Министерство образования и науки Украины
Кировоградский национальный технический университет

На правах рукописи

ДРЕЕВ АЛЕКСАНДР НИКОЛАЕВИЧ

УДК 621.391 (0.43)

МЕТОДЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ПЕРЕДАЧИ ИНФОРМАЦИИ В КОРПОРАТИВНЫХ СЕТЯХ

05.12.02 – Телекоммуникационные системы и сети

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
Смирнов Алексей Анатольевич
доктор технических наук, профессор

Кировоград – 2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На современном уровне развития телекоммуникационных сетей и систем возникает ряд проблем, связанных непосредственно с обеспечением необходимого уровня качества обслуживания по критериям QoS в соответствии стандартам Международного союза электросвязи, которые приводят к необходимости и актуальности проведения исследований и создания перспективных методов и алгоритмов обработки и передачи информации [24, 27]. Обеспечение качества обслуживания связи по критериям QoS подразумевает использование кодирование, сжатие, шифрование, лучшее распределение нагрузки каналов передачи, распределение нагрузки на вычислительные ресурсы, технологий цифровой обработки сигналов, и другие возможности управления и оптимизации.

Одним из приоритетных направлений является улучшение использования существующих аппаратных решений, что предоставляет повышение качества обслуживания связи по критериям QoS или увеличение пропускной способности канала связи.

Актуальность темы. Высокий уровень развития телекоммуникационных технологий и стремления современного социума к информационному объединению, создаёт условия создания и развития глобальной телекоммуникационной структуры, которая обеспечивала б свободный доступ к информационным ресурсам для каждого желающего.

К сожалению, известные технологические и протокольные решения в этой области, в условиях повышенной интенсивности информационного трафика, не обеспечивают заданных требований. Поэтому множество передовых концепций и технологий (Traffic Engineering (TE), DiffServ, FastReRouting и др.) не могут в полной мере реализовать функции, которые связаны с повышением качества обслуживания и масштабируемости в телекоммуникационных системах (ТКС).

Современный прогресс в отрасли телекоммуникационных технологий привёл к разработке множества методов, предназначенных для обеспечения

качества обслуживания передачи информации. Однако в дополнение к мультисервисным ТКС эти методы часто не имеют достаточной теоретической основы, поэтому описание свойств, преимуществ и недостатков обосновывается преимущественно на практическом опыте использования, что не гарантирует их успешности в условиях широкого спектра телекоммуникационных услуг. Вследствие этого становится актуальным дифференциальный подход к оптимизации использования аппаратных ресурсов в телекоммуникационных системах и сетях. Частичным случаем ТКС являются корпоративные сети, где наблюдаются значительные вариации требований эффективности передачи информации и отдельно выделяются системы со значительным объёмом часто востребованных данных.

В теории информации и передачи данных накоплен значительный теоретический материал и практический опыт. Наиболее значимыми работами в этой области являются исследования зарубежных и отечественных учёных, среди которых Вегешна Ш., Столлингс В., Арипов Н.М., Вишневский В.М., Захаров А.В., Зайченко Ю.П., Назаров А.Н., Конахович Г.Ф., Лемешко А.В., Юдин А.К. и другие. Однако, динамическое развитие телекоммуникационных устройств и средств управления, и также разнообразие программных технологических решений способствуют тому, что постановка задач повышения качества обслуживания передачи данных существенно видоизменяется по причине необходимости учёта новых телекоммуникационных услуг.

Вследствие обозначенного, получают повышенную актуальность вопросы разработки методов повышения эффективности передачи информации. При этом ключевой особенностью таких методов и средств есть адаптация средств управления и протоколов к требованиям и особенностям поведения отдельных видов телекоммуникационных услуг. В частности, динамическое управления объёмом графического и видиоконтента в процессе его доставки конечным получателям. Учёт этих факторов и особенностей телекоммуникационного трафика выходит за рамки существующих методов повышения эффективности передачи информации и требует как их модификации, так и повторного рассмотрения.

Таким образом, тема диссертационной работы, которая связанна с решением **научно-технической задачи** разработки методов повышения эффективности передачи информации в ТКС, актуальна.

Проблема повышения пропускной способности существующих каналов связи без изменений аппаратного обеспечения возникла при бурном росте количества клиентов глобальной сети, при котором обновить аппаратную часть стало затруднительно не только в финансовом ракурсе, но и физически, из-за нехватки человеко-часов. Основными путями уменьшения загруженности каналов связи, в таком случае, становятся более совершенные методы сжатия информации и более совершенные методы маршрутизации. В наше время основным потоком информации в телекоммуникационных сетях и системах остаётся графическая информация. Поэтому даже незначительное улучшения степени сжатия графической информации может значительно увеличить качество обслуживания [27, 29].

Связь работы с научными программами и темами. Диссертационная работа выполнена на кафедре программного обеспечения Кировоградского национального технического университета. Соискатель получил ряд результатов, как участник отдельных этапов исследований при выполнении госбюджетных тем МОН Украины: «Разработка методов повышения оперативности передачи и защиты информации в телекоммуникационных системах» (36Б113, №ДР 0113U003086), «Разработка методов синтеза тестовых моделей поведения программных объектов, повышения оперативности передачи и защиты информации в телекоммуникационных системах» (36Б115, №ДР 0115U003103), и научно-исследовательских роботах «Методы повышения оперативности передачи данных и защиты информации в телекоммуникационной сети» (№ДР 0112U006631), «Разработка методов повышения безопасности телекоммуникационных сетей» (№ДР 0112U006630).

Цель и задача исследования. Целью диссертационного исследования является повышение эффективности передачи графической информации путём прогрессирующего сжатия графической информации и прогнозирования отказов в обслуживании ТКС.

В соответствии поставленной целью для решения научно-технической задачи в диссертационной работе решить следующие взаимосвязанные задачи:

1. Провести анализ и сравнительные исследования методов повышения эффективности передачи информации в корпоративных сетях.

2. Разработать математическую модель передачи одно и многопакетного сообщения в ТКС, которая учитывает возможности и особенности использования промежуточного телекоммуникационного оборудования для дополнительного сжатия графического контента.

3. Усовершенствовать метод прогрессирующего сжатия графической информации, который в отличие от аналогов, обеспечивает приоритетность сохранения контурной информации и имеет возможность регулировки трафика графического контента.

4. Разработать метод предсказания трафика в ТКС, на основе разложения числовой последовательности на гармонические составляющие с некратными частотами и последовательным выделением трендов.

5. Провести исследование результативности разработанного метода повышения эффективности передачи информации в корпоративных сетях, обосновать практические рекомендации к его использованию.

Объект исследования – процесс повышения эффективности передачи информации в ТКС на основе совершенствования методов прогрессирующего сжатия графической информации и предсказания отказов в обслуживании.

Предмет исследования – метод повышения эффективности передачи информации в корпоративных сетях.

Методы исследования. Для решения научно-технической задачи был использован широкий спектр методов. При обосновании направления исследований повышения эффективности передачи данных в ТКС использован системный подход. При разработке прогрессирующего метода сжатия графической информации использованы основные положения теории информации и фрактального анализа, и также методы вейвлет преобразования. При разработке метода прогнозирования трафика сервера ТКС, использованы методы спектрального анализа временных рядов, их аппроксимации и

экстраполяции. Оценка эффективности теоретических и практических результатов была произведена на основе математической статистики.

Научная новизна полученных результатов обеспечена теоретическим обобщением и новым решением научно-технической задачи, которая состоит в разработке методов повышения эффективности передачи дополнительно сжатого графического контента и прогнозирования трафика в ТКС. Соответственно получены следующие **научные результаты:**

1. Впервые разработан метод предсказания трафика телекоммуникационных сетей, который отличается от известных использованием разложения временного ряда на гармонические составляющие с некратными гармониками и последовательным выделением трендов, что позволило регулировать значения вероятности отказов в обслуживании по причине превышения пропускной способности сервера дополнительного сжатия графической информации [24, 34, 25, 36, 41].

2. Усовершенствован метод прогрессирующего сжатия графической информации, который в отличие от известных имеет более качественное отображение контуров графической информации и даёт возможность регулировать трафик графического контента, что позволит повысить оперативность переданных данных в условиях значительной сетевой нагрузки графическим контентом [22, 26, 28, 35, 37, 116].

3. Получила дальнейшее развитие математическая модель процесса передачи сообщений в ТКС, которая, в отличие от известных, учитывает возможности и особенности использования промежуточного телекоммуникационного оборудования для дополнительного сжатия графической информации, что позволило оценить среднее время и дисперсию времени доставки многопакетных сообщений в условиях дополнительного сжатия низкоприоритетного графического трафика [24, 27, 29, 30, 36, 38, 42].

Практическое значение полученных результатов состоит в адаптации систем управления и протоколов к требованиям и особенностей поведения отдельных видов телекоммуникационных услуг для уменьшения времени передачи данных, а также в возможности использования предложенного метода

для разработки программных средств управления передачей данных в корпоративной сети.

1. Разработано специальное программное и математическое обеспечение для моделирования и управления информационными потоками в ТКС, оценки среднего времени доставки информационных пакетов с учётом особенностей механизма прогрессивного сжатия графической информации, произведены оценки оперативности передачи информации в ТКС. Показано, что их использование позволяет на 5% повысить оперативность передачи информационных пакетов в ТКС.

2. Разработано программное обеспечение, которое адаптирует систему кэш-сервера с дополнительным сжатием графической информации в ТКС. Это позволило на 12% повысить детализацию графической информации.

3. Разработанная программно-аппаратная система прогнозирования трафика в ТКС, которая позволила регулировать интенсивность потока графической информации через сервер дополнительного сжатия графической информации и снизить общий трафик графического контента в 1,6 раза.

4. Практическая значимость полученных результатов подтверждается их использованием:

– при усовершенствовании средств уменьшения мультимедийного трафика в сети Интернет сервис-провайдера «Империал», г. Кировоград (акт №351 от 10.01.2015);

– в учебном процессе Кировоградского национального технического университета (акт №24 от 10.09.2014).

5. Практическая значимость диссертационной работы определяется возможностью использования предложенных моделей, методов и способов не только при обеспечении оперативности передачи информации в ТКС, но и компьютерных и информационных управляющих системах и сетях общего и специального назначения.

Личный вклад соискателя. Все новые результаты диссертационной работы автором получены самостоятельно.

В работах, выполненных в соавторстве и опубликованных в изданиях, часть которых входит в перечень ВАК Украины, автору принадлежат: в [22]

проведено исследование неравномерности распределения единичных битов для дополнительного сжатия SPIHT кода, в [23] разработаны алгоритмы и программное обеспечение для исследования пути развёртки на степень энтропийного сжатия цифрового изображения, в [25] разработан метод экстраполяции квазипериодических временных рядов на неравномерной сетке по времени, в [26] частичное участие в формализации методики сравнения, в [27] произведена разработка математической модели, в [29] разработаны алгоритмы и программное обеспечение для автоматизации сбора статистических данных в [30] произведена теоретическая оценка оптимального размера блока кодирования, в [116] поставлено и проведено исследование на существующем программном обеспечении, разработано собственное программное обеспечение для нахождения сравнительных характеристик изображений, в [33] произведена теоретическая оценка ошибок прогнозирования, в [34] проведена проверка метода долгосрочного прогнозирования, в [36] предложен метод снижения нагрузки на телекоммуникационный сервер за счёт кэширования графической информации, в [38] определены условия целесообразности использования системы дополнительного сжатия мультимедийной информации, в [39] получена теоретическая оценка среднего времени доставки многопакетного сообщения, в [40] введена фрактальная размерность для оценки оптимального размера блока для битового арифметического сжатия, в [41] выполнено исследование влияния помех на качество прогнозирования, в [42] выполнено исследование влияния коэффициента сжатия изображения на оперативность их доставки в телекоммуникационной сети.

Апробация результатов диссертационных исследований. Основные результаты диссертационной работы доложены и обговорены на: научно-практических конференциях «Комбінаторні конфігурації та їх застосування» (2010, 2012, 2014, Кировоград, КНТУ); научно-практической конференции посвящённой 80-летию физико-математического факультета КГПУ им. В. Винниченко (2010, Кировоград, КГПУ); научно-технических конференциях «Наукові технології – для захисту повітряного простору» (2011, 2013, Харьков, ХУВС); научно-технической конференции «Дискретна

математика та її застосування у економіко-математичному моделюванні та інформаційних технологіях» (2012, Запорожье, ЗНУ); международной научно-практической конференции «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку» (2013, Харьков, АВВ); всеукраинской научной конференции «Проблеми і перспективи розвитку IT-індустрії» (2013, Харьков, ХНЕУ), V всеукраинской научно-практической конференции «Інформатика та системні науки» (2014, Полтава, ПУЕТ), V международной научно-практической конференции «Інформаційні технології та моделювання в економіці» (2014, Черкасы, ЧНУ), международной научно-практической конференции «Комп'ютерне моделювання в наукоємних технологіях (КМНТ-2014)» (2014, Харьков, ХНУ).

Публикации. По теме диссертации опубликовано 22 научных трудов, среди которых 1 коллективная монография, 8 статей, которые включены в перечень научных специальных издательств Украины (из которых 5 входят в международные наукометрические базы), 1 статья в зарубежном научном издании, и 12 тез докладов и материалов на всеукраинских и международных конференциях.

Структура и объёмы диссертации. Диссертационная работа состоит из введения, четырёх разделов, выводов, списка использованных источников и 5-ти приложений. Общий объём работы составляет 242 страницы, в том числе 123 страницы основного текста, 50 рисунков по тексту, 12 таблиц по тексту, 16 страниц списка использованных источников из 125 наименований и 103 страницы приложений.

РАЗДЕЛ 1
АНАЛИЗ И ИССЛЕДОВАНИЕ МЕТОДОВ УМЕНЬШЕНИЯ ТРАФИКА ГРАФИЧЕСКОЙ ИНФОРМАЦИИ ДЛЯ УВЕЛИЧЕНИЯ ОПЕРАТИВНОСТИ ПЕРЕДАЧИ ДАННЫХ В ТЕЛЕКОММУНИКАЦИОННЫХ СЕТЯХ. ОБОСНОВАНИЯ ВЫБОРА НАПРАВЛЕНИЯ ИССЛЕДОВАНИЯ И ПОСТАНОВКА НАУЧНОЙ ЗАДАЧИ

В данном разделе произведено исследование методов, которые применяются для увеличения оперативности передачи данных в телекоммуникационных системах и сетях. Также обоснованы критерии оценки их эффективности в повышении оперативности в телекоммуникационной сети. Проведён сравнительный анализ доступных критериев управления данными в сети, и на основе этого анализа сделан выбор наиболее значимых критериев. Сформулирована задача повышения оперативности передачи сообщений в телекоммуникационной сети за счёт дополнительного сжатия менее значимой графической информации.

1.1 Анализ и сравнительное исследование перспективных направлений развития телекоммуникационных систем и сетей

На современном этапе развития телекоммуникационные системы развились в сложную не структурированную сеть, в которой передаётся информация при помощи коммутаторов и маршрутизаторов между серверами и клиентами [14, 15, 20, 46]. Термином «трафик» обозначают количество переданной информации или количество пакетов за единицу времени. Это характеризует интенсивность использования сегмента сети [44]. В зависимости от программного и аппаратного обеспечения на участке телекоммуникационной сети является ограничения на трафик, которое называют пропускной способностью [45]. При интенсивном использовании сети, для гарантии

доставки информации используют алгоритмы перенаправления потоков информации по более долгим или медленным, но менее нагруженным маршрутам [12, 54, 55, 57]. Это приводит к снижению оперативности доставки информации, поэтому вводят алгоритмы сортировки трафика для выделения критических по времени доставки пакетов, с целью совершить их передачу первоочерёдно через более быстрый канал. Менее важные пакеты или перенаправляются по медленным путям, или эти пакеты отбрасываются в пользу более важного содержания.

Корпоративная сеть, которая есть частным случаем ТКС и имеет неопределённую структуру, может иметь в своём составе сегменты, проходящие через глобальную ТКС посредством туннелирования по технологиям VPN. Для туннелированных сегментов осуществляется шифрование трафика, что приводит к анонимности не только по данным, но и по типу трафика, вследствие чего, общие методы обеспечения качества связи QoS вынуждены работать по косвенным признакам определения вида трафика. Однако для корпоративной сети возможно явное представление типа переданных данных и временные особенности трафика, которые зависят от внутреннего распорядка корпорации и их вида деятельности [62]. В связи с указанными особенностями, по полученным статистическим данным нагрузки сервера корпоративной сети, был произведён анализ трафика по ежечасовым измерениям, для которого полученный критерий Хэрста редко был выше 0,55 [67, 108]. Это позволило использовать классические методы моделирования и исследования без учёта самоподобия трафика в корпоративной сети. Далее по тексту рассматривается обобщённая ТКС, для которой выполняются указанные особенности трафика.

Исследование ситуации показывает, что методы повышения эффективности передачи данных в телекоммуникационных системах с современными протоколами доступа и управления, учитывающих особенности передачи разной природы данных, исследованы недостаточно. Это указывает на необходимость разработки эффективных методик увеличения надёжности передачи и её оперативности [8, 56].

Процесс передачи информации по загруженным сегментам сети приводит к частичной потери мене значимой, по мнению маршрутизатора, информации. Такой подход позволяет освободить канал для передачи более важных сообщений. Однако факт передачи сообщения по предварительным сегментам сети уже состоялся, что приводит к занятости предварительных сегментов сети передачей информации, которая будет отброшена – это противоречие можно решить, если менее значимая информация не будет передаваться изначально. Связи с этим возникает задача уменьшения количества передаваемой менее значимой информации на этапе запроса, и если клиент подтвердит значимость информации, он может осознано принять решение о получении этой информации в полном виде.

1.2 Исследование информационного трафика телекоммуникационной сети, определение части графической информации

Разработка метода снижения загруженности каналов связи нацелена в основном на графический контент. Для подтверждения актуальности и полезности такого подхода необходимо установить объёмы трафика графического контента, оценить снижение загруженности сети и принять решение о целесообразности такого подхода. К сожалению, самостоятельно провести исследования подобного плана не представляется возможным без задействования аппаратных средств сбора статистической информации на глобальном уровне [8]. Поэтому приходиться доверять сторонним источникам, фирмам, которые обслуживают сети на разных этапах предоставления и передачи информации. Наиболее известной на сегодняшний день является программа сбора, обработки и прогнозирования развития сетевого трафика Cisco Visual Networking Index.

По открытым данным исследования Cisco Visual Networking Index составлена следующая таблица:

Таблица 1.1

Виды и объёмы трафика в глобальной сети

Вид трафика	(PB per Month)	Доля трафика в %
Видео	22,814	63,63%
Файлы	6,548	18,26%
http	6,492	18,11%
Всего	35,854	100,00%

Из таблицы видно, что графическая информация, на сегодняшний день занимает около 15% от общего трафика в сети. Сокращение этого трафика вдвое позволит снять загруженность во всей сети на 5%, что значительно сокращает вероятность переполнения очереди на загруженных точках доступа беспроводной связи.

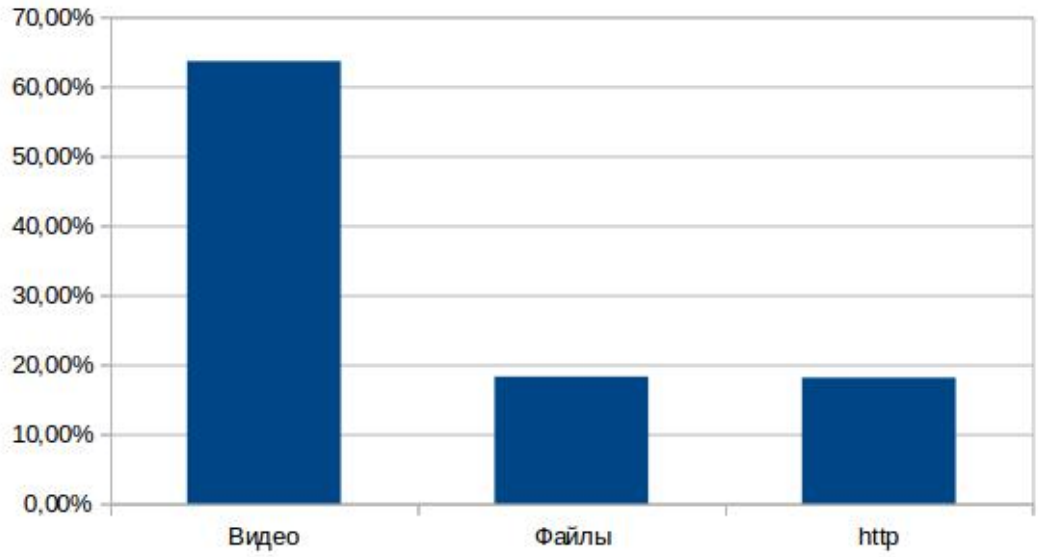


Рис. 1.1. Виды и объёмы трафика в глобальной сети

В таблице 1.2 показан результат моделирования при различных загруженностях телекоммуникационного канала с очередью (длина очереди принята 10 с целью подчеркнуть эффект). Из таблицы видно, что вероятность потери пакета с разгрузкой сети резко падает. С использованием протокола гарантированной доставки пакетов, следует понимать, что пакет будет

доставлен с задержкой из-за вынужденной повторной отправки потерянного в очереди пакета.

Таблица 1.2

Вероятность потери пакета в зависимости от загруженности телекоммуникационного канала

Относительная загруженность	0,99	0,95	0,9	0,85	0,75	0,7	0,65
Вероятность потери пакета	0,08	0,06	0,04	0,03	0,01	0,01	0,00
Реальная относительная пропускная способность канала	0,91	0,89	0,86	0,83	0,74	0,70	0,65

Таким образом, незначительное снижение нагрузки телекоммуникационного канала цифровой передачи информации приводит к значительному возрастанию качества связи и уменьшению латентности сети к передачи многопакетных сообщений. Сокращая объёмы передач графической и графической информации за счёт лучшего сжатия, отбрасывания незначимой информации на уровне осознанного решения пользователя, мы улучшаем характеристики сети по задержке передачи информации высокоприоритетных сообщений. Это подтверждает актуальность применения кэширования и дополнительного сжатия графической информации [8, 78, 79, 80].

1.3 Анализ требований к качеству обслуживания при передаче данных в телекоммуникационных системах и сетях, обоснование критериев и показателей эффективности

Одним из возможных методов повышения оперативности передачи информации в телекоммуникационной сети, является уменьшение нагрузки на сервера телекоммуникационной сети, за счёт использования промежуточных серверов. К таким системам серверов относятся системы хранения информации проходящей через них, с последующими повторными её выдачами без

обращения к основным серверам. Это уменьшит нагрузку на основные сервера телекоммуникационных сетей. При этом если промежуточный сервер дополнительно проводит сжатие информации, объём трафика значительно уменьшается. Благодаря более короткому пути запроса, ответа и уменьшенного объёма переданной информации, повышается оперативность обслуживания пользователей. В большинстве работ по качеству обслуживания в сети [12, 45, 54, 63, 66, 77, 95, 106], ограничиваются только рассмотрением гарантированного и оперативного предоставления информации, за счёт отказов в доставке менее важного содержимого.

Все пользователи, запрашивающие информацию, получают её или целиком, или часть информации, отбросив её часть по какому-нибудь критерию. Например, пропускают графические данные без учёта их важности и актуальности. Предлагается значительно уменьшить объём передаваемой информации за счёт значительных потерь качества графической информации с прогрессивным кодированием. Это даёт пользователю возможность оценить ценность информации и, при необходимости, догрузить её полностью без повторного приёма уже загруженной части [118, 119, 120].

Для внедрения такой системы необходимо заменить всю графическую информацию на всех серверах, перекодировав её в формат со свойством прогрессивности и обеспечить поддержку этого формата клиентским программным обеспечением. Однако на сегодняшний день такую работу выполнить представляется возможным только на корпоративном уровне при использовании собственного программного обеспечения. В общем случае, гораздо более реально обеспечить использование нового формата при формировании новых информационных серверов телекоммуникационных сетей. Поддержка формата на клиентском программном обеспечении проводиться установкой плагина или специализированного клиентского приложения, что поддерживается во всех современных средствах получения информации.

Обзор возможных средств прогрессирующего сжатия графического контента проведён в главе 1.5.

С целью определить возможность внедрения промежуточных серверов дополнительного сжатия графического контента и их влияния на качество обслуживания телекоммуникационной сети, необходимо построить модель потоков графической информации, которая будет учитывать предложенные изменения.

1.3.1 Моделирование потоков графической информации, учет возможности дополнительного сжатия графического контента на промежуточных серверах

С целью определения важных характеристик сети и методов их изменения, строится математическая модель системы сервер-клиент [18, 69, 70, 71].

Введём обозначения:

T_1 – среднее время на прямую загрузку графического контента с серверов;

T_2 – среднее время на загрузку пережатой информации с сильной потерей качества;

T_3 – среднее время на загрузку исходной графической информации на промежуточный сервер;

$T_п$ – время затраченное на перепаковывание информации, зависит от типа графической информации, кодека и вычислительных мощностей сервера;

k – доля графического контента, которая уже находится на промежуточных серверах;

S – общее количество графического контента;

S_k – количество графического контента, который уже находится на промежуточных серверах ($k=S_k/S$);

$L(t)$ – количество активных пользователей системы, а L – их максимальное значение.

Распределение времени передачи графической информации происходит неравномерно, за счёт использования быстрой глобальной между серверной связи и довольно медленными ответвлениями к конечному пользователю. Введём обозначения средней скорости передачи графической информации магистральной сети и скорости сегмента к конечному пользователю:

v_c – средняя скорость передачи информации между серверами;

v_k – средняя скорость передачи сегмента сети к конечному пользователю.

Соответственно

$$T_1/T_3=v_c/v_k. \tag{1.1}$$

В случае высококачественно соединения к глобальной сети конечного пользователя или использования прокси-сервера, можно принять, что скорости передачи соизмеримы и $T_1 \approx T_3$ (рис. 1.2, 1.3).

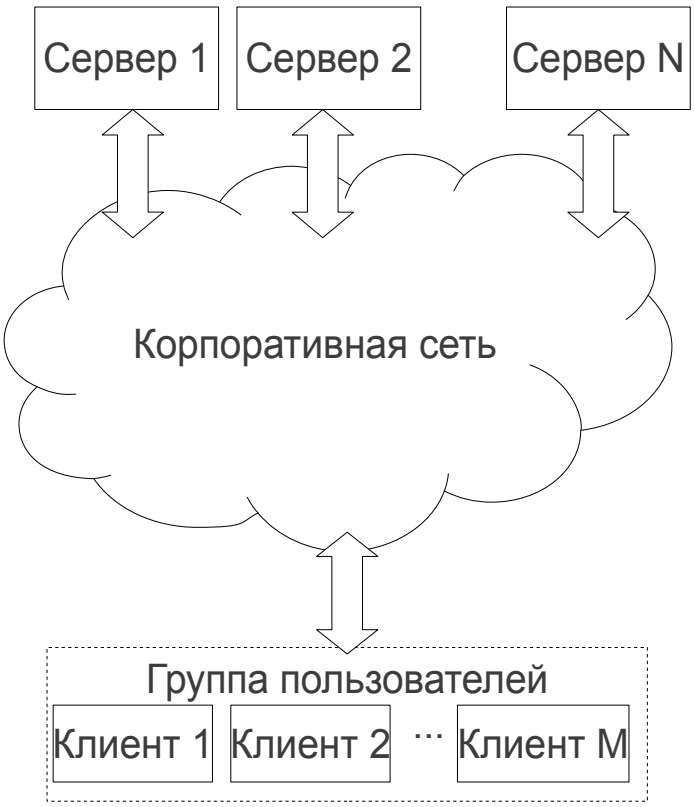


Рис. 1.2. Потоки графической информации в телекоммуникационной сети

В этих случаях, пользователь получит данные не раньше, чем за время T_1. Однако, предположим использование более совершенного алгоритма упаковки графической информации, тем самым уменьшив количество потока информации в g раз. В главе 2.1 доказывается, что и время загрузки уменьшиться в среднем в g раз. Это повышает показатель оперативности доставки информации и, при равной вероятности искажений передачи помехами, повышает её надёжность передачи.

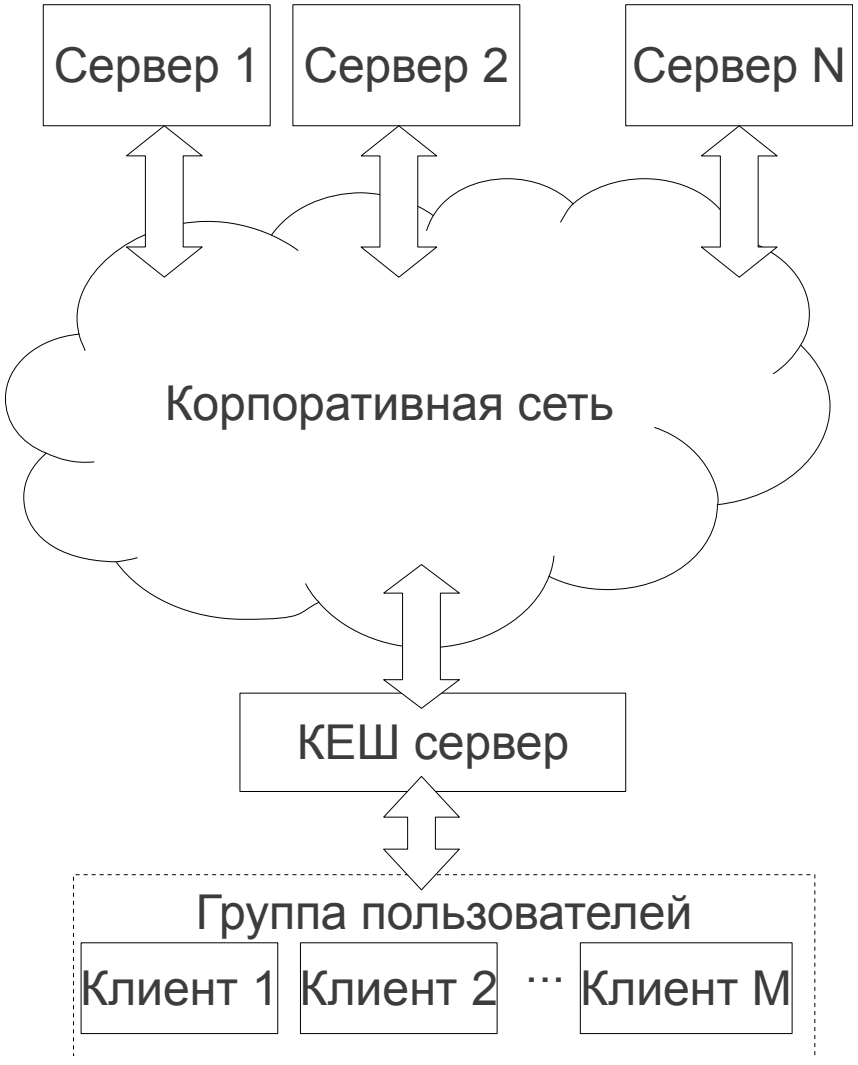


Рис. 1.3. Использование прокси-сервера в телекоммуникационной сети

Внедрение кодирования графической информации с более сильным коэффициентом сжатия, при аналогичном качестве восстановления, было бы более лучшим решением. Однако это подразумевает что на каждом ресурсе глобальной телекоммуникационной сети, включая архивные данные, будет

изменена вся информация. Такое глобальное изменение невыполнимо за короткий срок, с точки зрения привлечения человеческих и машинных ресурсов на обработку всей существующей информации.

Предлагается автоматизация такого процесса: рис. 1.4.

Подобная система не нова и внедрена в интернет браузер Opera, однако не пользуется популярностью. Причина редкого использования кроется в отсутствии ускорения загрузки страниц при довольно быстром подключении к сети. Причины такой неэффективности системы можно понять из модели потоков информации в сети. Поняв причины, нужно разработать методы для устранения этих причин [43, 63, 86, 88, 98, 102].

Пускай в определённый момент времени t, известны полное количество графической информации в сети S, количество перепакованной информации на промежуточных серверах S_k. Тогда вероятность того, что пользователь обратиться к уже перепакованной информации будет $k=S_k/S$, и эту информацию он получит за время $T_2=T_1/g$, где g степень сжатия по отношению к оригинальному источнику информации.

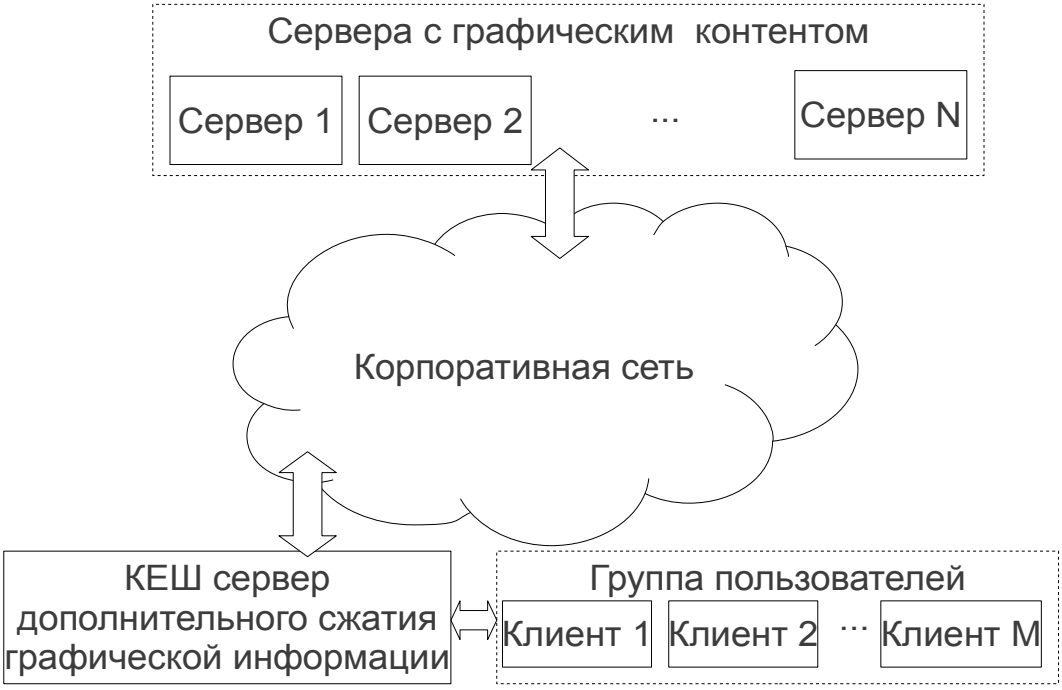


Рис. 1.4. Схема использования промежуточного сервера перепакованного графического контента

Если пользователь запросил новую информацию, которая ещё не попала в базу перепакованных данных, то полное время на получение информации будет составлять $T_2+T_3=T_1/g+T_3$, откуда видно более медленное получение за счёт дополнительного времени на загрузку информации на промежуточный сервер и её переупаковка T_3. В случае максимально быстрого подключения к глобальной сети, время T_1/g входит полностью как дополнительное, снижая оперативность доставки информации. При наличии более медленного сегмента подключения к глобальной сети в $r=v_k/v_c$ раз, соответственно время загрузки составят ($T_1 \approx T_3\ r$):

$$T_{1а}=T_3+T_3\ r/g, \tag{1.2}$$

в случае не попадания пользователя в уже дополнительно сжатый контент, время на доставку к серверу добавляется к времени доставки перепакованного содержимого, и

$$T_{1б}=T_3\ r/g, \tag{1.3}$$

когда передаются уже ранее перепакованные данные.

Среднестатистическое время ожидания доставки информации будет, с учётом равномерного распределения вероятности обращения пользователей (игнорируем популярность ресурсов): $T_{1с}=kT_{1б}+(1-k)T_{1а}$. Подставив (1.2) и (1.3), приведя подобные, получим среднестатистическое время загрузки графического контента с использованием промежуточного сервера дополнительного сжатия графического контента:

$$T_{1с}=T_3(1-k+r/g). \tag{1.4}$$

В этом случае от аппаратного обеспечения зависят величины T_3 и r, k и g могут быть результатом изменения программного обеспечения и логики работы системы. Для минимизации значения $T_{1с}$ необходимо достигнуть максимальных значений $0\le k\le 1$ и $g\ge 1$, которые отвечают за полноту перепакованного

графического контента и за степень сжатия информации. Коэффициент изменения скорости передачи в сети можно оценить по (1.5):

$$z(k,g)=(1-k+r/g)/r. \quad (1.5)$$

Из (1.3) следует, что программной реализацией повышения пропускной способности сети и оперативности доставки, будет улучшение принятых на сегодня методов упаковки графической информации. Также необходима организация работы промежуточных серверов с целью максимальной скорости добавления графического контента ($k \to 1$).

1.3.2 Отказоустойчивость системы промежуточных серверов дополнительного сжатия графического контента

На этапе внедрения системы дополнительного сжатия графического контента, коэффициент доли перепакованных данных будет равен нулю. Каждый запрос клиента будет сопровождаться обращением к оригинальным данным, дополнительным сжатием и отправкой результата. Это приведёт к сильной загруженности серверов, что может повлечь за собой отказ системы в обслуживании на некоторый процент запросов. Такая ситуация может полностью лишить систему стабильности. Сильно превышать средние запросы к аппаратной части производительности системы тоже не выгодно, по причине простоя значительных ресурсов на конечном этапе внедрения.

Серверная часть дополнительного сжатия графической информации должна содержать систему предсказания трафика, с целью предотвращения отказов в обслуживании. Механизмом снижения нагрузки может быть временное возвращение некоторого процента запросов клиентов в классическую схему прямого общения с серверами. Такая система предсказания должна в автоматическом режиме качественно определять возможность угрозы

перегрузки, а также иметь количественную оценку процента запросов, которые сервер не сможет обслужить [2, 52, 73, 82, 83, 92, 100, 105].

1.4 Сравнительное исследование методов анализа временных рядов с целью прогнозирования степени загруженности серверов телекоммуникационных систем и сетей

Надёжность, отказоустойчивость обслуживания серверами телекоммуникационных систем и сетей зависит от загруженности её серверов [72, 73, 92]. Это приводит к необходимости постоянного мониторинга состояния сервера по количеству занятых и свободных ресурсов. Под ресурсами сервера телекоммуникационной системы или сети тут и дальше понимается время выполнения программного обеспечения на процессоре, постоянная и оперативная память, исходящий и входящий трафики. При превышении запроса к ресурсам сервера или кластеру серверов, который будем считать единой структурной единицей, балансируемая внутренними алгоритмами и процедурами, пользователь получит отказ в обслуживании или будет поставлен в очередь на обслуживание, что приведёт к значительному снижению качества обслуживания по критериям надёжности и оперативности. Автоматизация процесса мониторинга загруженности и оценки прогноза загруженности серверов, даёт возможность предсказать нехватку ресурсов и предотвратить ухудшение качества обслуживания сети по критериям QoS. При опасности скорого превышения возможностей сервера, можно избежать отказа в обслуживании обновив оборудование сервера или перенаправив часть задач к другому серверу, тем самым распределив нагрузку телекоммуникационной системы или сети.

Исходными данными для построения прогноза загруженности сервера телекоммуникационной системы или сети (далее «сервера»), является статистика количества занятых и свободных ресурсов сервера за определённый

отрезок времени. Исходные данные представимы в виде дискретно во времени таблично представленной функцией. Это позволяет использовать методы экстраполяции дискретно заданной функции. Тут применимы методы анализа дискретных сигналов во времени [65, 74, 76, 87, 96]. Для решения главной задачи построения прогноза загруженности сервера, нужно решить ряд локальных задач:

а) построить аналитическое приближения сигнала, которое имело бы максимально сходные свойства объёма трафика сети, которые рассмотрены в п. 1.4.1;

б) расчет прогноза по аналитическому приближению;

в) оценка погрешности прогноза;

г) получение вероятности нехватки ресурсов сервера для обработки необходимого количества запросов.

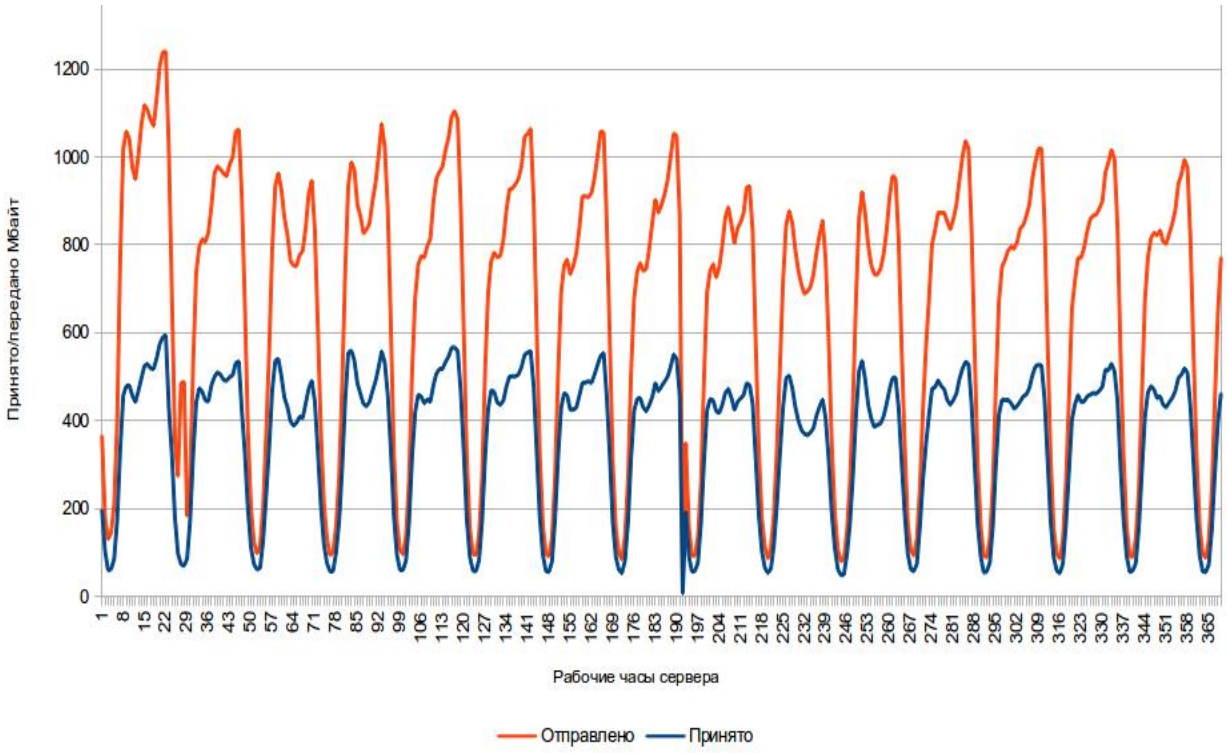
1.4.1 Характеристика временного ряда сетевого трафика типичного сервера корпоративной сети

Применимость методов прогнозирования и анализа временных рядов зависит от характера поведения базисных функций, который используется в начальном анализе. Идеальным вариантом является набор базисных функций, которые отражают физические и статистические свойства сигнала, который подвергнут анализу. При этом может оказаться, что применимы методы, которые изначально не предназначались для анализа другого рода сигналов.

