты»).

Очевидно, что невыгодно применять методы для *"элементов"* — к *"словам"* или *"битам"*. Менее очевидно, что невыгодно и обратное: применять методы для потоков *"слов"* к данным без значимых вероятностных взаимосвязей, к *"элементам"* или *"битам"*.
Все **названия и аббревиатуры методов** содержатся в тексте работы.

Базовых стратегий сжатия три:

1) Трансформация потока («Скользящее окно-словарь»).

Описание поступающих данных через уже обработанные. Сюда входят LZ-методы для потоков *"слов"*, т.е. когда комбинации поступающих элементов предсказуемы по уже обработанным комбинациям. Преобразование по таблице, RLE, LPC, DC, MTF, VQ, SEM, Subband Coding, Discrete Wavelet Transform, — для потоков *"элементов"*, т.е. когда не имеет смысла рассматривать комбинации длиной два и более элемента, или запоминать эти комбинации, как в случае Linear Prediction Coding.

Никаких вероятностей, в отличие от второй стратегии, не вычисляется. В результате трансформации может быть сформировано несколько потоков. Даже если суммарный объем потоков увеличивается, их структура улучшается, и последующее сжатие можно осуществить проще, быстрее и лучше.

2) Статистическая стратегия.

а) Адаптивная (поточная).

Вычисление вероятностей для поступающих данных на основании статистики по уже обработанным данным. Кодирование с использованием этих вычисленных вероятностей. Семейство PPM-методов — для потоков *"слов"*, адаптивные варианты методов Хаффмана и Шеннона-Фано, арифметического кодирования — для потоков *"элементов"*. В отличие от первого случая, давно собранная статистика имеет тот же вес, что и недавняя, если метод не борется с этим специально, что гораздо сложнее, чем в случае LZ. Кроме того, считаются вероятными все комбинации, даже те, которые еще не встречались в потоке, и, скорее всего, никогда не встретятся.

б) Блочная.

Отдельно кодируется и добавляется к сжатому блоку его статистика. Статические варианты методов Хаффмана, Шеннона-Фано и Арифметического Кодирования — для потоков *"элементов"*. Статическое СМ для *"слов"*.

3) Трансформация блока.

Входящие данные разбиваются на блоки, которые затем трансформируются целиком, а в случае блока однородных данных лучше брать весь блок, который требуется сжать. Это методы сортировки блоков (“BlockSorting”-методы: ST, BWT, PBS), а также Fourier Transform, Discrete Cosine Transform, фрактальные преобразования, Enumerative Coding.

Как и при первой стратегии, в результате могут формироваться несколько блоков, а не один. Опять же, даже если суммарная длина блоков не уменьшается, их структура значительно улучшается, и последующее сжатие происходит проще, быстрее и лучше.

Резюмируя одним предложением: метод сжатия может быть или статистическим, или трансформирующим, и обрабатывать данные либо поточно, либо блоками, причем

- чем больше и однороднее данные и память, тем эффективнее блочные методы;
- чем меньше и неоднороднее данные и память, тем эффективней поточные методы;
- чем сложнее источник, тем сильнее улучшит сжатие оптимальная трансформация;
- чем проще источник, тем эффективней прямолинейное статистическое решение (математические модели «источник Бернулли» и «источник Маркова»).

В первой главе рассматривается кодирование источников данных типа «аналоговый сигнал»: Линейно-предсказывающее кодирование и Субполосное кодирование. Описаны в основном те варианты методов, которые показали высокую эффективность на практике. Большинство из них разработано автором.

Метод LPC применяется обычно для сжатия аналоговых сигналов. И как правило, каждый элемент сигнала отклоняется от своего предсказываемого значения не только из-за «сильных» обусловленных изменений — эволюции, но и из-за «слабых» фоновых колебаний, то есть шума. Поэтому возможно **два противоположных типа моделей**:

- вклад шума невелик по сравнению с вкладом эволюции;
- вклад эволюции невелик по сравнению с вкладом шума.

В первом случае мы будем предсказывать значение $S[i]$ на основании сложившейся линейной тенденции, во втором — как равное среднему арифметическому h предыдущих элементов.

Если ресурсов достаточно, можно вычислять предсказываемые значения, даваемые несколькими моделями, и затем либо синтезировать эти не-

сколько предсказаний, либо выбирать (через каждые P шагов) ту модель, которая оптимальна на заданном числе предыдущих элементов.