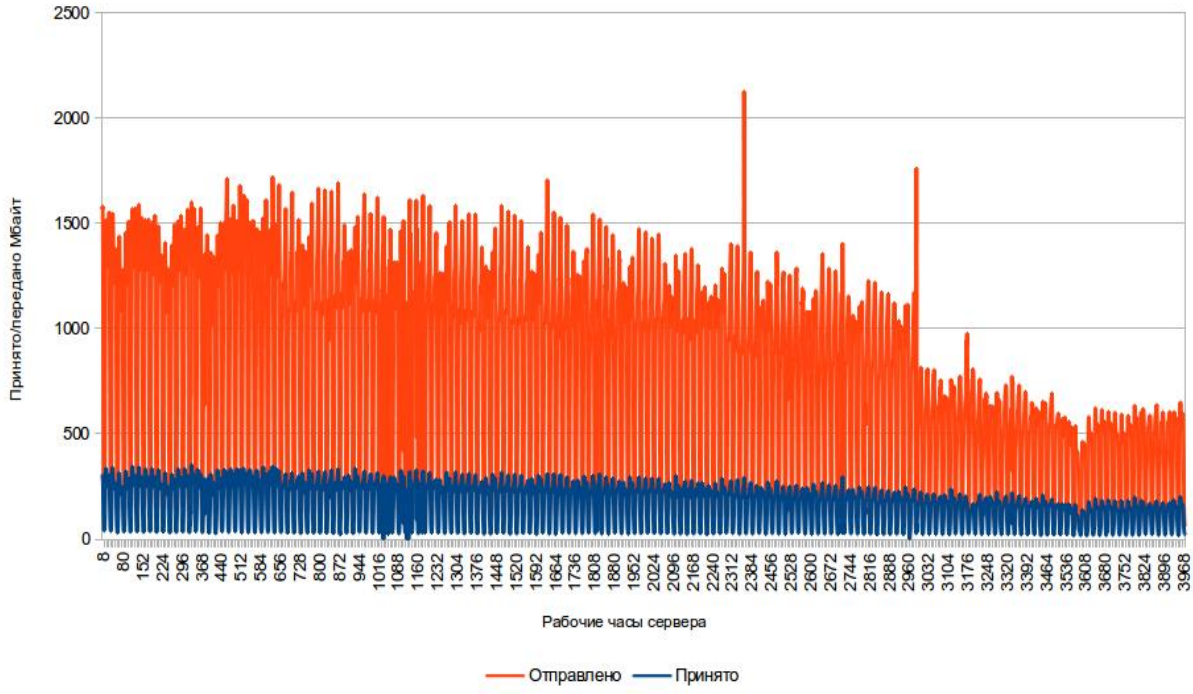
Типичный график временного ряда, который получен в результате обработки статистики графического трафика из сервера телекоммуникационной сети показан на рис. 1.5.

На рисунке видно, что сигнал имеет циклический характер, который отображает суточные изменения количества клиентов, зависимость от дня недели, праздников. В общем случае естественные циклы загруженности

сервера могут иметь и другие механизмы синхронизации, в том числе и естественные, спонтанно возникающие сервисные диалоги “запрос-ответ”.



а) 370 часов работы



б) 4000 часов работы

Рис. 1.5. Поток графической информации сервера телекоммуникационной сети

Сигналы имеют явный колебательный характер с медленно изменяющейся амплитудой и нестабильной фазой. Математически такие квазипериодические колебания можно задать моделью [1, 3, 87]:

$$f(t) = A_0 + \sum_{k=1}^{N} \left(A_k + \psi_k(t)\right) \sin(\omega_k t + \varphi_k + \xi_k(t)), \tag{1.6}$$

где $f(t_i)$ – функция которая моделирует сигнал, t – время, A_k, ω_k, φ_k – константы модели, $\psi_k(t)$, $\xi_k(t)$ – случайные функции от времени с нулевым математическим ожиданием и ограниченной дисперсией δ_k. В общем случае $\psi_k(t)$, $\xi_k(t)$ имеют распределение отличное от нормального. Сам сигнал можно классифицировать как ***квазипериодический*** и ***стационарный*** со стохастической составляющей, которая не может быть прогнозирована.

Циклические изменения нагрузки могут проявляться при разном масштабировании сигнала по времени [4, 60, 90, 104]. Также одни и те же синхронизирующие факторы использования ресурсов серверов могут в разных случаях приводить к нарастанию интенсивности использования серверных ресурсов, так и к спаданию. Это не позволяет использовать заданный жёстко регрессионный базис функций в общем случае.

Классические методы регрессии для поиска коэффициентов приближения (1.6) дают возможность определить значения коэффициентов A_k, φ_k, однако для их поиска необходимо иметь значения циклических ω_k, которые можно получить, использовав гармонический анализ исходного сигнала или более сложными методами.

1.4.2 Обзор методов поиска гармонических составляющих процесса изменения трафика в телекоммуникационной сети

При анализе временных рядов широко используют гармонический анализ – преобразование Фурье и разложение в ряд Фурье [1]. Однако

полученные частоты при разложении в ряд Фурье имеют общий кратный период равный промежутку исследования. При экстраполяции исследуемого сигнала полученными наборами гармонических составляющих, получим циклическое продолжение сигнала, которое не представляет интереса при построении прогноза.

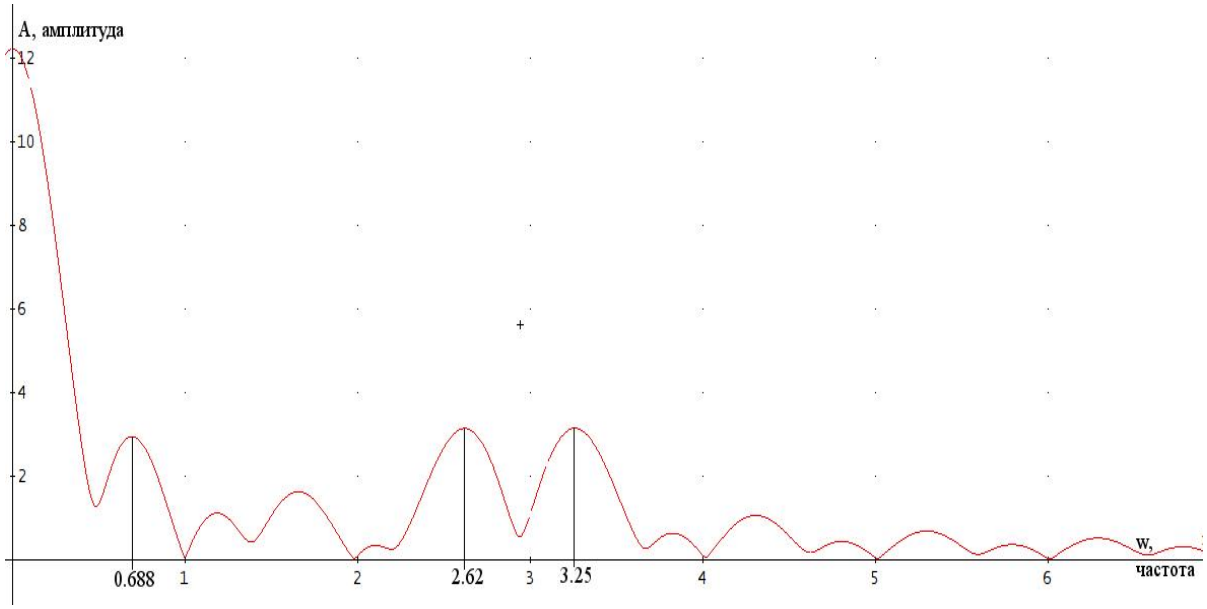


Рис. 1.6. Частотно-амплитудная диаграмма преобразования Фурье функции $1-2\sin(3t)+\sin(3.1t)+\cos(0.1t)$

Преобразование Фурье предназначено для выделения энергии гармонических составляющих сигнала, объедение которых позволяет восстановить исходный сигнал. Однако использование преобразования на ограниченном промежутке во времени дискретного сигнала не позволяет адекватно выделить доминирующее гармоники, это показано конкретным примером на рис. 1.6.

Конкретный пример показывает неточность определения основной частоты $\omega_1=3$, и полностью отсутствуют признаки наличия частот $\omega_2=0{,}1$ и $\omega_3=3{,}1$.

В работах по частотно амплитудному анализу [87] используют интегральное вейвлет-преобразование, которое, как и преобразование Фурье,

дает усредненную энергию гармонической составляющей конкретной частоты. Однако вейвлет анализ был разработан как метод для обнаружения момента времени возникновения частотного всплеска, поэтому применение тут вейвлет-преобразования оправданно только меньшей сложностью алгоритма расчета вейвлет коэффициентов. Результатом применения интегрального вейвлет-преобразования для частотного анализа, будет набор частот и амплитуд гармонических составляющих сигнала, которые будут менее точными, чем при использовании преобразования Фурье. В другом случае, для прогнозирования процесса, необходимо строить прогнозы для вейвлет-коэффициентов, что основано на предположении возможности более точного прогнозирования этих коэффициентов против самого временного ряда [3, 4, 74].

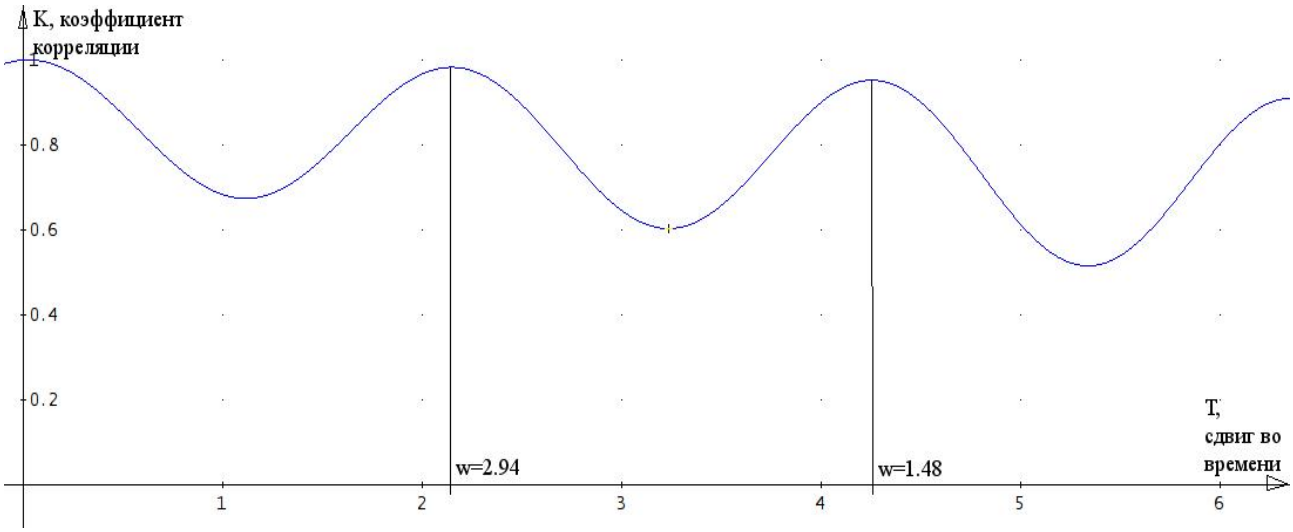


Рис. 1.7. Результат автокорреляционного анализа функции $1-2\sin(3t)+\sin(3.1t)+\cos(0.1t)$

Автокорреляционный анализ также часто применяют для поиска периодичностей в исследуемом сигнале. Суть метода заключается в нахождении корреляционного коэффициента сигнала с собой, но при этом со сдвигом по времени. Естественно, при наличии в функции периода T, $f(t)=f(t+T)$, коэффициент корреляции $f(t)$ и $f(t+T)$ будет равен единице. При наличии квазипериодичности, коэффициент корреляции будет приближаться к единичному значению рис. 1.7.

Главным недостатком перечисленных выше методов поиска периодов гармонических составляющих сигнала является невозможность получить оценку наличия колебаний с периодом большим, чем время наблюдения сигнала.

При использовании авторегрессии [75, 101], для заданного набора $g_k(t)$, ищут приближение:

$$f(x) = \sum_{k=0}^{N} a_k g_k(t) + Y(t),$$

где a_k – искомые коэффициенты, N+1 – количество наперед заданных функций базиса разложения, $Y(t)$ – случайный шум, который невозможно аппроксимировать. Главным недостатком метода является зависимость качества приближения от наперёд заданного базиса $g_k(t)$. Поэтому базис должен быть максимально полным и не иметь линейно зависимых и почти линейно зависимых функций. При наличии почти линейно зависимых базисных функций, решение системы линейных уравнений для поиска a_k, даёт неправдоподобно большие значения коэффициентов (как пример можно взять явление биения близких гармоник, где на промежутке приближения амплитуды взаимно отнимаются, а на промежутке прогноза складываются, приводя к абсурдным результатам). Этот метод не применим к поставленной задаче, по причине невозможности подготовить достаточно полный базис для общего случая. Для полностью автоматически разворачиваемой системы любой ограниченный базис не может быть использован. Также наличие N базисных функций приводит к решению системы линейных уравнений с N неизвестных, алгоритм решения которой имеет факториальную сложность [51].

Генетические алгоритмы [51] подобно авторегрессии требуют наличия базовых функций $g_k(t)$, однако в аппроксимационный полином входят только некоторые из них, которые прошли через «искусственный отбор».

Головной идеей алгоритма является выборка подмножества из головного множества аппроксимирующих функций, которые лучше соответствуют критерию отбора (максимальная корреляция, минимальность отклонения и другие критерии). Из полученного подмножества строят полный набор линейных комбинаций, из которой повторяют отбор лучших приближений. Цикл отбора повторяют до прекращения заметного улучшения значений критерия отбора.

Алгоритм похож на метод авторегрессии, в котором решение одной огромной системы линейных уравнений заменили на решения многих систем линейных уравнений с двумя неизвестными. С вязи с этим, метод принципиально не может дать лучшее приближение, однако получает хорошее приближение при значительно меньших вычислительных затратах. Также генетические алгоритмы более стойкие к использованию почти линейно зависимых функций в начальном базисе.

Поиск составляющих гармоник производиться выборкой из заготовленного базиса, который содержит максимально возможное количество гармонических функций с различными периодами.

Результат анализа принятых методов гармонического анализа и прогнозирования поведения систем показал, что методы не дают полной возможности получать надёжные прогнозы даже на качественном уровне.

1.4.3 Применимость классических методов анализа к поставленной задаче

Обзор методов для получения набора частот ω_k экстраполяционного многочлена (1.5) для последующего применения метода наименьших квадратов поиска значений коэффициентов A_k и φ_k показывает, что выше рассмотренные методы имеют ограниченное применение. Методика поиска составляющих гармоник квазипериодического сигнала требует изменения, и для

прогнозирования подобных временных рядов необходима методика выделения гармонических составляющих с некратными периодами.

1.5 Анализ и оценка существующих и перспективных методов сжатия графической информации

С целью обеспечения качества обслуживания в телекоммуникационных системах и сетях графическая информация передаётся в сжатом виде, что в свою очередь снижает время передачи содержимого и освобождает часть канала для других пользователей. В связи с этим, как показано в главе 1.2, качество сжатия графической информации существенно влияет на качество обслуживания телекоммуникационной системы или сети. Критически важной стоит задача выбора максимально эффективного алгоритма сжатия графического контента.

Сегодня наиболее конкурентными по степени сжатия и качества изображения лидируют методы основанные на вейвлет-преобразованиях. Однако проблемы лицензирования технологий, а также отсутствие поддержки прогрессирующей передачи информации, которые использованы в формате JPEG2000 не позволили широко использовать этот формат в Интернете. Формат сжатого изображения PNG предназначен для изображения с большими однотонными полями, такие изображения могут использоваться в мультипликации, построении диаграмм и подобных целях и не поддерживает прогрессивность передачи данных об изображении [49, 53, 85, 107, 114, 117, 124].

Для сравнения методов основанных на вейвлет-преобразовании выбрано формат JPEG2000, который предоставляется без исходного кода в виде свободной библиотеки, и SPIHT кодек, реализованный автором с использованием вейвлетов Хаара и целочисленного вейвлета Добеши 1/8. Для преобразования Хаара использован целочисленный алгоритм с

коэффициентами фильтров {½; ½} и {1; -1}. Для преобразования 1/8 использованы фильтры {-1/8;2/8;6/8;2/8;-1/8} и {0;-1/2;1;-1/2;0}. Вейвлет Добеши используется также при кодировании без потерь в формате JPEG2000, поэтому стоит ожидать от кодеков сходные характеристики [9, 10, 11, 17, 23, 31, 32, 50, 99].

В случае прогрессивного сжатия графической информации, количество переданных данных вырастет за счёт передачи дополнительных данных на запись порядка передачи вейвлет коэффициентов. Перед кодированием изображения были получены их статистические характеристики, на основе которых можно предположить применимость того или иного метода [23]:

а) математическое ожидание значения яркости пикселя (M);

б) дисперсия значений яркости пикселя (D);

в) математическое ожидание разности яркости между соседними пикселями (SM);

г) дисперсия разности яркости соседних пикселей (SD);

д) четвёртый момент разности яркости соседних пикселей (SQ);

е) соотношение SD/SQ которое для дискретного целочисленного значения яркости, больше единице, за исключением абсолютно однотонного изображения когда соотношение принимает единичное значение.

Все эти характеристики цифрового растрового изображения статистические и имеют линейную сложность вычисления по отношению к количеству пикселей. Все эти характеристики рассчитываются однопроходными алгоритмами, что максимально подходит для большинства архитектурных особенностей доступа к оперативной и постоянной памяти. Другие характеристики изображений имеют довольно сложные алгоритмы расчётов, которые могут быть полезными в теоретических изысканиях, но на практике сложно применимы, и также используют не целочисленные алгоритмы, что ограничивает платформы их применения.

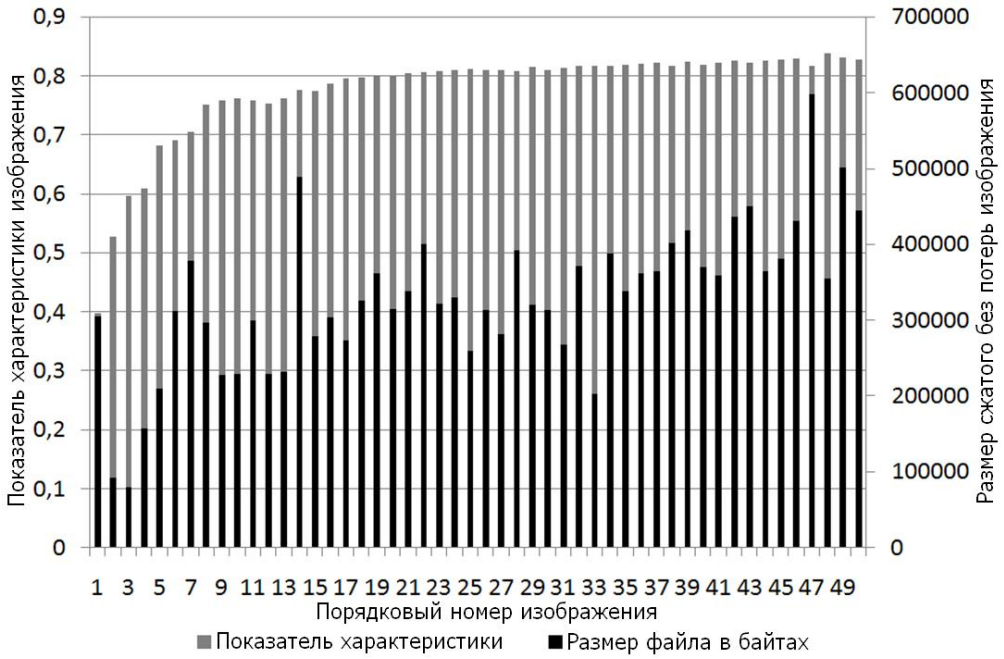


Рис. 1.8. Соответствие размера сжатого изображения (SPIHT, вейвлет Хаара) к характеристике SD/SQ, коэффициент корреляции 0,54

Таблица 1.3

Корреляция значений статистических характеристик изображения с размером сжатого изображения

Характеристика	M	D	SM	SD	SQ	SD/SQ	Ентропія	SM/SQ
SPIHT(Haar)	-0,54	-0,19	0,62	0,54	0,48	0,54	0,46	0,63
SPIHT(1/8)	-0,48	-0,2	0,55	0,46	0,39	0,46	0,39	0,56
JPEG2000(Loweless)	-0,47	-0,18	0,54	0,45	0,39	0,46	0,39	0,56

Таблица 1.3. отображает корреляции между статистическими характеристиками изображения и размером файла при упаковке без потери качества. Также была подсчитана энтропия информации исходного изображения, которая берётся за основу оценки прогноза степени сжатия информации [17], но, как показал эксперимент, более правдоподобный показатель возможности сжатия изображения является отношение математического ожидания модуля разности соседних пикселей к четвёртому моменту этих разностей (эксцессу) SM/SQ. Для построения таблицы было использовано 50 разнотипных фотографических изображений.

Для моделирования условий работы кодеков распространён способ замены изображения статистическим эквивалентом, где изображение имеет заданную корреляцию между соседними строками [11]. Однако в ходе этого эксперимента был получен результат корреляции всего на уровне 0.38, чего явно не достаточно для оценки сжимаемости изображения тем или иным методом.

Величина корреляции SM/SQ со степенью сжатия изображения даёт наибольшее значение из опробованных. Поэтому принято решение использовать результат линейной регрессии методом наименьших квадратов в качестве модели предсказания эффективности сжатия исходного изображения различными алгоритмами или на основе различных вейвлетов.

В результате линейной регрессии получено:

$R_{sh} = 511900 \cdot SM / SQ + 73221$ – при кодировании SPIHT вейвлетом Хаара;

$R_{s1/8} = 423083 \cdot SM / SQ + 93254$ – при кодировании SPIHT вейвлетом 1/8;

$R_{jpeg2000} = 396807 \cdot SM / SQ + 82141$ – при loweless кодировании JPEG2000.

График зависимости SM/SQ и размера выходного файла показано на рис. 1.9, на котором видно изменения размеров выходного файла и соответствующей характеристики SM/SQ. В статистическом смысле величину SM можно трактовать как средне ожидаемую разность межу соседними пикселями. SQ содержит четвёртые степени этих разностей, что означает сильное преувеличение значения более резких переходов в изображении. Фактически, отношение SM/SQ тем ближе к нулю, чем больше элементов изображения имеет резких переходов. Про такие изображения говорят «насыщенны мелкими деталями». В то же время такое изображение может иметь довольно высокую корреляцию между соседними строками/столбцами, поэтому конкретный пример из многоклеточного изображения даст по корреляционной характеристике неоправданно низкие значения. Однако та же характеристика корреляции будет адекватной, если квадраты расположить диагонально.

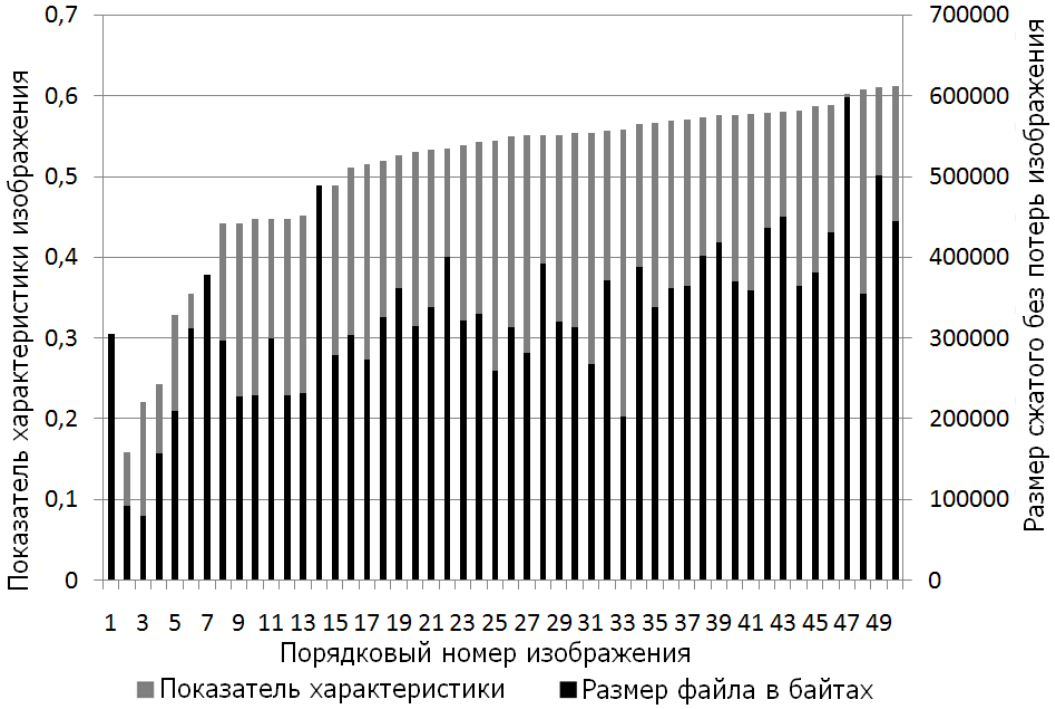


Рис. 1.9. Соответствие размера сжатого изображения (SPIHT, вейвлет Хаара) по отношению характеристики SM/SQ, коэффициент корреляции 0,63

Использование линейной регрессии позволило выделить тенденцию улучшения качества кодирования по размеру файла в случае использования более простого вейвлета Хаара при SM/SQ<0,22. Это даёт возможность перед кодированием выбрать более подходящее вейвлет-преобразование с большей надёжностью, чем при использовании междустрочной корреляции изображения.

На рис. 1.10 показаны линейные графики, которые показывают изменение степени сжатия рисунка по изменению характеристики SM/SQ. Расположения двух прямых почти параллельно объясняется тем, что при кодировании были использованы одинаковые вейвлеты. Меньший размер файла JPEG2000 обеспечен более качественным кодированием полученных вейвлет-коэффициентов при помощи алгоритма математического кодирования. Введение дополнительного математического сжатия в алгоритм SPIHT, может уравнять степень сжатия изображений с форматом JPEG2000, а в некоторых случаях превзойти его.

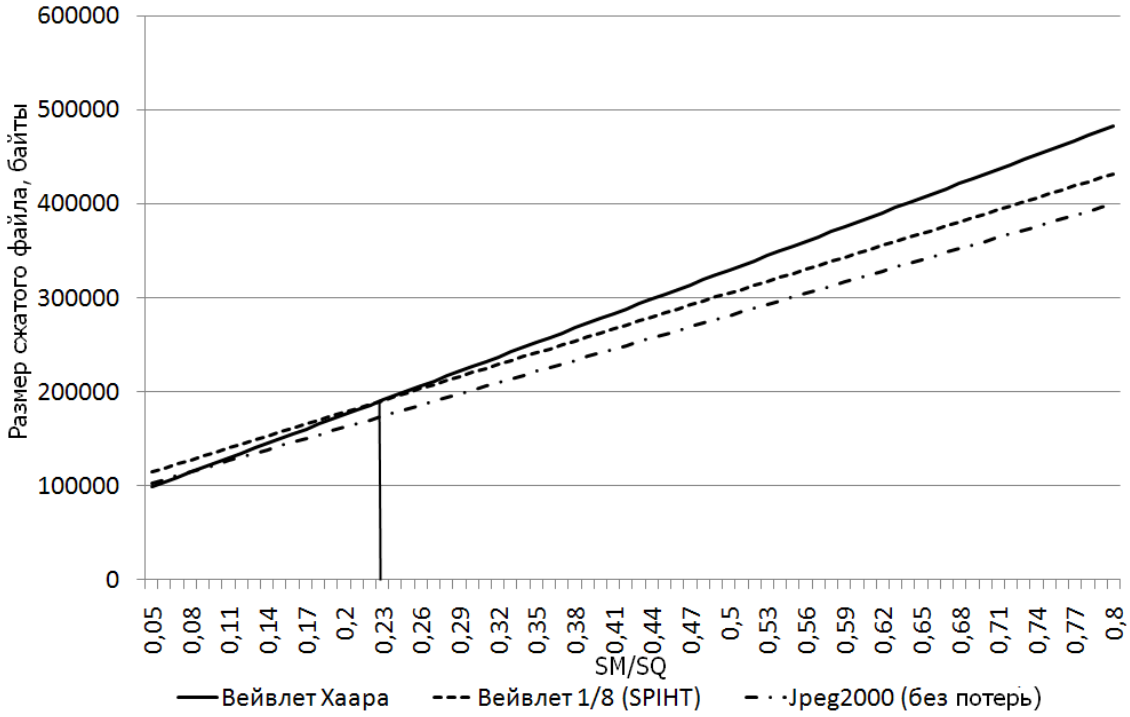


Рис. 1.10. Линейная регрессия зависимости SM/SQ и выходного размера файла сжатого изображения

Также для высоких коэффициентов сжатия, при значительных потерях информации, SPIHT алгоритм должен быть улучшен, что бы сохранить конкурентоспособность при высоких коэффициентах сжатия.

1.6 Постановка задачи повышения оперативности передачи данных

На основе вышеизложенной информации выбрана схема доставки часто востребованной графической информации при помощи промежуточного сжатия графических данных прогрессивным кодеком с потерей информации, кэширующего сервера, с возможностью восстановления первоначального качества только для актуальной информации, посредством дозагрузки недостающей информации.

Продуктивность выбранной системы зависит от измеряемых параметров системы T_1, T_2, T_3, $T_п$, S, S_k, $L(t)$, v_c, v_k; и от управляемых параметров k и g. Необходимо получить оценку и условия увеличения оперативности передачи графических данных в телекоммуникационной корпоративной системе или

сети, используя как базу оценку (1.5); обеспечить устойчивость системы к возможным перегрузкам и отказам в обслуживании, за счет перехода части запросов на стандартную схему по критерию, полученному из прогноза загруженности кэширующего сервера; выбрать кодек для графической информации, на примере графического контента, для получения оптимального значения g.

Таким образом, на основе указанных зависимостей, сформулирована оптимизационная задача, которая заключается в минимизации времени доставки сообщения:

$$T_{1c} \to \min,\ T_{1c} = f(P_{отк}, K_{кач}, g),$$

с ограничениями $0 \le P_{отк} \le 1$, $K_{кач} > 0$, $g \ge 1$, где $P_{отк}$ – вероятность отказа в обслуживании сервером дополнительного сжатия графической информации контента, которая имеет верхний предел в соответствии рекомендациям международного союза электросвязи (МСЕ) Е.845, Е.846; $K_{кач}$ – коэффициент качества визуализации графической информации, который имеет нижнюю границу по возможности оценивания актуальности информации и задаётся заказчиком ТКС услуг; g – коэффициет дополнительного сжатия графической информации.

Выводы

Проведённые исследования показали, что QoS характеристиками, которые влияют на качество передачи данных в телекоммуникационной сети, является достоверность, своевременность и оперативность. Для решения задачи необходимо решить оптимизационную задачу с достижением времени доставки информации T_{1c} к минимуму. Определено, что задачу можно решить путём добавления в алгоритм доставки графической информации дополнительной компрессии со значительной потерей информации, которая может быть восстановлена по запросу пользователя. За счёт отсечения невостребованной информации появляется возможность освобождения дополнительных ресурсов, которые можно использовать для передачи других данных (повышения пропускной способности каналов передачи данных телекоммуникационных систем и сетей), которые в другом случае были бы перенаправлены по медленному каналу, или сброшены из очереди доставки.

Показано практическую нецелесообразность повсеместного внедрения новых более совершенных кодеков по причине инертности внедрения новых технологий и поддержке обратной совместимости. Предложена система с серверами промежуточного дополнительного сжатия графической информации.

Проведён анализ системы промежуточного дополнительного сжатия графического контента телекоммуникационной корпоративной сети на основе построенной математической модели изменения времени доставки сообщения (1.5). Получены математические зависимости между основными параметрами, на основе которых сделаны следующие выводы:

1. Установлена зависимость между уменьшением общего объёма трафика и оперативностью его доставки в обслуживании телекоммуникационной системой [27, 29].

2. Повышение оперативности системы и уменьшение общего трафика путём промежуточного дополнительного сжатия прогрессивным кодированием

графического контента гарантируется использованием более сильного сжатия и увеличением количества пользователей [36].

3. Установлена необходимость учёта загруженности серверов дополнительного сжатия графического контента, с целью предотвращения нарушений действия системы в виде отказа в обслуживании, путём прогнозирования загруженности посредством анализа временных рядов трафика через систему [30].

На основе полученных результатов сформулировано ряд задач, которые решаются в следующих разделах. Во втором разделе решается задача снижения трафика графической информации в телекоммуникационной корпоративной сети на основе построения улучшенного прогрессивного кодека графической информации с большим коэффициентом сжатия и выборочной потерей качества несущественной информации.

В третьем разделе решается задача разработки метода предсказания возможности отказов в обслуживании серверами телекоммуникационных сетей, по причине превышенного количества запросов, при помощи анализа временных рядов.

В четвёртом разделе рассмотрены рекомендации по практическому использованию разработанных методов повышения оперативности передачи данных и качества обслуживания в телекоммуникационных системах и сетях.

РАЗДЕЛ 2
РАЗРАБОТКА МЕТОДА ПОВЫШЕНИЯ ОПЕРАТИВНОСТИ ПЕРЕДАЧИ ДАННЫХ В ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМАХ И СЕТЯХ НА ОСНОВЕ УМЕНЬШЕНИЯ ТРАФИКА ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

В данном разделе произведено построение критериев оперативности передачи сообщений в телекоммуникационной системе или сети, сделаны теоретические и практические обоснования ключевых предположений, которые стоят в основе статистической модели сети. Предлагается схема с КЭШ-серверами, которые поддерживают сжатие с прогрессивной передачей графической информации.

2.1 Разработка критериев оперативности передачи данных в корпоративных сетях с графическим трафиком

В статье [109] приведено исследование поиска среднего времени доставки одного пакета в компьютерной сети. В качестве результата приведены формулы нахождения среднего математического времени доставки пакета, а также дисперсии времени доставки этого пакета:

$$D_{\frac{T_{ср}}{T_n}} = \frac{\left(T_{TA} + T_П\right)^2 \cdot \left(1 - P_{кв}\left(P_{пр} + P_{но}\right)\right)}{T_П^2 \cdot \left(P_{кв}\left(P_{пр} + P_{но}\right)\right)^2}, \quad (2.1)$$

где $\tau = \frac{T_{TA} + T_П}{T_П}$ – приведённое время на одну попытку передачи информации, а $p = P_{кв}\left(P_{пр} + P_{но}\right)$ – приведённая вероятность удачной доставки пакета информации. Однако для оценки статистических величин необходимо иметь значения многих коэффициентов непрямого измерения. Поэтому предлагается другой подход для получения вероятности количества попыток передачи пакетов доставки многопакетного сообщения. На основе практически измеренного времени доставки сообщения предлагается экспериментально

установить статистические характеристики сети, а также доказать пропорциональную зависимость времени доставки сообщения от его размера.

2.1.1 Построение математической модели зависимости времени передачи сообщения от количества пакетов передаваемой информации

Предположим что в сети, состоящей условно из одного сегмента, передаётся сообщение. Процесс передачи состоит из установки канала связи за время $\tau_к$ и передачи самих пакетов с данными. Вероятность успешного принятия пакета p, величина которой зависит от качества и вида канала связи. Вероятность утери пакета равна q=1-p, после утери производится повторная передача пакета. Существуют ситуации, когда пакет может передаваться несколько раз. В случае передачи множества пакетов, передающая сторона посылает следующий пакет, не дожидаясь получения подтверждения о правильном приеме предыдущего пакета. Потому, при неудачной передаче пакета, общее время увеличивается только на время повторной посылки одного пакета. Этот механизм отображен в формулах (2.2) как последовательность независимых испытаний до первой удачной попытки:

$$p_1(i) = pq^{i-1},\ \tau(i) = i\tau_p,\ \text{для } i \geq 1. \qquad (2.2)$$

Тут $p_1(i)$ – вероятность доставки одного пакета с i-той попытки; $\tau(i)$ – время занятости канала передачи при доставке пакета за i попыток. Зная значения времени и их вероятности (2.2), можно оценить общее среднестатистическое время [13, 21, 91] доставки сообщения (2.3):

$$\tau(i) = \tau_k + N\sum_{i=1}^{i_{\max}} pq^{i-1}\tau_p i, \qquad (2.3)$$

где N – количество пакетов в сообщении, которое пропорционально количеству информации в этом сообщении. Вследствие того, что τ_k, $\sum_{i=1}^{i_{\max}} pq^{i-1}\tau_p i$ постоянны для конкретного пути в телекоммуникационной сети, то время доставки

информации пропорционально количеству информации в сообщении, плюс постоянная установления канала связи и маршрутизации (2.4):

$$\tau(S) = SA + B\,, \tag{2.4}$$

где S – количество байтов в сообщении; A, B – искомые постоянные характеристики телекоммуникационного соединения. Проведём проверку и дополнение гипотезы (2.4).

2.1.2 Распределение вероятности доставки сообщения из *N* пакетов

Используя формулы (2.2) можно построить распределение вероятности доставки одного пакета сообщения. Такое распределение называется распределением Паскаля, которое задаёт вероятность первой удачной попытки в серии экспериментов.

Для построения распределения вероятности двух пакетов, нужно объединить распределения [21]. Два пакета требуют уже поочередной передачи, поэтому вероятность, что два пакета придут за i квантов времени τ_p есть сумма произведений всех вариантов $p(j)p(k)$ при $j+k=i$. Тогда для двух пакетов с распределениями (2.2) получим:

$$p_2(i) = \sum_{j=1}^{i-1} p_1(j)\cdot p_1(i-j) \to p_2(i) = \sum_{j=1}^{i-1} pq^{j-1}\cdot pq^{i-j-1} \to p_2(i) = p^2\sum_{j=1}^{i-1} q^{i-2}\,,$$

где p_2 – вероятность того что два пакета будут доставлены за время $i\tau_p$.
Окончательно:

$$p_2(i) = p^2 q^{i-2}(i-1),\ i\geq 1. \tag{2.5}$$

Аналогично можно получить распределение для передачи трёх пакетов:

$$p_3(i) = \sum_{j=1}^{i-1} p_2(j)\cdot p(i-j) \to p_3(i) = p^3\sum_{j=1}^{i-1}(j-1)q^{i-3} \to p_3(i) = p^3 q^{i-3}\sum_{j=1}^{i-1}(j-1),$$

Сумма представляет арифметическую прогрессию, сумма которой равна:

$$S_1 = \sum_{j=1}^{i-1}(j-1) \rightarrow S_1 = \frac{(i-1)(i-2)}{2}.$$

Откуда получаем общее выражение, пригодное для использования без суммирования:

$$p_3(i) = p^3 q^{i-3} \frac{(i-1)(i-2)}{2}. \qquad (2.6)$$

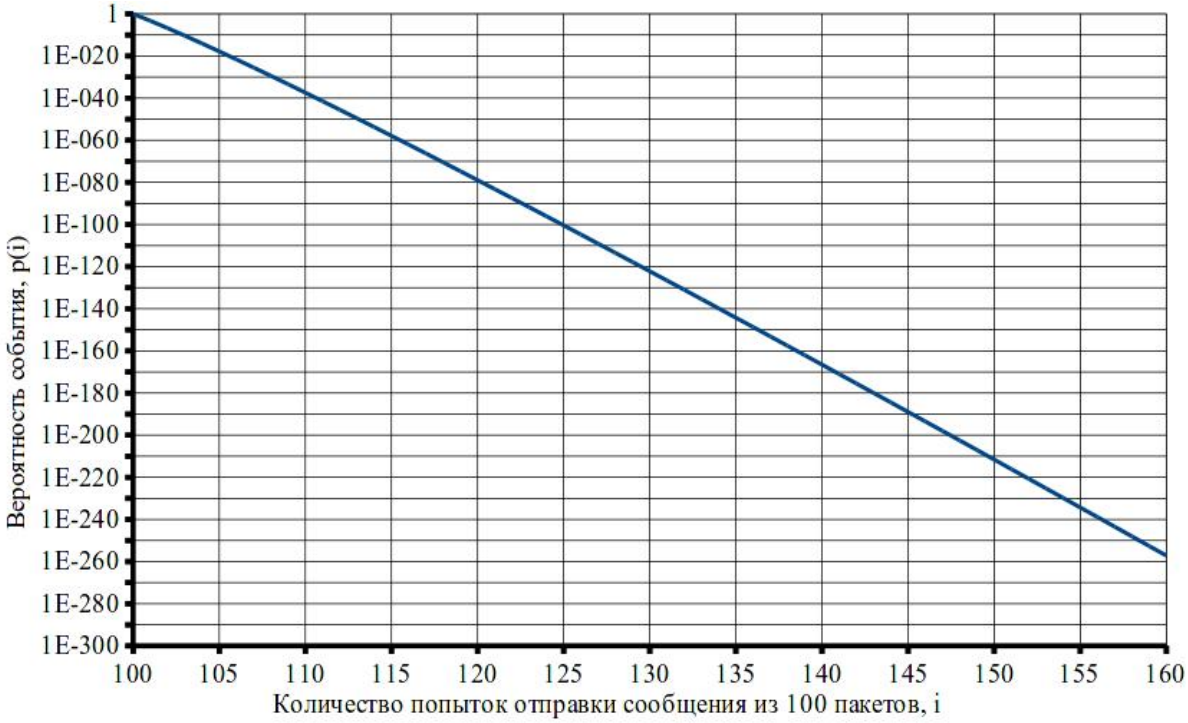


Рис. 2.1. Распределение вероятности доставки 100 пакетов при вероятности правильной доставки одного пакета p=0,99999

Теперь, для облегчения догадки про вид общей формулы для нахождения вероятности передачи N пакетов за i попыток, найдем $p_4(i)$. Для этого опять используем свёртку распределений вероятностей:

$$p_4(i) = \sum_{j=1}^{i-1} p_3(j) p_1(i-j) \rightarrow p_4(i) = \sum_{j=1}^{i-1} p^3 q^{j-3} \frac{(j-1)(j-2)}{2} pq^{i-j-1},$$

$$p_4(i) = p^4 \sum_{j=1}^{i-1} q^{j-3+i-j-1} \frac{(j-1)(j-2)}{2} \rightarrow p_4(i) = \frac{p^4 q^{i-4}}{2} \sum_{j=1}^{i-1} (j-1)(j-2).$$

Найдём значение суммы $S_4(i) = \sum j(j-1)$ с изменёнными границами для более простых выкладок. Для этого примем что $f(j) = j^2 - j$, и $S_4(i) = f(0) + f(1) + f(2) ... + f(i-2)$. Чтобы найти сумму, построим $F(j) = Aj^3 + Bj^2 + Cj$ такое, что $f(j) = F(j+1) - F(j)$. Тогда $f(j) = A(j+1)^3 + B(j+1)^2 + C(j+1) - Aj^3 - Bj^2 - Cj$, раскрыв скобки, получим $f(j) = j^2(3\text{A}) + j(3\text{A} + 2\text{B}) + (A + B + C)$. Из последнего равенства получим: $j^2 - j = j^2(3\text{A}) + j(3\text{A} + 2\text{B}) + (A + B + C)$. Теперь можно записать систему уравнений для определения искомых коэффициентов: $\begin{cases} 3\text{A} = 1 \\ 3\text{A} + 2\text{B} = -1 \\ A + B + C = 0 \end{cases} \rightarrow \begin{cases} A = 1/3 \\ B = -1 \\ C = 2/3 \end{cases}$, откуда $F(j) = j(j-1)(j-2)/3$. Заменим сумму *f*(*j*) разностями *F*(*j*+*1*)-*F*(*j*):

$$S_4(i) = (F(1) - F(0)) + (F(2) - F(1)) + ... + (F(i-1) - F(i-2)).$$

Сокращая *F*(*i*) одинаковыми аргументами, получим выражение для поиска суммы:

$$S_4(i) = F(i-1) - F(0) \rightarrow S_4(i) = \frac{(i-1)(i-2)(i-3)}{3}.$$

На основе полученного выражения для поиска суммы $S_4(i)$, запишем общую формулу для поиска вероятности отправки четырёх пакетов за *i* квантов времени (2.7).

$$p_4(i) = p^4 q^{i-4} \frac{(i-1)(i-2)(i-3)}{2 \cdot 3}. \qquad (2.7)$$

На основе формул (2.5), (2.6) и (2.7) можно предположить, что общий вид формулы имеет вид:

$$p_N(i) = p^N q^{i-N} \frac{(i-1)(i-2)\ldots(i-(N-1))}{(N-1)!}. \qquad (2.8)$$

Докажем что формула (2.8) даёт вероятность отправки N пакетов информации за i квантов времени.

Нам уже известна справедливость формул (2.5), (2.6) и (2.7). Поэтому, утверждение можно доказать методом математической индукции установив справедливость равенства: $p_{N+1}(i) = \sum_{j=1}^{i-1} p_N(j) p_1(i-j)$. Проведём подстановки и вынесем константы за знак суммирования:

$$p_{N+1}(i) = \sum_{j=1}^{i-1} p^N q^{j-N} \frac{(j-1)(j-2)\ldots(j-(N-1))}{(N-1)!} pq^{i-j-1},$$

$$p_{N+1}(i) = \frac{p^{N+1} q^{i-(N+1)}}{(N-1)!} \sum_{j=1}^{i-1} (j-1)(j-2)\ldots(j-(N-1)).$$

Аналогично выводу формулы (2.7) примем за $f(j) = j(j-1)(j-2)\ldots(j-(N-2))$, тогда $F(j) = j(j-1)(j-2)\ldots(j-(N-1))/N$. Проверим правильность догадки про вид $F(j)$, которая есть антиразность обобщённой степени в разностном исчислении:

$$f(j) = F(j+1) - F(j) \to \prod_{k=0}^{N-2}(j-k) = \frac{1}{N}(j+1)\prod_{k=0}^{N-2}(j-k) - \frac{1}{N}\prod_{k=0}^{N-1}(j-k),$$

$$\prod_{k=0}^{N-2}(j-k) = ((j+1)-(j-(N-1)))\frac{1}{N}\prod_{k=0}^{N-2}(j-k),$$

$$\prod_{k=0}^{N-2}(j-k) = (j+1-j+N-1)\frac{1}{N}\prod_{k=0}^{N-2}(j-k),$$

$$\prod_{k=0}^{N-2}(j-k) = N\frac{1}{N}\prod_{k=0}^{N-2}(j-k),$$

что подтверждает правильность предположения о виде $F(j)$. Теперь есть возможность заменить суммирование выражением $F(i\text{-}1)\text{-}F(0)$:

$$p_{N+1}(i) = \frac{p^{N+1} q^{i-(N+1)}}{(N-1)!} \sum_{j=1}^{i-1} f(j-1),$$

$$\sum_{j=1}^{i-1} f(j-1) = \sum_{j=1}^{i-1} \left(F(j+1-1) - F(j-1)\right),$$

$$\sum_{j=1}^{i-1} f(j-1) = F(i-1) - F(0),$$

$$\sum_{j=1}^{i-1} f(j-1) = \frac{(i-1)(i-2)\ldots(i-N)}{N},$$

$$p_{N+1}(i) = \frac{p^{N+1} q^{i-(N+1)}}{(N-1)!} \cdot \frac{(i-1)(i-2)\ldots(i-N)}{N},$$

и окончательно упрощая:

$$p_{N+1}(i) = p^{N+1} q^{i-(N+1)} \frac{(i-1)(i-2)\ldots(i-N)}{N!},$$

что соответствует формуле (2.8) при подстановке N:=N+1.

Утверждение доказано.

Используя распределение вероятности (2.8), построим математическое ожидание и дисперсию количества попыток на отправку N пакетов информации.

2.1.3 Среднее статистическое время и дисперсия доставки сообщения из *N* пакетов

В пункте 2.2.1 сделано предположение о линейности времени доставки информации от количества пакетов, которые эта информация занимает. Для полной уверенности нужно получить математическое ожидание времени доставки пакетов аналитически из распределения вероятности (2.8).

Ранее показано, что вероятность доставки сообщения из одного пакета за i попыток (1) $p_1(i) = pq^{i-1}$. Математическое ожидание количества попыток для успешной передачи пакета будет $M(p_1) = \sum_{i=1}^{\infty} pq^{i-1} i$. Для нахождения аналитического представления значения суммы ряда примем за $f(i) = q^i i$ выражение под знаком суммы, и найдём $F(i)$ такое что $F(i+1) - F(i) = f(i)$. Искать $F(i)$ будем в виде $F(i)=(Ai+B)q^i$. Тогда

$$F(i+1) - F(i) = (Ai + A + B)q^{i+1} - (Ai + B)q^i \rightarrow f(i) = q^i(iA(q-1) + q(A+B) - B).$$

Для равенства необходимо чтобы $\begin{cases} A(q-1) = 1 \\ q(A+B) - B = 0 \end{cases}$, откуда $\begin{cases} A = -\dfrac{1}{1-q} \\ B = -\dfrac{q}{(1-q)^2} \end{cases}$. Далее подставляя и сводя подобные получаем:

$$\sum_{i=1}^{\infty} f(i) = F(\infty + 1) - F(1) \rightarrow \sum_{i=1}^{\infty} f(i) = 0 - \left(-\frac{1}{1-q} - \frac{q}{(1-q)^2} \right) q,$$

$$M(p_1) = \frac{p}{q} \sum_{i=1}^{\infty} f(i),$$

$$M(p_1) = \frac{p}{q} \left(\frac{1}{1-q} + \frac{q}{(1-q)^2} \right) q,$$

$$M(p_1) = (1-q) \left(\frac{1}{1-q} + \frac{q}{(1-q)^2} \right).$$

Упростив последнее выражение, получаем среднее математическое количество попыток для передачи одного пакета данных, без учёта времени на установления канала связи и маршрутизации:

$$M(p_1) = \frac{1}{1-q}. \qquad (2.9)$$

С учётом того факта, что при получении распределений p_N использовано сложение независимых случайных величин, математические ожидания и дисперсии суммируются:

$$M(p_N) = \frac{N}{1-q}, \tag{2.10}$$

что согласуется с ранее полученным (2.3). Однако, с целью доказательства правильности получения обобщённой формулы распределения вероятности (2.8), найдём значения среднего времени ожидания доставки информации аналитически.

По определению среднестатистического

$$M(p_N) = \sum_{i=1}^{\infty} i p^N q^{i-N} \frac{(i-1)(i-2)...(i-(N-1))}{(N-1)!}.$$

С целью упрощения записей в дальнейших выкладках перепишем последнее выражение

$$M(p_N) = \frac{p^N}{q^N (N-1)!} \sum_{i=0}^{\infty} q^i i^{(N)},$$

где $i^{(N)}=i(i\text{-}1)(i\text{-}2)...(i\text{-}(N\text{-}1))$ – разностная степень, изменена граница суммы от значения 0. Изменение границы суммирования не изменит значение суммы, так как при $i=0$ значение под знаком суммы имеет значение ноль. Используя $F(i)$, для которого $F(i+1)\text{-}F(i)=f(i)$, где $f(i)=q^i i^{(N)}$, можно выразить значение суммы: $\sum_{i=0}^{\infty} f(i) = F(\infty) - F(0)$. Искать $F(i)$ нужно в виде:

$$F(i) = q^i \left(A_0 + A_1 i + A_2 i^{(2)} + ... + A_{N-1} i^{(N-1)} + A_N i^N \right).$$

С целью использования более короткой записи введём обозначение $\Delta F(i) = F(i+1) - F(i)$. Тогда $\Delta F(i) = \Delta G(i) \cdot V(i) + G(i) \cdot \Delta V(i) + \Delta G(i) \cdot \Delta V(i)$, обозначив $G(i) = A_0 + A_1 i + A_2 i^{(2)} + \ldots + A_{N-1} i^{(N-1)} + A_N i^N, V(i) = q^i$. Найдём разности:

$$\Delta G(i) = A_1 + 2A_2 i^{(1)} + \ldots + (N-1)A_{N-1} i^{(N-2)} + NA_N i^{N-1}, \Delta V(i) = -(1-q)q^i .$$

$$\Delta F(i) = q^i \Big((qA_1 - (1-q)A_0) + (q2A_2 - (1-q)A_1) i^{(1)} + \ldots$$
$$\Big) \ldots + (qkA_k - (1-q)A_{k-1}) i^{(k-1)} + \ldots + (qNA_N - (1-q)A_{N-1}) i^{(N-1)} - (1-q)A_N i^{(N)} .$$

Приравниваем коэффициенты $\Delta F(i)$ и $f(i)$ при равных степенях и получаем систему уравнений:

$$-(1-q)A_N = 1,$$
$$qNA_N - (1-q)A_{N-1} = 0,$$
$$q(N-1)A_{N-1} - (1-q)A_{N-2} = 0,$$
$$\ldots$$
$$q(k-1)A_{k-1} - (1-q)A_{k-2} = 0,$$
$$\ldots$$
$$q3A_3 - (1-q)A_2 = 0,$$
$$q2A_2 - (1-q)A_1 = 0,$$
$$qA_1 - (1-q)A_0 = 0.$$

Решив систему определяем что $A_0 = \dfrac{-q^N N!}{(1-q)^{N+1}}$. Учитывая, что $F(\infty) \to 0$, находим значение суммы

$$M(p_N) = \frac{p^N}{q^N (N-1)!} \left(0 - \frac{-q^N N!}{(1-q)^{N+1}} \right),$$

$$M(p_N) = \frac{N}{1-q},$$

которая совпадает с формулой (2.10). Этот результат доказывает правильность нахождения аналитического выражения (2.8) для поиска распределения вероятности.

Для дисперсии аналогично можно получить $D(p_N)=\sum_{n=1}^{N} D(p_1)$, откуда $D(p_N)=N\cdot D(p_1)$. Определим значение дисперсии для отправки одного пакета:

$$D(p_1)=\sum_{i=1}^{\infty} p\cdot q^{i-1}i^2-M^2(p_1)\rightarrow D(p_1)=\frac{p}{q}\sum_{i=1}^{\infty} q^i i^2-M^2(p_1).$$

Далее нужно найти значение суммы ряда $\sum_{i=1}^{\infty} q^i i^2$. Примем что $f(i)=q^i i^2$. Будем искать $F(i)$ такое, что $f(i)=F(i+1)-F(i)$. Антиразность $F(i)$ имеет вид: $F(i)=(Ai^2+Bi+C)q^i$. Откуда получим

$$F(i+1)-F(i)=q\left(A(i+1)^2+B(i+1)+C\right)q^i-\left(Ai^2+Bi+C\right)q^i.$$

Упрощая

$$f(i)=\left(q\left(Ai^2+2\mathrm{A}i+A+Bi+B+C\right)-\left(Ai^2+Bi+C\right)\right)q^i,$$
$$f(i)=\left(\left((q-1)Ai^2+\left(2\mathrm{A}q+B(q-1)\right)i+\left(qA+qB+C(q-1)\right)\right)q^i,$$

получаем систему уравнений для нахождения искомых коэффициентов:

$$\begin{cases}(q-1)A=1\\ 2\mathrm{A}q+B(q-1)=0\\ qA+qB+C(q-1)=0\end{cases},\quad \begin{cases}A=-1/(1-q)\\ B=-2q/(1-q)^2\\ C=-q(1+q)/(1-q)^3\end{cases}.$$

Теперь можно подставить найденные коэффициенты для поиска суммы ряда:

$$\sum_{i=1}^{\infty} f(i)=F(\infty+1)-F(1),\ \sum_{i=1}^{\infty} f(i)=0-\left(\frac{-1}{1-q}-\frac{2q}{(1-q)^2}-\frac{q(1+q)}{(1-q)^3}\right)q,$$

$$\sum_{i=1}^{\infty} f(i) = \left(\frac{1}{1-q} + \frac{2q}{(1-q)^2} + \frac{q(1+q)}{(1-q)^3} \right) q .$$

После этого можно записать и упростить выражения для вычисления дисперсии:

$$D(p_1) = (1-q)\left(\frac{1}{1-q} + \frac{2q}{(1-q)^2} + \frac{q(1+q)}{(1-q)^3} \right) - \frac{1}{(1-q)^2},$$

$$D(p_1) = 1 + \frac{2q}{1-q} + \frac{q(1+q)-1}{(1-q)^2},$$

$$D(p_1) = \frac{q}{(1-q)^2}.$$

И согласно свойству суммирования дисперсий, для передачи N пакетов информации имеем дисперсию:

$$D(p_N) = N \frac{q}{(1-q)^2}. \qquad (2.11)$$

где q – вероятность отложенной передачи пакета, по причине ненадёжности канала передачи или ожидания в очереди.

Также существует возможность восстановить вероятность утери пакета или его задержки в очереди на промежуточных узлах сети по экспериментально определённой дисперсии времени доставки пакетов информации, выразив эту вероятность из выражения (2.11):

$$q = 1 + \frac{N - \sqrt{N(4D_N + N)}}{2D_N}. \qquad (2.12)$$

В реальной сети пакеты передаются по цепи сегментов, в каждой из которых происходят задержки в передаче пакета. Однако, как показано раньше, каждый сегмент имеет характеристику среднего времени на доставку пакета,

что даёт для полного времени передачи пакета от отправителя к получателю сумму времён ожиданий на отдельных сегментах. Также этот вывод можно подтвердить теоремой о последовательных случайных событиях с распределением времени ожидания $f_1(t)$ и $f_2(t)$. В [13] доказано, что математическое ожидание объединённого распределения $M(f_1(t)*f_2(t))$ последовательного наступления обоих событий, будет сумма математических ожиданий распределений $M(f_1(t))$ и $M(f_2(t))$:

$$M\left(f_1(t) * f_2(t)\right) = M\left(f_1(t)\right) + M\left(f_2(t)\right). \qquad (2.13)$$

Практически это означает, что для цепи сегментов будет получено аналогичное распределение со средним временем ожидания сообщения, которое равно сумме средних времён ожидания сообщения на отдельных сегментах сети (при условии неизменности маршрута).

Закон сложения математических ожиданий при сложении распределений последовательных случайных величин даёт важный вывод: ***участок многосегментной сети можно заменить статистически равноценной односегментной связью.***

Факт соответствия теоретических результатов с экспериментально полученными характеристиками сети при многократной загрузке файлов подтверждён в главе 4.1, в разделе посвящённому экспериментальной работе.

Как результат можно выделить:

1. Время доставки информации в телекоммуникационной сети имеет линейную зависимость от количества этой информации.

2. Уменьшение количества передаваемой информации в k раз, во столько же раз уменьшает среднее время доставки информации, а также в k раз уменьшает дисперсию времени доставки сообщения.

3. Экспериментальное нахождение дисперсии времени доставки сообщения позволяет оценить вероятность отложенной передачи пакета информации по выражению (2.12).

4. Использование дополнительного сжатия информации и отбрасывание незначительной информации не только сократит время её доставки, но и за счёт уменьшения дисперсии, также снизит отклонения от среднего времени

доставки. Этот результат можно использовать практически, например, для разработки протокола более точной синхронизации часов посредством телекоммуникационной сети.

2.2 Разработка метода повышения эффективности передачи информации в телекоммуникационных корпоративных системах и сетях при помощи усовершенствованного прогрессивного кодека графического контента на основе SPIHT

Как показано в главе 1.4, на качество работы системы сильно влияет размер дополнительно сжатой графической информации [5, 6, 8]. Уменьшить количество передаваемой информации можно используя более совершенные методы кодирования изображений. Например, трёхмерный вариант вейвлет преобразования был использован для создания более совершенного кодирования видео (результаты и сервис для кодирования-проигрывания видео представлены на сайте codec.ru).

2.2.1 Сравнение битовых плотностей при использовании различных методов кодирования графической информации

Основными методами, которые претендуют на использование в системе дополнительного сжатия, является JPEG2000 и SPIHT. Однако можно показать, что избыточность информации в упакованном SPIHT изображении имеет большую степень. Это можно показать, установив энтропию информации результата упаковки изображения. Учитывая, что кодирование в обоих методах побитовое, установим отношение между единичным битом и нулевым. Для удобства значения отношений можно взять на каждый килобайт файла, рис. 2.2, 2.3, 2.4.

Из приведённых графиков видна неравномерность распределения нулевых и единичных битов в исходном не кодированном изображении. При получении кодированного изображения JPEG2000, происходит избавление от

избыточной информации, что приводит к равномерному распределению вероятностей считывания единичного и нулевого бита.

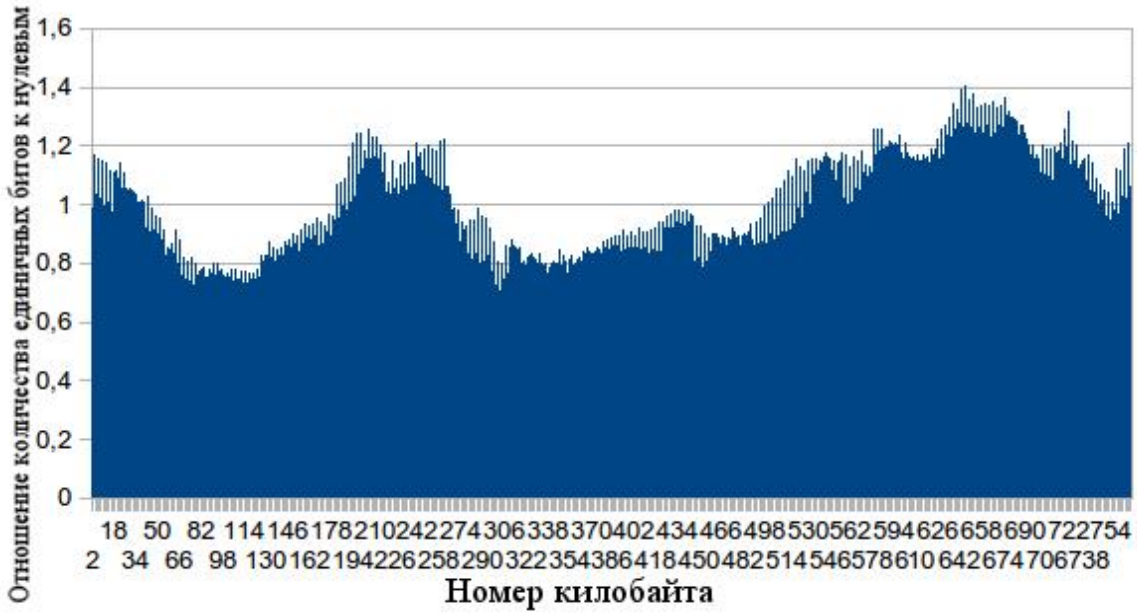


Рис. 2.2. Отношение между количеством нулевого и единичного бита для не кодированного изображения bmp

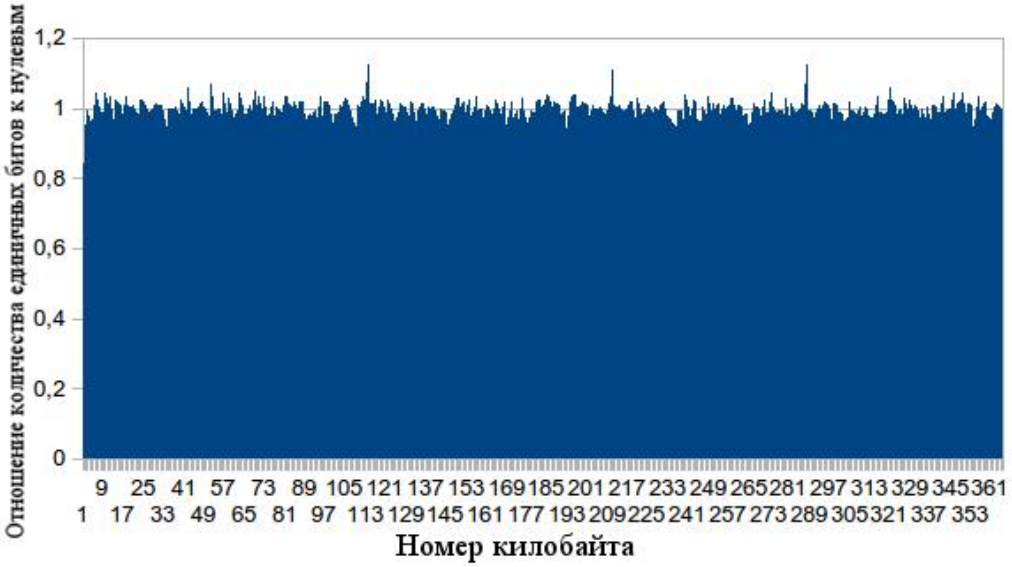


Рис. 2.3. Отношение между количеством нулевого и единичного бита для кодированного изображения JPEG2000

Это говорит о качественном сжатии JPEG2000, но SPIHT кодирование по своей сути предпочитает передачу множествами по битовым слоям, что

приводит к более частой передачи нулевого бита, а после, когда существенных пикселей становиться больше – чаще передаётся единица (рис. 2.4).

Ситуация с неравномерным распределением соотношения нулевых и единичных битов является следствием особенностей SPIHT кодирования и отсутствия дополнительного сжатия, которое присутствует в JPEG2000. Для ситуации неравномерного побитового кодирования, когда один бит имеет дробное значение количества информации в результирующем файле, подходит реализация математического сжатия [9, 11, 19, 22, 23].

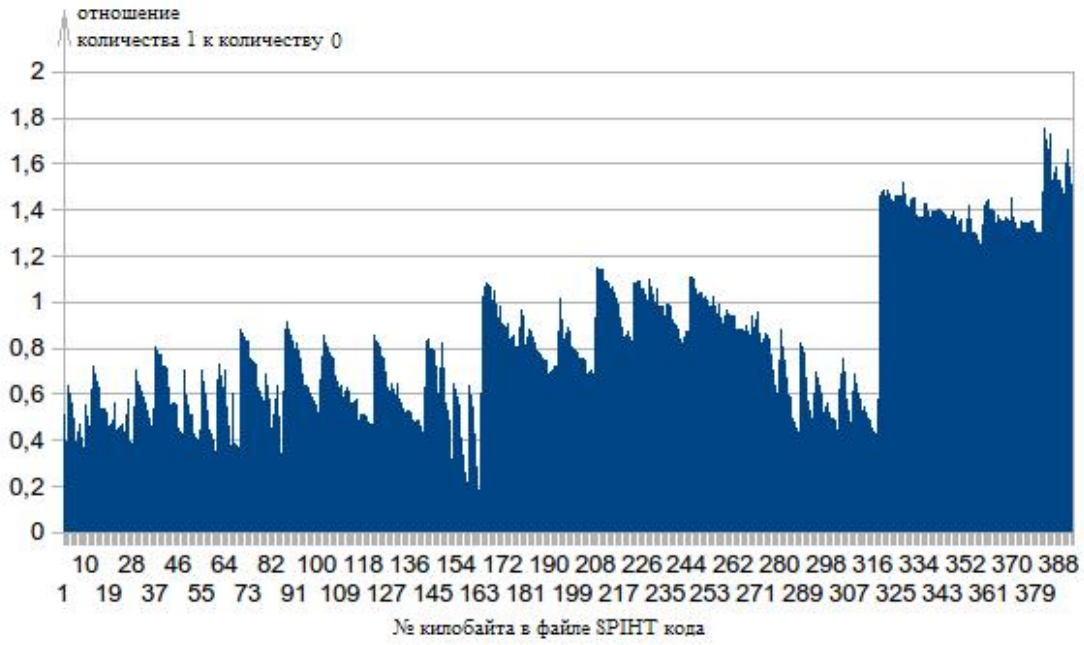


Рис. 2.4. Отношение между количеством нулевого и единичного бита для кодированного изображения SPIHT

На рис. 2.4 показано соотношение единичных битов к нулевым, которое не имеет постоянное значение и довольно резко изменяется не только при переходе от одного битового поля к другому, но и от одной зоны изображения к другой. Поэтому использование постоянного коэффициента для вероятностей попадания 0 или 1, для получения минимального размера выходного файла эти вероятности должны меняться. Такое кодирование будет называться адаптивным, одним из таких алгоритмов является Q-кодирование. Однако в ходе адаптивного кодирования параметры кодека меняются с запаздыванием, что негативно сказывается на степени сжатия. Поэтому необходимо предусмотреть блочное кодирование, для которого отдельно определяют

вероятности знаков. Для этого необходимо определиться с размером блока и точностью задания вероятности поступления единичного бита.

Пусть для записи отношения доли единичных битов на N_i битов файла используется запись с фиксированной точкой размером в b бит: $p(1)=1-p(0)$. Тогда размер кода будет

$$N_i^* = -N_i\left[p(0)\log_2 p(0) + p(1)\log_2 p(1)\right] + b, \qquad (2.14)$$

где N^*_i размер упакованной части N_i, $p(0)$ вероятность появления нулевого бита равного N_{i0}/N_i, $p(1)$ вероятность появления единичного бита равного N_{i1}/N_i. Используя формулы определения вероятностей, исключаем из (2.14) количества единичных и нулевых бит, оставив только вероятность единичного бита и размер кодированной части изображения:

$$N_i^* = -N_i\left((1-p(1))\log_2\left((1-p(1))\right) + p(1)\log_2\left(p(1)\right)\right) + b, \qquad (2.15)$$

где первое слагаемое зависит от свойств информации, а второе является константой с фиксированной запятой размером в 16 или 8 бит, которая содержит значение вероятности $p(1)$. Полный размер файла при этом будет суммой размеров упакованных блоков:

$$N^* = -\sum N_i\left((1-p(1))\log_2(1-p(1)) + p(1)\log_2 p(1)\right) + n\cdot b, \qquad (2.16)$$

где n – количество частей для упаковки изображения. Коэффициент сжатия энтропийным методом в зависимости от $p(1)$ показано на рис. 2.5. При этом в части для кодирования наблюдается колебания вероятности появления 1, поэтому степень сжатия маленьких кусков данных должны дать лучший результат. Однако, при увеличении количества частей, нужно использовать большее количество информации для записи значения вероятностей, слагаемое nb в формуле (2.16).

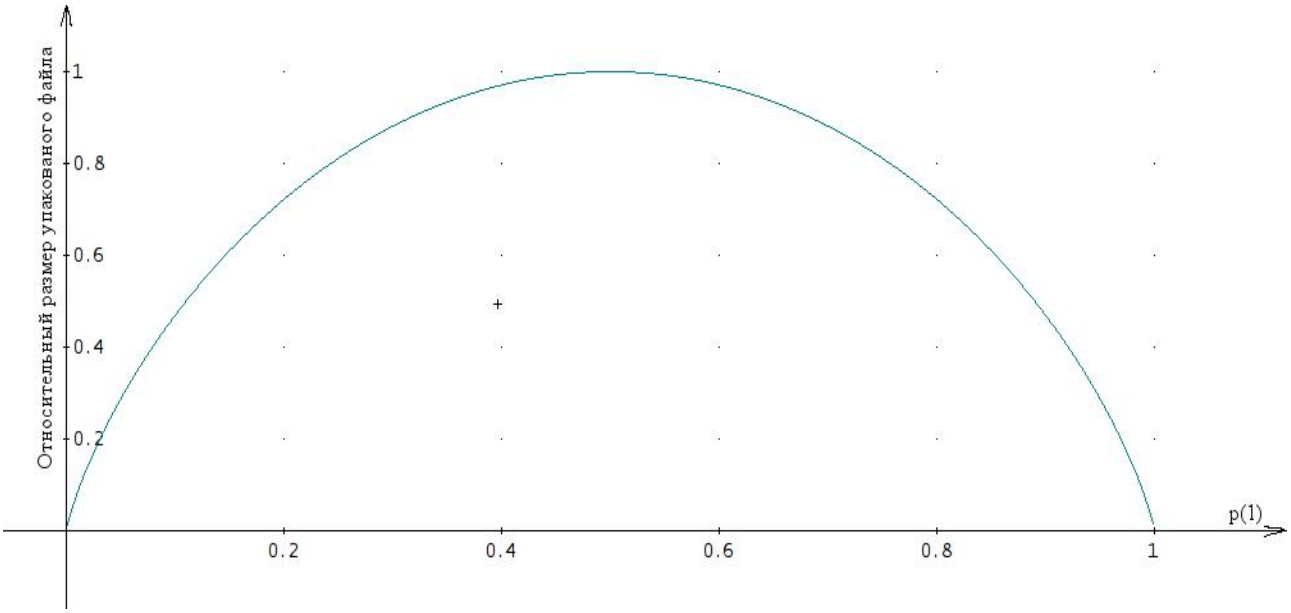


Рис. 2.5. Зависимость коэффициента сжатия от вероятности попадания единицы

Это приводит к задаче оптимизации:

$$N^*(n) = \frac{-N}{n} \sum \left[(1 - p_i(1)) \log_2 (1 - p_i(1)) + p(1)_i \log_2 p_i(1) \right] + n \cdot b \to min. \qquad (2.17)$$

Смысл такой оптимизации состоит в более точном определении вероятности появления единичного бита на участке кода, но для записи такого коэффициента тоже нужно выделять определённое количество информации. При увеличении количества промежутков *n* вдвое, количество информации для записи коэффициентов тоже вырастает вдвое. Этот рост должен быть компенсирован улучшением степени сжатия изображения. Пример подбора количества промежутков показано на рис. 2.6. Определённо, что при избыточном делении потока информации на блоки, степень сжатия может не только ухудшиться, но и в результате можно получить файл большего размера, чем исходный.

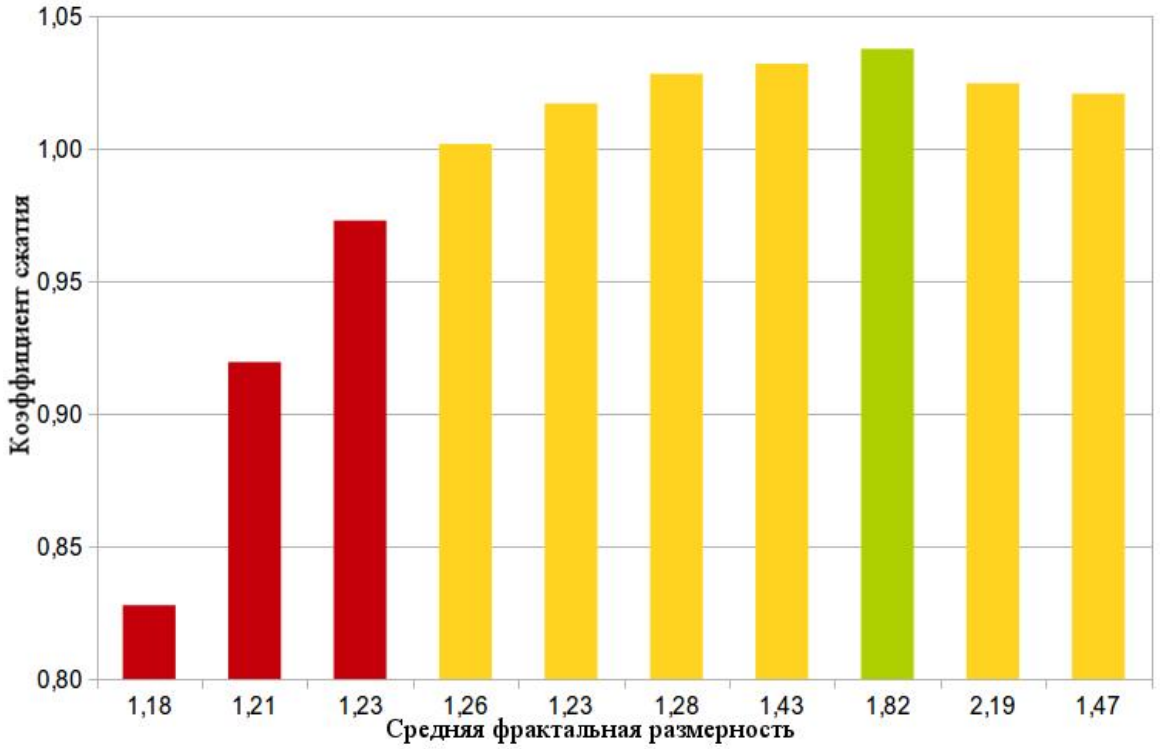


Рис. 2.6. Эксперимент сжатия по блокам
(по вертикали степень сжатия, по горизонтали фрактальная размерность изменения оценки вероятности выборки единичного бита; промежутки уменьшались вдвое)

Принять решение о дальнейшем уменьшении блоков с обобщённой информацией о доли единичных битов можно из анализа формулы (2.15). Допустим, что размер блока уменьшается вдвое, при этом обозначим:

$$p_{2i,2n}(1) = p_a,\ p_{2i+1,2n}(1) = p_b,\ p_{i,n} = p,\ p = (p_a + p_b)/2.$$

Понять во сколько раз измениться степень сжатия информации одного блока можно из соотношения:

$$K = \frac{2\left(p\log_2 p + (1-p)\log_2(1-p)\right)}{p_a\log_2 p_a + (1-p_a)\log_2(1-p_a) + p_b\log_2 p_b + (1-p_b)\log_2(1-p_b)}. \qquad (2.18)$$

$K(p_a,p_b)$ – симметричная относительно аргументов функция, которая равна 1 при $p_a=p_b$, что означает отсутствие изменения в размере сжимаемой информации и в этом случае пользы от разбиения потока данных на более мелкие части не будет. Когда $p_a \neq p_b$, то K>1, и это (рис. 2.7) означает уменьшение исходного количества информации на передачу.

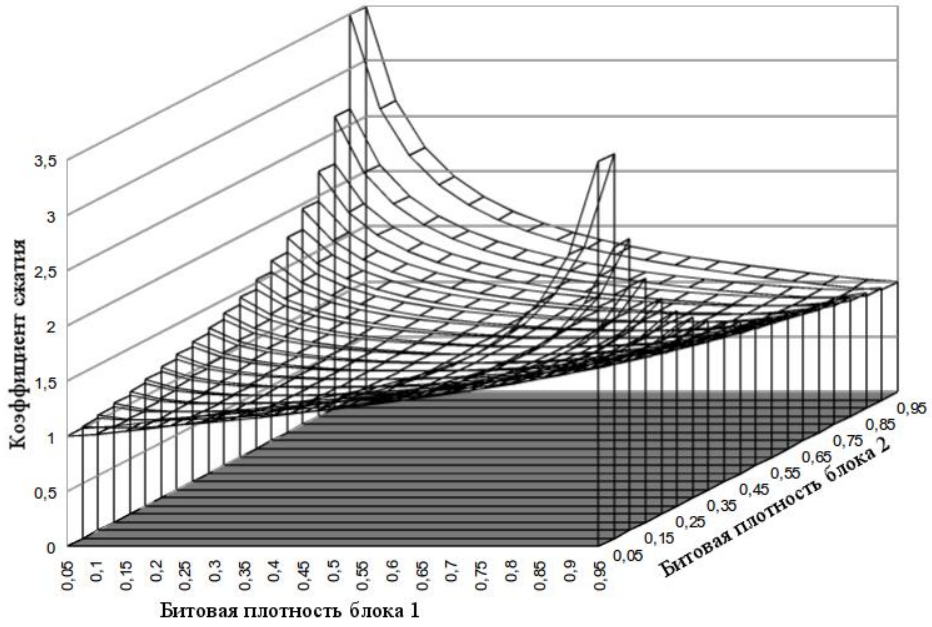


Рис. 2.7. Зависимость коэффициента уменьшения количества информации при разбиении блока кодирования $p=p_a+p_b$

Из рисунка 2.7 с графиком $K(p_a,p_b)$ видно, что степень сжатия информации значительно возрастает при максимальной разнице между p_a и p_b. Для принятия решения об уменьшении блоков кодирования необходимо иметь оценку $|p_a-p_b.|$. Произвести оценку можно при помощи понятия фрактальной размерности числовой последовательности [104] (рис. 2.8, 2.9).

С целью показать применимость определения фрактальной размерности для оценки улучшения сжатия информации рассмотрим два крайних случая:

1) Фрактальная размерность F близка к 2 (рис. 2.8). При увеличении числа отчётов кривой, эта кривая постепенно заполняет плоскость – новые изломы имеют размах соизмеримый с предыдущими. Как следствие, из (2.17) мы получаем высокий коэффициент сжатия.

2) Фрактальная размерность F ближе 1 (рис. 2.8). В этом случае уточнённая кривая будет практически повторять предыдущую, и уточняя вероятность выборки единицы увеличить коэффициент сжатия информации не удастся.