Требуется минимизировать ошибки предсказаний, поэтому применим такой критерий: оптимальна та модель, которая дает наименьшую сумму абсолютных значений этих ошибок:

$$\min(\sum_i | D_i |)$$

В алгоритме PNG для сжатия изображения можно выбрать одну из следующих LPC-моделей (называемых также «фильтрами»):

1) E=0 (нет фильтра)
2) E=D (элемент слева)
3) E=B (элемент сверху)
4) E=(B+D)/2
5) алгоритм Paeth

Все пять — варианты шумовой модели, и только последняя модель является комбинированной, учитывающей эволюцию.

В алгоритме Lossless JPEG может быть задан один из восьми предсказателей-фильтров:

1) E=0 (нет фильтра)
2) E=B (элемент сверху)
3) E=D (элемент слева)
4) E=A (выше и левее)
5) E=B+D–A
6) E=B+(D–A)/2
7) E=D+(B–A)/2
8) E=(B+D)/2

Пятый, шестой и седьмой — варианты эволюционной модели, остальные — шумовой.

Выбор фильтра в общем случае. Имеем множество фильтров $Sp_n[X,Y]=S_n(K_{i,j}\cdot S[X–i,Y–j])$ дающих оценки-предсказания Sp_n для значения элемента в позиции (X,Y) как функцию S_n от значений элементов контекста, то есть элементов, соседних с текущим $S[X,Y]$.

Два самых очевидных пути дальнейших действий:

1. выбрать одно значение Sp_n , даваемое той функцией S_n , которая точнее на заданном числе предыдущих элементов,

2. выбрать значение, вычисляемое как сумма всех Sp_n , взятых с разными весами: $S_p = \Sigma(W_n \cdot Sp_n)$, $0 \le W_n \le 1$.

Кроме довольно очевидного способа — корректировки W_n, весов, с которыми берутся предсказания, даваемые разными фильтрами, есть и еще варианты:

3. брать те K значений Sp_n, которые дают S_n, лучшие на K (задаваемом числе) ближайших элементов контекста (некоторые из этих S_n могут совпадать, и их останется меньше, чем K).
4. все функции S_n сортируются по значению точности предсказания на ближайших элементах контекста и выбирается несколько функций, дающих лучшую точность.

Четвертый вариант является простым расширением первого. Третий же предполагает сопоставление каждому элементу одной S_n, дающей для него самое точное предсказание. В обоих случаях, если требуется выбрать одну S_n из нескольких S_i с одинаковым значением точности, можно посмотреть значения точности этих S_i на большем числе элементов.

Во 2-м, 3-м и 4-м часто полезно уменьшить шум описанным в работе способом. И в конечном итоге сложить оставшиеся $S_p = \Sigma(W_n \cdot Sp_n)$. В простейшем случае $W_n = 1/H$, где H — количество оставшихся S_n.

Субполосное кодирование. Цель метода — сжатие потока R-битных элементов, в предположении, что значение каждого из них отличается от значений соседних элементов незначительно: $S_i \approx S_{i-1}$.

Основная идея состоит в том, чтобы формировать два потока: для каждой пары S_{2i} , S_{2i+1} сохранять полусумму $(S_{2i} + S_{2i+1})/2$ и разность $(S_{2i} - S_{2i+1})$. Далее эти потоки следует обрабатывать раздельно, поскольку их характеристики существенно различны.

К получаемым потокам можно и дальше рекурсивно применять разложение на полусуммы (Average) и разности (Delta). При сжатии аналоговых сигналов, как правило, полезно дальнейшее разложение полусумм.

Есть два пути создавать три выходных потока или блока:

1. Original↓

   Average1+Delta1↓

   DA+DD

(**DA** — Average2, получаемая при разложении разностей Delta1, **DD** — Delta2, получаемая при разложении Delta1. Аналогично с остальными обозначениями).

2. Original↓

   Average1↓+**Delta1**

AA+AD

Пять вариантов создания четырех, 14 вариантов создания пяти, и так далее. При принятии решения о целесообразности разложения некоторого потока или блока S на A и D, оказывается полезен следующий прием. Будем заглядывать на несколько шагов вперед: если непосредственные результаты разложения A и D сжимаемы суммарно хуже, чем исходный S, может оказаться, что если разложить дальше A и/или D, то только тогда результат — три или четыре потока, по которым восстановим S, — сжимаем лучше, чем S. Если же и эти AA, AD, DA, DD дают в сумме худшее сжатие, заглянем еще на шаг вперед, то есть попытаемся разложить эти вторичные результаты, и так далее, пока глубина такого просмотра не достигнет заданного предела, или же дальнейшее разложение окажется невозможным из-за уменьшения длин в два раза на каждом шаге.

Во второй главе обсуждаются относительно алгоритмически простые методы обхода плоскости:

- змейка (зигзаг-сканирование),
- обход строками и столбцами, в т.ч. с разворотами,
- полосами, в т.ч. с разворотами,
- решетками, в т.ч. с учетом значений элементов,
- контурный обход, в т.ч. с неизвестными контурами,
- другие методы (по спирали, квадратная змейка).

Задача обхода плоскости возникает при обработке двухмерных данных. Самым рапространенным видом таких данных являются изображения.
Цель метода: создание одномерного массива D из двухмерного S. Причем если предполагается последующее сжатие D, то желательно создавать его так, чтобы «разрывов» было как можно меньше: каждый следующий элемент d_i, заносимый в D на i–ом шаге, является соседним (в плоскости) для предыдущего, занесенного в D на (i-1)–ом шаге, d_{i-1}.
В любом случае можно начать с одного из четырех углов, и дальше двигаться в одном из двух направлений: по вертикали и по горизонтали. Первое, то есть положение первого угла, влияния на степень сжатия почти не оказывает, особенно при сжатии изображений. А вот второе, выбор направления, может существенно улучшить сжатие, поскольку области в этих двух случаях (основное направление по вертикали или по горизонтали) будут сгруппированы по-разному.

Например, отсканированный текст лучше сжимать, обходя по горизонтали, поскольку в нем больше длинных сплошных горизонтальных линий, чем.вертикальных.
В случае контура сложной формы может возникнуть необходимость помечать уже обработанные точки плоскости, чтобы избежать лишних вычислений, предотвращающих повторное их занесение в D. Тогда есть два основных варианта: либо добавить по одному «флаговому» биту для каждой точки плоскости, либо выбрать (или добавить) значение для «флага», показывающего, что точка уже внесена в D, и записывать это значение на место уже внесенных точек.
Материалы этой главы, так же как изложенные в первой и третьей главах, существенно используются в пятой, при описании семейства методов сжатия изображений без потерь ERI97.

В третьей главе описываются методы сортирующих преобразований: Сортировка параллельных блоков и Фрагментирование. Рассматриваются как простейшие «демонстрационные» варианты, так и достаточно сложные, более эффективные в некоторых важных частных случаях. Весь материал этой главы разработан и впервые описывается автором.
Два блока A и B называются параллельными, если каждому элементу A[i] первого блока поставлен в соответствие один элемент B[i] второго блока, и наоборот. Длины блоков L_A и L_B равны: $L_A = L_B = L$. Размеры элементов блоков R_A и R_B могут быть разными.
Основная идея метода PBS состоит в сортировке элементов In[i] входного блока In и их раскладывании в несколько выходных блоков Out_j на основании атрибутов A[i] этих элементов. Атрибут A[i] есть значение функции A, определяемой значениями предшествующих элементов In[j] и/или элементов P[k] из параллельного блока P:

```
A[i]=A(i, In[j], P[k]), i=0...L-1; j=0,...,i-1; k=0,...,L-1
```

При декодировании осуществляется обратное преобразование: элементы из нескольких блоков Out_j собираются в один результирующий, соответствующий несжатому блоку In.
Чем лучше значения In[i] предсказуемы по значениям A[i], тем эффективнее последующее сжатие блоков Out_j с помощью простых универсальных методов.
В простейшем случае сортирующего преобразования ST(1) значение атрибута вычисляется так: A[i]=In[i-1] ; при ST(2): A[i]=In[i-1]$\cdot 2^R$ + In[i-2] , и так далее.

В простейшем же случае PBS A[*i*]=P[*i*], то есть значение атрибута равно значению элемента из параллельного блока. Выходных блоков столько, сколько значений принимают P[*i*]. Делая проход по P, считаем частоты значений атрибута, и в результате получаем размеры сортированных блоков (далее «контейнеров»), в которые будем раскладывать элементы из входного блока In:

```
for( a=0; a<n; a++) Attr[a]=0;  //инициализация (1)
for( i=0; i<L; i++) Attr[P[i]]++;   //подсчет частот    (2)
```

In — входной сортируемый блок длины L;
P — параллельный блок длины L;
L — количество элементов во входном блоке;
$n=2^R$ — число всех возможных значений атрибута;
Attr[n] — частоты значений атрибута.