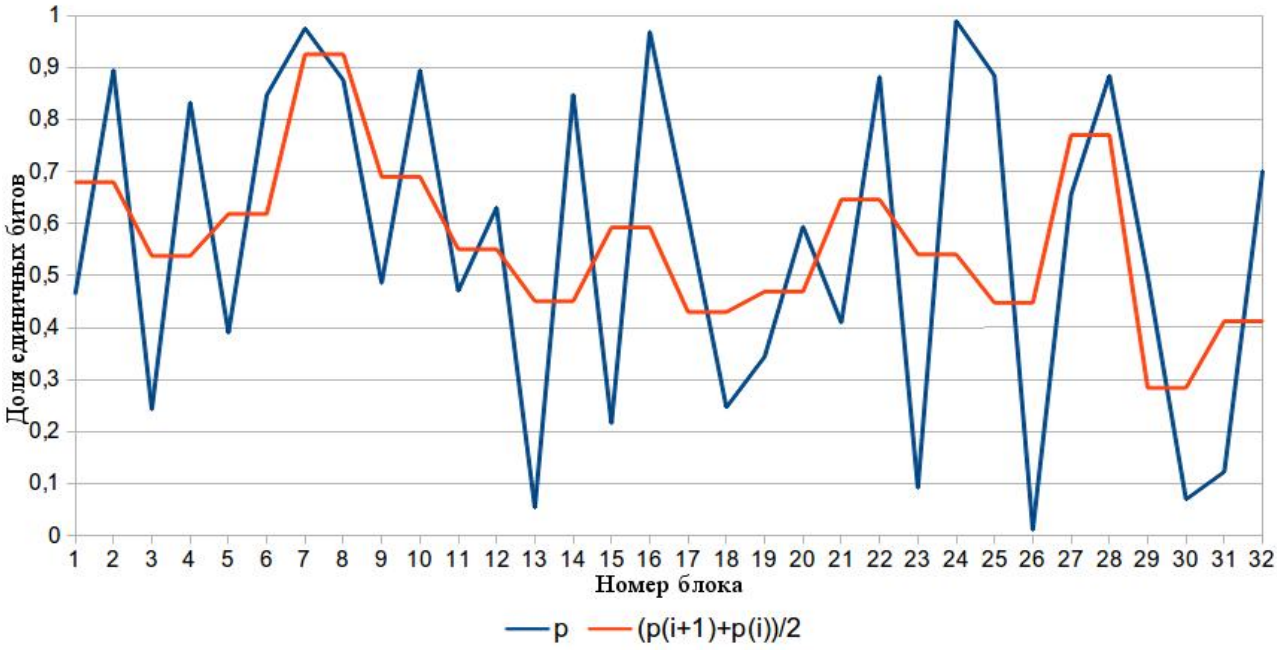


Рис. 2.8. Уточнение линии с фрактальной размерностью 1,8

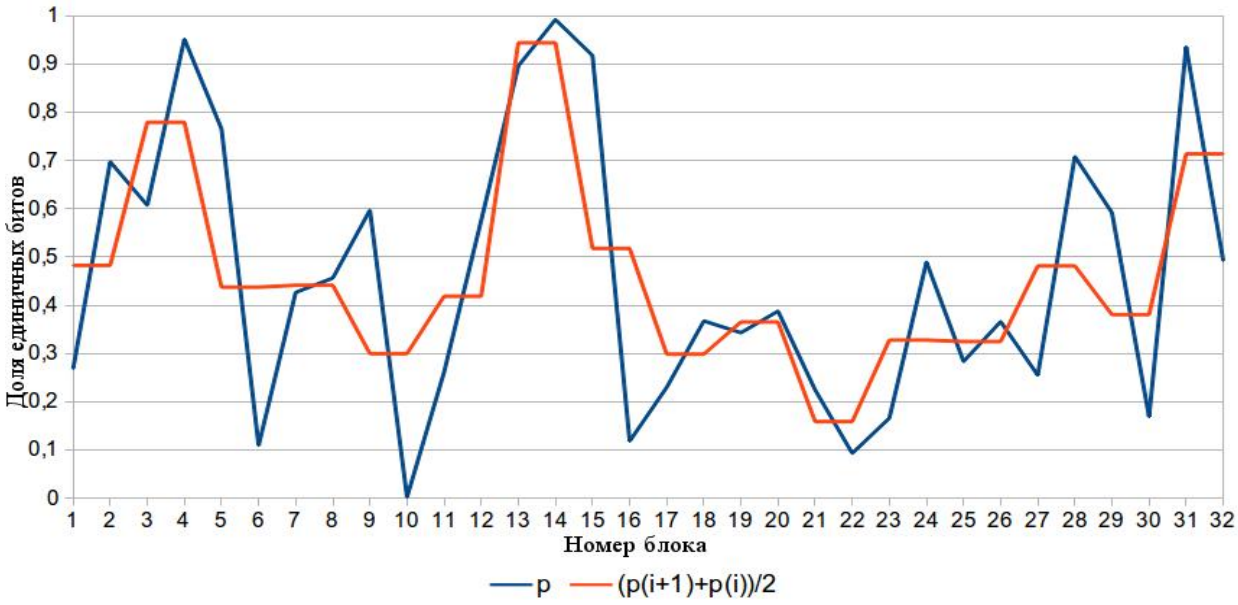


Рис. 2.9. Уточнение линии с фрактальной размерностью 1,12

Для последовательности вероятностей $\{p_0,p_1,...,p_{2n-1}\}$ на более крупных блоках получим набор вероятностей $\{(p_0+p_1)/2,(p_2+p_3)/2,...,(p_{2n-2}+p_{2n-1})/2\}$ на условном единичном отрезке. Длины кривых для первой и второй последовательностей будут:

$$L_1 = \sum p_i \frac{1}{2\mathrm{n}},\ L_2 = \sum \frac{p_{2\mathrm{i}} + p_{2\mathrm{i}+1}}{2} \cdot \frac{1}{n}.$$

Локальная фрактальная размерность кривой будет

$$F(n) = -\log_2(L_2/L_1).$$

Размерность тут понимается как число F, которое показывает, в какую степень нужно возвести a, что бы получить коэффициент удлинения кривой, уменьшив «линейку» в a раз. При длине L линия имеет n отрезков $\{(p_i,i/n);(p_{i+1},(i+1)/n)\}$. Средняя длина одного отрезка будет $<L>=L/n$. При увеличении количества блоков от n_1 до n_2 ($n_1<n_2$), тогда удлинение линии будет примерно равно $L_2 \approx L_1(n_2/n_1)^F$ и $<L_2>=L_1(n_2/n_1)^F/n_2$. Здесь и далее обозначение $<L>$ используется для среднего арифметического или математического ожидания. Окончательно, средняя разность между последовательными значениями вероятностей появления единичного бита будет

$$\langle|\Delta p_2|\rangle = \langle|\Delta p_1|\rangle\left(\frac{n_2}{n_1}\right)^{F-1}. \qquad (2.19)$$

Для начальной разбивки последовательности битов, которая состоит из n_1 сегментов фрактальной размерности F, перешли к разбивке из $n_2=zn_1$ сегментов. Из (2.19) следует

$$\langle|\Delta p_2|\rangle = \langle|\Delta p_1|\rangle z^{F-1} . \quad (2.20)$$

Смысл этого выражения заключается в постоянном возрастании различия плотности единичных бит при рассмотрении всё более коротких подпоследовательностей из главной последовательности. Из соотношений (2.18) и (2.20) следует увеличение коэффициента сжатия K к некоторому пределу. Наиболее пессимистичной оценкой изменения коэффициента сжатия будет, если принять p=0,5, p_a=p-Δp/2, p_b=p+Δp/2:

$$K(F,z) = \frac{z\log_2 0{,}5}{\sum\left(0{,}5+z^{F-1}/2\right)\log_2\left(0{,}5+z^{F-1}/2\right)+\left(0{,}5-z^{F-1}/2\right)\log_2\left(0{,}5-z^{F-1}/2\right)}, \quad (2.21)$$

.

Однако, при дополнении каждого блока информацией о доле единичных битов, потребуется дополнительная информация размером в zb. Задача оптимизации (2.17) сводится к отысканию минимума уравнения с одной натуральной переменной z:

$$N / K(F,z) + zb \to min . \quad (2.22)$$

Целевая функция (2.22) не линейная, поэтому её минимум ищется при помощи численных методов. Задача является простой, поскольку z может принимать только натуральные значения и имеет только один минимум по причине монотонности второй производной (2.21) и монотонности zb.

2.2.2 Принцип разделения битовых полей в SPIHT кодировании и его модификация по принципу минимизации относительных погрешностей

Просмотр графической информации в телекоммуникационных сетях в большинстве случаев происходит в два этапа, выбор актуального контента, а потом его просмотр. При оценке актуальности графического контента, пользователь коммуникационной сети не нуждается в максимально допустимом качестве. В особых случаях, когда пропускная способность канала передачи существенно низкая, возникает проблема оперативной доставки изображения с максимальным качеством и ограничением по времени доставки. В таком случае заранее неизвестно количество информации, которую получит приёмник, и нужно обеспечить передачу наиболее информативных частей в первую очередь. SPIHT кодирование наиболее отвечает поставленным требованиям, по критерию первоочерёдности передачи максимальной энергии сигнала [99].

Классическое использование алгоритма для прогрессивной передачи вейвлет коэффициентов преобразованного изображения разделяет массив коэффициентов на существенные и не существенные коэффициенты и не существенные множества. В не существенных множествах каждый из коэффициентов меньший определенной величины. Результатом является передача коэффициентов только старших битов значимых вейвлет коэффициентов. Следующим шагом список существенных пикселей увеличивается, и передается информация о значении битов следующей битовой плоскости. Фактически, для байтовых коэффициентов, передается информация о содержании бита 128, следующим шагом о наличии битов для 64, и так далее. При этом наличие маломощных составляющих всплесков передается в последнюю очередь, группировка малых коэффициентов в не существенное множество позволяет одним битом передать нулевой бит для всего множества пикселей. За счет этого и достигается уменьшение количества битов для передачи. В случае прекращения передачи содержимого файла, будут приняты коэффициенты в неполном виде с равномерной потерей младших битов. На

рис. 2.10 серым выделением обозначено еще не полученные биты, которые взяты как среднее возможное значение [35, 37, 49]:

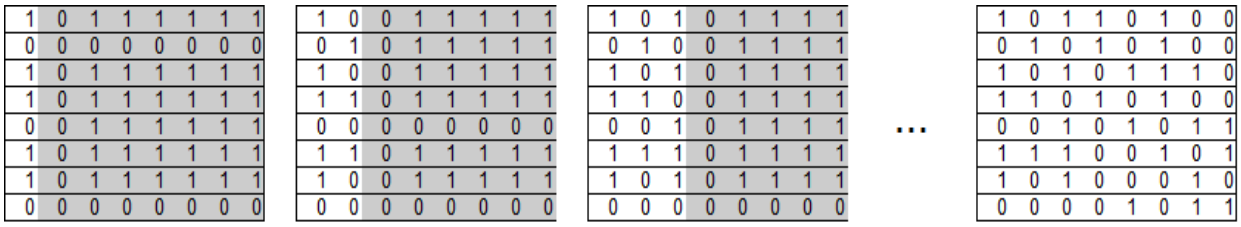


Рис. 2.10. При SPIHT передаче, первоочерёдно передаются старшие биты вейвлет коэффициентов

Анализ работ позволяет сформулировать заключение, которое подтверждает актуальность улутшенного отображения контурной информации за счёт снижения общего качества изображения [64, 118, 119, 120, 121, 122, 125].

Для улучшения детализации изображения предлагается передавать следующие значимые биты, сохранив относительную погрешность коэффициентов.

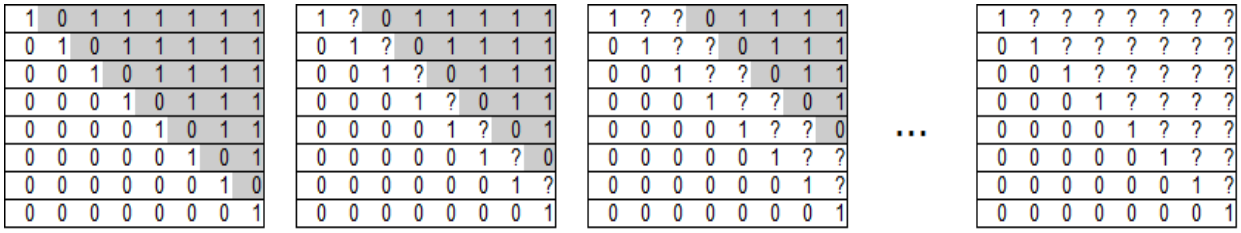


Рис. 2.11. Передача вейвлет коэффициентов с сохранением относительной погрешности

В этом случае схема передачи значимых битов будет изображаться схеме на рис. 2.11. При этом уточнения амплитуды длинных всплесков происходит только после получения информации про все детали изображения. Визуально неполно принято изображение будет иметь зональные искажения яркости и цветности без потерь малоконтрастных деталей.

Фактически это означает, что при меньшем количестве принятой информации глаз человека увидит больше деталей на изображении, в котором сохранялась в первую очередь относительная погрешность вейвлет коэффициентов.

Следствием перестановки последовательности бит передачи размер переданной информации и степень сжатия не измениться, однако возрастает количество переданных деталей при неполной передаче данных или при сильных коэффициентах сжатия. В режиме предварительной оценки актуальности изображения это может сократить трафик передачи на 1%-50% в зависимости от типа изображения.

а) отсечение младших битов б) дифференциация отсечения

Рис. 2.12. Зрительное сравнение изображение с сохранение абсолютной погрешности и относительной погрешности вейвлет коэффициентов

Классический метод SPIHT кодирования представлен блок-схемой, которая показана на рис. 2.13. Модификация проводиться на этапе передачи значимых битов. При этом значимые биты передаются только после того, как на приёмнике уже будет получена информация о знаке и старшем единичном бите всех вейвлет коэффициентов (алгоритм на рис. 2.14).

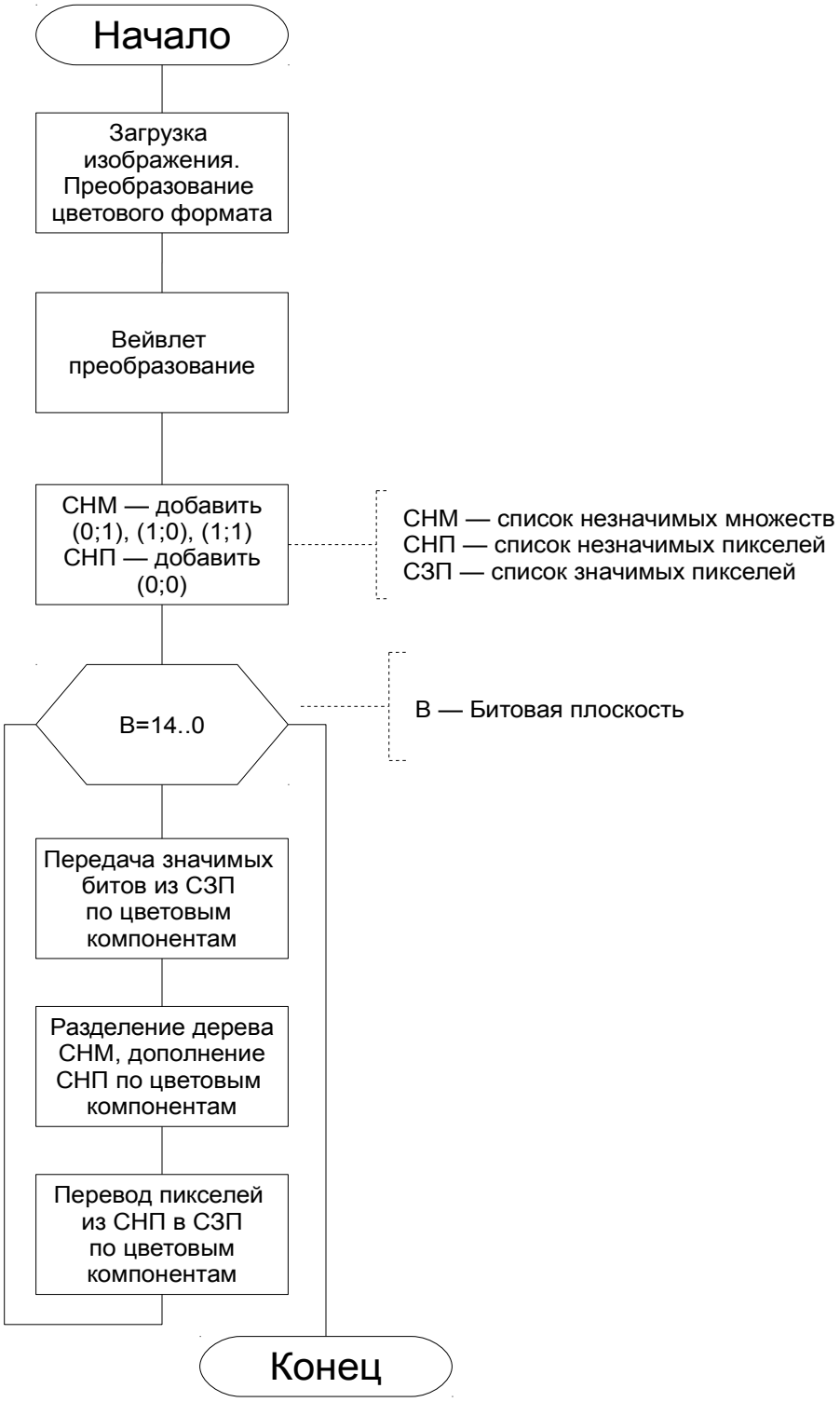


Рис. 2.13. Разновидность алгоритма SPIHT кодирования

Такая модификация возможна благодаря уже созданным спискам значимых вейвлет коэффициентов с однозначной их последовательностью. Поэтому очерёдность передачи значимых битов полностью повторяется на приёмнике графической информации.

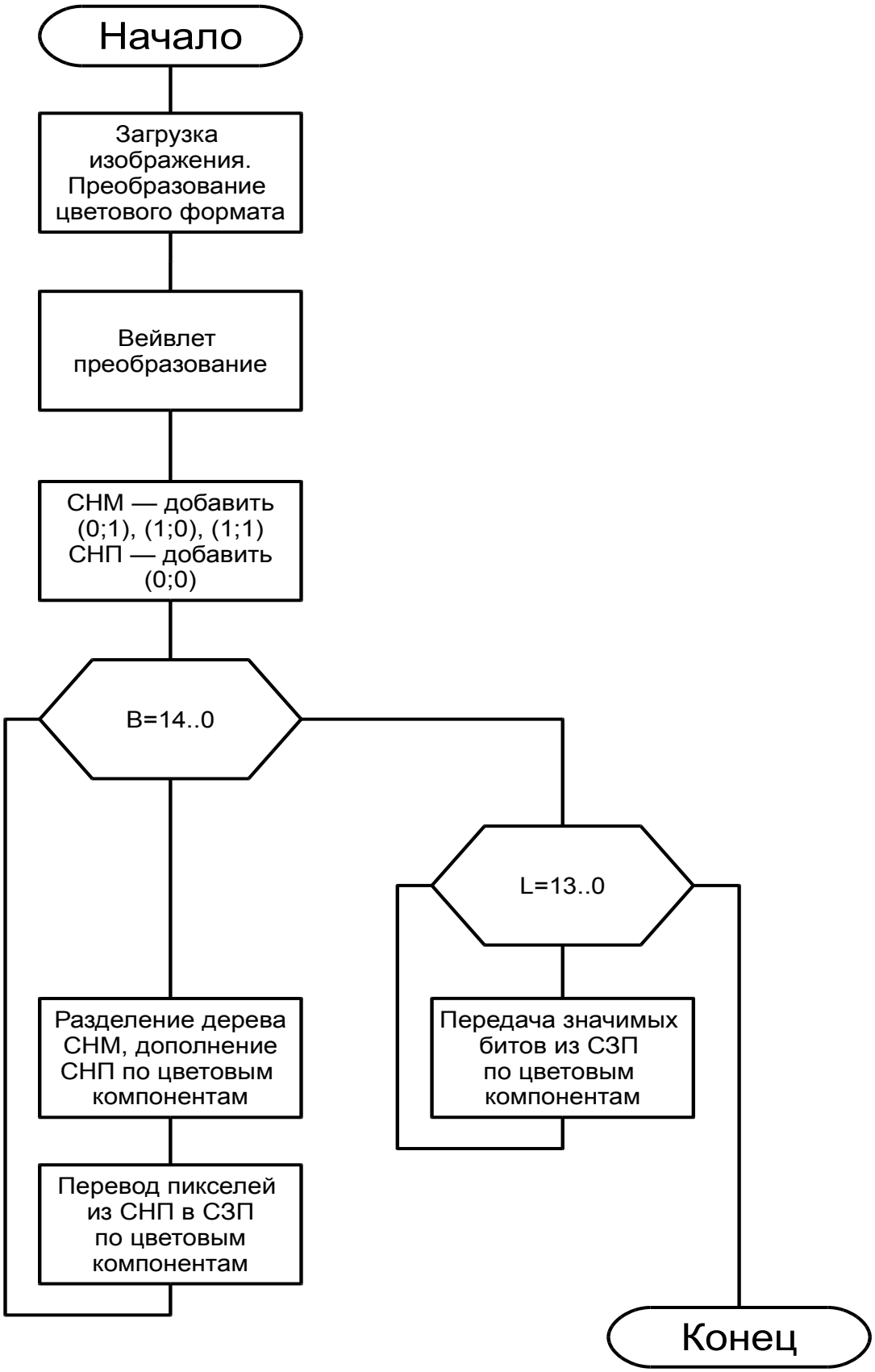


Рис. 2.14. Модификация SPIHT_O с отложенной передачей битов значимых пикселей

2.3 Разработка структуры и алгоритмов работы серверов дополнительного сжатия графического контента телекоммуникационных корпоративных систем и сетей

Разработка порядка работы системы повышения оперативности доставки часто используемой информации за счёт снижения количества передачи низко приоритетной информации требует более подробного описания. Более детальная схема показана на рис. 2.15, прототип которой представлен на рис. 1.4.

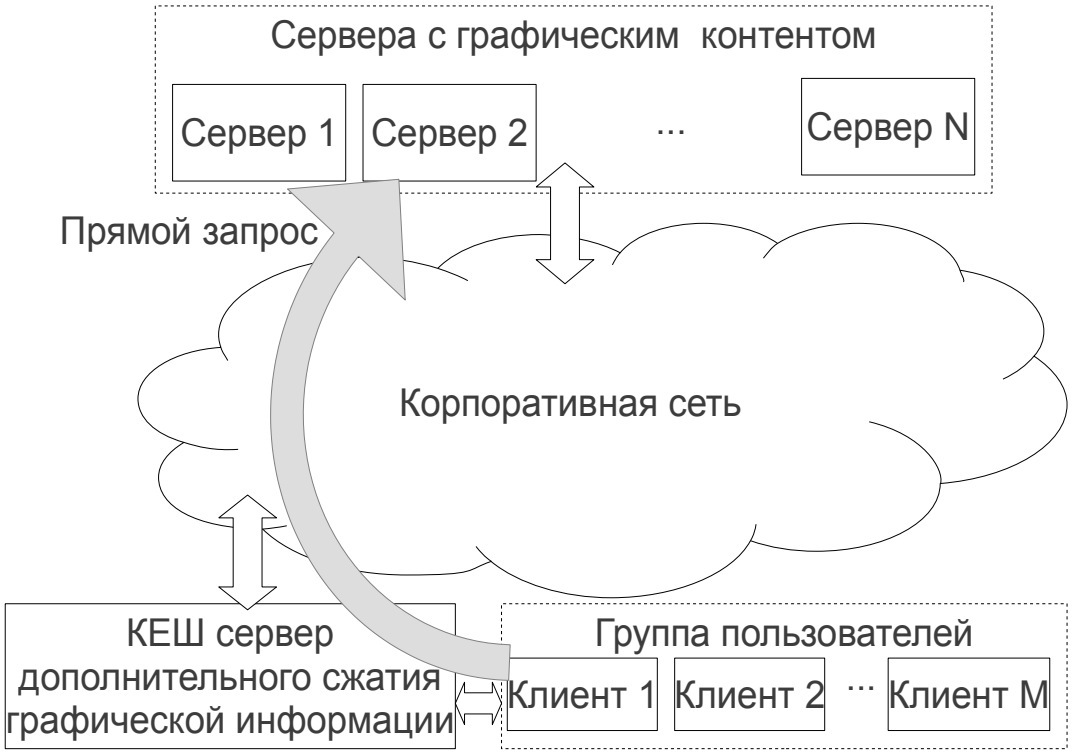


Рис. 2.15. Структура системы уменьшения трафика низкоприоритетной графической информации

Наряду с использованием более качественного прогрессивного сжатия информации (выводы раздела 1) необходимо учитывать ситуации, при которых информация отсутствует на промежуточном сервере и запрос идёт к оригинальным данным. При этом конечный пользователь не будет ожидать дополнительного сжатия, возможно длительного, по причине более сложного алгоритма кодирования. Наполненность сервера и частота запросов к информации, которая отсутствует на сервере, сильно влияет на

результирующую оперативность доставки информации. Потому нужно учитывать динамику наполнения сервера.

Структура, представленная на рис. 2.15, диктует алгоритм взаимодействия клиента с предложенной системой. Алгоритм показан на блок-схеме рис. 2.16.

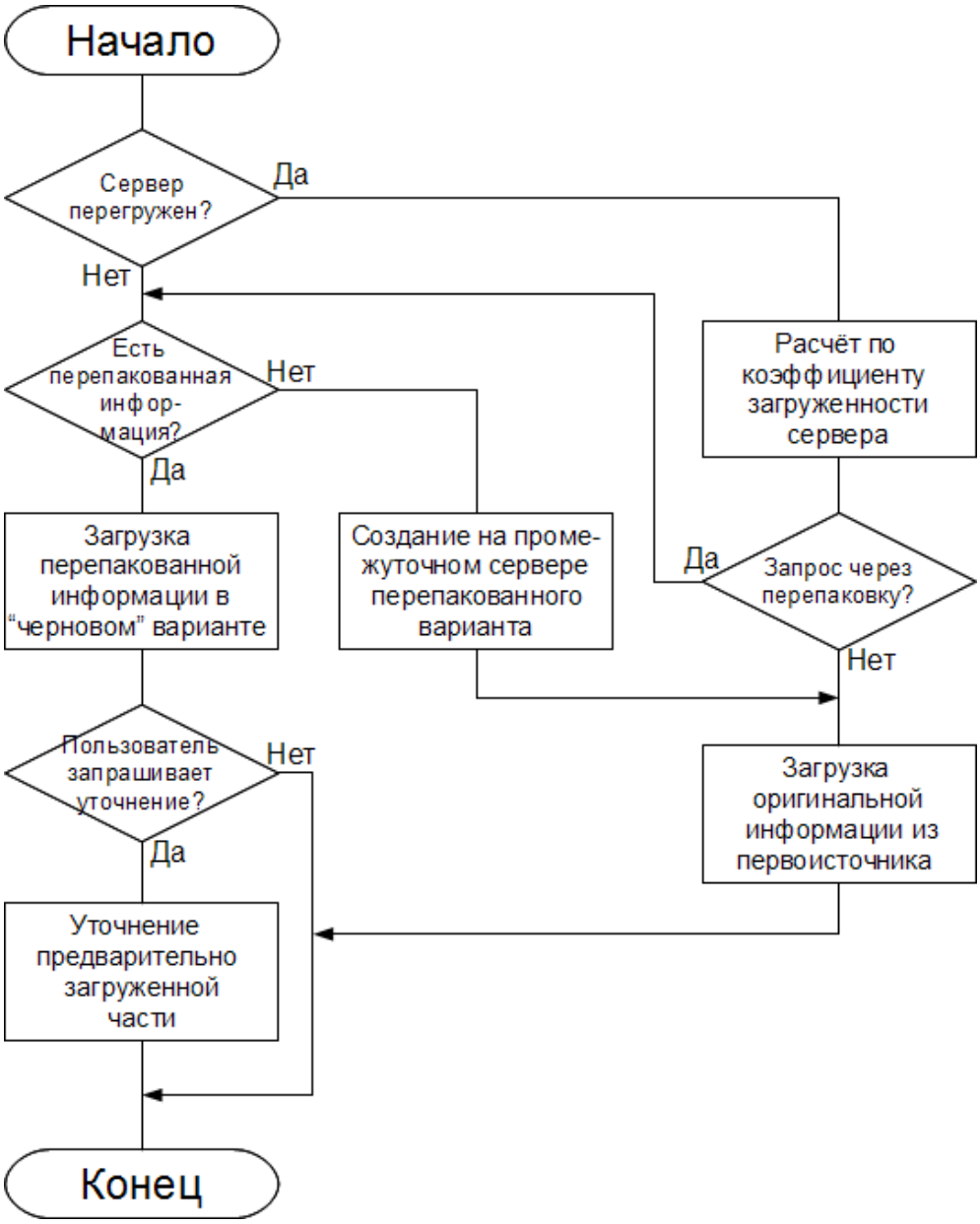


Рис. 2.16. Блок-схема алгоритма роботы системы уменьшения трафика низкоприоритетной графической информации со стороны сервера

Согласно алгоритму, взаимодействие начинается с оценки загруженности по вычислительным возможностям и пропускной способностью, которая согласовывается с сервером автоматически в фоновом режиме. По результатам оценки, клиент автоматически должен часть запросов проводить в классическом

режиме, с целью не допустить глобального отказа в обслуживании. Также, по результатам оценки загруженности, администраторам будет обеспеченно амортизационное время для наращивания вычислительных возможностей оборудования.

Рис. 2.17. Блок-схема алгоритма роботы системы уменьшения трафика низкоприоритетной графической информации со стороны пользователя

В случае низкой загруженности серверов, или принятия решения обращения к серверу перепакованной информации при высокой загруженности, клиент отправляет запрос к серверу дополнительного сжатия. На сервере производиться проверка наличия запрошенной информации в перепакованном виде и, при её наличии, черновая копия отправляется клиенту. Иначе, клиенту отсылается оригинальный вариант без дополнительного сжатия и, отдельно создаётся перепакованная копия оригинальной информации. При повторном запросе, от этого или другого клиента будет переданный черновой вариант информации, экономя трафик на передаче низкоприоритетной информации.

Получив черновой вариант, клиент имеет возможность оценить ценность полного качества или тонко регулировать требуемое качество получаемой информации. Благодаря прогрессивному кодированию, имеется возможность избежать полной загрузки информации, достаточно получить её недостающую часть. Порядок работы клиентской программы можно увидеть на блок-схеме изображенной на рис. 2.17.

2.4 Разработка математической модели роботы серверов дополнительного сжатия графического контента телекоммуникационных корпоративных систем и сетей

Изучение поведения системы дополнительного сжатия графической информации возможно или на математической модели или её реализации, что означает разработку и внедрение без уверенности в перспективности метода. Это накладывает дополнительные условия моделирования поведения системы в процессе её работы с учётом динамического изменения контента сети и количества клиентов этой системы, структурная схема которой приведена на рис. 1.4.

Математическая модель наряду с введёнными в пункте 1.3.1 обозначениями: T_3, T_n, k, S, S_k, включает в себя и $L(t)$ – количество активных пользователей системы, а L – их максимальное значение; отношение более медленного сегмента подключения к глобальной сети в r раз, которое

соответственно увеличит время загрузки T_3 r; и уже полученное соотношение определения коэффициента увеличения скорости передачи (1.5).

Коэффициент k не постоянен. Он выражает вероятность обращения клиента к информации, которая уже находится на сервере и необходимости обращения к оригинальному контенту нет. Учитывая тот факт, что информация постоянно добавляется со средней скоростью $v_д$ и устаревает со скоростью $v_у$, контент графической информации постоянно меняется, $dS = S(t+dt) - S(t)$. Этот показатель есть характеристикой сети.

Показатель скорости добавления информации на сервер дополнительного сжатия графической информации зависит от частоты попадания мимо загруженного контента. Такое непопадание происходит с вероятностью 1-k от общего количества запросов пользователей. Соответственно:

$$dS_k \sim (1-k)L(t)dt\,.$$

С целью более полного описания добавим обозначения:

S_n – количество изображений не в кэш-сервере, $S_n=S(1-k)$, $S_n=S-S_k$;

v_n – скорость добавления новых изображений в сети, $v_n=dS/dt$;

v_o – скорость увеличения количества устарелой информации, тут принята как постоянная часть от общего количества изображений $v_o=\gamma S$, $0<\gamma<1$ – коэффициент пропорциональности;

v_k – скорость добавления новых изображений в базу данных, зависит от количества пользователей системы. Каждый запрос к новой информации, которой ещё нет на сервере, вероятность такого запроса $1-k=S_n/S$, сопровождается добавлением нового контента в базу. Соответственно, количество добавленного контента за единицу времени пропорционально количеству активных пользователей системы $L(t)$, или $v_k=dS_k/dt$.

Доли кэшированного контента можно описать следующим отношением $k(t) = S_k / S$ и $k(t+dt) = \dfrac{S_k + dS_k}{S + dS}$. Получим соотношение прироста доли графической информации в базе данных:

$$k(t+dt)-k(t)=\frac{Sk(t)+(v_k-k(t)v_o)dt}{S+(v_n-v_o)dt}-k(t).$$

Сократив и опустив бесконечно малые, получим

$$\frac{dk}{dt}=\frac{v_k-kv_n}{S}.$$

Для моделирования изменения количества активных пользователей системы была использована модель распространения слухов (уравнение Ферхюльста), согласно которой вероятность сообщить новость незнающему пропорциональна количеству неосведомлённых людей и также пропорциональна количеству осведомлённых, которые эту новость распространяют:

$$\frac{dL(t)}{dt}=\alpha L(t)(L-L(t)).$$

Используя ранее введённые обозначения, запишем систему уравнений динамики развития системы дополнительного сжатия, что и будет её математической моделью:

$$\begin{cases} \dfrac{dk}{dt}=\dfrac{v_k-kv_n}{S} \\ \dfrac{dS}{dt}=v_n-v_o \\ v_k=\beta(1-k)L(t) \\ \dfrac{dL(t)}{dt}=\alpha L(t)(L-L(t)) \end{cases},$$

где α,β – коэффициенты пропорциональности модели. Модель применима в случае превышения скорости добавления графического контента в базу данных системы скорости появления нового контента в сети.

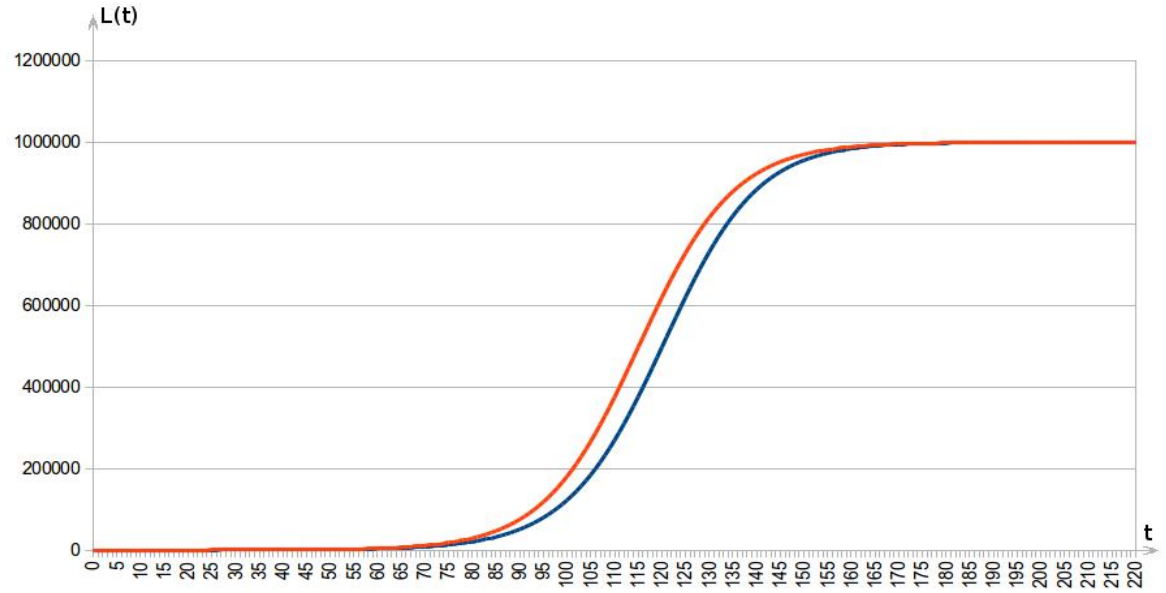


Рис. 2.18. Примеры решения модели распространения слухов

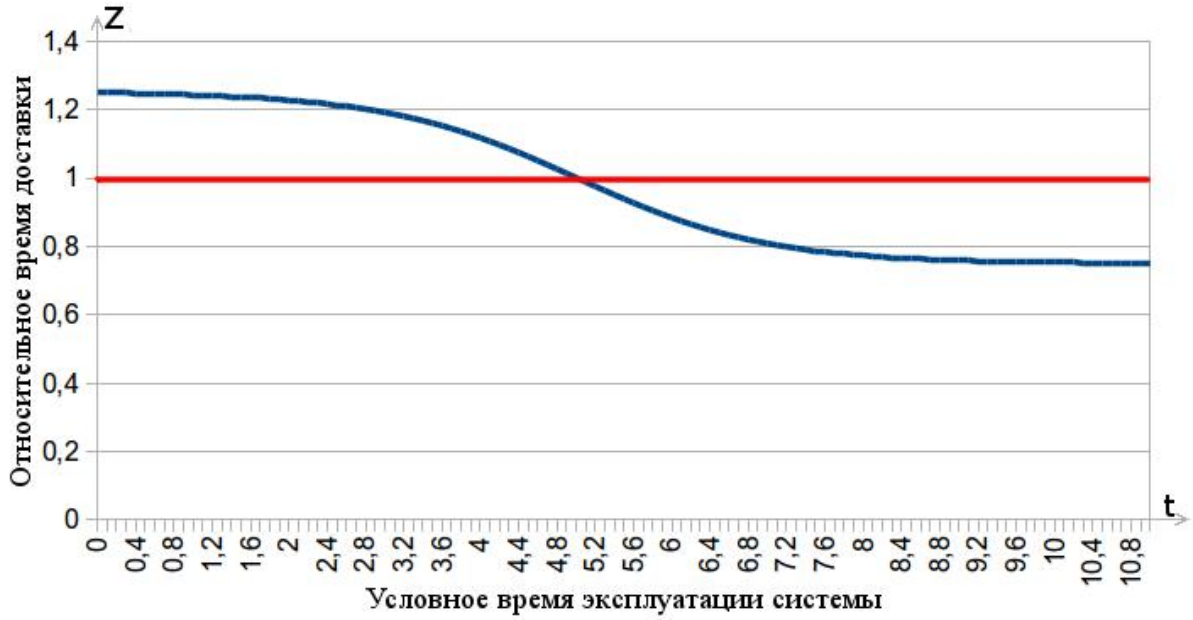


Рис. 2.19. Пример решения модели изменения среднего времени получения графической информации в случае равных пропускных возможностей основной сети и линии подключения

Сведение уравнений к общему даёт дифференциальное уравнение развития системы:

$$\frac{dk}{dt} = \left(\frac{(1-k)\beta L}{1+e^{-\alpha L t + C}} - k v_n \right) / S . \qquad (2.23)$$

Зная изменение во времени k, можно определить изменение среднего времени получения графической информации из сети по формуле (1.5). Уравнение (2.23) решалось численно, по причине гладкости функции, методом Эйлера. Для расчётов принято, что все постоянные коэффициенты раны единице, а коэффициент изменения размера графического контента v_k равен 0,5. При этом коэффициент среднего времени доставки сообщения в зависимости от времени эксплуатации системы был рассчитан по (1.5).

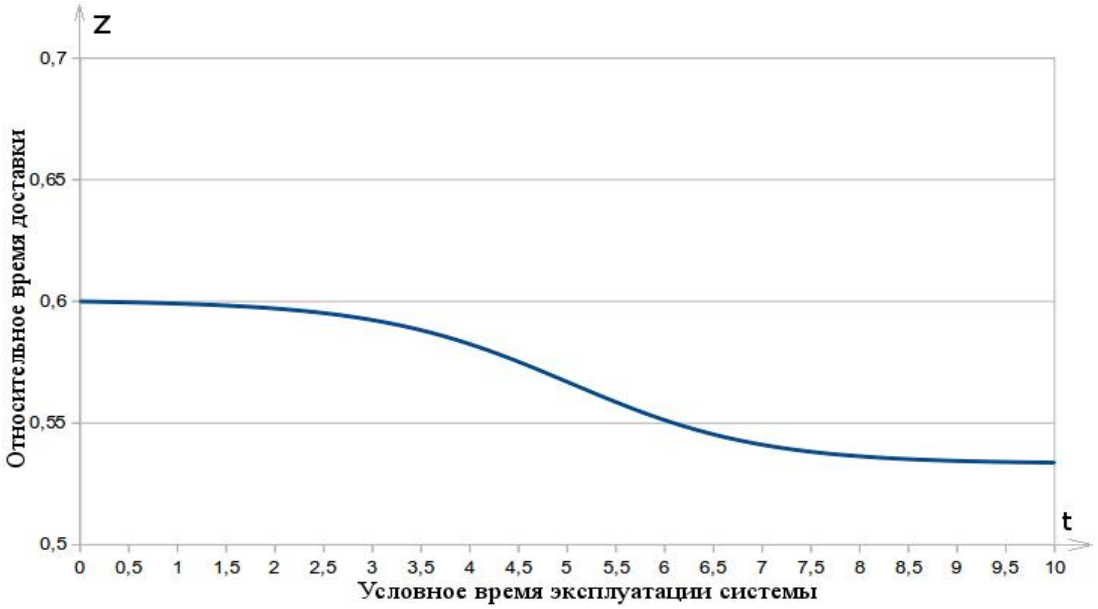


Рис. 2.20. Пример решения модели изменения среднего времени получения графической информации в случае низкой скорости подключения к основной сети в отношении 1:10

В случае ограниченной скорости получения информации, к примеру 1:10, при тех же условиях, график относительного времени доставки сообщения будет иметь вид, показанный на рис. 2.20.

Произведём оценку полученных результатов по модели. Модель иллюстрирует неэффективность использования системы с малым количеством

пользователей при их скоростном подключении к сети. Это приводит к преждевременному снижению популярности сервиса и остановке его развития. На этапе становления системы, то того как будет наблюдаться прирост скорости доставки сообщений, необходимо передавать клиенту не изменённый контент. При этом клиенты будут получать информацию без ожидания её дополнительного сжатия. Однако при значительном снижении скорости передачи информации на этапе провайдер-клиент, система будет эффективной сразу, и выигрыш по времени будет только увеличиваться по мере использования системы.

Существует порог скорости подключения к глобальной сети, при котором система даёт выигрыш в скорости доставки сообщений даже без использования промежуточного хранения информации. В этом случае экономия времени на передачу меньшего количества информации окупает затраты на получение контента из основного источника и дополнительному сжатию информации.