Например, в P содержатся старшие 8 битов массива 16-битных чисел, а в In — младшие 8. Заметим, что этот процесс «дробления» можно продолжать и дальше, создавая вплоть до 16-и параллельных блоков: содержащий первые биты, вторые, и так далее. Кроме того, при сжатии мультимедийных данных часто уже имеются параллельные блоки: например, левый и правый каналы стереозвука, красная, зеленая и синяя компоненты изображения, и т.п.
Если функция A задана иначе, то и при подсчете частот формула будет иной:

```
// A[i]=P[i]&252 :
for( i=0; i<L; i++) Attr[ P[i]&252 ]++; // (2a)

// A[i]=255-P[i] :
for( i=0; i<L; i++) Attr[ 255-P[i] ]++; // (2b)

// A[i]=(P[i]+P[i-1])/2 :
Attr[ P[0] ]++;              // (2c)
for( i=1; i<L; i++) Attr[ (P[i]+P[i-1])/2 ]++;
```

Итак, после двух циклов — инициализации и подсчета частот —мы получили в массиве Attr длины контейнеров (сортированных блоков). Теперь можно вычислить, где какой контейнер будет начинаться, если их последовательно сцепить в один блок в порядке возрастания значения атрибута:

```
for( a=0, pos=0; a<n; a++) {         // (3)
  tmp=Attr[a]; // длина текущего a-го контейнера
```

```
  Attr[a]=pos; // теперь там адрес-указатель на его начало
         // (указатель на начало свободного места в нем)
  pos+=tmp;// начало следующего — дальше на длину текущего
}
```

В том же массиве Attr[n] теперь адреса-указатели на начала контейнеров. Остается только пройти по входному блоку, раскладывая элементы из него по контейнерам:

```
for(i=0; i<L; i++) Out[Attr[P[i]]++]=In[i];      //(4c)
```

Out — выходной отсортированный блок (sorted, transformed), содержащий все контейнеры. Подробнее:

```
for( i=0; i<L; i++) { // На каждом шаге:
 s=In[i];// берем следующий элемент s из входного блока In,
  a=P[i];// берем его атрибут a из параллельного блока P,
// и адрес свободного места в контейнере, задаваемом этим a;
  pos=Attr[a];
// кладем элемент s в выходной блок Out по этому адресу,
  Out[pos]=s;
  pos++;//теперь в этом контейнере на один элемент больше,
  Attr[a]=pos;//a свободное место — на один элемент дальше.
}
```

Функция A — та же, что и во втором цикле. То есть, для выше рассмотренных примеров:

```
(2a) a=P[i]&252;
(2b) a=255-P[i];
(2c) a=(P[i]+P[i-1])/2;
       //причем i=1...L-1, а когда i=0, a=P[0].
```

Обратное преобразование. Совершенно идентичны первые три цикла: инициализируем, в результате прохода по параллельному блоку находим длины контейнеров, затем определяем адреса начал контейнеров во входном отсортированном блоке In. Именно эти адреса — цель выполнения первых трех циклов. И только в четвертом цикле — наоборот: берем из отсортированного блока с контейнерами In, кладем в единый выходной блок Out:

```
for(i=0;i<L; i++) Out[i]=In[Attr[P[i]]++];      //(4d)
```

In — входной отсортированный блок со всеми контейнерами
Out — создаваемый выходной блок

Подробнее:

```
for( i=0; i<L; i++) {
  a=P[i];
  pos=Attr[a];
//так было при прямом (сжимающем) преобразовании:
//Out[pos]=In[i];
        //(в текущих обозначениях это записывается так:
        //In[pos]=Out[i];)
//а так делаем сейчас, при разжимающем преобразовании:
  Out[i]=In[pos];
  pos++;
  Attr[a]=pos;
}
```

Фрагментирование. Цель: разбиение исходного потока или блока на фрагменты с разными статистическими свойствами. Такое разбиение должно увеличить степень последующего сжатия. В простейшем случае битового потока необходимо находить границы участков с преобладанием нулей, участков с преобладанием единиц, и участков с равномерным распределением нулей и единиц. Если поток символьный, может оказаться выгодным разбить его на фрагменты, отличающиеся распределением вероятностей элементов: например, на фрагменты с русским текстом и фрагменты с английским.

Основная идея состоит в том, чтобы для каждой точки потока X (лежащей между парой соседних элементов) находить значение функции отличия FO(X) предыдущей части потока от последующей. В базовом простейшем варианте используется «скользящее окно» **фиксированной длины**: Z элементов до точки X, и Z элементов после нее.

Иллюстрация (Z=7):

…8536429349586436542 | 9865332 | X | 6564387 | 58676780674389…

Цифрами обозначены элементы потока, вертикальными чертами «|» границы левого и правого окон. X — не элемент потока, а точка между парой элементов, в данном случае между ”2“ и ”6“.

При фиксированной длине окон Z и **одинаковой значимости**, или **весе,** всех элементов внутри окон (независимо от их удаленности от точки X), значение функции отличия может быть легко вычислено на основании разности частот элементов в левом и правом окнах. Это сумма по всем 2^R возможным значениям V_i элементов. Суммируются абсолютные значения раз-

ностей: число элементов с текущим значением V в **левом** окне длиной Z **минус** число элементов с данным значением V в **правом** окне длиной Z. Суммируя 2^R модулей разности, получаем значение FO(X) в данной точке X:

$$FO(X) = \sum_{V=0}^{2^R-1} \left| CountLeft[V] - CountRight[V] \right|,$$

где CountLeft[V] — число элементов со значением V в **левом** окне;
CountRight[V] — число элементов со значением V в **правом** окне.
Видно, что если в обоих окнах элементы одинаковые, то сумма будет стремиться к нулю, если же элементы совершенно разные — к $2 \cdot Z$.
Для приведенного выше примера:

V =	2	3	4	5	6	7	8	9
FO(X) =	+|1-0|	+|2-1|	+|0-1|	+|1-1|	+|1-2|	+|0-1|	+|1-1|	+|1-0|
=6								

После того как все значения FO(X) найдены, остается отсортировать их по возрастанию и выбрать границы по заданному критерию (заданное число границ и/или пока FO(X)>$F_{\min}$).

В четвертой главе рассмотрены некоторые методы сжатия источников данных без памяти: SEM, VQ, ENUC. Показано, что SEM (разделение мантисс и экпонент) является большим семейством методов сжатия, содержащим методы универсального кодирования как частный случай. Обсуждаются четыре основных варианта SEM:

- Фиксированная длина экспоненты — фиксированная длина мантиссы,
- Фиксированная длина экспоненты — переменная длина мантиссы,
- Переменная длина экспоненты — переменная длина мантиссы
- Переменная длина экспоненты — фиксированная длина мантиссы.

Описано, как коды Элиаса и Голомба можно обобщить и задавать с двумя параметрами.
В пятой главе подробно описано семейство методов сжатия изображений без потерь ERI97. Существенно используется материал, изложенный в предыдущих главах.
Основные свойства описываемой группы алгоритмов:

1. Размер получаемых данных равен размеру задаваемых данных плюс A байт служебной информации (A зависит только от параметров, задаваемых алгоритму).
2. Скорость работы ERI97 сравнима со скоростью аналогов – JPEG, PNG, LOCO – но качество достигаемого сжатия – уже сейчас лучше на 10%-70% (не вариант JPEG-а без потерь, а именно JPEG с потерями, но лучшим качеством).
3. Алгоритм состоит из последовательности четырех независимых действий, причем
   - каждое из них работает с каждым пикселом (элементом изображения);
   - любые из них могут быть пропущены при прямой и затем при обратной трансформации;
   - "сжимающая" трансформация (компрессор) осуществляет их в прямом порядке, а "разжимающая" трансформация (декомпрессор) – в обратном.
4. Скорость работы компрессора в общем случае равна скорости декомпрессора и зависит только от размера данных, но не от их содержания: Ск=O(размер).
5. Памяти требуется 2*(размер входных данных)+C.
6. В общем случае отсутствуют операции умножения и деления.
7. Число N компонент каждого пиксела, как и разрядность пикселов R, являются двумя из множества задаваемых параметров, и могут быть произвольными.

Четыре действия, из которых состоит ERI97, компрессор выполняет в следующем порядке:

1. Первичная обработка
2. Контекстная обработка
3. Обход плоскости
4. PBS, Сортировка параллельных блоков

При разжатии эти действия выполняются в обратном порядке.