Выводы

Исследование статистической модели участка сети с учётом использования дополнительного сжатия графической информации позволило определить методы повышения эффективности передачи информации и также определить основные требования к методу, при помощи которого будет производиться дополнительное сжатие. По результатам сформулированы выводы:

1. Участок многосегментной сети можно заменить статистически равноценной односегментной связью. Дисперсия времени ожидания сообщения зависит от вероятности удачной передачи единичного пакета p, где неудача при передаче может быть вызвана как физическими помехами, так и простоями в очередях или другими факторами [42, 48].

2. Среднее время ожидания сообщения пропорционально объёму передающейся информации [29, 38, 39].

3. Существенно упростить регулировку объёма трафика графической информации можно используя прогрессивное сжатие с потерей информации [27].

4. Улучшить визуальное восприятие фотографического изображения при значительных коэффициентах сжатия SPIHT методом, можно используя передачу вейвлет коэффициентов при сохранении не абсолютных, а относительных погрешностей [37].

5. Прогрессивное сжатие даёт возможность конечному пользователю принимать решение о важности повышения качества получаемой графической информации для соблюдения баланса качество-оперативность [24, 27].

6. Получена оценка дополнительного сжатия SPIHT кода блочным арифметическим сжатием с оптимизацией размера блока. Размер блока оптимизирован по рассчитанной фрактальной размерности битовой последовательности сжимаемого кода [40].

РАЗДЕЛ 3
РАЗРАБОТКА МЕТОДА ПРЕДСКАЗАНИЯ ВОЗМОЖНОСТИ ОТКАЗОВ В ОБСЛУЖИВАНИИ СЕРВЕРОВ ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМАХ И СЕТЯХ ПРИ ПОМОЩИ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ

3.1 Разработка критериев оценки эффективности предотвращения отказа обслуживания серверов телекоммуникационной системы и сетей

Главным критерием эффективности работы системы предотвращения отказа в обслуживании есть изменение величины вероятности отказа $p_{отк}$.

Сервер дополнительного сжатия графической информации в телекоммуникационной системе или сети можно представить в виде системы массового обслуживания с физической пропускной способностью μ запросов в единицу времени [94]. Также, по причине ограничения времени обработки запроса, можно принять, что система имеет ограниченную очередь запросов длиной m. Тогда вероятность того, что на запрос к серверу на получение графической информации получит отказ, равна [18]:

$$p_{отк} = \lambda^{m+1} \frac{\mu - \lambda}{\mu^{m+2} - \lambda^{m+2}}, \tag{3.1}$$

где λ количество запросов от клиентов в единицу времени.

По физическому смыслу величина μ зависит от свойств сети, к которой подключён сервер, и продуктивности оборудования и программного обеспечения из которого собран сервер. Повлиять на эту величину можно лишь физической заменой оборудования или разработкой более совершенного программного обеспечения. Оба варианта будут сравнительно продолжительными во времени и в нашем случае принимаются за постоянную величину.

Напротив, величина λ зависит от количества активных клиентов сервера. Эта величина переменна от времени, зависит от времени суток, дня недели, сезона и других подобных субъективно-объективных факторов.

Связи с вышеупомянутыми факторами (3.1), вероятность отказа в обслуживании тоже будет переменной во времени. Показателем системы предотвращения отказов в обслуживании будет вероятность P, что эта вероятность превысит заданный порог p_n:

$$P\left(p_{отк} > p_n\right). \tag{3.2}$$

Оценка вероятности превышения количества отказов в обслуживании есть основным критерием качества работы системы предотвращения в обслуживании сервером дополнительного сжатия графической информации в телекоммуникационной системе или сети.

Однако в реальности происходит изменение интенсивности потока входных заявок, которые влияют на вероятность отказа в обслуживании. Очень низкое значение вероятности отказа означает значительные простои сервера (системы обслуживания), в связи которым использование оборудования становиться нерентабельным. С другой стороны, завышенная вероятность отказа в обслуживании приводит к падению качества обслуживания сети.

Выходом из возникшего противоречия есть указание диапазона допустимых значений вероятности отказа в обслуживании:

$$P\left(\left(p_{отк} - p_n\right) > \Delta p\right). \tag{3.3}$$

Вероятности выхода из указанного диапазона вероятности отказа в обслуживании (3.3) при наличии алгоритма предотвращения отказа и её отсутствии будут характеризовать целесообразность такой системы.

3.2 Разработка алгоритма предотвращения отказов в обслуживании дополнительного сжатия графической информации

В первом разделе (п. 1.4) была показана ограниченность существующих методов построения прогноза на долгий период изменения загруженности серверов. Это приводит к необходимости разработки нового алгоритма для экстраполяции квазипериодических временных рядов.

3.2.1 Построение частотно-корреляционного спектра сигнала

Задача состоит в аппроксимации с последующей экстраполяцией сигнала заданного значениями на, возможно неравномерной, сетке по времени непериодических колебательных процессов, которые содержат в себе гармонические составляющие с не кратными частотами. Для описания алгоритма регрессии в тригонометрический ряд за приемлемое машинное время, рассмотрим приближение экспериментальных данных функцией $g(t) = a_0 + a_1 \sin(\omega t) + a_2 \cos(\omega t)$, где a_0, a_1, a_2, ω – искомые коэффициенты.

Дано: $f(t_i)$ – значения функции, результаты измерений в моменты времени t_i, где i=1..N, N – натуральное число; необходимо определить коэффициенты a_0, a_1, a_2, ω такие, что бы

$$S(\omega) = \frac{1}{N}\sum_{i=1}^{N}\left(a_0 + a_1 \sin(\omega t_i) + a_2 \cos(\omega t_i) - f(t_i)\right)^2 = \min. \qquad (3.4)$$

Для нахождения a_0, a_1, a_2 можно использовать метод наименьших квадратов.

Найдём частные производные $S(\omega)$ и приравняем их к нулю, получив систему уравнений, решения которой даёт минимум среднеквадратического отклонения:

$$\begin{aligned} S'_{a_0}(\omega) &= 2a_0 + 2a_1\overline{\sin(\omega t)} + 2a_2\overline{\cos(\omega t)} - 2\overline{f(t)} = 0, \\ S'_{a_1}(\omega) &= 2a_0\overline{\sin(\omega t)} + 2a_1\overline{\sin^2(\omega t)} + 2a_2\overline{\sin(\omega t)\cos(\omega t)} - 2\overline{f(t)\sin(\omega t)} = 0, \\ S'_{a_2}(\omega) &= 2a_0\overline{\cos(\omega t)} + 2a_1\overline{\sin(\omega t)\cos(\omega t)} + 2a_2\overline{\cos^2(\omega t)} - 2\overline{f(t)\cos(\omega t)} = 0. \end{aligned} \qquad (3.5)$$

Тут для упрощения записей введено обозначение среднего арифметического: $\frac{1}{N}\sum_i v(t_i) = \overline{v(t)}$, где N – количество слагаемых в сумме, а i – индекс суммирования. Именно расчёт суммы многих значений занимает основное машинное время. Однако время расчётов можно сократить, если временной интервал измерения h постоянное. Тогда суммы синусов косинусов и их произведения можно рассчитать по конкретным формулам.

Приведём вывод формулы нахождения значений средних арифметических для поиска коэффициентов системы (3.5). Вывод остальных значений аналогичен и тут приводятся сразу готовые выражения. По формуле Муавра $e^{jx}=\cos(x)+j\cdot\sin(x)$, де $j=\sqrt{(-1)}$, сумма

$$\sum_{i=0}^{N}\sin(\omega t_0+\omega h\cdot i)=\sum_{i=0}^{N}\mathrm{Im}(e^{j(\omega t_0+\omega h\cdot i)})=\mathrm{Im}\left[e^{jt_0\omega}\sum_{i=0}^{N}\left(e^{j\omega h}\right)^i\right],$$

является геометрической прогрессией с коэффициентом $e^{j\omega h}$. Использовав соотношение нахождения суммы геометрической прогрессии, получим:

$$\sum_{i=0}^{N}\sin(\omega t_0+\omega h\cdot i)=\mathrm{Im}\left[e^{jt_0\omega}\frac{e^{j\omega h(N+1)}-1}{e^{j\omega h}-1}\right].$$

Для получения конечного вида формул, снова воспользуемся формулой Муавра, вернувшись снова к использованию синусов и косинусов, оставив только мнимую, а потом только действительную части:

$$\frac{1}{N+1}\sum_{i=0}^{N}\sin(\omega t_0+\omega h\cdot i)=\frac{\cos(\omega(t_0-h/2))-\cos(\omega(t_0+h(N+1/2)))}{2(N+1)\sin(\omega h/2)},$$

$$\frac{1}{N+1}\sum_{i=0}^{N}\cos(\omega t_0+\omega h\cdot i)=\frac{-\sin(\omega(t_0-h/2))+\sin(\omega(t_0+h(N+1/2)))}{2(N+1)\sin(\omega h/2)}.$$

Аналогично ищутся формулы для нахождения сумм квадратов тригонометрических функций и сумм произведения синуса на косинус, для этого предварительно воспользуемся тригонометрическими формулами понижения степени:

$$\frac{1}{N+1}\sum_{i=0}^{N}\sin^2(\omega t_0+\omega h\cdot i)=\frac{1}{N+1}\sum_{i=0}^{N}\frac{1}{2}\left(1-\cos(2\omega t_0+2\omega h\cdot i)\right)=$$

$$=\frac{1}{2}+\frac{\sin(2\omega(t_0-h/2))-\sin(2\omega(t_0+h(N+1/2)))}{2(N+1)\sin(\omega h)};$$

$$\frac{1}{N+1}\sum_{i=0}^{N}\cos^2(\omega t_0+\omega h\cdot i)=\frac{1}{N+1}\sum_{i=0}^{N}\frac{1}{2}\left(1+\cos(2\omega t_0+2\omega h\cdot i)\right)=$$

$$=\frac{1}{2}+\frac{-\sin(2\omega(t_0-h/2))+\sin(2\omega(t_0+h(N+1/2)))}{2(N+1)\sin(\omega h)};$$

$$\frac{1}{N+1}\sum_{i=0}^{N}\cos(\omega t_0+\omega h\cdot i)\sin(\omega t_0+\omega h\cdot i)=$$

$$=\frac{1}{N+1}\sum_{i=0}^{N}\frac{1}{2}\sin(2\omega t_0+2\omega h\cdot i)=$$

$$=\frac{\cos(\omega(2t_0-h))-\cos(\omega(2t_0+h(2N+1)))}{4(N+1)\sin(\omega h)}.$$

Благодаря этим соотношениям расчёт коэффициентов для системы уравнений (3.5) происходит значительно быстрее. Фактически, полностью проходит поиск суммы только для трёх составляющих, которые используют измеренные значения.

Решение системы (3.5) даёт возможность определить коэффициенты a_0, a_1, a_2, что определяет функцию, которая имеет минимальное среднеквадратичное отклонение при искомой частоте ω. Однако, относительно ω система нелинейная, поэтому минимум $S(\omega)$ ищется при помощи численного метода.

При поиске частоты ω невозможно учесть весь набор частот $(0;+\infty)$, поэтому поиск сверху ограничивается частотой, которая задаётся теоремой Котельникова-Найквиста: при дискретизации сигнала с шагом во времени h, сигнал можно восстановить однозначно, если он не содержит составляющие частоты выше чем $\omega=\pi/(2h)$ (на один период должно быть не меньше четырёх точек измерения). Теоретически нижняя граница частот ограничена нулём, однако для практических расчётов нижняя граница частот берётся или в зависимости от природы сигнала, или за наибольший период берётся от трёх до десяти временных промежутков исследования сигнала.

3.2.2 Поиск частоты гармонического колебания-приближения

Сумма синусоид с кратными периодами колебаний $\omega_i=2\pi i/(b-a)$ ($[a;b]$ – интервал исследования сигнала), то есть ряд Фурье, описывает любой непрерывный сигнал. По этой причине корреляционная функция $S(\omega)$ имеет вид колебаний с локальными минимумами при частотах кратных частоте, которая по периоду совпадает со временем исследования сигнала. Для наглядности, покажем на примере вид $S(\omega)$ на графиках 3.1 и 3.2. На рис 3.1 показан вид $S(\omega)$ при исследовании синусоиды $a_1\cdot\sin(\omega_1 t)$ без гармоник. Тут $S(\omega)$ является колебательной функцией, при этом частота колебаний зависит только от промежутка исследования, а количество точек разбиения, только на амплитуду локальных минимумов. Влияние количества точек разбиения на вид графика $S(\omega)$ показано на рисунке 3.1.

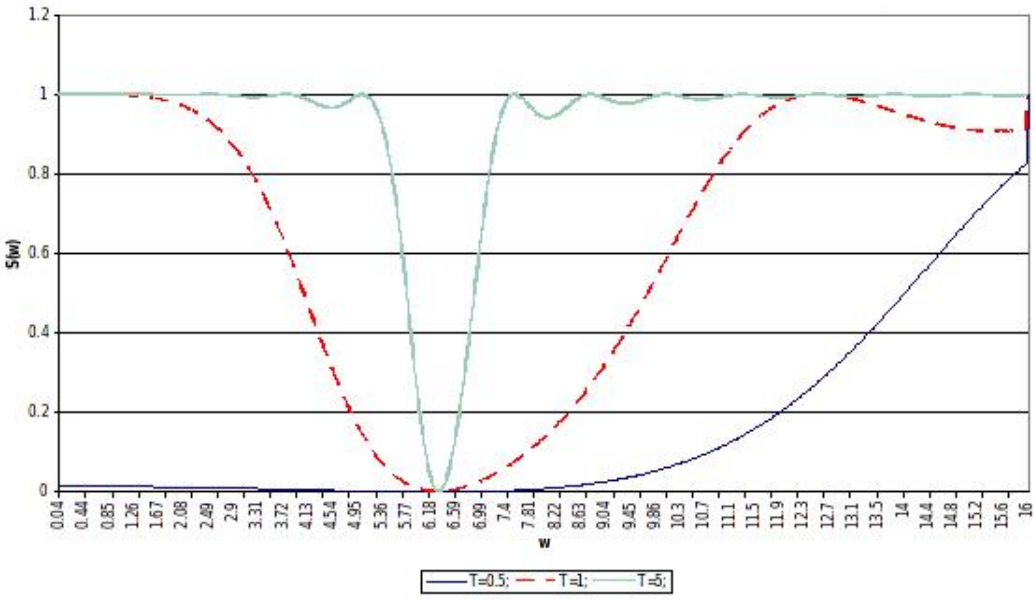


Рис. 3.1. Явление ложных минимумов в частотах с кратной гармоникой ко времени наблюдения

Локальные минимумы появляются при совпадении ω с частотой, кратной частоте колебания с периодом равным промежутку исследования. Например, примем что экспериментальные исследования проводились на промежутке времени от t_0 до t_n (n+1 – количество точек измерения). Тогда шаг по времени h влияет только на значение частоты Найквиста. Время наблюдения сигнала

составляет $T=t_n-t_0$, откуда находится, что паразитные локальные минимумы будут локализованы около $\omega_k=2\pi k/T$, где k – натуральное число.

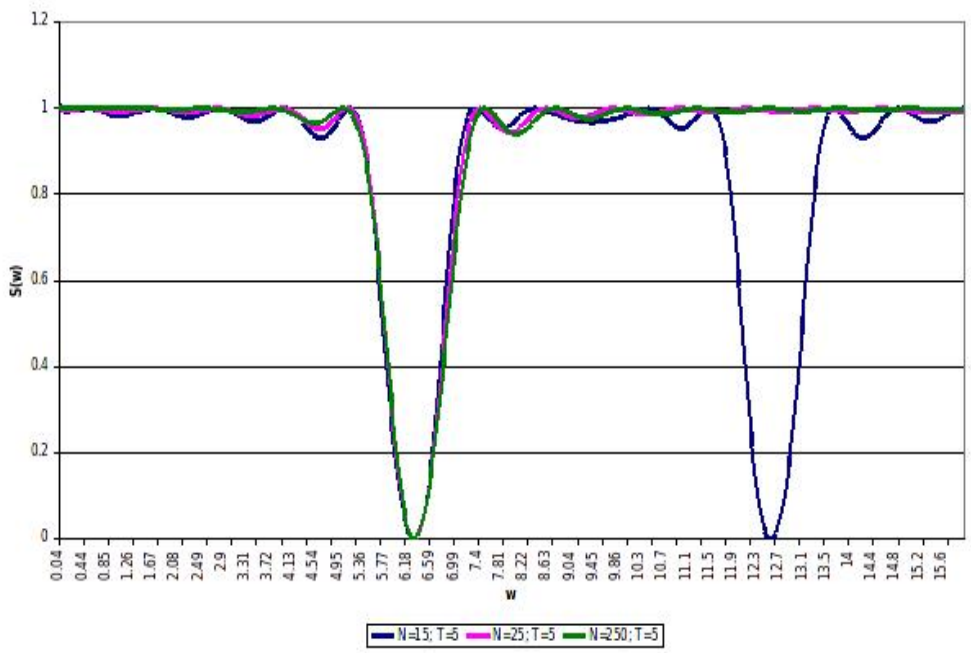


Рис. 3.2. Зависимость интенсивности ложных минимумов от количества точек разбиения

На рисунке 3.2 при количестве точек измерения N=15, показано паразитный минимум, появление которого является следствием превышения частоты Найквиста, эта частота обозначена на рисунке как $\omega_н$.

Перед использованием метода уточнения значения частоты, необходимо найти минимальное значение в таблично представленной $S(\omega)$, используя разбиения $[\omega_0;\omega_н]$ частот на равномерные промежутки по количеству

$$N_\omega = 3\frac{\omega_н - \omega_0}{2\pi}(t_n - t_0). \qquad (3.6)$$

Определив необходимое количество точек разбиения для уточнения частот аппроксимирующей синусоиды N_ω, строим таблицу частот со значениями $S(\omega)$. Для уточнения частоты, из таблицы нужно взять частоту, при которой получено минимальное среднеквадратичное отклонение вместе с двумя соседними значениями. Далее следует уточнение частоты, тут автор использовал метод простой дихотомии.

3.2.3 Поиск частотного спектра

Большая часть природных процессов не состоит из одного колебания, системы имеют разные резонансные частоты, гармоники, существенные влияния могут оказывать и другие колебательные системы. Поэтому нужно использовать для аппроксимации сумму гармонических колебаний разных амплитуд, частот и фаз.

На основе алгоритма, предложенного в предыдущем пункте, ищем приближение:

$$f(t) \approx \sum_{i=1}^{K} g_i(t), \tag{3.7}$$

где

$$g_i(t) = a_{0,i} + a_{1,i}\sin(\omega_i t) + a_{2,i}\cos(\omega_i t).$$

Можно использовать алгоритм:

1. Для функции *f(t)* ищем приближение $g_1(t)$ по методу поиска наилучшего приближения синусоидой (пункт 3.3.2.) и строим функцию отклонения от приближения $f_1(t)=f(t)-g_1(t)$.
2. Повторяем п.1 для функции отклонения от приближения $f_1(t)$ и получаем $f_2(t)$, $g_2(t)$.
3. Действия п.1 повторяем до достижения желаемой точности приближения.

Выполнение алгоритма даёт гарантированное среднеквадратичное приближение (3.7). Однако этот алгоритм не даёт гарантии сходимости такого процесса, происходит это при отсутствии в последовательности значений периодичных составляющих и временной ряд представляет собой белый шум.

3.2.4 Использование приближения тригонометрическим многочленом для экстраполяции

Главной целью разработки метода построения приближения тригонометрическим многочленом квазипериодического сигнала, есть прогнозирование. Однако использовать полученный тригонометрический

многочлен можно только после оценки погрешностей и надёжности полученных прогнозов.

При исследовании надёжности приближения, нужно оценить точность нахождения периода и фазы, составляющих сигнала. При исследовании почти периодических сигналов, период не стабильный и может медленно меняться с течением времени. Поэтому тут лучше оценить доверительный интервал прогноза на заданный промежуток времени.

Будем считать исследуемый сигнал случайным, он содержит случайный набор гармонических колебаний. Такой процесс является эргодическим и для него справедливо соотношение:

$$\tau_p = \frac{1}{R_y(0)} \int_0^{\infty} \left| R_y(s) \right| ds \,. \tag{3.8}$$

Тут τ_p – порядок времени, на который имеет смысл прогнозирование процесса, а

$$R_y(s) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} y(x)y(x-s)dx - \left(\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} y(x)dx \right)^2$$

есть корреляционной функцией исследуемого сигнала [1].

При поиске выражения регрессии в виде тригонометрического многочлена, автоматически будем иметь среднее квадратичное отклонение $S_k(\omega_k)$ и, также, таблицу функции отклонений $f_{k+1}(x_i)$. Если принять отклонения $f_{k+1}(x_i)$ случайной величиной с нормальным законом распределения с нулевым математическим ожиданием и средним квадратичным отклонением $S_k(\omega_k)$, можно оценить время прогнозирования с заданной точностью.

Пускай допустимое отклонение от реального сигнала от прогноза есть b, и вероятность того что эта норма не будет превышена будет $P_н$, тогда вероятность получения прогноза в границах допустимых отклонений, будет

$$P_H \le \left(\frac{1}{\sqrt{2\pi S_k(\omega_k)}} \int_{-b}^{b} \exp\left(-l^2 / \left(2\mathrm{S}_k(\omega_k)\right)\right) dl \right)^n , \qquad (3.9)$$

где n – количество прогнозируемых шагов; l – переменная интегрирования. При этом, промежуток прогнозирования $t_H = hn,$ с вероятностью $P_н$ является надёжным для разрешённого отклонения $[-b;b]$. Для более надёжной оценки нужно учесть, что среднеквадратичное отклонение $S_k(\omega_k)$ полученное из N испытаний тоже не точное, но лежит с определённой надёжностью возле значения $S_k(\omega_k)$. Практически, при превышении за 100 пробных измерений, надёжность среднего квадратичного отклонения высокая и прогноз не требует дополнительных коррекций.

В процессе построения регрессионного уравнения усредняются значения частот и амплитуд гармоник исследуемого сигнала. Поэтому прогноз может иметь значительную ошибку, которую необходимо оценить.

Процесс построения базиса для регрессионного уравнения даёт значения циклических частот колебаний, на основе которых находиться характеристическое уравнение. Соответствующее дифференциальное уравнение содержит в множестве решений и построенное приближение:

$$y^{(n)} + b_{n-1} y^{(n-1)} + \ldots + b_2 y'' + b_1 y' + b_0 y = 0.$$

Обобщением ряда Тейлора [103] есть равенство

$$f(t) = f(t_0)v_0(t-t_0) + f'(t_0)v_1(t-t_0) + \ldots + f^n(t_0)v_n(t-t_0) + \int_{t_0}^{t} F(\tau)v_n(t-\tau)d\tau,$$

где $f(t)$ – функция, к которой строят приближение, $v_i(t-t_0)$ – частные решения линейного дифференциального уравнения с постоянными коэффициентами D[v]=0 порядка n с начальными условиями $v_i^{(k)}(t-t_0) = \delta_{i,k}$. Тут $\delta_{i,k} = 1$ при $i=k$, иначе $\delta_{i,k} = 0$. F(t) – результат действия дифференциального оператора D[] на $f(t)$. Пускай функция $f(t)$ представлена выборкой по времени f_i однозначно,

что означает выполнение условий теоремы Найквиста. Существует приближение колебательными функциями:

$$f(t) \approx \sum_{k=1}^{N} a_k g_k(t),$$

где каждой функции $g_k(t)$ отвечает дифференциальный оператор $\mathbf{D}_k[]$ минимального порядка. Оператор $\mathbf{D}[]$ имеет характеристическое уравнение, которое получено произведением всех часных характеристических уравнений операторов $\mathbf{D}_k[]$. По построению оператора $\mathbf{D}[]$ существует эквивалентное приближение по обобщению ряда Тейлора, и $v_i(t-t_0)$ являются линейной комбинацией $v(t-t_0)=\sum_{k=1}^{N} a_k^* g_k(t)$. В результате:

$$f(t) - \sum_{k=1}^{N} a_k g_k(t) = \int_{t_0}^{t} F(\tau) v_N(t-\tau) d\tau,$$

где $v_N(t-t_0)$ ищется как решение задачи Коши D[v]=0, v(0)=0, v'(0)=0, ..., $v^{(n)}(0)=1$. Выбор t_0 производят исходя свойствам интеграла, который можно разложить на интервалы:

$$f(t) - \sum_{k=1}^{N} a_k g_k(t) = \int_{t_0}^{t_n} F(\tau) v_N(t-\tau) d\tau + \int_{t_n}^{t} F(\tau) v_N(t-\tau) d\tau.$$

Значение интеграла на интервале $[t_0;t_n]$ находится как разность между измеренными и прогнозируемыми значениями, обозначим эту разность Δ_n. По теореме о среднем, существует такое c, для которого

$$f(t) - \sum_{k=1}^{N} a_k g_k(t) = \Delta_n + F(c) \int_{t_n}^{t} v_N(t-\tau) d\tau.$$

Как результат, оценкой ошибки прогнозирования будет $R(t)$ есть:

$$\left|R(t)\right| \le \left|\Delta_n\right| + \max\left(\left|F(t)\right|\right)\cdot\left|\int_{t_n}^{t} v_N(t-\tau)d\tau\right|,$$

при справедливости предположения дальнейшей ограниченности $F(t)$ на время прогнозирования, что естественно для почти периодических временных рядов. По причине использования для приближения исключительно гармонических составляющих, модуль значений функции $v_N(t)$ ограничен, поэтому оценка ошибки прогнозирования возрастает пропорционально времени, на который строится прогноз.

Если принять, что функция $v_N(t) = A_0 + \sum_{k=1}^{N}\left(B_k \sin(\omega_k t) + C_k \cos(\omega_k t)\right)$, с учётом начальных условий задачи Коши, можно использовать равенства функции и её производных нулю, что даёт систему для определения неизвестных коэффициентов:

$$A_0 + \sum_{k=1}^{N} C_k = 0;\ \sum_{k=1}^{N} \omega_k B_k = 0;\ \sum_{k=1}^{N} \omega_k^2 C_k = 0;\ldots$$

$$\sum_{k=1}^{N}\left(B_k \sin^{(2N+1)}(\omega_k t) + C_k \cos^{(2N+1)}(\omega_k t)\right) = 1.$$

Это даёт возможность оценить максимальное значение модуля $\left|v_N(t)\right| \le \left|A_0\right| + \sum_{k=1}^{N}\left(\sqrt{B_k^2 + C_k^2}\right)$. Тогда оценкой ошибки прогнозирования будет:

$$\left|R(t)\right| \le \left|\Delta_n\right| + \max\left(\left|F(t)\right|\right)\cdot\left(\left|A_0\right| + \sum_{k=1}^{N}\sqrt{B_k^2 + C_k^2}\right)t.$$

Особенностью оценки $R(t)$ есть универсальность, необходимым условием является только полнота представления функции $f(t)$, по теореме Найквиста, и принадлежность базиса функций $g_k(t)$ к множеству линейно независимых решений дифференциального уравнения $\mathbf{D}[g_k(t)]=0$.

3.3 Исследование влияния разработанного метода предотвращения отказов в обслуживании на качество работы системы

Вследствие сказанного в главе можно сделать вывод, вероятность отказа в обслуживании зависит от входящего потока заявок λ, продуктивности системы обслуживания (сервера сети) μ и длины очереди m. По причине постоянности значений величин μ и m, управление вероятностью отказа в обслуживании должно осуществляться через λ. На прямую эта величина не подаётся регулированию, однако можно изменить на стороне клиента долю запросов, которые будут обрабатываться сервером дополнительного сжатия информации. Запросы, которые не попали в эту долю, пойдут минуя промежуточный сервер, по классической схеме к первоисточникам графической информации, тем самым уменьшив значение λ.

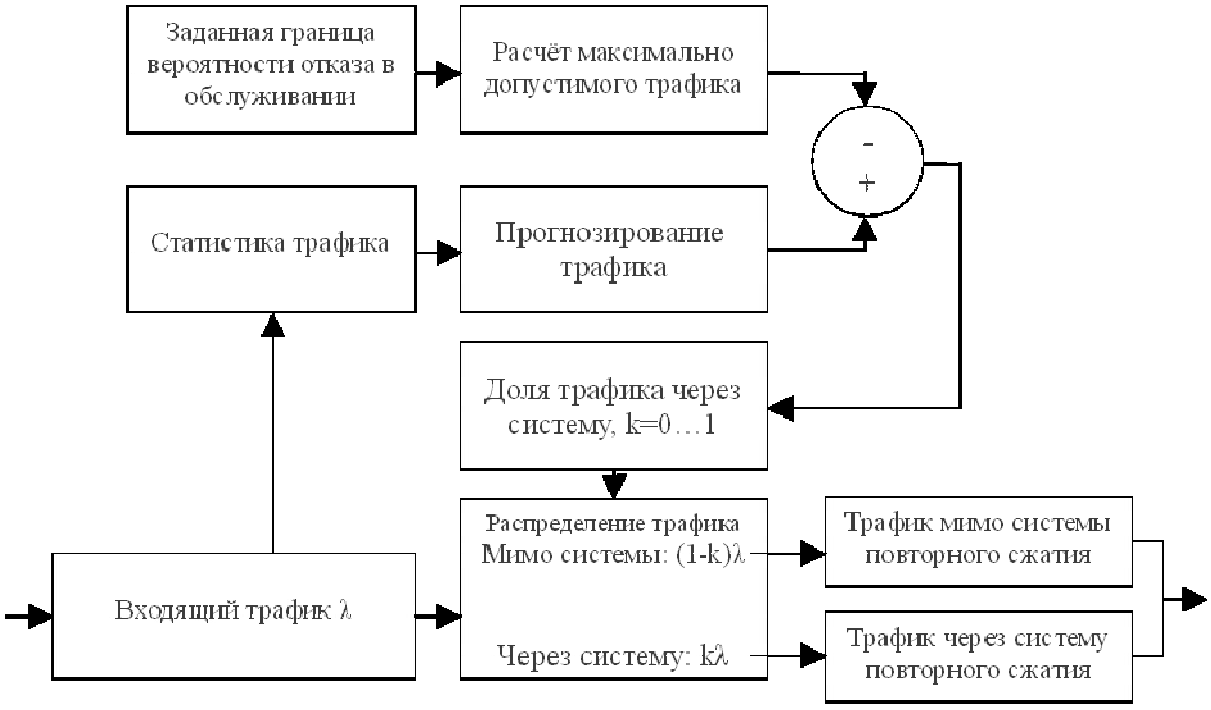


Рис. 3.3. Схема коррекции доли трафика через систему дополнительного сжатия графической информации для регулирования вероятности отказа в обслуживании

Таким образом, λ будет переменной во времени величиной, которую можно уменьшить за счёт частичного снижения качества обслуживания. Однако такое изменение будет довольно инерционным (система управления с запаздываниями), и сервер будет сталкиваться с довольно частыми повышенными нагрузками, с повышенной вероятностью отказа в обслуживании и сопровождаться необоснованными снижениями качества обслуживания. Система предсказания решает проблему запаздывания. Использование такой системы показано на рис. 3.3.

Схема демонстрирует, что установив долу запросов k, результат уменьшения входного трафика будет обнаружен не сразу, а по мере распространения этого коэффициента к пользователям системы. Пользователи могут получать k вместе с перепакованной графической информацией.

Пускай в системе вероятность отказа в обслуживании не должна превышать $p_{от}$. Тогда система должна поддерживать входной поток запросов на уровне не большем чем $\lambda_{от} \approx k \cdot \lambda$. В идеальном случае система гарантировано будет удерживать вероятность отказа в допустимых рамках. Однако система предсказания основана на феноменологическом подходе без объяснения причин предсказанных изменений. Такие предсказания носят вероятностный характер и вероятность такого прогноза можно оценить по формуле (3.9).

Пусть сделан прогноз изменения входящего потока $\lambda_n \pm \Delta\lambda$ с вероятностью $P_н$. Тогда при фактических значениях сервер будет работать в оптимальном режиме. Иначе с вероятностью $1\text{-}P_н$ сервер будет работать с пониженной нагрузкой или вероятность отказа в обслуживании будет вне допустимых пределах. Таким образом надёжность системы предотвращения превышения заданной вероятности отказа в обслуживании характеризуется надёжностью прогнозирования (3.9).

Выводы

Введение прогнозирования трафика телекоммуникационного сервера дополнительного сжатия графической информации позволило использовать упреждающее регулирование части запросов с целью не допустить превышения аппаратных возможностей системы обслуживания. В свою очередь, прохождение части запросов по классической схеме позволило повысить надёжность обслуживания за счёт меньшей сжимаемости графического трафика на участке сети. Применимость разработанной системы долгосрочного прогнозирования обусловлена выводами:

1. Существующие методы построения прогнозов квазиэргодических с нестабильным периодом и сильными аддитивными помехами сигналов, на основе феноменологического подхода, не удовлетворяют требования надёжности и возможности оценки вероятности ошибочности этих прогнозов. Это обосновывает разработку нового метода на основе разложения временного сигнала на гармонические составляющие с некратными частотами последовательным выделением трендов.

2. Использование минимизации среднеквадратичного отклонения для поиска частоты гармонического колебания, которое есть приближением к заданным данным, даёт лучшие результаты для построения прогнозов. Благодаря построению приближения в среднем, методика прекрасно работает и для исходных данных, которые имеют значительную часть ошибок измерения и шумы, также метод применим для неравномерных временных сеток входных данных.

3. Для методики прогнозирования почти периодических сигналов выведено вероятностную оценку ошибки прогнозирования, что есть более естественным для построения прогнозов дальнейшего развития процессов [25, 33, 34, 41].

РАЗДЕЛ 4
РЕКОМЕНДАЦИИ ПО ПРАКТИЧЕСКОМУ ИСПОЛЬЗОВАНИЮ РАЗРАБОТАННЫХ МЕТОДОВ ПОВЫШЕНИЯ ОПЕРАТИВНОСТИ ПЕРЕДАЧИ ДАННЫХ И КАЧЕСТВА ОБСЛУЖИВАНИЯ В ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМАХ И СЕТЯХ

4.1 Оценка достоверности теоретически полученных результатов в отношении оперативности передачи сообщения и его размерами

Экспериментальная проверка проводилась закачкой 12-ти файлов разного размера более 3000 раз каждый. При этом засекалось время полного получения закачиваемого файла. Полученные оценки среднего статистического времени получения информации отображены на рис. 4.1:

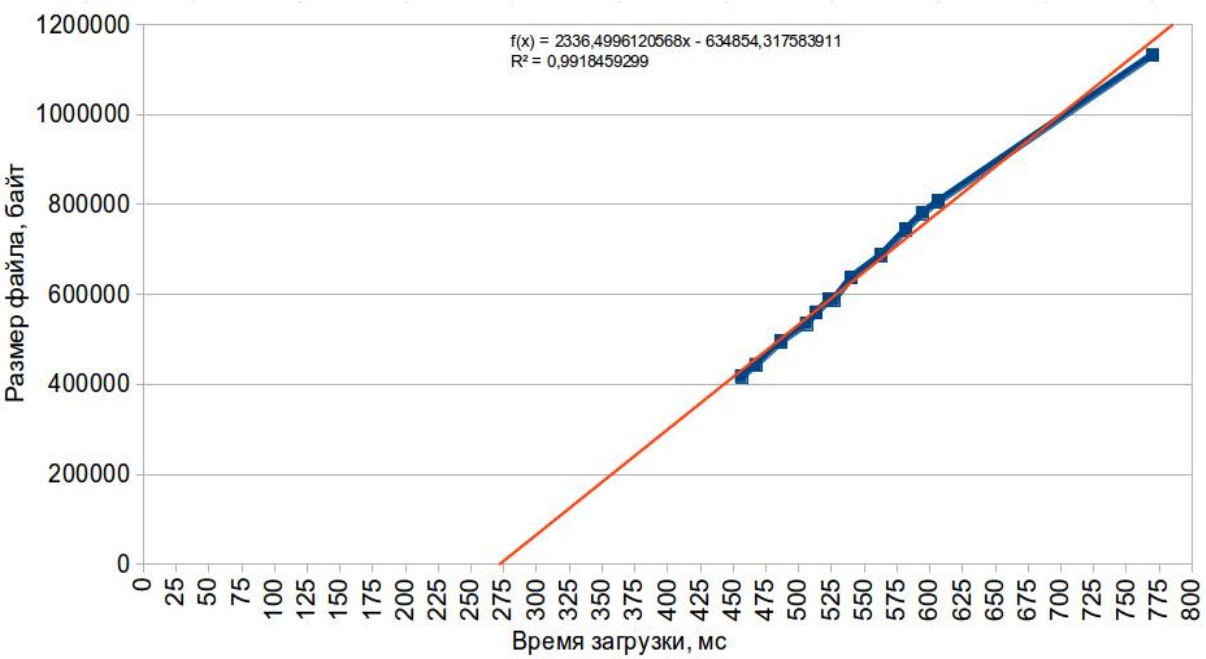


Рис. 4.1. Экспериментально полученные среднестатистическое время доставки сообщения в зависимости от размера этого сообщения

На рис. 4.2 показан график изменения дисперсии времени загрузки файлов. Обе зависимости показывают значительную близость к линейному закону, что согласуется с ранее полученными результатами (2.10) и (2.11).

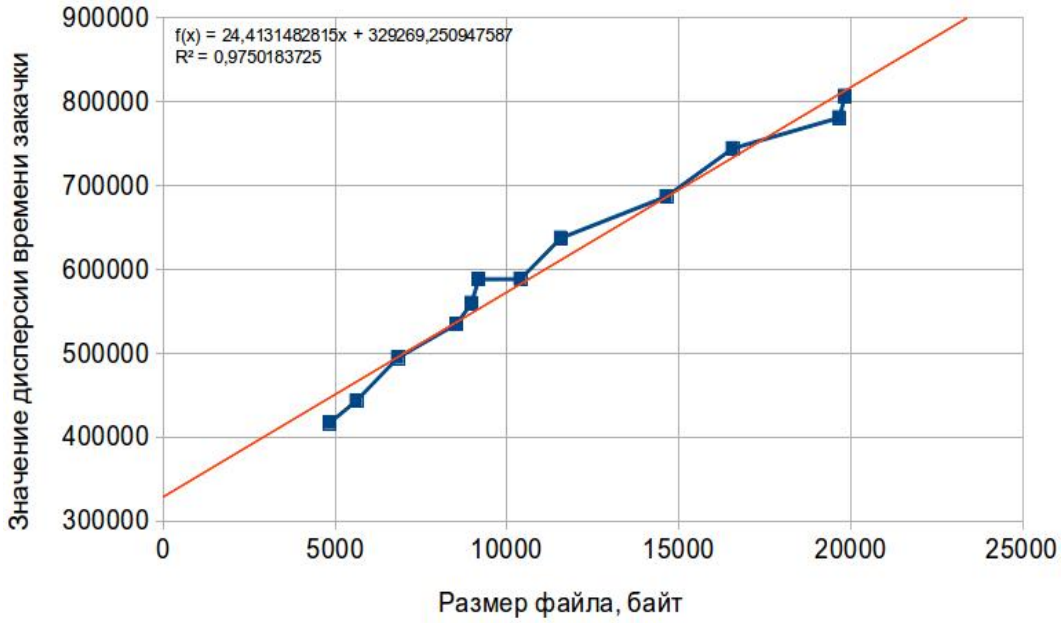


Рис. 4.2. Зависимость дисперсии времени доставки сообщения от размера этого сообщения

4.2 Разработка рекомендаций по практическому применению метода предотвращения отказов в обслуживании серверов дополнительного сжатия графической информации в телекоммуникационной системе или сети

Удостоверится в правильности теоретических построений предложенным методом корреляционного анализа, проведём несколько тестовых примеров. Для этого возьмём две функции, которые будут источником двух последовательностей значений сигналов:

$$y^{*}(t) = \sin(\pi t) + \cos((\pi + 0.1)t),$$
$$y^{**}(t) = (5 + 0.02\sin(0.01t))\cos(3t + 0.01\sin(0.1t)). \quad (4.1)$$

Функция $y^{*}(t)$ не периодическая, ибо отношение частот двух составляющих даёт иррациональное число; функция $y^{**}(t)$ периодична, но со временем постепенно изменяется амплитуда и фаза колебаний.

Для обеих функций строится таблица «отсчётов» с шагом по времени h=0.05 от t=0 до t=10. Фрагмент таблицы показан в табл. 4.1.

Таблица 4.1

Значения функций

$$y^{*}(t) = \sin(\pi t) + \cos((\pi + 0.1)t),\ y^{**}(t) = (5 + 0.02\sin(0.01t))\cos(3t + 0.01\sin(0.1t)).$$

t	y*(t)	y**(t)
0	1	5
0.05	1.143328	4.943828
0.10	1.256936	4.776554
0.15	1.338087	4.501936
0.20	1.384886	4.126146
…	…	…
9.90	0.471164	-0.68199
9.95	0.512576	0.066311
10.00	0.540302	0.813124

Графики этих функций показаны на рис. 4.3.

Аппроксимация проводиться с учётом только 10-ти частотных составляющих, границы поиска ω составляют [0.1;10]. Результатом расчётов переставлены в виде таблицы значений частот, их амплитуд и фаз. Як второстепенный результат работы алгоритма, мы имеем среднее квадратичное отклонение и таблицу отклонений функции-приближения от эталонных данных (рис. 4.3).

Синтетический пример с известными частотами показал на применимость метода к детерминированным процессам. Однако процесс обращения пользователя к серверу в сети не может быть детерминированным, но имеет и свои закономерности в связи суточными ритмами, выходными днями и т.д. С целью определить применимость анализа и экстраполяции трафика, который зависит от объективно-субъективных факторов, проводится эксперимент на реальных данных с сетевого сервера. Типичный результат эксперимента представлен на рис. 4.4.