Первые три действия в том или ином виде присутствуют во всех аналогичных методах. Первые два действия не имеют отношения к PBS и могут совершаться отдельно, практически никак не ориентируясь на последующую PBS. Более того, каждое из них может выполняться параллельно, распадаясь на P процессов, причем возможные значения P – от 1 до S, где S – число пикселов в обрабатываемом блоке. Первое действие настолько тривиально, что в практической реализации легко может быть совмещено со вторым.

Тем не менее, поскольку логически это разные действия, в описании алгоритма они идут отдельно.
Алгоритм был реализован в компьютерной программе ERI в 1997-ом году и активно совершенствуется в течение последних четырех лет.
Основные отличия реализации – программы ERI (последняя версия на текущий момент – 21-е мая 2002-го года – 5.1fre) :

1. Примерно десять внешне задаваемых параметров.
2. В настоящее время поддерживается только режим N=3, R=8, то есть каждый пиксел состоит из трех 8-битных компонент.
3. Байт-ориентированность: операции над битами по возможности сведены к минимуму, что многократно увеличивает скорость работы, но немного ухудшает качество сжатия.

В настоящее время ERI достигает лучших результатов среди всех известных аналогов по сжатию без потерь малошумных 24-битных изображений. Причем как по качеству сжатия, так и по общему суммарному показателю, учитывающему также и скорости сжатия и разжатия.
В приложениях даны результаты сравнения работы программы ERI с другими программами, сжимающими полноцветные изображения: лучшие из известных программ, а также программы, реализующие промышленные стандарты (PNG, LOCO-I/JPEG-LS, baseline JPEG).

Основные результаты и выводы

В работе показано, что

- разработанный автором класс ЛПК-фильтров оказывается на практике выгоднее, чем несколько фильтров, используемых в алгоритмах PNG и LOCO-I и являющихся лишь частными случаями из множества ЛПК-фильтров описываемого класса;
- в случае разложения сжимаемых данных на высокочастотную и низкочастотную компоненты с помощью субполосного кодирования и дискретного вэйвлетного преобразования, возможно не просто рекурсивное применение такого разложения, но и «рекурсия с просмотром на шаг вперед»;
- разработаны и впервые описаны обощенные методы сортировки; оптимизированные варианты методов PBS и фрагментирования существенно лучше «демонстрационных» по скорости и требованиям к объему рабочей памяти.

- широко известные коды Элиаса и Голомба можно обобщить и задавать с двумя параметрами, вследствие чего степень сжатия многих типов данных (этими кодами) существенно улучшится;
- многие методы, не считавшиеся принадлежащими к группе методов «Представление целых чисел» (в частности DAKX исследователя Д.А.Копфа и «структурная модель» Фенвика) реально используют ту же идею (разделение мантисс и экспонент) и принадлежат одному из четырех основных вариантов;
- использование векторного квантования и нумерующего кодирования являются очень перспективными направлениями исследований, в частности в задаче сжатия изображений без потерь;
- достаточно сложные для реализации методы обхода плоскости, по этой причине не описывавшиеся ранее столь подробно, дают увеличение степени сжатия изображений в большом числе случаев;

Основные публикации по теме диссертации

1. Д.Ватолин, А.Ратушняк, М.Смирнов, В.Юкин. Методы сжатия данных. Учебное пособие. // Москва, «Диалог-Мифи» - 2002. 384 стр. http://compression.graphicon.ru
2. Ратушняк О.А. Методы фрагментирования, анализа структуры и сжатия импульсно-сигнальной информации, а также устройство для их реализации. Патент РФ No. 96103325
3. Ратушняк О.А. О сжатии информации. // КомпьютерПресс. - 2000. - №.5 – стр. 160-163.
4. Ратушняк О.А. Сжатие мультимедийной информации. // Hard’n’Soft. - 2001. - №.4 – стр. 78-79.
5. Ратушняк О.А. Высокоэффективные алгоритмы сжатия изображений без потерь. // Сборник материалов 5-й Международной Конференции "Распознавание-2001", Курск 2001– стр. 155-158.
6. Ратушняк О.А. Алгоритмы сжатия изображений без потерь с помощью сортировки параллельных блоков. // Тезисы докладов конференции молодых учёных по математике, мат. моделированию и информатике, Новосибирск, 4-6 декабря 2001 г. – стр. 48-49.