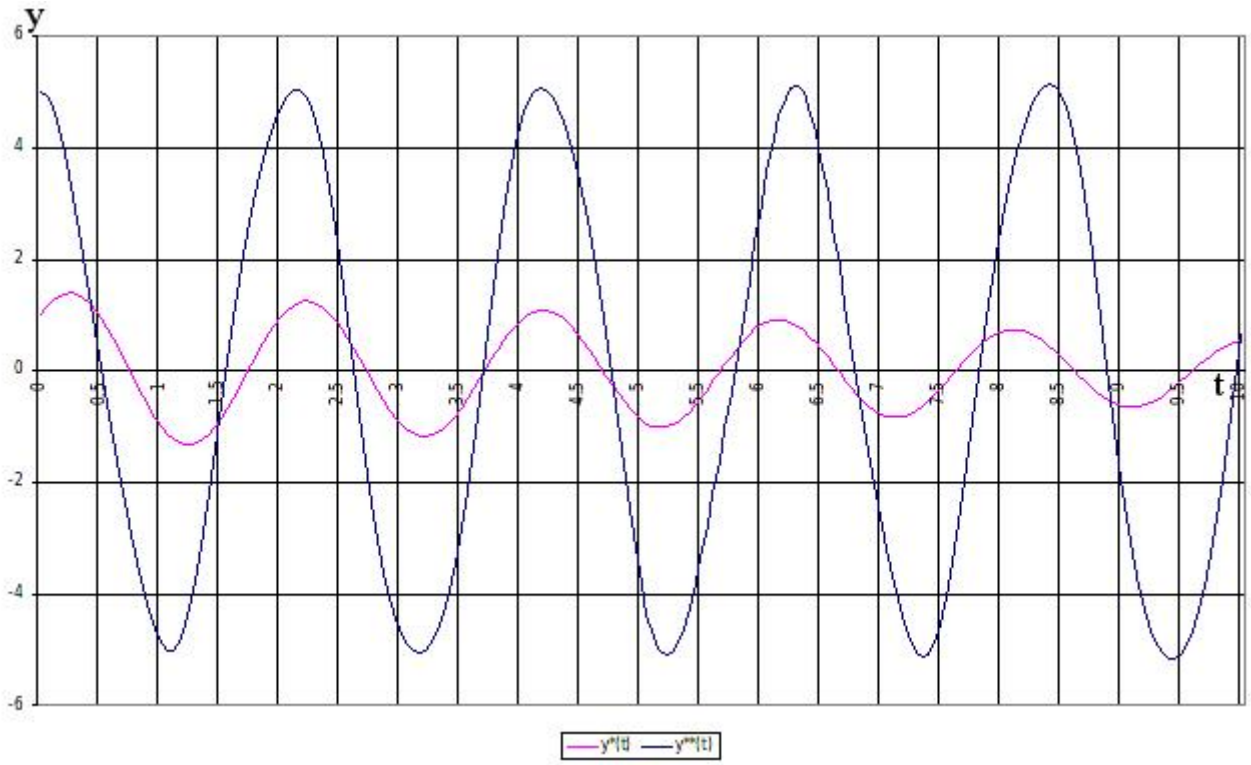


Рис. 4.3. Графики функций

$$y^{*}(t) = \sin(\pi t) + \cos((\pi + 0.1)t),\ y^{**}(t) = (5 + 0.2\sin(0.1t))\cos(3t + 0.1\sin(5t)).$$

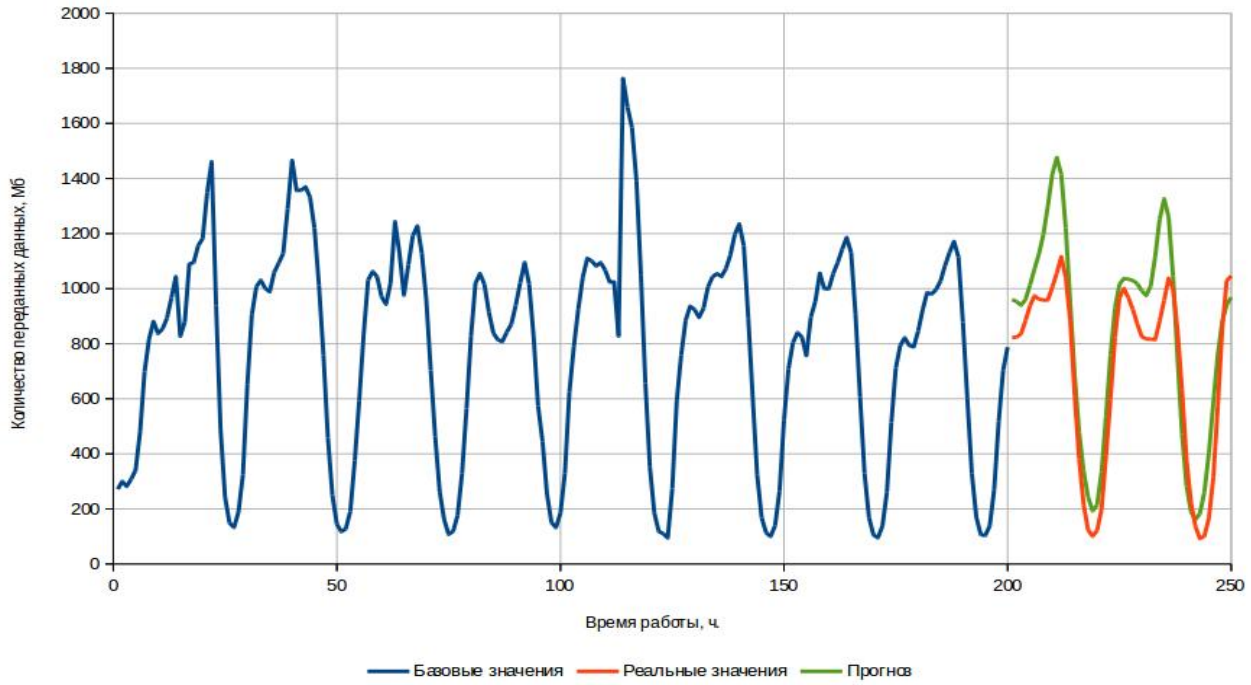


Рис. 4.4. Исходящий трафик сервера и его прогноз

Количество исходящего трафика используется как базовая последовательность для построения тригонометрической аппроксимации

удовлетворяющей (3.4). По полученному аппроксимационному тригонометрическому полиному была построена экстраполяционная кривая обозначенная зелёным цветом. Качество экстраполяции можно наглядно оценить по реальным данным обозначенным красным цветом. Для построения тригонометрического полинома было использовано 8 гармонических составляющих с разными частотами.

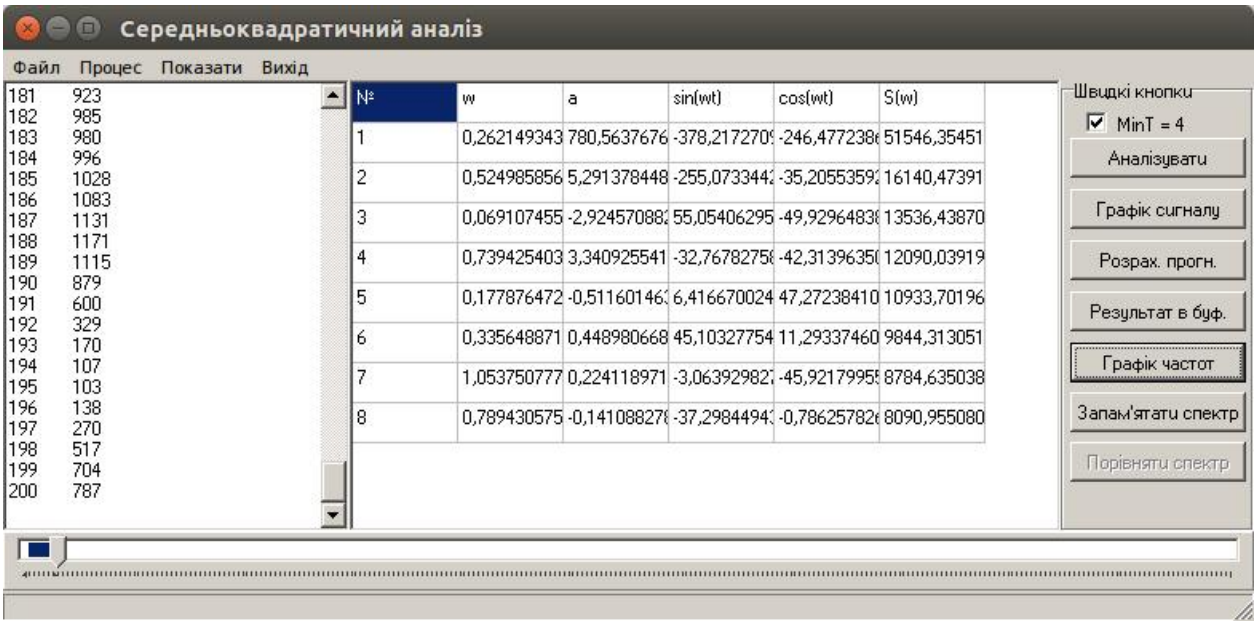


Рис. 4.5. Главное окно программного обеспечения для построения тригонометрического аппроксимационного полинома

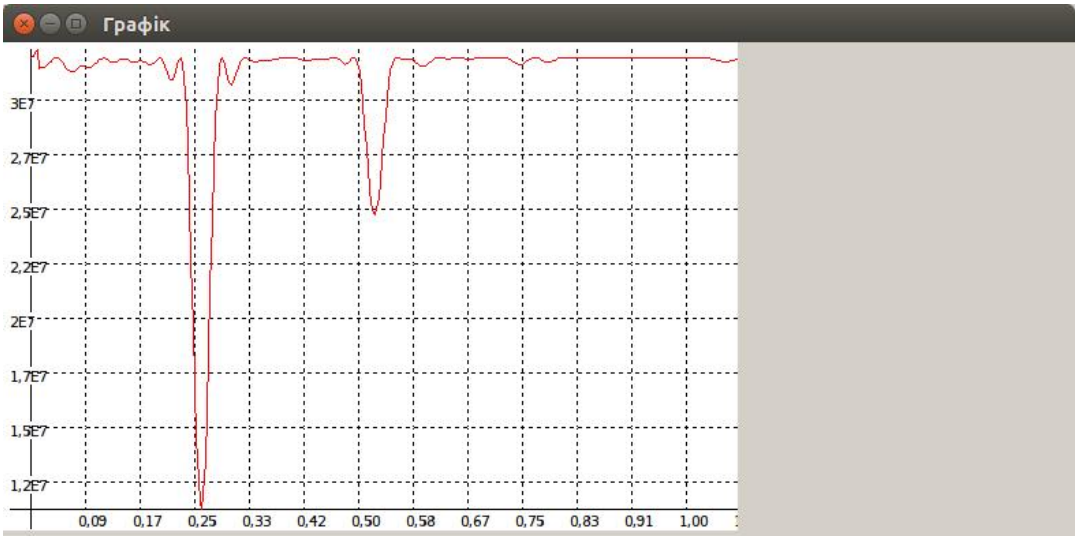


Рис. 4.6. График зависимости суммы квадратов отклонения при заданной частоте тригонометрической компоненты

Внешний вид тестового программного обеспечения на разных этапах построения прогноза показанный на следующих рисунках: рис. 4.5, 4.6, 4.7.

На следующем рисунке изображён процесс построения прогнозированных значений. Также программа предлагает формулу (нижняя правая часть рис. 4.7) пригодную для использования в качестве аппроксимации или экстраполяции в электронных таблицах Excel или Open Calc:

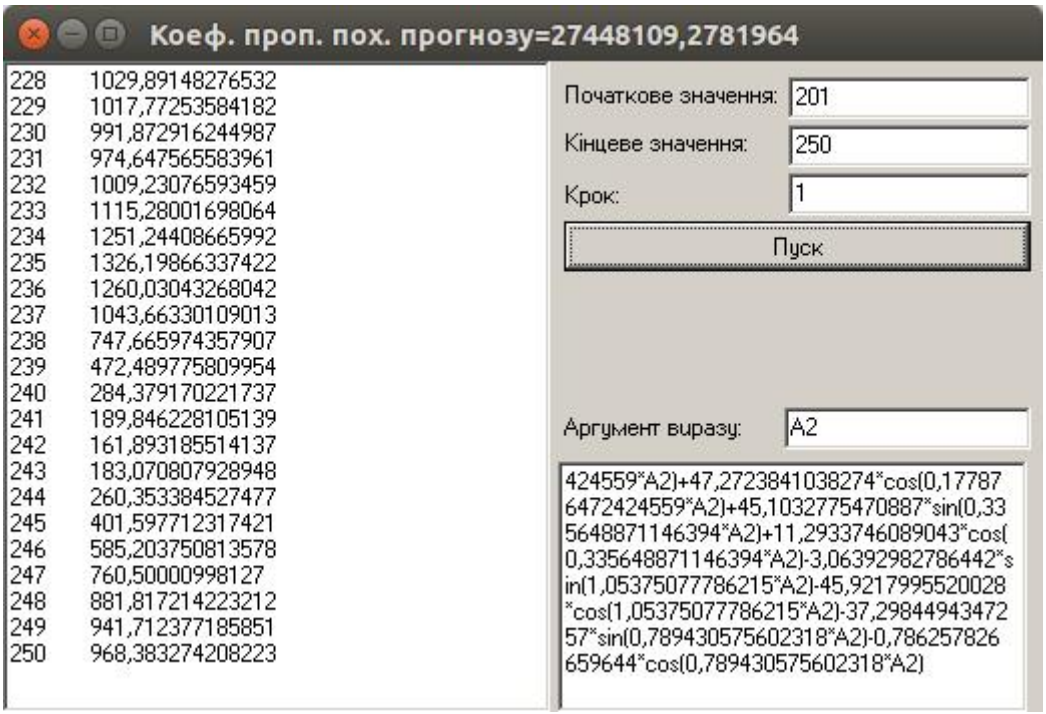


Рис. 4.7. Окно программного обеспечения для построения прогнозов на основе экстраполяции тригонометрическим полиномом

Однако, такая программа непригодна для использования с целью мониторинга краткосрочного прогнозирования на несколько шагов. Для этой цели разработан консольный вариант с использованием языка программирования Free pascal (приложение Г, второй листинг), которое получает со стандартного потока ввода исходные статистические данные и выдаёт на стандартный поток вывода прогнозируемое значение с модулем ожидаемого отклонения и дисперсией. Из дисперсии можно получить доверительный интервал прогнозированного значения, если считать это

прогнозирование случайным числом с математическим ожиданием в реальном значении и полученной дисперсией.

Экспериментальная проверка показывает достаточную надёжность прогнозирования и высокую степень надёжности оценки погрешности полученных прогнозов. Эксперимент проводился прогнозированием скользящим окном с выдерживанием прогноза в коридоре 90%-й надёжности, при которой реальные значения по вероятностной оценке, с вероятностью 90% не выходили за предсказанный коридор.

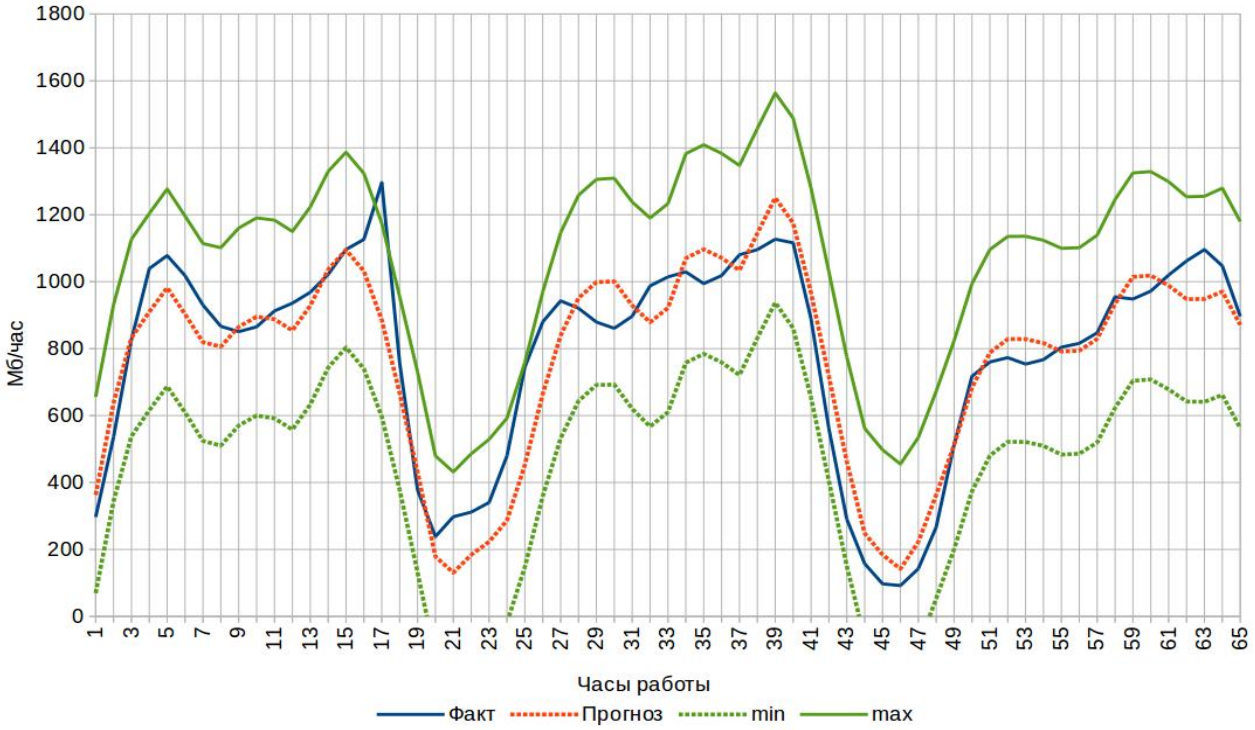


Рис. 4.8. Результаты прогнозирования на фоне реальных, полученных позднее, значений и доверительный 90% интервал

Данные для построения графика показаны в приложении В, таблице 2.

Применимость метода предсказания трафика для ограничения вероятности отказа в обслуживании демонстрируется на основе системы массового обслуживания с очередью, для которой вероятность отказа в обслуживании при относительной интенсивности входящих заказов находится из отношения [18]:

$$p_{отк} = \frac{1-p}{1-p^{N+2}} p^{N+1}.$$

Тогда для N=25 и N=50 вероятность отказа показана на графике рис. 4.9

Рис. 4.9. Вероятности отказа в обслуживании в зависимости от интенсивности поступления заявок

Для сервера с установленным пределом вероятности отказа в обслуживании необходимо чтобы интенсивность трафика была ограничена. Например, из предыдущего графика видно, что для серверов с верхней границей вероятности отказа $p_{отк}$=0,005 ограничениями интенсивности обращений будут p_{25}=0,88 и p_{50}=0,95 соответственно. Фактически, для обеспечения надёжности обслуживания необходим механизм ограничения количества заявок в единице времени, один из которых предложено в главе 3.3.

Предложена система прогнозирования интенсивности заявок к серверу кэширования и дополнительной упаковки изображений. Как показано в [33, 34, 41] предложенный метод даёт возможность получать прогнозы с оценкой вероятности попадания развития процесса в прогнозированный интервал. Также есть возможность поиска доверительного интервала по заданной надёжности прогнозирования. В примере показан прогноз изменения интенсивности запросов к серверу с надёжностью прогнозирования 0,90.

Пример: пускай в результате эксплуатации сервера за время t относительная интенсивность заявок не превышает допустимый порог $t_{доп}$ времени. Тогда, в течение времени $t_{кр}=t-t_{доп}$, вероятность отказа в обслуживании будет выше допустимого порога.

В случае использования прогнозирования интенсивности поступления заявок, мы имеем возможность с вероятностью P_H снизить интенсивность заявок, пустив их в обход системы или иным способом. В таком случае среднее время превышения порога интенсивности потока заявок будет

$$t_{кр}=(1-P_H)(t-t_{доп}),$$

где $(1-P_H)$ — вероятность ошибочности прогнозирования. Результат применения системы прогнозирования показан на рис. 4.10.

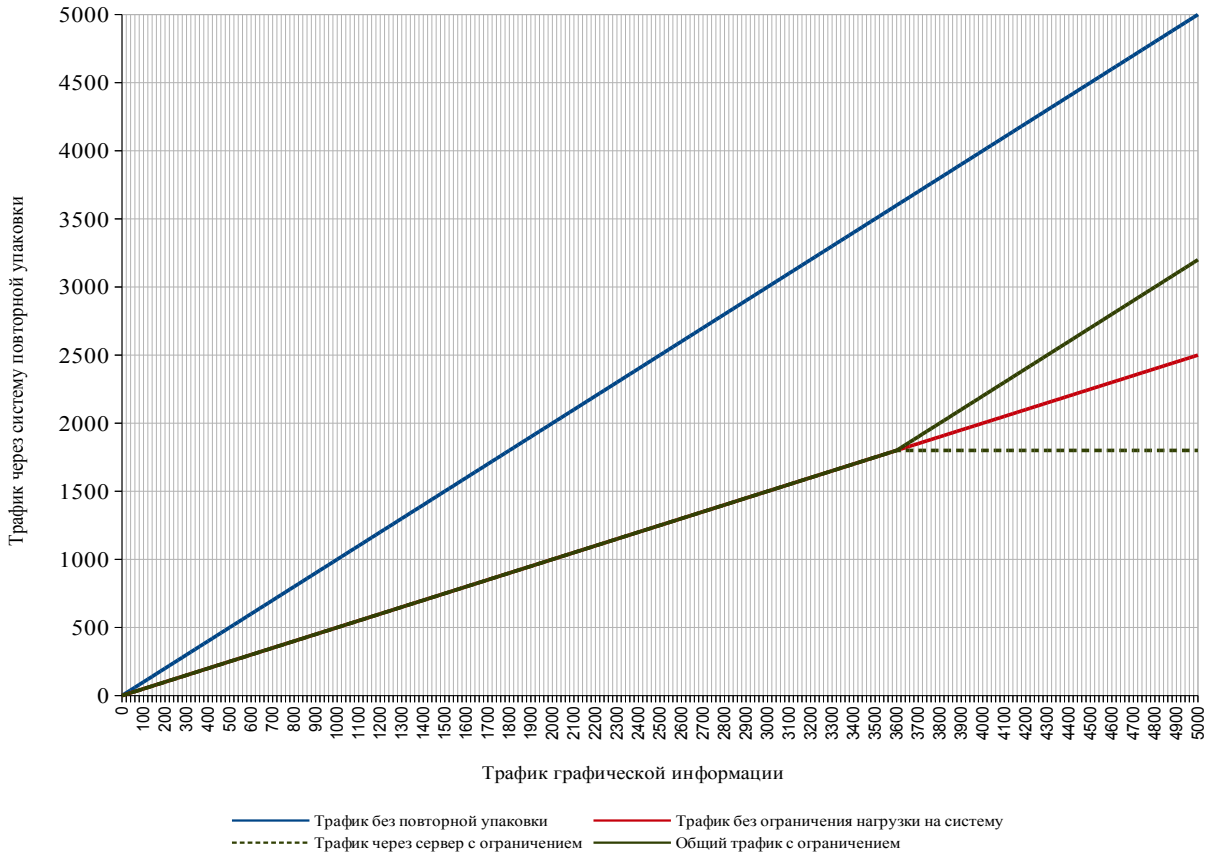


Рис. 4.10. Режимы работы сервера дополнительного сжатия графической информации в зависимости от исходящего трафика

Максимум времени работы сервера в границах допустимой нагрузки может быть достигнут при максимальной точности прогнозирования. Однако, при прогнозировании доверительный интервал прогнозируемых значений пропорционально вероятности попадания в этот интервал, что при высоких вероятностях даёт меньшую точность прогнозируемых значений. В связи с этим, точность прогнозирования или доверительный интервал значений выбирают по индивидуальным параметрам надёжности системы.

Исходя из оценки погрешности прогнозирования (3.9), используется взаимная зависимость вероятности выхода фактического значения P_H как сверху, так и снизу за пределы доверительного интервала [a-b;a+b]. В результате, время соизмеримое с $t_{кр}$, серверное оборудование будет использовано с недогрузкой b, в результате которой часть клиентов будет работать в обход сервера дополнительного сжатия графического контента. Однако за счёт этого время работы с недопустимой нагрузкой сократится в (1–P_H) раз.

Использование системы прогнозирования с допуском недогрузки системы в b=300 Мб/ч и для допустимой пропускной способностью A=2000 Мб/ч, получено прогнозирование на срок в 1 час с 90% надёжностью (фрагмент рис. 4.8, более полный фрагмент в приложении В).

4.3 Разработка по практическому применению метода повышения оперативности доставки часто востребованного графического контента за счёт дополнительного сжатия усовершенствованным кодеком SPIHT

С целью практического сравнения качества сжатия изображений было выбрано 1322 изображения. Все изображения имеют разное количество крупных и мелких деталей. К тестируемым изображениям относятся пейзажные архитектурные и студийные иллюстрации, 3D-арт, фракталы, необработанные ночные пейзажи с высокой долей шумов, и некоторые другие. Также все

полученные изображения были преобразованы к размеру 512x512 пикселей с цветовой передачей RGB 24 бита на пиксель. Изменение размера отсекло первоначальные артефакты полноразмерных изображений jpeg формата, что первоначально ставило в невыгодное положение другие алгоритмы сжатия.

В первую очередь для всех изображений проведена оценка использования арифметического сжатия с битным алфавитом и классическим алфавитом из 256 символов (1 байт на символ в не запакованном виде). Результаты показаны в табл. 4.2.

Таблица 4.2

Результат упаковки несжатых изображений арифметическим побитным и побайтовым блочным сжатием

Тип кодирования	Размер блока с фиксированными вероятностями словаря																Средний размер файла
	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536	13107	26214	52428	10485	
Побитно	206	313	443	184	0	1	2	21	84	42	2	1	0	0	0	0	719643
Побайтно	0	0	0	0	0	0	0	12	127	690	425	60	5	1	3	0	651945

В таблице под размерами блоков указано количество изображений, которые получили наименьший размер упакованной информации. Также для всех упакованных файлов указано среднее значение их размеров. Из полученной таблицы видно, что байтовое кодирование тут есть более выгодным. Однако в побайтном кодировании приходится рассчитывать вероятности для больших блоков, что приводит к несколько большим затратам оперативной памяти. Также можно отметить, что побитовое кодирование имеет большую степень сжатия при блоках размером в 32-256 байт, что значительно меньше блоков для побайтного сжатия в 8-32Кб.

В следующей таблице 4.3 представлены результаты кодирования изображений SPIHT алгоритмом с последующим блочным арифметическим побайтовым кодированием. Однако результаты будут более информативны при сравнении с побитным кодированием, табл. 4.4.

Таблица 4.3

Результат упаковки изображений без потерь алгоритмом SPIHT с последующим блочным арифметическим побайтовым сжатием

Исполь-зованный вейвлет	Размер блока с фиксированными вероятностями словаря														Средний размер
	64	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	
Хаара	0	0	1	0	0	0	1	228	955	121	15	2	0	0	441597
5/3	0	1	0	0	0	0	1	105	1102	95	17	1	1	0	451656
Доб. p=4	0	0	1	0	0	0	0	2	308	658	168	182	4	0	321789

Соответственно, при кодировании изображений сжатых SPIHT алгоритмом более выгодно использовать по битное сжатие, которое и более простое в реализации и менее требовательно в объёме промежуточного буфера памяти.

Более полной проверки требует сжатие с потерями. Для проверки использовались сжатые в 50 раз по сравнению с несжатой информацией (BMP формат размером 512x512 пикселей), изображения jpeg2000, SPIHT, SPIHT с отложенной передачей значимых битов. Для сравнения использованы показатели среднего абсолютного отклонения, максимального абсолютного отклонения, взаимной корреляции, дисперсии, PSNR [7, 97, 110, 112, 115].

Таблица 4.4

Сравнение упаковки изображений без потерь алгоритмом SPIHT с последующим арифметическим побитовым блочным сжатием с побайтовым сжатием

Размер блока	32	64	128	256	512	1024	2048	4096	Средний размер	
									Побитно	Побайтно
Хаара	3	449	808	63	0	0	0	0	437758	441597
Целочисловой 5/3	0	23	793	506	1	0	0	0	446722	451656
Добеши p=4	0	2	4	878	439	0	0	0	317099	321789

Таблица 4.5

Сравнение результатов сжатия с потерями изображений в 50 раз

Кодек	Характеристика				
	Максимальное отклонение	Среднее отклонение	Корреляция	Дисперсия отклонений	PSNR
jpeg2000	106,56	6,67	0,9790	10,087	29,763
SPIHT	112,89	7,94	0,9715	12,572	27,952
SPIHT с отложенной передачей	135,81	20,96	0,9165	29,108	19,530

Из таблицы 4.5 видно естественное отставание SPIHT алгоритмов кодирования, поскольку они используют дополнительную информацию для обеспечения прогрессивности закодированной информации [113]. Практически это означает, что начальный фрагмент файла позволяет восстановить изображение полностью, однако с худшим качеством.

а) оригинал б) SPIHT в) отложенная передача

Рис. 4.11. Зрительное сравнение изображений SPIHT кодирования 0,48 бит на пиксель

Ожидаемым результатом есть худшие показатели кодирования с отложенной передачей значимых битов. Однако, для изображений, в которых отсутствуют монотонные значительные по площади участки, отложенная

передача позволяет отобразить большее количество деталей (рис. 4.11, рис. 4.12).

Несмотря на зональные нарушения цветопередачи (фрагмент в)), на заднем фоне видны детали и более чёткие границы, которых нет на изображении при классическом кодировании (фрагмент б)). Для сравнения приведены исходное изображение а).

В случаях наличия больших монотонных участков, визуально принимаются значительно лучше классическое кодирование. Это связано со значительным квантованием низкочастотных значимых вейвлет коэффициентов.

Вследствие, метод отложенной передачи значимых битов применим в ситуации повышенной значимости наличия деталей изображения, такие как топографические снимки поверхности Земли.

а) оригинальное в) классический SPIHT в) с отложенной передачей

Рис. 4.12. Зрительное сравнение фрагмента изображений SPIHT кодирования 0,48 бит на пиксель

Объективным показателем показания использования модификации метода кодирования SPIHT может быть соотношение сигнал-шум (PSNR) по отношению контура исходного и восстановленного изображения. Для получения показателя для оригинальных и восстановленных после сжатия изображений был выделен контур. Сравнение контуров производилось подсчётом совпавших контурных пикселей L по отношению общего количества

контурных пикселей *N*. Совпавшие фоновые пиксели не учитывались. По причине равности единице модуля максимального отклонения, формула для расчёта PSNR=10·lg(|*max*|^2/*D*) приводится в виду: PSNR=10·lg(*N*/*L*). Результаты сравнения сохранения контурной информации представлены в следующей таблице (табл. 4.6):

Таблица 4.6

Сравнение сохранения контуров при сжатии изображений в 50 раз

50x	JPEG2000		JPEG		SPIHT		OSPIHT		SPIHT vs OSPIHT	
	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
0	20,1637	21,1153	91,9784	8,85709	71,1533	11,0869	69,8467	11,2479	1,3066	0,161
1	19,9478	22,1329	65,2136	11,844	34,4995	17,3745	30,6383	18,4055	3,8612	1,031
2	50,4414	14,0751	65,0204	11,8698	50,0077	14,1501	42,7969	15,5026	7,2108	1,3525
3	13,2798	25,667	52,0512	13,8022	27,9243	19,2112	17,5836	23,2286	10,3407	4,0174
4	74,6169	10,6741	117,297	6,74507	106,463	7,58681	105,144	7,69508	1,319	0,10827
5	55,9114	13,1808	116,326	6,81727	72,255	10,9534	72,2415	10,9551	0,0135	0,0017
6	77,124	10,387	93,7755	8,68901	74,9477	10,6356	74,7435	10,6593	0,2042	0,0237
7	46,0978	14,8572	62,979	12,1469	53,5669	13,5529	45,4544	14,9793	8,1125	1,4264
8	74,5607	10,6806	95,7753	8,50573	68,2311	11,4512	61,8828	12,2994	6,3483	0,8482
9	63,4374	12,0839	89,6693	9,07793	61,0235	12,4209	58,9669	12,7186	2,0566	0,2977
10	79,3656	10,1382	102,107	7,94971	87,1305	9,3274	82,5333	9,79822	4,5972	0,47082
11	29,7793	18,6525	66,107	11,7259	37,452	16,6613	28,5138	19,0297	8,9382	2,3684
12	50,9612	13,986	90,8081	8,96831	55,5794	13,2325	52,0103	13,809	3,5691	0,5765
...	...	...	...	...	...	...	...	...	...	...
1322	63,6272	12,058	97,8666	8,31811	66,774	11,6387	66,6848	11,6503	0,0892	0,0116
									5,1367201	0,8354473

Таблица 4.7

Сравнение количество оценок вейвлет коэффициентов

№	50x		Отно-шение	33x		Отно-шение	25x		Отно-шение
	SPIHT	OSPIHT		SPIHT	OSPIHT		SPIHT	OSPIHT	
0	17191	17826	1,036938	18134	21626	1,192566	32104	36847	1,147739
1	17094	19901	1,16421	21043	27138	1,289645	34896	42729	1,224467
2	17639	17951	1,017688	19997	23233	1,161824	34972	39901	1,140941
3	11802	14363	1,216997	26034	25992	0,998387	25992	32662	1,256617
4	12768	15011	1,175674	29491	31864	1,080465	33751	34448	1,020651
5	20285	22045	1,086764	26953	34001	1,261492	42573	51649	1,213187
6	17307	18162	1,049402	18162	20422	1,124436	28890	35534	1,229976
7	16065	21173	1,317958	25243	29517	1,169314	34090	45330	1,329715
8	12513	15574	1,244626	28899	29242	1,011869	29242	38793	1,326619
...	...			...			...		
1322	9085	10957	1,206054	24377	26721	1,096156	39933	41072	1,028523
Ср.			1,095815			1,1112			1,126414

Также для более уверенного результата исследования было использование протоколирование количество оценок вейвлет коэффициентов при восстановлении упакованного изображения, результаты которого представлены в следующей таблице 4.7.

По результатам исследования можно заключить, что использование отложенной передачи значимых битов позволило повысить соответствие контура на 20% (по количеству пикселей) и на 9-12% количество оценки значимых вейвлет коэффициентов.

4.4 Результаты имитационного моделирования системы прогрессивного сжатия графического контента

С целью определение перспективности дальнейшего внедрения системы дозагрузки дополнительно сжатых изображений было проведено имитационное моделирование. Узким местом локальной сети является беспроводная связь Wi-Fi, которая обеспечивается роутером D-Link R-615 стандарта IEEE 802.11n с максимальной фактической скоростью передачи-приёма данных 150 Мбит/с.

На сегодняшний день распространены стандарты беспроводной связи IEEE 802.11b, IEEE 802.11a и IEEE 802.11g, также был утверждён 11 сентября 2009 года стандарт IEEE 802.11n. Применение последнего стандарта позволяет повысить скорость передачи данных до 600 Мбит/с, однако большинство устройств на сегодняшний день поддерживают скорость соединения до 150 Мбит/с и эта скорость достигается только при идеальных условиях. С 2011 по 2013 разрабатывался стандарт IEEE 802.11ac, окончательное принятие стандарта запланировано на 2014 год. Скорость передачи данных при использовании 802.11ac может достигать нескольких Гбит/с. 27 июля 2011 года институт инженеров электротехники и электроники (IEEE) выпустил официальную версию стандарта IEEE 802.22. Системы и устройства,

поддерживающие этот стандарт, позволят передавать данные на скорости до 22 Мбит/с в радиусе 100 км от ближайшего передатчика.

В экспериментальной сети находились актуальные на сегодняшний день устройства, которое по большей части не поддерживали стандарт IEEE 802.11n и осуществляли соединение на скорости максимум в 54 Мбит/с. Также ситуация усугублялась наличием большого количества соседних беспроводных сетей, что привело к средней скорости обмена данными на уровне до 10Мбит/с. Схема топологии установленной сети показана на рис. 4.13.

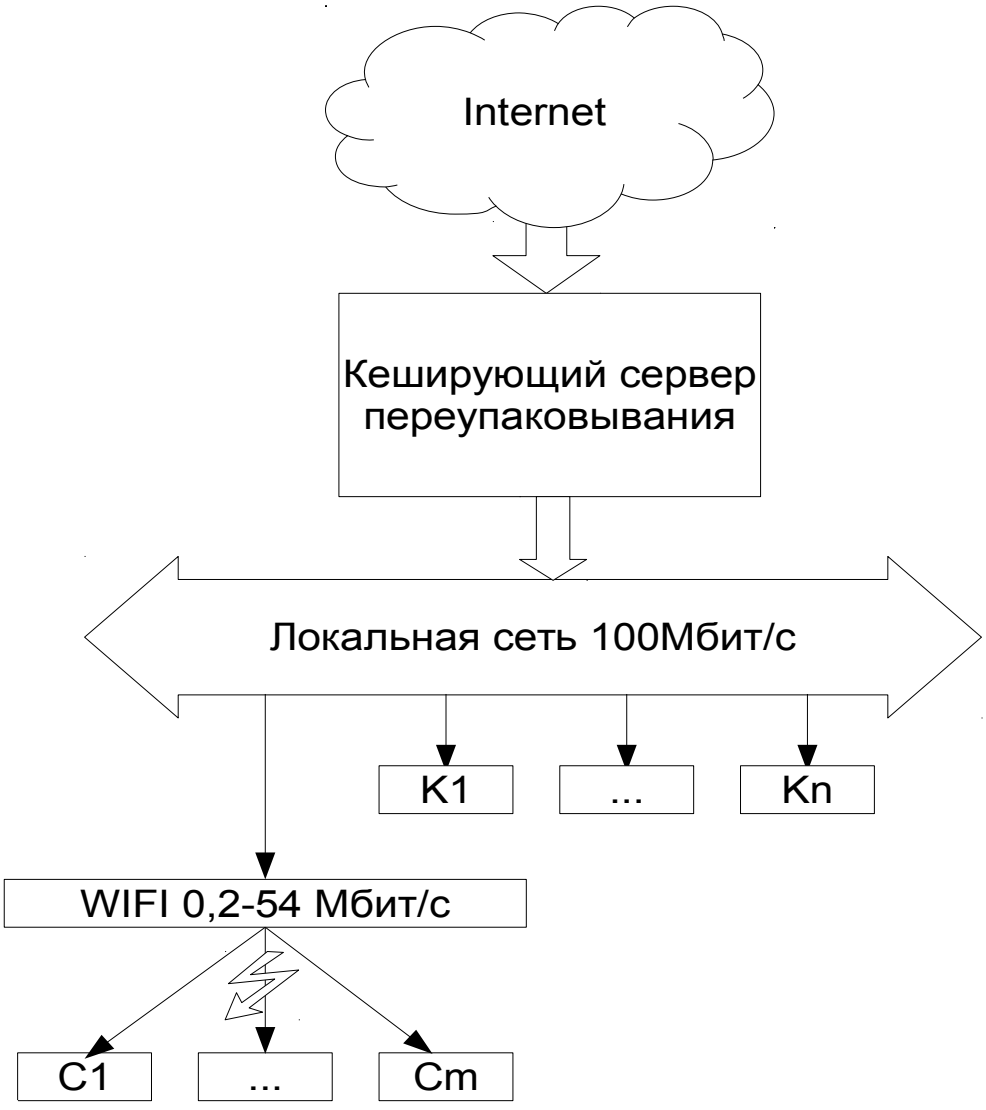


Рис. 4.13. Схема топологии сети с использованием технологии дополнительного сжатия графической информации

Проблемный моментом эксплуатации сети является просмотр графической информации одним или несколькими клиентами, в этот момент система обеспечения QoS, в случае превышения пропускной способности беспроводной сети, отбрасывает из очереди пакеты более низкого приоритета. Такое поведение, при поддержке качества мультимедиа и IP-voice, существенно снижает качество обслуживания других пользователей. Предложенная автором система позволяет отбросить большую часть пакетов с решением о существенности самим пользователем не на основе срочности пакетов, а на основе семантической наполненности фотографического изображения по её черновикам.

Исследование проводилось в границах правдоподобных значениях, в которых мультимедийный трафик составлял 75% пропускной способности сети. Остальные 15% процентов составлял http контент с 85%-ю графической информации. По причине подавляющего количества трафика от единой точки доступа к клиентам сети, коллизии пакетов были относительно редкими и при теоретических оценках не учитывались. Общая пропускная способность сети была занята на 85-90%.

В результате задействования кэширующего сервера дополнительного сжатия информации, трафик графической информации в локальной сети упал в три раза (в результате дополнительного сжатия клиентам отправлялись файлы с черновым содержанием имеющими размер вчетверо меньший по сравнению с оригиналом, пользователи получали оригинальное качество только при дополнительном запросе). После чего занятость канала передачи упала до с 90% до 80% от пропускной способности. Также, снижение загруженности WiFi канала связи на 10% значительно снизило вероятность отброса пакетов низкого приоритета, что уменьшило показатель латентности в сети. Поскольку сеть принадлежала коллективу с объединённым видом деятельности, наблюдалось однообразие контента для пользователей локальной сети. Поэтому графическая информация из Интернета получалась один раз, и раздавалась по требованию из сервера клиентам; это позволило снизить внешний трафик графической

информации пропорционально количеству клиентов локальной сети, в общем случае экономия составляла до 20% от первоначальных значений.

Использование системы дополнительного сжатия графического контента возможно внедрять и на локальных сетях, с топологией аналогичной показанной на рис. 4.13. Оценка изменения времени загрузки страниц при http просмотре прямому измерению не поддаётся, по причине изменения в широких пределах объёмов информации. Поэтому произведено измерение временных задержек по доставке отдельных пакетов обычного и высокоприоритетного трафиков. Также проведено имитационное моделирование существующей сети с внедрённой технологией дополнительного сжатия графического контента. Имитационное моделирование тут есть необходимым этапом, подтверждающим теоретические ожидания и как дополнительная база для обоснования внедрения разработанной системы.

В реализованной схеме на рис. 4.13 через общий промежуточный сервер предоставляется доступ к внешней сети. На этом сервере происходит фильтрация http запросов низкоприоритетного трафика, в результате которой запросы на изображения во внешнюю сеть транслируются только при отсутствии локальной копии в КЕШ базе данных. Пользователь получает частично загруженное изображение для оценки значимости содержимого, после чего изображение может быть дополнено для восстановления первоначального качества. В зависимости от настроек системы дополнительного сжатия изображения общий трафик графического контента может быть уменьшен на 10..90%.

Внедрение системы на локальном и корпоративном уровне, где информационные потоки связаны в большей части едиными тематиками, могут привести к большей экономии внешнего трафика, за счёт более частого повторения запросов разными пользователями в уже созданный КЕШ. Также загрузка сниженного качества изображения с возможностью восстановления качества по дополнительному требованию пользователя, уменьшают поток внутреннего трафика. Однако влияние интенсивности трафика на

оперативность доставки отдельных пакетов не прямая, эта зависимость является следствием уменьшения очередей пакетов на устройствах маршрутизации, шлюзах и других устройствах ТКС. Оценить такое изменение возможно при помощи теории массового обслуживания с ограниченной очередью и отказами в обслуживании.

Для теоретической оценки среднего времени пребывания пакета в очереди использовано два основных параметра: относительная нагрузка на сеть p, как отношение фактического трафика к пропускной способности коммуникационного канала; и максимальная длина очереди N, которая зависит от величины буферов приёма/передачи в устройствах ТКС.

В [18] подана теория систем массового обслуживания, из которой выводится соотношение зависимости среднего времени пребывания пакета в системе в зависимости от нагрузки на эту систему и длины очереди:

$$t \sim \frac{1-(N+1)p^N + Np^{N+1}}{(1-p)(1-p^{N+1})(1-p^N(1-p)/(1-p^{N+1}))}.$$

График $t(p)$ зависит от длины очереди, однако тенденция поведения системы сохраняется при всех значения параметра N. В качестве примера на рис. 4.14 показан график при N=25 и N=50.

Также одним из следствий моделирования есть очевидность уменьшения времени нахождения пакета в системе при уменьшении нагрузки на телекоммуникационный коммутатор или шлюз, при этом время задержки снижается тем сильнее, чем более первоначально была загружена телекоммуникационная сеть. Поэтому уменьшение количества заявок на обслуживание есть актуальным направлением и для систем ТКС.

Рис. 4.14. Относительное время ожидания обслуживания заявки

Для более детального изучения влияния объёма трафика на оперативность доставки пакетов информации было использовано имитационное моделирование локальной компьютерной сети.

Как инструмент для имитационного моделирования был выбран программный пакет Opnet Modeller в редакции для образовательных целей (отличается ограничением на коммерческое использование), который имеет в своём распоряжении средства для построения компьютерных сетей разных масштабов и топологий. Программный пакет имеет широкую базу данных готовых сетевых элементов, которые уже имеют в себе алгоритмы имитации и этапов обработки сетевых данных с учётом затрат времени на их выполнение реальными физическими устройствами.

На основе структуры реальной локальной офисной сети построена модель с беспроводным сегментом. Полная модель топологии сети показана на рис. 4.15.

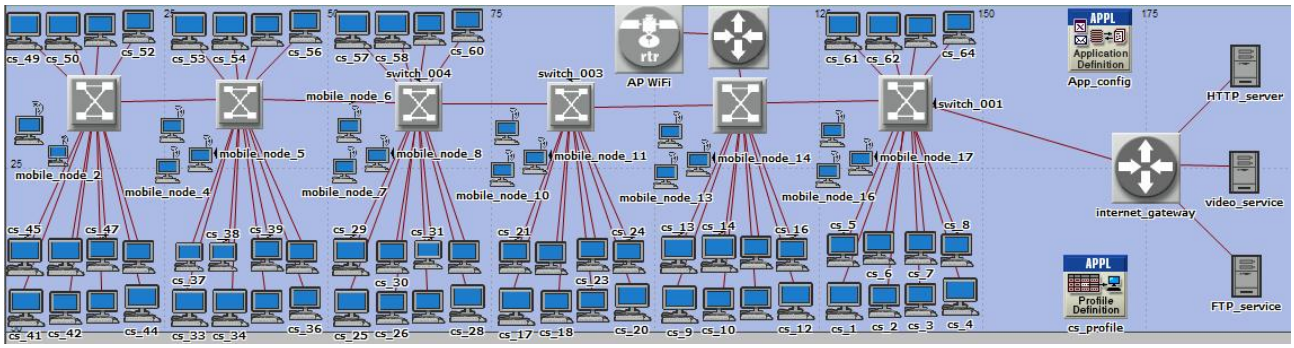


Рис. 4.15. Модель офисной компьютерной сети

На модели видно расположение стационарных хостов с проводным соединение, расположенных в отделениях. Структурные отделения хоть и находятся в одном адресном пространстве, но разделены отдельными коммутаторами. Беспроводная сеть доступна через отдельный маршрутизатор как и сервисные сервера, которые в модели играют роль Интернет-ресурсов.

По причине наличия бетонных перекрытий и зашумлённости эфира, беспроводная связь имеет показатели скорости передачи данных в пределах 11Мбод. В модель входят 64 стационарных персональных компьютера и 18 компьютеров с беспроводном подключением к сети. Восемь компьютеров используют протокол повышенного приоритета IP телефонии, остальные компьютеры используют http и ftp протоколы обмена пакетами общего приоритета.

Моделирование проводилось в условии средней интенсивности обмена файлами по ftp протоколу, использованием IP телефонии ограниченным количеством клиентов во избежание превышения пропускной способности сети, и интенсивном просмотре WEB страниц насыщенных графической информацией. Средний размер графического файла можно указать в параметрах http сервера модели показанной на рис. 4.15. В процессе моделирования проводилось измерение времени доставки, как пакетов повышенного приоритета, так и пакетов данных общего назначения. Результаты роботы модели в течение виртуальных 10 мин. представлены на рис. 4.16 в виде графиков.

На графике с рис. 4.16 видно, что задержка трафика повышенного приоритета при указанных условиях составляет в среднем 0.20 сек. В ethernet сети задержка значительно ниже и составила в среднем 1.5 миллисекунд с наличием временных спадами до 10-22 миллисекунд.

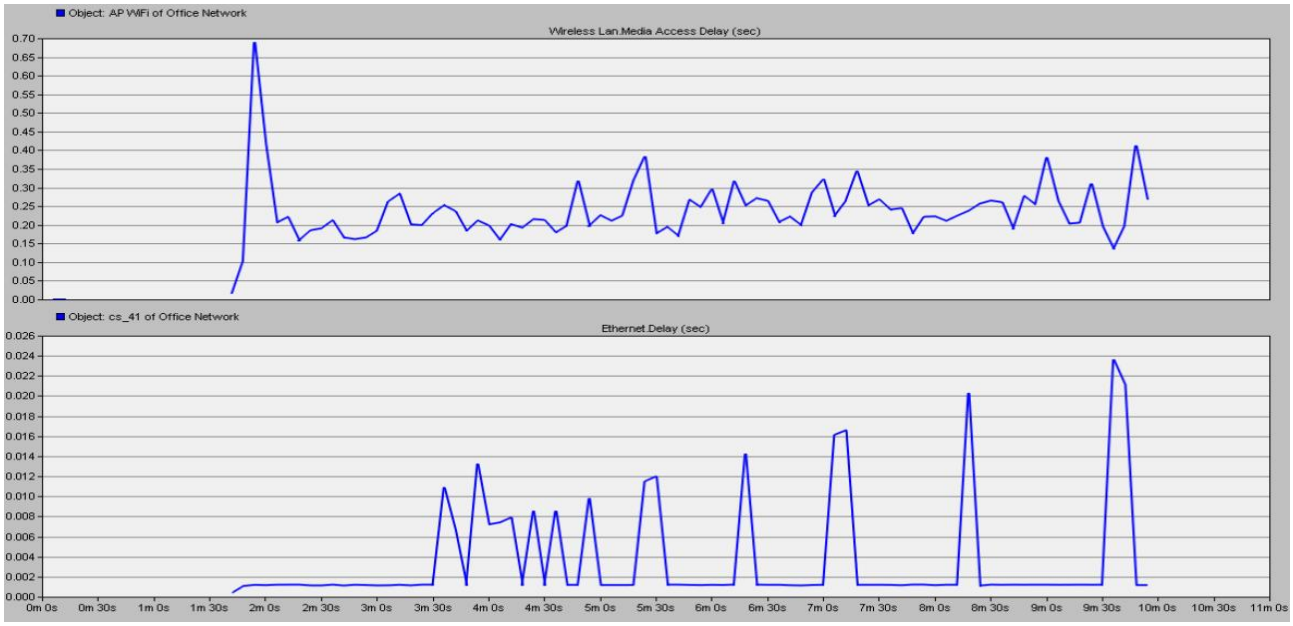


Рис. 4.16. Временные задержки передачи пакетов данных общего назначения (нижний график) и пакетов повышенного приоритета (верхний график)

Повторное моделирование, результаты которого показаны на рис. 4.17, проводилось с изменением настроек http сервера, в котором был изменён средний размер файла изображения до 50% от первоначального. В реальной системе этот показатель может быть плавающим для удержания по возможности параметров сети в предварительно заданных пределах. В конкретном примере задержка пакетов повышенного приоритета в беспроводном соединении составила 0.17 сек., а в ethernet соединении 0.7 миллисекунд.

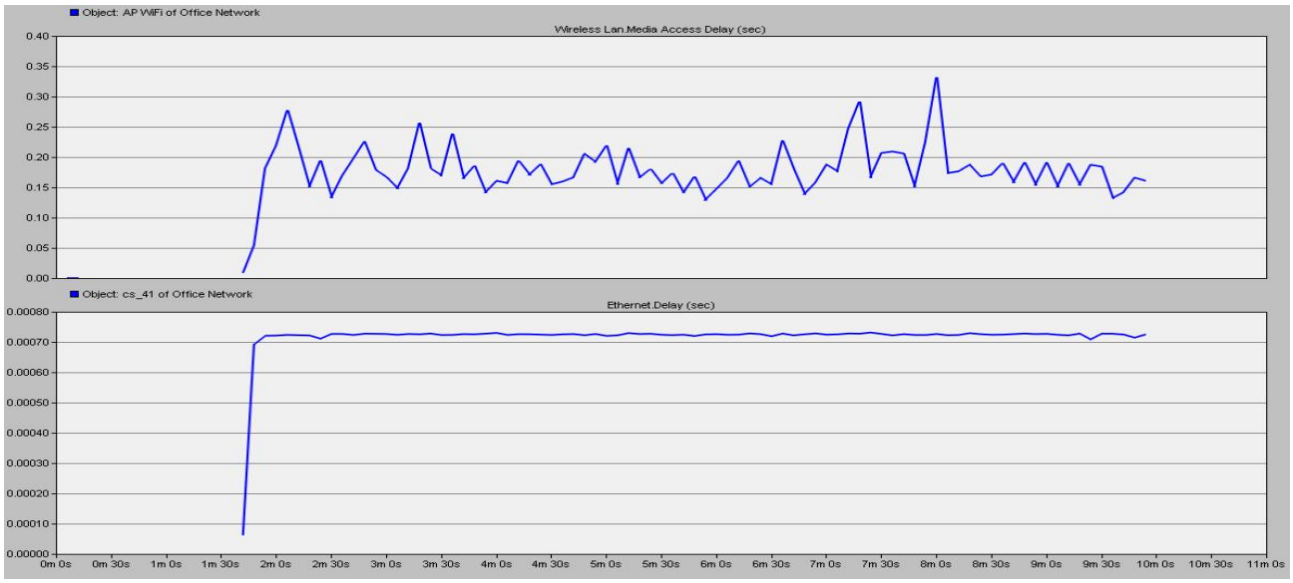


Рис. 4.17. Временные задержки передачи пакетов при уменьшенном трафике графического контента

По результатам моделирования можно сделать выводы, что снижение интенсивности общего трафика на 4-6% дало результаты снижения временных интервалов доставки пакетов не только для высокоприоритетной части трафика. Ключевые результаты моделирования можно сравнить со следующей таблицы:

Таблица 4.8

Изменение задержек доставки пакетов в сети

	Без дополнительного сжатия, сек.	С дополнительным сжатием, сек.	Относительное изменение задержки пакетов,
WiFi	0.22	0.17	77 %
Ethernet	0.0017	0.0008	47 %

Окончательно получено подтверждение для загруженных сетей: при незначительном уменьшении трафика в сети наблюдается значительное уменьшение времени передачи пакета в этой сети, результат более значимый для высокоприоритетной части трафика.

Практическая проверка влияния загруженности сети на оперативность передачи пакетов данных было проведено на стационарном компьютере с программным ограничением трафика со стороны провайдера в 60 Мбит/сек. Сеть нагружалась при помощи скачивания группы файлов из внешней сети, при этом проводились замеры прохождения ICMP пакетов утилитой ping.

Таблица 4.9

Результаты измерения задержек ICMP пакетов

Нагрузка сети, Кб/сек.	Нагрузка сети, %	t, мс. минимум	t, мс. среднее	t, мс. максимум
5000	83	7,8	23,6	77,2
4000	66	8,2	15,3	77,6

Результат оценивался по 1000 измерений времени ответа на ICMP-эхо запрос к удалённому серверу, который не относился к файловым серверам для загрузки тестовых файлов. Получены результаты для сети с пропускной способностью 6000Кб/с (табл. 4.9).

В результате снижения нагрузки на сеть на 20% было улучшено среднее время прохождения пакетов на 35%. Изменения максимального и минимального времени прохождения пакетов остались в прежних пределах.

Выводы

В результате практических экспериментов и математического моделирования получены результаты:

1. На практике установлена линейная зависимость между размером сообщения и временем его передачи в телекоммуникационной системе или сети.

2. Установлена линейная зависимость среднего квадратического отклонения времени получения сообщения от качества связи и размера сообщения.

3. Система прогнозирования загруженности сервера даёт возможность предупредить отказы в обслуживании с заданной вероятностью.

4. Введение в стандартное кодирование изображения методом SPIHT процесса отложенной передачи значимых битов позволяет значительно повысить количество передаваемых деталей изображения, однако ухудшает характеристику соотношения сигнал-шум. Использование отложенной передачи значимых битов необходимо решать с учётом конкретной значимости изображений и сферой их использования.

5. Внедрение кэширующего сервера с дополнительным сжатием графического контента позволяет снизить нагрузку на канал связи на 5-10%. В зависимости от количества повторных запросов к часто востребованной информации также может значительно уменьшить графический трафик из внешней сети.

ВЫВОДЫ

В диссертационной работе решена актуальная научно-техническая задача, которая состоит в разработке метода повышения эффективности передачи информации в корпоративных сетях. Проведённые в диссертационной работе исследования, результаты решения научно-технических и частично научных задач, а также результаты расчётов и сравнительного анализа, дали возможность получить следующие научные и практические результаты.

1. Проведённый анализ моделей, методов и инструментов управления информационным потоком в современных корпоративных сетях показал, что в условиях повышенной интенсивности графического трафика используемые в наше время методы обработки информации в ТКС не позволяют обеспечить необходимый уровень эффективности передачи информации. Исследования основных моделей и методов повышения эффективности передачи информации позволили определить основные направления диссертационного исследования и пути решения научных задач.

2. Впервые разработан метод прогнозирования трафика сервера ТКС, который отличается от известных использованием разложения на гармонические составляющие с некратными частотами последовательным выделением трендов, что позволило уменьшить вероятность отказа в обслуживании и уменьшить интенсивность графического трафика в корпоративной сети в 1,6 раза.

3. Усовершенствован метод прогрессивного сжатия информации, который в отличие от известных имеет лучшие характеристики сохранения контуров в изображении и даёт возможность регулирования трафика графического контента, что позволило при сравнимых степенях сжатия повысить детализацию изображения на 12%.

4. Получила дальнейшее развитие математическая модель процесса передачи многопакетных сообщений в ТКС, которая в отличие от известных, учитывает возможности и особенности использования промежуточного телекоммуникационного оборудования для дополнительного сжатия графического контента, что позволило оценить время доставки многопакетных

сообщений в условиях дополнительного сжатия низкоприоритетного графического трафика.

5. Проведено исследование эффективности разработанного метода повышения эффективности передачи данных, которое показало уменьшение времени передачи отдельных пакетов на 4%-6% за счёт уменьшения графического трафика; обоснованы практические рекомендации к использованию разработанного метода.

Таким образом, полученные в диссертационной работе результаты дают основания полагать, что поставленная научно-техническая задача уменьшения графического трафика с регулированием вероятности отказа в обслуживании решена, а цель повышения эффективности передачи информации в корпоративных сетях, достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Айфичер Э.* Цифровая обработка сигналов. Практический подход / Э. Айфичер – М.: «Вильямс», 2004. – 992 с.
2. *Амиантов И.Н.* Избранные вопросы статистической теории связи / Амиантов И.Н. – М.: Советское радио. – 1971. – 416 с.
3. *Артеменко М.Е., Касымов Р.Р.* Прогнозирование временных рядов для задач управления с использованием аппарата нейронных сетей и вейвлет анализа / М.Е. Артеменко, Р.Р. Касымов // Холодильна техніка і технологія. – 2010. – №3. – С. 74-76.
4. *Артеменко М.Е, Касымов Р.Р.* Использование вейвлетов для повышения качества прогноза телекоммуникационного трафика/ М.Е. Артеменко, Р.Р. Касымов // Зв'язок. – 2010. – №3.- С. 57-60.
5. *Баранник В.В.* Обоснование базовой технологии компрессии изображений с заданным качеством визуализации / В.В. Бараник, И.Е. Рогоза, А.А. Краснорущкий // Інформаційно-керуючі системи на залізничному транспорті. – 2013. – Вип. 1. – С. 17-23.
6. *Баранник В.В.* Теоретичні основи та методи стиску зображень в телекомунікаційних системах на підставі біноміально-поліадичного представлення: Автореф. дис. д.т.н.: 05.12.02 / В.В. Баранник //. – Харків, Українська державна академія залізничного транспорту 2006. – 36 с.
7. *Бараннік В.В.* Метод стиснення НН-квадратури вейвлет-перетвореного зображення на основі поліадичного кодування/ В.В. Бараннік, А.В. Ширяєв, П.Н. Гуржий // Наукоємні технології, 2011. № 1-2 (9-10) – С. 69-72.
8. *Богданова Н.В.* Метод та способи підвищення ефективності управління телекомунікаційними мережами: Автореф. дис. кандидата техн. наук: 05.12.02/ Н.В. Богданова // Державний університет інформаційно-комунікаційних технологій. – К. 2008. – 20 с.

9. *Бомба А.Я*. Застосування арифметичного кодування у растровому графічному форматі PNG / А.Я. Бомба, О.В. Шпортько // Вісник Нац. ун-ту водного господарства та природокористування: серія «Технічні науки». – Вип. 2(50). – 2010. – С. 246-247.
10. *Ватолин Д.С*. Алгоритмы сжатия изображений: метд. пособие / Д.С. Ватолин – М.: МГУ им. В.М. Ломоносова, 1999. – 76 с.
11. *Ватолин Д.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юркин. – М.: ДИАЛОГ-МИФИ, 2003. – 384 с.
12. *Вегешна Ш.* Качество обслуживания в сетях IP / Ш. Вегешна – Пер. с англ.. – М.: Издательский дом «Вильямс», 2003. – 386 с.
13. *Вентцель Е.С.* Теория вероятностей: учеб. для вузов / Е.С. Вентцель. – М.: Высш. шк., 1999. – 576 с.
14. *Вишневский В.М.* Теоретические основы проектирования компьютерных сетей / В.М. Вишневский – М.: Техносфера, 2003. - 512 с.
15. *Галкин В.А.* Телекоммуникации и сети / В.А. Галкин, Ю.А. Григорьев. – М.: МГТУ имени Н.Э. Баумана, 2003. – 608 с.
16. *Гмурман В.Е.* Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.
17. *Гонсалес Р.* Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1072 с.
18. *Городецкий А.Я.* Информационные системы. Вероятностные модели и статистические решения. Учебн. пособие / А.Я.Городецкий СПб: Изд-во СПбГПУ, 2003. 326 с.
19. *Гриньов Д.В*. Методи стиснення зображень в системах цифрової обробки даних / Д.В. Гриньов, З.З. Закіров // Системи обробки інформації. – Х.: ХУПС, 2010. – Вип. 2(83). – С. 66-70.

20. *Гук М.Ю.* Аппаратные средства локальных сетей. Энциклопедия / Михаил Юрьевич Гук – СПб.: Издательство “Питер”, 2000. – 576 с.

21. *Донченко В.С.* Теорія ймовірностей та математична статистика: навчальний посібник / В.С. Донченко, М.В.-С. Сидоров, М.М. Шарапов – К.: ВЦ «Академія», 2009. – 288 с.

22. *Дреев А.Н.* Использование неравномерного распределения единичных битов для дополнительного сжатия SPIHT кода / А.Н. Дреев, А.А. Смирнов // Информационные системы в управлении, образовании, промышленности: монография. Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014. – С. 498.

23. *Дрєєв О.М.* Дослідження впливу шляху розгортки на ступінь ентропійного стиснення цифрового зображення / О.М. Дрєєв, О.В. Слюсар // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. Випуск 21. – Кіровоград: КНТУ. – 2008 – С. 115-118.

24. *Дрєєв О.М.* Метод розвантаження телекомунікаційного сервера за рахунок кешування зображень / О.М. Дрєєв // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. Випуск 25. Ч. І. – Кіровоград: КНТУ. – 2012 – С. 419-424.

25. *Дрєєв О.М.* Метод прогнозування завантаженості серверу телекомунікаційної мережі / О.М. Дрєєв, О.А. Смірнов, Є.В. Мелешко, О.В. Коваленко // Системи обробки інформації. Випуск 3(101) Том 2. – Х.: ХУПС. – 2012. – С. 181-188.

26. *Дрєєв О.М.* Оцінка якості стиснення зображень на основі дискретного перетворення Хартлі / О.В. Коваленко, О.П. Доренський, О.М. Дрєєв // Системи озброєння і військова техніка. Науковий журнал 2(34) – Х.: ХУПС – 2013. С. 99-102.

27. *Дрєєв О.М.* Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі / О.А. Смірнов, О.М. Дрєєв, О.П. Доренський // Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

28. *Дреев А.Н.* Сравнение битовых плотностей при использовании различных методов кодирования информации / А.Н. Дреев, А.А. Смирнов // Системи обробки інформації, 2014, випуск 2 (118), том 2 – Харків: ХУПС – 2014. С 64-66.

29. *Дрєєв О.М.* Моделювання впливу інтенсивності трафіку на оперативність доставляння інформації / О.М. Дрєєв // Науково-виробничій журнал "Зв'язок". – Київ: ДУТ, 2014. – № 2 (108) С. 24-29.

30. *Дреев А.Н.* Повышение вероятности доставки сообщений в телекоммуникационных системах и сетях для обеспечения информационной безопасности / А.Н. Дреев, А.А. Смирнов //«Безпека інформації» Том 21, №1 2015 р. – Київ: НАУ – 2015. – С. 22-28.

31. *Дрєєв О.М.* Узагальнення вейвлету Хаара / О.М. Дрєєв, Г.М. Дрєєва // Збірник тез доповідей Комбінаторні конфігурації та їх застосування, 15-16 жовтня 2010 р. – Кіровоград – С. 58

32. *Дрєєв О.М.* Узагальнення вейвлету Хаара / О.М. Дрєєв // Матеріали науково-практичної конференції, присвяченої 80-річчю фізико-математичного факультету КДПУ ім. В. Винниченка 26 листопада 2010 р. – Кіровоград – С. 12

33. *Дрєєв О.М.* Метод прогнозування завантаженості серверу телекомунікаційної мережі / О.М. Дрєєв, О.В. Коваленко // Тези доповідей Новітні технології – для захисту повітряного простору. Дев'ята наукова конференція. 18-19 квітня 2011 р. – Х.: ХУПС. – 2012. – С. 206

34. *Дрєєв О.М.* Метод довгострокового прогнозування навантаження серверу телекомунікаційної мережі / О.М. Дрєєв, Г.М. Дрєєва // Комбінаторні

конфігурації та їх застосування. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: “Екслюзів-систем”. – 2012. – С. 50

35. *Дрєєв О.М.* Вдосконалення стиснення зображень SPIHT методу шляхом додаткового кодування та відкладеної передачі уточнення вейвлет коефіцієнтів / О.М. Дрєєв // Дискретна математика та її застосування у економіко-математичному моделюванні та інформаційних технологіях. 11-13 жовтня 2012 р. – Запоріжжя: ЗНУ – 2012. – С. 22-23.
36. *Дрєєв О.М.* Методи підвищення якості обслуговування у телекомунікаційних системах та мережах / О.М. Дрєєв, Г.М Дрєєва, О.А. Смірнов // Збірник тез доповідей. Академія внутрішніх військ МВС України “Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку” 20-21 березня 2013р. – Харків: АВВ. – 2013. С. – 18-19
37. *Дреев А.Н.* SPIHT кодирование с отложенной передачей значимых битов / А.Н. Дреев // Тези доповідей. Новітні технології – для захисту повітряного простору. Дев’ята наукова конференція 17 квітня 2013 р. – Х.: ХУПС. – 2013. – С. 206
38. *Дреев А.Н.* Повышение оперативности доставки данных повышенной востребованности в телекоммуникационных системах и сетях / А.Н. Дреев, А.А. Смирнов, Е.В. Мелешко // Проблеми і перспективи розвитку ІТ-індустрії 25-26 квітня 2013 р. Системи обробки інформації. – Випуск 3 (110). Том 2. – Харків: ХУПС. – 2013. С. – 199.
39. *Дрєєв О.М.* Середньостатистичний та найімовірніший час доставки багатопакетного повідомлення в телекомунікаційній системі або мережі / О.М. Дрєєв, О.А. Смірнов // V Всеукраїнська науково-практична конференція "Інформатика та системні науки" ІСН – 2014, 13-15 березня 2014 року, м. Полтава – С. 92

40. *Дрєєв О.М.* Визначення оптимального розміру блоку при бітовому арифметичному кодуванні / О.М. Дрєєв, Г.М. Дрєєва // Збірник тез доповідей Комбінаторні конфігурації та їх застосування, 11-12 квітня 2014 р. – Кіровоград – С. 44

41. *Дреев А.Н.* Экстраполяция квазипериодических процессов с аддитивными помехами / А.Н. Дреев, А.А. Смирнов // П’ята Міжнародна науково-практична конференція "Інформаційні технології та моделювання в економіці" 15-16 травня 2014 р. – Черкаси – С. 59

42. *Дреев А.Н.* Статистическая модель передачи многопакетного сообщения в телекоммуникационной системе или сети / А.Н. Дреев, А.А. Смирнов // «Компьютерное моделирование в наукоемких технологиях (КМНТ-2014)» Харьков, 28-31 мая 2014 года – С. 137-140

43. *ДСТУ* 2481 – 94 Системи оброблення інформації інтелектуальні інформаційні технології. Терміни та визначення. – Х.: ДЕРЖСТАНДАРТ УКРАЇНИ, 1994. – 33 с.

44. *ДСТУ* В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.:УкрНДІССІ, 1995. – 23 с.

45. *Дымарский Я.С.* Управление сетями связи: принципы, протоколы, прикладные задачи / Я.С. Дымарский., Н.П. Крутякова, Г.Г. Яновский – М.: ЭкоТрендз, 2003. – 384 с.

46. *Ершов В.А.* Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов – М.: Изд. МГТУ им. Н.Э. Баумана, 2003. – 432 с.

47. Закон України «Про Концепцію Національної програми інформатизації» (Відомості Верховної Ради (ВВР), 1998, N 27-28, ст.181) (Із змінами, внесеними згідно із Законом N 5463-VI (5463-17) від 16.10.2012, ВВР, 2014, N 4, ст.61). – http://zakon1.rada.gov.ua/laws/show/74/98-%D0%B2%D1%80.

48. *Зайченко Ю.П.* Компьютерные сети / Ю.П. Зайченко. – К.: Слово, 2003. – 256 с.

49. *Зубко Р.А*. Алгоритми стиснення зображень в системах цифрової обробки даних / Р.А. Зубко // Восточно-Европейский журнал передовых технологий. – 2013. – №1/2(61). – С. 40-44.

50. *Иванов В.Г.* Прогрессивные информационные технологии сжатия изображений / Иванов В. Г., Ломоносов Ю. В., Любарский М. Г. // Проблеми інформатики та комп'ютерної техніки (ПІКТ – 2014) : пр. ІІІ-ї Міжнар. наук.-практ. конф., 27–30 трав. 2014 р. – Чернівці : Родовід, 2014. – С. 172–174.

51. *Ивахненко А.Г.* Долгосрочное прогнозирование и управление сложными системами / А.Г. Ивахненко – Киев: «Техніка». – 1975. – 312 с.

52. *Игнатенко Е.Г., Бессараб В.И.* Алгоритм адаптивного мониторинга загрузки кластерных web-серверов / Е.Г. Игнатенко, В.И. Бессараб // Збірник тез. Нові технології в телекомунікаціях. VI міжнародний науково-технічний симпозіум. – Карпати, Вишків: ДУІКТ. – 2011. – С.34.

53. *Карпенко С.В.* Метод компактного представлення зображень в телекомунікаційних системах на основі тривимірного поліадичного кодування: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.12.02 –телекомунікаційні системи та мережі / С.В. Карпенко. – Харків: Харківський національний університет радіоелектроніки, 2009. – 22 с.

54. *Касимов Р.Р.* Вдосконалення алгоритмів QoS маршрутизації в мережах з технологією IP/MPLS на основі прогнозу трафіка: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.12.02 – телекомунікаційні системи та мережі / Р.Р. Касимов. – Київ: Київський державний університет інформаційно-комунікаційних технологій, 2011. – 24 с.

55. *Кириченко Л.*О. Влияние методов маршрутизации на QOS в мультисервисных сетях при самоподобной нагрузке / Л.О. Кириченко, Т.А. Радивилова, Э. Кайали // Восточно-Европейский журнал передовых технологий. Информационные технологии. 1/2 (49) 2011. – С. 15-18

56. *Квєтний Р.Н.* Методи та засоби передавання інформації у проблемно-орієнтованих розподілених комп'ютерних системах: монографія / Р.Н. Квєтний, А.Я. Кулик – Вінниця: ВНТУ, 2010. – 362 с.

57. *Коваленко А.А.* Методи та засоби підвищення оперативності передачі даних у мультисервісних мережах: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.13.05 «компьютерні системи та компоненти» / А.А. Коваленко. – Харків: Харківський національний університет радіоелектроніки, 2008. – 20 с.

58. *Комарова Л.О.* Алгоритми управління потоками даних у телекомунікаційній мережі реального часу / Л.О. Комарова // Телекомунікаційні та інформаційні технології. – 2014. – №2 – С. 13-18.

59. *Кормен Т.* Алгоритмы: построение и анализ / Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. – М.:"Вильямс", 2005. – 1296 с.

60. *Конахович Г.Ф.* Сети передачи пакетных данных / Г.Ф. Конахович, В.М.Чуприн. – К.:МК-Пресс, 2006. – 272 с.

61. *Кононенко В.О.* Колебательные системы с ограниченным возбуждением / В.О. Кононенко – М.: Наука, 1964. – 232 с.

62. *Королев А.В.* Адаптивная маршрутизация в корпоративных сетях / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2003. – 224 с.

63. *Королев А.В.* Управление сетевыми ресурсами / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2004. – 272 с.

64. *Красильников Н.Н.* Теория передачи и восприятия зображений / Н.Н. Красильников – М.: Радио и Связь, 1986. – 248 с.

65. *Кузнецов О.О.* Методи обробки сигналів даних та зображень. Навчальний посібник / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов – Х.: НТУ «ХПІ», 2011. – 301 с.

66. *Кучерявый Е.А.* Управление трафиком и качество обслуживания в сети Интернет / Е.А. Кучерявый – М.: Наука и Техника, 2004. – 336 с.

67. *Кучук Г.А.* Метод дослідження фрактального мережевого трафіка / Г.А. Кучук // Системи обробки інформації. – Х.: ХВУ, 2005. – Вип. 5 (45). – С. 74-84.

68. *Кучук Г.А.* Метод агрегування фрактального трафіка / Г.А. Кучук, А.А. Можаев, О.В. Воробьов // Радіоелектронні та комп'ютерні системи. – 2006. – №6(18). – С. 181-188.

69. *Лемешко А.В.* Потоковые модели многоадресной и широковещательной маршрутизации в телекоммуникационных сетях [Электронный ресурс] / А.В. Лемешко, К.М. Арус // Проблеми телекомунікацій. – 2013. – No 1 (10), с. 38 - 45. – Режим доступа: http://pt.journal.kh.ua/2013/1/1/131_lemeshko_multicast.pdf.

70. *Лемешко А.В.* Особенности математического описания процессов многоадресной маршрутизации потоковыми моделями / А.В. Лемешко, Арус Кинан // Труды Северо-кавказского филиала Московского технического университета связи и информатики, часть 1. – Ростов-на-Дону: СКФ МТУСИ, 2014. - С. 94-98.

71. *Лемешко О.В.* Результати порівняльного аналізу потокових моделей маршрутизації в телекомунікаційних мережах / О.В. Лемешко, О.А. Дробот, Д.В. Симоненко // Збірник наукових праць Харківського університету Повітряних Сил. Вип. 1(13), 2007. – С. 66-69.

72. *Лемешко А.В.* Тензорная модель многопутевой маршрутизации с гарантиями качества обслуживания одновременно по множеству разнородных показателей [Электронный ресурс] / А.В. Лемешко,

О.Ю. Евсеева // Проблеми телекомунікацій. – 2012. – No 4 (9). – С. 16 - 31. – Режим доступа: http://pt.journal.kh.ua/2012/4/1/124_lemeshko_tensor.pdf.

73. *Лемешко А.В.* Модель отказоустойчивой маршрутизации многоадресных и широковещательных потоков в MPLS-сети / Лемешко А.В., Арус К.М. // Системи обробки інформації. – 2013. – №9 (116). – С. 160 - 163.

74. *Малла С.* Вейвлеты в обработке сигналов: пер. с англ./ С. Малла – М.: Мир, 2005. — 671 с.

75. *Марчук Г.И.* Методы вычислительной математики / Г.И. Марчук – М.: Наука, 1989. – 608 с.

76. *Муранов О.С.* Експериментальні дослідження механізмів прогнозування пульсацій пакетного трафіку / Муранов О.С. // Защита информации : Сб. научн. трудов Национального авиационного университета. – К. : НАУ, 2008. – Специальный выпуск. – С.137-142.

77. *Муранов О.С.* Підвищення якості технології адаптивного управління трафіком пакетних телекомунікаційних мереж: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.13.06 – інформаційні технології / О.С. Муранов. – Київ: Національний авіаційний університет, 2010. – 29 с.

78. *Новиков Ю.В.* Локальные сети: архитектура, алгоритмы, проектирование / Ю.В. Новиков, С.В. Кондратенко – М.: ЭКОМ, 2000. – 312 с.

79. *Остерлох Х.* Маршрутизация в IP-сетях. Принципы, протоколы, настройка / Х. Остерлох – С.Пб.: BHV-С.Пб., 2002. – 512 с.

80. *Олифер В.Г.* Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 2-е изд./ В.Г. Олифер, Н.А. Олифер – СПб: Питер, 2005. – 864 с.

81. *Пантелеев А.В.* Методы оптимизации в примерах и задачах / А.В. Пантелеев, Т.А. Летова. – М.: Высшая школа, 2002. – 544 с.

82. *Приоров А.Л.* Обработка и передача мультимедийной информации: учебное пособие / А.Л. Приоров, В.В. Хрящев – Яросл. гос. ун-т им. П.Г. Демидова. – Ярославль: ЯрГУ, 2010. – 188 с.

83. *Прокис Дж.* Цифровая связь. Пер. с англ./Под ред. Д. Д. Кловского. М.: Радио и связь, 2000. – 800 с.

84. *Руккас К.М.* Потоковая модель сети MPLS. / К.М. Руккас, К.А. Овчинников // Системи обробки інформації. Випуск 3(101) Том 2. – Х.: ХУПС. – 2012. – С. 211-214

85. *Рябий М.О.* Технологія підвищення рівня стиснення цифрових зображень на основі сплайн-моделі: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.13.06 – інформаційні технології / М.О. Рябий // Київ, Національний авіаційний університет, 2013. – 20 с.

86. *Семенов С.Г.* Модели и методы управления ресурсами в информационно-телекоммуникационных системах / С.Г. Семенов, А.А. Смирнов, Е.В. Мелешко – Х.: Вид-во Віровець А.П. «Апостроф», – 2012. – 212 с.

87. *Сергиенко А.Б.* Цифровая обработка сигналов / А.Б. Сергиенко – СПб.: Питер, 2002. – 608 с.

88. *Свами М.Н.* Графы, сети и алгоритмы: пер. с англ. / М.Н. Свами, К. Тхуласираман; под ред. В.А. Горбатова. – М.: Мир, 1984. – 454 с.

89. *Семенов А.Б.* Структурированные кабельные системы / А.Б. Семенов, С.К. Стрижаков, И.Р. Сунчелей – 3-е изд. – М.: “Компьютер-Пресс”, 2001. – 608с.

90. *Семенов С.Г.* Анализ методов прогнозирования в телекоммуникационных сетях автоматизированных систем управления / С.Г. Семенов // Збірник наукових праць «Системи управління, навігації та зв’язку», - К.:ЦНДІ навігації і управління, – 2008.-Вип. 2(6). – С.134-137

91. *Семенов С.Г.* Математическая модель процесса доставки информационных пакетов в компьютерной сети системы критического применения / С.Г. Семенов, И.В. Ильина // Науково-технічний журнал «Радіоелектронні і комп'ютерні системи» Х.:ХАІ, – 2008.-Вип. 1(28) – С.162-165

92. *Семенов С.Г.* Оптимизация трафика на основе сбалансированной загрузки информационно-телекоммуникационной сети // Системи обробки інформації. – Х.: ХВУ, 2004. – № 8(36). – С.206-210

93. *Семенов Ю.А.* Сети Інтернет. Архитектура и протоколы / Ю.А. Семенов. – М.: Блик плюс, 1998. – 424 с.

94. *Смирнов А.А.* Разработка методики оценки среднего времени обслуживания информационных пакетов в телекоммуникационной сети / А.А. Смирнов, В.В. Босько, Е.В. Мелешко // Системи управління, навігації та зв'язку. – Київ: ДП «Центральний науково-дослідний інститут навігації і управління», 2009. – Вип. 2(10). – С.162-165.

95. *Смирнов А.А.* Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А. Смирнов, В.В. Босько, Е.В. Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

96. *Смірнов О.А*. Методи та засоби обробки сигналів і даних в інформаційних системах / О.А. Смірнов, Є.В. Мелешко, С.Г. Семенов – Кіровоград: Вид. КНТУ, 2012. – 252 с.

97. *Смірнов О.А*. Аналіз процесів стиснення та відновлення зображень на основі цифрових методів / О.А. Смірнов, О.П. Доренський, О.М. Дрєєв // Наку і техніка повітряних сил збройних сил України. Науково-технічний журнал 3(12) 2013. ISSN 2223-456X. С. 122-127.

98. *Столингс В.* Современные компьютерные сети / В. Столингс – СПб.: Питер, 2003.- 783 с.

99. *Сэломон Д.* Сжатие данных, изображений и звука / Д. Сэломон. – М.: Техносфера, 2004. – 368 с.

100. *Таненбаум Э.* Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. — 848 с.

101. *Турчак Л.И.* Основы численных методов: Учеб. Пособие / Л.И. Турчак – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 320 с.

102. *Уолрэнд Дж.* Телекоммуникационные и компьютерные сети / Дж. Уолрэнд. – М.: Постмаркет, 2001. – 480 с.

103. *Филер З.Е.* Обобщение формулы Тейлора, Ньютона и Лагранжа и их применение к решению дифференциальных и разностных уравнений / З.Е. Филер // Приближенные методы исследования дифференциальных и разностных уравнений и их приложения. – Куйбышев: Куйбышевский государственный университет, 1982. – С. 147-157.

104. Фрактальний аналіз процесов, структур и сигналов. Коллективная монографія / Под. ред. Р.Э. Пащенко. – Харьков: ХООО «НЭО «ЭкоПерспектива», 2006. – 348 с.

105. *Хелеби С.* Принципы маршрутизации в Internet. 2-е издание / С. Хелеби, Д. Мак-Ферсон – Пер. с англ. – М.: Издательский дом ”Вильямс”, 2001. – 448 с.

106. *Холодкова А.В.* Динамические модели распределённого управления сетевыми ресурсами ТКС / А.В. Холодкова // Проблеми і перспективи розвитку ІТ-індустрії 25-26 квітня 2013 р. Системи обробки інформації. – Випуск 3 (110). Том 2. – Харків: ХУПС. – 2013. С. – 199.

107. *Чобану М.К.* Синтез двумерных ортогональных банков фильтров и вейвлетов. / М.К. Чобану, О.В. Большакова, А.Г. Плахов // 2003, Труды Всероссийской научной удаленной конференции по информационным и телекоммуникационным технологиям, 2003.

108. *Шелухин О.И.* Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

109. *Шматков С.И.* Метод оценки времени доставки сообщения в компьютерных сетях / С.И. Шматков, // Збірник наукових праць. Системи обробки інформації. Випуск 6(113) – м. Харків, ХУПС – 2013. – С. 258-261.

110. *Юдін О.К.* Розробка сучасних стандартів стиснення відеоданих / О.К. Юдін, К.О. Курінь // Захист інформації. – 2010. – №3. – С. 74-81

111. *Юдін О.К.* Сучасні технології організації автоматизованої системи управління транспорту / О.К. Юдін, О.Л. Яковенко // Збірник наукових праць Інституту проблем моделювання в енергетиці ім.Г.Є.Пухова НАН України. – К.: ІПМЕ ім. Г.Є.Пухова НАН України, 2009. – Вип. 50. – С. 73-80.

112. *Яковенко О.В.* Методологічні основи комплексного представлення зображень з контрольованою погрішністю /О.В. Яковенко // Системи озброєння і військова техніка, 2008, № 2(14) – С. 128-131.

113. *Amir Said.* A New Fast and E cient Image Codec Based on Set Partitioning in Hierarchical Trees / Amir Said, William A. Pearlman // IEEE Transactions on Circuits and Systems for Video Technology , Vol. 6, June 1996.

114. *Clifford Lui.* A study of the JPEG-2000 image compression standard / Lui Clifford // Queen's University Kingston, Ontario, Canada May 2001. 97 p.

115. *Dorensky O.P.* Reseach results on the dependence of infocomm system compressed imagwes delivery effictiency on compression ratio / O.P. Dorensky // «Компьютерное моделирование в наукоемких технологиях (КМНТ-2014)» Харьков, 28-31 мая 2014 года – С. 118-120

116. *Dreyev A.N.* Block Mathematical Coding Method of Images Compressed by a SPIHT Algorithm / A.N. Dreyev, A.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 03, Issue 5. USA,

Indiana, Riley: Science and Engineering Publishing Company. – May 2013 – P.34-39. ISSN: 2250-3005

117. Introduction to data compression/ Khalid Sayood.—3rd ed – Morgan Kaufmann is an imprint of Elsevier, 2006. – 680 p.

118. *Kassim A.A., Lee W.S.* Embedded color image coding using SPIHT with partially linked spatial orientation trees. / A.A. Kassim, W.S. Lee // IEEE Transactions on Circuits and Systems for Video Technology 2003;13:203–6.

119. Kekade R.D. Progressive coding for image compression / R.D. Kekade // International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 7, July – 2013. C. 1642-1645.

120. *Vasuki A*. Progressive image compression using contourlet transform / A. Vasuki, , P.T. Vanathi // International Journal of Recent Trends in Engineering, Vol 2, No. 5, November 2009. – p. 193-197.

121. *Rajpoot N.* Progressive image coding using augmented zerotrees of wavelet coefficients / Nasir Rajpoot, Roland Wilson // Department of Computer Science, University of Warwick, Coventry (September 18, 1998).

122. *Sprljan N.* Modified SPIHT algorithm for wavelet packet image coding / Nikola Sprljan, Sonja Grgic, Mislav Grgic // Real-Time Imaging 11 – 2005. p. 378–388.

123. *Stefan Soucek, Member, IEEE, and Thilo Sauter, Member.* Quality of Service Concerns in IP-Based Control Systems // IEEE Transactions on industrial electronics. – 2004. – №51(6), P. 1249-1258.

124. *Taubman D.* JPEG2000 image compression: fundamentals, standards and practice / D. Taubman, M.W. Marcellin // Dordrecht: Kluwer Academic Publishers; 2002.

125. *Tchobanou M.* Optimization and development of algorithm SPIHT / M. Tchobanou, A. Chernikov, A. Plakhov // Труды Первой международной

научно-технической школы-семинары «Современные проблемы оптимизации в инженерных приложениях» (IWOPE-2005). т. 1, Ярославль: 2005, с. 45-49.

ПРИЛОЖЕНИЕ А

Описание основных алгоритмов сжатия фотографических изображений

1. JPEG кодирование

JPEG кодирование сегодня наиболее применимо не только при кодировании неподвижного изображения, но и при кодировании видео, как отдельное кодирование каждого кадра. Стандарт JPEG принят в 1991 году по спецификации ISO. В то время это был алгоритм, который оптимально сочетал скорости кодирования-декодирования изображения, и также поддерживал сжатие данных с разной степенью потерей информации в 2-200 раз. Основными шагами алгоритма кодирования JPEG является:

а) Цветное 24-битное изображение из формата представления по компонентам RGB переводиться в формат YCbCr – яркость/синий/красный. Благодаря пониженной чувствительности человеческого глаза к мелким деталям изменения цвета и повышенной чувствительности к изменению яркости, кодек использует возможность с пониженным качеством записывать информацию про цветность рисунка, а информацию про изменения яркости предавать с максимальным качеством. Такой подход позволил значительно уменьшить размер кодированного изображения без видимых изменений качества.

2) Синяя и красная составляющие объединяются в одно изображения чередованием строк. Тем самым, на этом этапе уже достигается сжатие в 2/3 раза, потому что из трёх компонентов изображения остаётся только два – яркости и объединённого цвета. Каждый из компонентов считается отдельным серым изображением, которые дальше обрабатываются отдельно.

3) Далее изображение разбивается на кластеры 8x8 пикселей, если размер входного изображения не кратное 8, то изображение дополняется пустыми пикселями до ближайшего кратного. Такой кластер тут называется единицей данных.

4) Для каждой единицы данных используют дискретное косинус-преобразование DCT. При этом значения яркости пикселей в поле 8x8 трактуются как единая одномерная последовательность чисел, которую можно получить, если обойти поле 8x8 зигзагом. Для выполнения преобразования используется целочисленная арифметика.

5) Каждое полученное число делиться на число квантования, которые берутся из спецификации стандарта. Это позволяет тонко регулировать количество потерянной энергии (информации) сигнала на разных частотных интервалах.

6) Все 64 квантованных частотных коэффициента кодируются при помощи комбинации RLE та кодами Хафмана. Таблица кодов Хафмана задана стандартом и неизменна для формата JPEG.

Таблица А.1

Таблица коэффициентов квантования

канал яркости							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

канал цветности							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Декодирование проводят в обратном порядке, поэтому алгоритм называется симметричным.

К основным недостаткам алгоритма можно отнести сильно выраженную блочность и «рябь» возле резких переходов вызванную явлением Гиббса. Формат рекомендовано к не полиграфическим фотоизображениям с отсутствием надписей. При максимальном качестве изображения потери информации присутствуют.

Модификации алгоритма к применению прогрессивной передачи изображения не получили распространения, по причине значительного увеличения полного размера файлов.

2. JPEG2000 алгоритм

JPEG2000 алгоритм сжатия изображений основан на использовании вейвлет-преобразования, имеет возможность кодирования-декодирования изображений с потерей и без потери информации. Этот алгоритм постоен как продолжение классического JPEG алгоритма, но тут не используется разделение изображения на блоки и дискретное косинус-преобразование заменено на дискретное вейвлет преобразование. При сжатии полученных коэффициентов используют более эффективных алгоритм арифметического кодирования. Последовательность действий при проведении кодирования изображения:

1) Смещение по яркости. Для всех цветовых компонентов выравнивается математическое ожидание (средняя яркость) к нулю, это позволяет сместить гистограммы в одну область и использовать общий словарь при арифметическом кодировании. При восстановлении изображения проводят обратное смещение яркости по каналам.

2) Переход из формата RGB к формату YUV. Здесь использовано целочисленный вариант перерасчета компонент изображения. Это позволяет делать преобразование без потери информации:

$$\begin{vmatrix} Y \\ U \\ V \end{vmatrix} = \begin{pmatrix} \left\lfloor \frac{R + 2\mathrm{G} + B}{4} \right\rfloor \\ R - G \\ B - G \end{pmatrix}. \qquad \text{(A.1)}$$

Тут квадратными скобками обозначена функция отсекания дробной части. Обратное преобразование при восстановлении изображения без потери информации имеет вид:

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{pmatrix} U + G \\ Y - \left\lfloor \frac{U + V}{4} \right\rfloor \\ V + G \end{pmatrix}. \qquad \text{(A.2)}$$

3) Дальше выполняют дискретное вейвлет-преобразование (DWT) с вейвлетом для сжатия с потерями, где берутся действительные коэффициенты, или с целочисленным вейвлетом для сжатия без потерь информации. Преобразование выполняют для всего расширенного квадрата до стороны степени двойки.

4) Для сжатия с потерями также используют квантование, однако значения коэффициентов квантования не стандартизированы, а записываются в файл сжатого изображения.

5) Полученный массив данных сжимается при помощи арифметического кодирования.

3. SPIHT алгоритм

SPIHT алгоритм сжатия изображений тоже использует вейвлет-преобразование, и его эффективность по малости размера выходного файла зависит от выбранного вейвлета, при помощи которого преобразуют изображение. Алгоритм довольно сложный и ресурсоёмкий, что не позволило, на ряду с патентными ограничениями, получить ему широкого распространения.

Сегодня, благодаря развитию мощностей вычислительной техники, ресурсоёмкость алгоритма перестала быть ключевым фактором, и по истечению на сегодня патентных ограничений, SPIHT становиться популярным среди исследователей. SPIHT кодирование имеет ряд преимуществ которые позволят получить выгоду от затраченных средств на его внедрение: поддержка прогрессивной передачи изображения, возможность упаковки изображения с утратой информации и без утраты, высокий коэффициент сжатия сравнимый с форматом JPEG2000, значительно менее ресурсоёмкий алгоритм восстановления изображения, благодаря прогрессивности имеется возможность

управлять степенью утраты информации на уровне отдельного пользователя. Последовательность действий при упаковке однотонного изображения:

1) Изображение дополняется до квадрата со стороной степени двойки.

2) Производится вейвлет преобразование. По причине не установленных стандартов на формат SPIHT, применяют различные вейвлеты, которые выбирают реализаторы конкретного программного обеспечения. Для обеспечения возможности кодирования изображения для передачи его без потери информации, используют дискретные вейвлет-преобразования, а для получения максимального коэффициента сжатия, используют вейвлеты с действительными коэффициентами с максимальным количеством нулевых моментов.

3) Полученные вейвлет-коэффициенты при надобности делят на коэффициенты квантования, и организуют в древовидную структуру, где каждому вейвлет-коэффициенту низкой частоты соответствуют четыре коэффициента более высокой частоты (см. рис. А.1). Каждое подмножество получает оценку как максимальное значение модуля его коэффициентов.

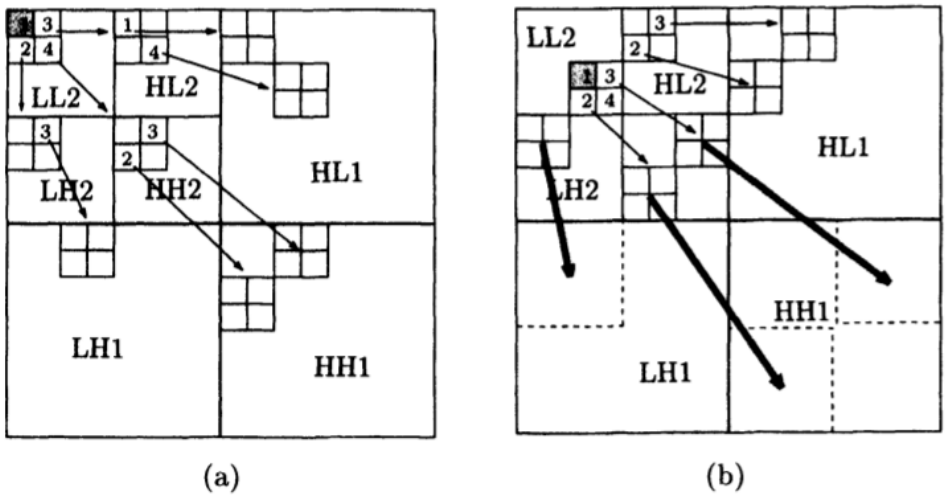


Рис. А.1. Пространственно ориентированные деревья для SPIHT

4) Создается *список несущественных множеств*, который содержит на начальном этапе три начальных множества «2», «3», «4» рис. А.4 а). *Список несущественных пикселей*, который содержит пиксель отмеченный на рис. А.4 а) черным квадратом. *Список существенных пикселей*, который на начале роботы алгоритма берут пустым.

5) Значения коэффициентов передаются по битовым плоскостям, начиная со старшего бита. Проверяется список несущественных множеств, где каждому несущественному множеству отвечает переданный 0. В случае попадания на существенное множество, оно извлекается из списка, корневой коэффициент переходит в конец списка несущественных пикселей, а четыре подмножества переходят в конец списка несущественных множеств. Для существенных пикселей, по очереди списка, передаются значения соответствующего бита. Далее для списка несущественных пикселей передаётся 0, если пиксель не

имеет в текущем разряде 1, иначе передаётся 1 с меткой знака числа и такой пиксель добавляется в конец списка существенных пикселей.

6) Пункт 5 повторяется до конца передачи всех битовых плоскостей. Высокий коэффициент сжатия достигается за счет передачи нулевых более старших бит зразу для всего множества. Например, для рисунка 1024x1024 основное подмножество коэффициента «2» с рис. А.1 а), будет содержать более 80000 коэффициентов. Из этого следует, что меньшее количество существенных высокочастотных коэффициентов вейвлет-образе изображения позволят значительно уменьшить размер выходного файла. Вместе с этим, обрыв передачи информации оставит приближенные значения наиболее существенных коэффициентов, и это позволит восстановить изображение полностью, но с зональной потерей контрастности переходов.

7) Полученные данные дополнительно могут быть сжаты при помощи, например, арифметического кодирования.

ПРИЛОЖЕНИЕ Б

Статистика результатов кодирования тестовых изображений кодеками jpeg2000, SPIHT, усовершенствованным SPIHT

Случайным выбором создана база изображений фотографического характера, 3D арт и сгенерированных изображений фракталов.

Время кодирования 1322 файла bmp → jpeg2000 (без потерь): близко к 17 мин.

Время кодирования 1322 файла bmp → SPIHT (progressive без потерь): близко к 20 мин.

Полученные результаты представлены в следующих фрагментах таблиц.

Таблица Б.1

Статистические характеристики исходных изображений

№	Среднее значение	Дисперсия	Байтовая энтропия	Битовая энтропия	Межстрочная горизонтальная корреляция
1	34,0546	2604,76	543369	628336	0,994181
2	68,5307	2158,34	711116	777115	0,983169
3	151,592	1958,27	521276	786324	0,973796
4	98,509	2429,48	732110	784889	0,884222
5	113,243	5191,63	768960	785496	0,959122
6	72,1596	2336,95	713959	778899	0,971206
7	43,2273	2469,98	650184	733908	0,989065
8	117,747	4570,11	776233	786113	0,979154
9	67,6096	2905,66	729052	770553	0,968583
10	109,003	5744,57	772555	783807	0,973529
11	183,998	3972,26	716082	769720	0,988864
12	194,38	1737,95	702180	770551	0,948144
13	169,18	2524,89	735500	782310	0,969254
14	161,18	2228,68	734597	784531	0,955617
15	76,6606	3374,69	743016	774580	0,942698
16	97,481	7352,48	745812	774139	0,964289
17	60,6904	2890,68	696106	767988	0,997122
18	91,5767	4958,98	759920	779642	0,954397
19	144,251	1998,56	730374	786249	0,965931
20	53,6721	2364,15	689709	765120	0,984909
21	92,2531	4642,96	755330	776725	0,970155
22	104,606	4894,62	767322	785895	0,976138
23	130,941	4357,25	766030	786426	0,981398
24	40,953	1829,74	665944	735875	0,991896
25	92,3805	5553,08	762169	776573	0,968659
26	162,742	3102,84	746858	782959	0,976181
27	94,0143	7282,53	741376	783642	0,977006
28	139,118	4267,51	769984	785291	0,95725
...	...	...	...	...	...

Таблица Б.2

Оценка энтропийного блочного побитового сжатия исходных изображений

№	Размер блока в байтах												Минимальное значение	Оптимальный размер блока
	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536		
1	545820	546267	569781	600212	615003	622524	624158	624466	624519	624593	624888	625639	545820	32
2	784839	774008	769607	768602	769694	770811	770952	770922	770899	770909	772065	772235	768602	256
3	763397	756405	758734	767968	772482	776753	777803	778370	779412	780367	781593	781919	756405	64
4	801444	791410	787144	785511	785076	785058	784863	784729	784652	784629	784633	784748	784629	16384
5	792137	782416	779239	779237	781629	783335	783568	783534	783488	783467	783484	783541	779237	256
6	794957	784010	779145	777557	777577	777959	777878	777765	777697	777665	777672	777772	777557	256
7	714829	703600	699616	698656	701852	711190	713189	713599	713670	713692	713805	714126	698656	256
8	789979	780163	776542	775937	777588	781128	781845	781946	781970	782048	782267	782524	775937	256
9	770023	760732	758500	761442	764085	766366	766736	766757	766741	766728	766789	766889	758500	128
10	777861	768522	766809	767305	767675	772263	773213	773365	773394	773458	773590	774577	766809	128
11	735741	728299	731375	740862	746493	756239	758379	758851	758939	759002	759149	759497	728299	64
12	785739	775140	770818	769245	768685	769382	769363	769300	769242	769262	769337	769533	768685	512
13	795238	784643	780181	778467	778102	778335	778198	778079	778018	778016	778073	778201	778016	16384
14	801835	791332	786786	785024	784786	784453	784177	784029	783958	783939	783972	784053	783939	16384
15	785279	774522	769787	767917	767714	769927	770280	770283	770248	770229	770229	770277	767714	512
16	747260	740631	741607	744512	748944	760893	763671	764295	764420	764518	764842	765277	740631	64
17	780694	769779	766178	764440	764476	767125	767586	767621	767580	767554	767544	767570	764440	256
18	785754	776562	773758	774765	776178	777647	777826	777786	777754	777769	777850	778175	773758	128
19	800959	789860	785361	784004	783730	784253	784191	784118	784079	784129	784460	784836	783730	512
20	773555	762678	757619	756956	757409	758278	758318	758241	758179	758157	758175	758367	756956	256
21	775289	764966	761315	761128	760679	762408	762659	762635	762596	762606	762686	762875	760679	512
22	788457	778526	775724	776191	776805	778062	778203	778150	778103	778091	778142	778427	775724	128
23	794427	784037	779685	777938	777289	777940	777930	777848	777809	777824	777902	778133	777289	512
24	727923	717758	716260	717012	718119	730680	733523	734145	734255	734279	734341	734474	716260	128
25	767474	758016	755946	756227	757464	761485	762315	762455	762455	762494	762651	763616	755946	128
26	798890	787554	782391	780701	780682	780680	780495	780360	780282	780244	780236	780251	780236	32768
27	730955	722386	722345	725698	738016	745690	747443	747821	747936	748185	748416	748635	722345	128
28	792750	782084	777482	775849	776187	776405	776279	776179	776145	776162	776338	776508	775849	256
29	765565	757568	756179	758839	760616	766000	767107	767359	767409	767534	767753	767849	756179	128
30	782756	772425	768469	766944	766726	767269	767253	767179	767164	767181	767286	767819	766726	512
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Таблица Б.3

Оценка энтропийного блочного побайтного сжатия исходных изображений

№	Размер блока в байтах											Минимальное значение	Оптимальный размер блока
	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	1048576		
1	702344	617268	573049	551084	540528	536044	534982	535938	539563	541693	543646	534982	65536
2	810555	727337	683043	661262	651166	650484	655924	666242	676333	696556	711422	650484	32768
3	481640	410520	372610	358302	360015	371331	389386	412630	440288	484026	521575	358302	8192
4	898955	812172	768327	746616	736380	731978	731157	731723	732662	732537	732415	731157	65536
5	919037	837429	794160	772556	762393	758507	758232	761506	763132	767603	769274	758232	65536
6	869434	782729	738511	716494	705720	700962	699998	702105	707048	707149	714311	699998	65536
7	755387	675637	632192	610328	599790	595659	596001	601687	612620	623913	650496	595659	32768
8	886115	811935	770744	751073	743628	743234	746199	755779	765977	772957	776559	743234	32768
9	848499	769647	726438	704961	695077	691835	693512	698490	703861	714587	729356	691835	32768
10	908516	824919	781516	759875	749671	745352	746345	749499	756509	758531	772856	745352	32768
11	813598	741963	699437	678454	669471	667820	671472	677234	692625	699316	716397	667820	32768
12	851408	766628	722985	701424	692176	688876	690983	694847	698698	699889	702540	688876	32768
13	857251	771688	727361	705686	696156	693232	695002	701407	712003	726739	735823	693232	32768
14	889449	802701	758495	736775	727251	723723	724777	728563	731679	732561	734952	723723	32768
15	885618	802617	759130	737203	726396	721214	719200	719457	723660	723817	743332	719200	65536

16	869111	792437	750882	730621	721807	719850	721766	727366	733378	742794	746103	719850	32768
17	840929	758349	714176	691962	681189	676637	675589	677097	681864	683881	696444	675589	65536
18	904519	821239	777902	756657	746815	743166	743222	745258	751655	751904	760228	743166	32768
19	787937	702987	658824	637948	630575	633087	643491	665189	694867	716242	730701	630575	16384
20	806102	718831	674156	651946	641250	637041	637813	644765	658122	678557	690061	637041	32768
21	835984	756445	713300	692339	683527	681807	684912	689184	703679	717189	755644	681807	32768
22	893128	809270	765584	744020	733855	729748	729348	732929	737583	742892	767656	729348	65536
23	833451	751451	707506	686438	677970	677063	680814	694769	713989	746105	766346	677063	32768
24	805143	727151	683675	662116	652083	648447	648646	652066	660922	661685	666239	648447	32768
25	874550	791798	748377	726869	716991	713918	716288	721457	733811	744791	762469	713918	32768
26	881095	795604	751583	729657	719159	714576	713951	716518	721133	723347	747210	713951	65536
27	839916	757574	714176	692723	682979	679573	680498	689436	698281	707216	741681	679573	32768
28	871851	787209	743749	722792	713121	711237	712532	720506	728697	729939	770330	711237	32768
29	872133	788467	745418	724268	714776	711341	710521	713932	719813	725092	757305	710521	65536
...	...	...	...	...	...	...	...	...	...	...	...	...	…

Таблица Б.4

Оценка энтропийного блочного побайтного сжатия исходных JPEG2000 изображений

№	Размер блока в байтах					Минимальное значение	Размер оптимального блока
	65536	131072	262144	524288	1048576		
1	225302	224838	224606	224606	224606	224606	262144
2	348042	347341	347110	346875	346875	346875	524288
3	89465	89231	89231	89231	89231	89231	131072
4	576532	575594	575129	574895	574659	574659	1048576
5	493018	492087	491622	491386	491386	491386	524288
6	452147	451449	450987	450754	450754	450754	524288
7	316851	316389	316156	315924	315924	315924	524288
8	424582	423884	423413	423180	423180	423180	524288
9	378175	377480	377248	377016	377016	377016	524288
10	516543	515609	515138	514908	514908	514908	524288
11	245604	245137	244903	244903	244903	244903	262144
12	358636	357934	357700	357466	357466	357466	524288
13	350119	349418	349184	348952	348952	348952	524288
14	402865	402168	401704	401467	401467	401467	524288
15	583620	582691	582227	581998	581765	581765	1048576
16	393515	392817	392582	392349	392349	392349	524288
17	274558	274091	273862	273631	273631	273631	524288
18	489723	488796	488332	488102	488102	488102	524288
19	370284	369593	369361	369131	369131	369131	524288
20	391680	390978	390748	390515	390515	390515	524288
21	406532	405829	405365	405132	405132	405132	524288
22	474221	473285	472820	472591	472591	472591	524288
23	404324	403629	403170	402935	402935	402935	524288
24	249332	248868	248635	248635	248635	248635	262144
25	493382	492458	491996	491762	491762	491762	524288
26	480495	479563	479098	478863	478863	478863	524288
27	470766	469839	469371	469143	469143	469143	524288

№	Размер блока в байтах					Минимальное значение	Размер оптимального блока
	65536	131072	262144	524288	1048576		
28	535146	534214	533747	533514	533281	533281	1048576
29	537871	536946	536482	536251	536015	536015	1048576
30	449584	448888	448426	448195	448195	448195	524288
...	...	...	...	...	...	...	...

Таблица Б.5

Оценка энтропийного блочного побитного сжатия исходных JPEG2000 изображений

№	Размер блока в байтах					Минимальное значение	Размер оптимального блока
	65536	131072	262144	524288	1048576		
1	224380	224379	224378	224378	224378	224378	262144
2	346648	346645	346644	346643	346643	346643	524288
3	89005,6	89005	89005	89005	89005	89005	131072
4	574433	574430	574428	574427	574426	574426	1048576
5	491160	491156	491154	491153	491153	491153	524288
6	450526	450523	450522	450521	450521	450521	524288
7	315698	315696	315695	315694	315694	315694	524288
8	422952	422950	422948	422947	422947	422947	524288
9	376786	376784	376783	376782	376782	376782	524288
10	514683	514679	514677	514677	514677	514677	262144
11	244680	244678	244677	244677	244677	244677	262144
12	357239	357237	357236	357235	357235	357235	524288
13	348724	348722	348721	348720	348720	348720	524288
14	401240	401238	401236	401235	401235	401235	524288
15	581542	581539	581537	581536	581535	581535	1048576
16	392123	392121	392120	392119	392119	392119	524288
17	273404	273402	273401	273401	273401	273401	262144
18	487875	487872	487870	487869	487869	487869	524288
19	368906	368904	368903	368902	368902	368902	524288
20	390289	390287	390286	390285	390285	390285	524288
21	404907	404905	404903	404902	404902	404902	524288
22	472368	472364	472362	472361	472361	472361	524288
23	402707	402704	402702	402702	402702	402702	262144
24	248406	248404	248403	248403	248403	248403	262144
25	491536	491533	491531	491530	491530	491530	524288
26	478638	478635	478633	478632	478632	478632	524288
27	468917	468914	468912	468911	468911	468911	524288
28	533054	533050	533048	533047	533046	533046	1048576
29	535788	535787	535785	535784	535783	535783	1048576
30	447971	447968	447966	447965	447965	447965	524288
31	312514	312512	312511	312510	312510	312510	524288
32	330880	330878	330877	330876	330876	330876	524288
33	351158	351155	351155	351154	351154	351154	524288
...	...	...	...	...	...	...	...

ПРИЛОЖЕНИЕ В

Данные для построения прогноза и прогноз загруженности сервера телекоммуникационной сети

Таблица В.1

Пример долгосрочного прогнозирования трафика сервера сети

№	Мб/час			№	Мб/час		
	Реальные данные	Прогноз	База		Реальные данные	Прогноз	База
1			272	38			1128
2			300	39			1290
3			282	40			1465
4			308	41			1357
5			340	42			1357
6			480	43			1370
7			698	44			1332
8			816	45			1219
9			880	46			1014
10			837	47			759
11			853	48			458
12			892	49			250
13			966	50			145
14			1044	51			118
15			828	52			128
16			880	53			191
17			1089	54			375
18			1095	55			593
19			1156	56			834
20			1182	57			1031
21			1349	58			1062
22			1461	59			1043
23			938	60			971
24			488	61			943
25			244	62			1019
26			149	63			1243
27			133	64			1130
28			188	65			976
29			324	66			1087
30			654	67			1192
31			905	68			1228
32			1007	69			1129
33			1030	70			960
34			1001	71			703
35			989	72			462
36			1059	73			261
37			1093	74			161

№	Мб/час		
	Реальные данные	Прогноз	База
75			108
76			118
77			176
78			328
79			565
80			827
81			1020
82			1054
83			1016
84			914
85			839
86			815
87			807
88			842
89			870
90			939
91			1021
92			1095
93			1017
94			826
95			574
96			445
97			253
98			150
99			132
100			185
101			330
102			624
103			789
104			926
105			1042
106			1110
107			1100
108			1082
109			1095
110			1067
111			1024
112			1024
113			828
114			1763
115			1659
116			1584
117			1392
118			1048
119			657
120			351

№	Мб/час		
	Реальные данные	Прогноз	База
121			184
122			118
123			110
124			94
125			273
126			591
127			759
128			885
129			936
130			924
131			896
132			929
133			1006
134			1042
135			1054
136			1044
137			1072
138			1122
139			1197
140			1234
141			1156
142			894
143			604
144			323
145			168
146			113
147			100
148			139
149			265
150			523
151			708
152			804
153			840
154			824
155			757
156			897
157			954
158			1056
159			1000
160			999
161			1054
162			1095
163			1143
164			1185
165			1131
166			905

№	Мб/час		
	Реальные данные	Прогноз	База
167			607
168			329
169			169
170			106
171			95
172			138
173			260
174			514
175			712
176			794
177			822
178			794
179			789
180			846
181			923
182			985
183			980
184			996
185			1028
186			1083
187			1131
188			1171
189			1115
190			879
191			600
192			329
193			170
194			107
195			103
196			138
197			270
198			517
199			704
200			787
201	822	960	
202	824	954	
203	836	939	
204	884	960	
205	937	1013	
206	974	1071	
207	962	1126	
208	958	1196	
209	959	1301	
210	1006	1416	
211	1057	1476	
212	1116	1415	

№	Мб/час		
	Реальные данные	Прогноз	База
213	1041	1220	
214	883	949	
215	611	683	
216	380	477	
217	215	337	
218	123	243	
219	101	192	
220	121	214	
221	198	333	
222	392	535	
223	597	758	
224	823	929	
225	966	1016	
226	1000	1037	
227	968	1034	
228	925	1030	
229	872	1018	
230	825	992	
231	817	975	
232	817	1010	
233	815	1116	
234	882	1252	
235	958	1326	
236	1038	1260	
237	1001	1043	
238	848	747	
239	633	472	
240	378	284	
241	224	190	
242	136	162	
243	92	183	
244	101	260	
245	166	402	
246	314	585	
247	572	760	
248	855	882	
249	1029	942	
250	1046	968	

Таблица В.2

Результат мониторинга трёхчасового прогнозирования трафика сервера сети с указанием доверительного интервала 90% надёжности

Факт, Мб/час	Прогноз, Мб/час	min	max	Относительная ошибка прогноза %
297	363	69	656	22,1
536	637	344	931	19,0
827	832	538	1127	0,7
1039	910	616	1204	12,4
1078	982	687	1277	8,9
1018	903	610	1196	11,3
930	819	524	1114	11,9
867	806	510	1102	7,0
851	865	570	1160	1,7
865	895	600	1191	3,5
913	888	592	1184	2,7
936	854	558	1150	8,7
968	928	633	1223	4,1
1022	1036	743	1330	1,5
1096	1095	803	1387	0,1
1126	1032	740	1324	8,4
1296	885	596	1175	31,7
758	667	379	955	12,0
378	431	132	730	14,2
238	179	-122	480	25,0
298	131	-170	432	56,1
311	184	-118	486	41,0
340	223	-83	529	34,5
480	287	-19	592	40,2
745	453	147	759	39,2
880	664	359	970	24,5
943	839	532	1146	11,0
921	951	644	1259	3,3
879	999	692	1306	13,6
861	1001	693	1309	16,3
897	929	620	1238	3,6
988	879	567	1190	11,0
1014	922	610	1233	9,2
1029	1070	758	1383	4,0
994	1097	784	1409	10,3
1018	1071	759	1383	5,2
1080	1034	721	1347	4,3
1096	1145	832	1458	4,5
1127	1251	938	1564	11,0
1116	1175	861	1489	5,3
889	967	654	1280	8,7
562	717	405	1030	27,7
291	463	151	776	59,4
157	248	-66	562	57,4

Факт, Мб/час	Прогноз, Мб/час	min	max	Относительная ошибка прогноза %
97	184	-129	498	89,5
92	142	-171	455	54,3
142	223	-90	535	56,2
267	363	53	673	36,1
513	512	201	824	0,1
717	684	373	994	4,7
760	788	480	1096	3,7
773	828	522	1135	7,2
754	828	521	1136	9,9
767	816	509	1124	6,4
804	792	484	1100	1,6
816	794	486	1102	2,7
847	829	519	1139	2,1
955	934	623	1245	2,2
948	1015	704	1325	7,0
972	1018	708	1329	4,8
1020	988	678	1299	3,1
1062	948	642	1254	10,8
1096	948	641	1255	13,5
1047	970	662	1279	7,3
897	871	562	1180	2,9

ПРИЛОЖЕНИЕ Г

Листинг програмного обеспечения для построения прогнозов

```
//---------------------------------------------------------------------------

#include <vcl.h>
#include <math.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

double __fastcall sqr(double x){ return x*x; }

double __fastcall TForm1::Find_W_S(double &w,double &a,double &b,double &c)
{
   //Find code
   int i,j,k;
   double sf,ff,si,co,sisi,coco,sico,fsi,fco,d;
   double ts,tc,tf,x,s;
   sf=ff=si=co=sisi=coco=sico=fsi=fco=0.0;
   for(i=0;i<LONG;i++)
   {
     x=XX[i];
     tf=GK[i];
     ts=sin(w*x);
     tc=cos(w*x);
     sf+=tf;
     ff+=tf*tf;
     si+=ts;
     co+=tc;
     sisi+=ts*ts;
     coco+=tc*tc;
     sico+=ts*tc;
     fsi+=tf*ts;
     fco+=tf*tc;
   }
   d=sisi*coco*LONG+2.0*si*sico*co-co*co*sico-si*si*coco-sico*sico*LONG;
   a=sf*sisi*coco+fsi*sico*co+si*sico*fco-co*sico*fco-fsi*si*coco-sf*sico*sico;
   b=fsi*coco*LONG+si*fco*co+sf*sico*co-co*fsi*co-si*sf*coco-fco*sico*LONG;
   c=sisi*fco*LONG+si*fsi*co+si*sico*sf-sf*sisi*co-si*si*fco-sico*fsi*LONG;
   a/=d; //a
   b/=d; //b
   c/=d; //c
   s=0.0;
   for(i=0;i<LONG;i++)
   {
     s+=sqr((GK[i]-(a+b*sin(w*XX[i])+c*cos(w*XX[i]))));
   }
   //s=ff+a*a+b*b*sisi+c*c*coco-2.0*(a*sf+b*fsi+c*fco)+2.0*(a*b*si+a*c*co+b*c*sico);
   s/=LONG;
   return s;
}

void __fastcall TForm1::Button5Click(TObject *Sender)
{
   LoadData();
   int n;
   unsigned int MAX,MIN;
   int i;
   double w,s,wmin,wmax,a,b,c,*DAT;
   MAX=0;
```

```
   MIN=0;
   wmin=2.0*M_PI/((XX[LONG-1]-XX[0])*15.0);
   wmax=LONG*0.5*M_PI/(XX[LONG-1]-XX[0]);
   n=6/M_PI*(wmax-wmin)*(XX[LONG-1]-XX[0]);
   DAT=new double[n];
   for(i=0;i<n;i++)
   {
     w=wmin+(wmax-wmin)*i/n;
     s=Find_W_S(w,a,b,c);
     DAT[i]=floor(s*200.0);
     if(DAT[i]>DAT[MAX]) MAX=i;
     if(DAT[i]<DAT[MIN]) MIN=i;
   }
   MyGr->SetDATA(DAT,n);
   MyGr->LineColor=clRed;
   MyGr->FoneColor=clWhite;
   MyGr->SetRectangle(wmin,DAT[MIN],wmax,DAT[MAX]);
   MyGr->Plot();
   Form3->Show();
   delete[] DAT;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
   XX=YY=GK=NULL;
   StringGrid1->Cells[0][0]="№";
   StringGrid1->Cells[1][0]="w";
   StringGrid1->Cells[2][0]="a";
   StringGrid1->Cells[3][0]="sin(wt)";
   StringGrid1->Cells[4][0]="cos(wt)";
   StringGrid1->Cells[5][0]="S(w)";
   this->LONG=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
   delete MyGr;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::LoadData()
{
  LONG=Memo1->Lines->Count;
  if (XX!=NULL) {delete[] XX; XX=NULL;}
  if (YY!=NULL) {delete[] YY; YY=NULL;}
  if (GK!=NULL) {delete[] GK; GK=NULL;}
  XX=new double[LONG];
  YY=new double[LONG];
  GK=new double[LONG];
  int i,n;
  String s1,s2;
  for (i=0;i<LONG;i++)
  {
    s1=Memo1->Lines->operator [](i);
    n=s1.Pos("\t");
    if(n==0) n=s1.Pos(" ");
    if(n==0)
    {
      MessageBox(NULL,"Не вірний формат даних!","ERROR!",MB_OK);
      return;
    }
    s2=s1;
    s2.Delete(1,n);
       s1.Delete(n,s2.Length()+1);
    XX[i]=s1.ToDouble();
    YY[i]=GK[i]=s2.ToDouble();
  } //Начальные данные считаны

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
   LoadData();
   int n;
   int ko=0;
   int MIN=0;
   int i,j;
   int Auto = 0;
   double w,s,wmin,wmax,*DAT,a,b,c;
```

```
    double PR,DR;
    StringGrid1->RowCount=1;
    DR=1e400;
    wmin=2.0*M_PI/((XX[LONG-1]-XX[0])*15.0);  //Обмеження частоти 15 періодов
    wmax=0.5*M_PI/(XX[1]-XX[0]);              //Обмеження за частотою Найквіста-Котельникова
    if( !CheckBox1->Checked ) wmax += wmax;
    n=2.0/M_PI*(wmax-wmin)*(XX[LONG-1]-XX[0]);  //Початкове розбиття частотної області
                                                                          //з           врахуванням
паразитних мінімумів
    DAT=new double[n];                        //Масив початкових наближень по частотам
    if(TrackBar1->Position==1)
    {
           TrackBar1->Position = 256;
           Auto = 1;
    }
    for(j=0;j<TrackBar1->Position;j++)
    {
      PR=DR;
      wmin=2.0*M_PI/((XX[LONG-1]-XX[0])*15.0);
          wmax=0.5*M_PI/(XX[1]-XX[0]);
          if( !CheckBox1->Checked ) wmax += wmax;
      //n=2/M_PI*(wmax-wmin)*(XX[LONG-1]-XX[0]);
      MIN=0;
      for(i=0;i<n;i++)
      {
        w=wmin+(wmax-wmin)*i/n;
           DAT[i]=Find_W_S(w,a,b,c);  //Заповнення масиву відхиленнями від частот

           if(DAT[i]<DAT[MIN]) MIN=i; //Запам'ятати мінімальне відхилення
        }
        w   =wmin+(wmax-wmin)*(MIN-1)/n; //
        wmax=wmin+(wmax-wmin)*(MIN+1)/n; //
      wmin=w;
      w=0.5*(wmax+wmin);
      double w1,w2,am[5];
      am[0]=DAT[MIN-1]; am[2]=Find_W_S(w,a,b,c); am[4]=DAT[MIN+1];
          while( (wmax-wmin)>(w*0.000001) )//Уточнення методом дихотомії
          {
            w1=(wmin+w)*0.5;
            am[1]=Find_W_S(w1,a,b,c);
            w2=(wmax+w)*0.5;
            am[3]=Find_W_S(w2,a,b,c);
            if( (am[1]<am[2])&&(am[1]<am[3]) )
            {
          wmax=w; w=w1;
          am[4]=am[2];
          am[2]=am[0];
            }else
            if( (am[2]<am[1])&&(am[2]<am[3]) )
            {
          wmax=w2; am[4]=am[3];
          wmin=w1; am[0]=am[1];
            }else
            if( (am[3]<am[2])&&(am[3]<am[1]) )
            {
          wmin=w; w=w2;
          am[0]=am[2];
          am[2]=am[3];
            }
      }
      StringGrid1->RowCount++;
          StringGrid1->Cells[0][StringGrid1->RowCount-1]=IntToStr(StringGrid1->RowCount-1);
          StringGrid1->Cells[1][StringGrid1->RowCount-1]=FloatToStr(w);
          StringGrid1->Cells[2][StringGrid1->RowCount-1]=FloatToStr(a);
          StringGrid1->Cells[3][StringGrid1->RowCount-1]=FloatToStr(b);
          StringGrid1->Cells[4][StringGrid1->RowCount-1]=FloatToStr(c);
          StringGrid1->Cells[5][StringGrid1->RowCount-1]=FloatToStr(am[2]);
          DR=a*a;
          double DDD,ddd;
          DDD=ddd=0.0;
          for(i=0;i<LONG;i++)  //Виключення з початкових даних знайдену гармоніку
          {
            DDD+=(GK[i]-a)*(GK[i]-a);
            GK[i]-=a+b*sin(w*XX[i])+c*cos(w*XX[i]);
            ddd+=GK[i]*GK[i];
          }
          DDD = sqrt(DDD/ddd);

          if ( (DDD<1.05) &&(ko==0) )
```

```
     {
       TrackBar1->SelStart=1;
       TrackBar1->SelEnd=j;
          StatusBar1->SimpleText="Рекомендується "+IntToStr(j+1)+" гармонік.";
          ko++;
          if( Auto==1 )
          {
         TrackBar1->Position = j+1;
        break;
          }
     }
   }
   delete[] DAT;
   DAT=NULL;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  LoadData();
  int i,MIN,MAX;
  MIN=MAX=0;
  for(i=1;i<LONG;i++)
  {
    if (YY[i]>YY[MAX]) MAX=i;
    if (YY[i]<YY[MIN]) MIN=i;
  }
  MyGr->SetDATA(YY,LONG);
  MyGr->LineColor=clRed;
  MyGr->FoneColor=clWhite;
  MyGr->SetRectangle(XX[0],YY[MIN],XX[LONG-1],YY[MAX]);
  MyGr->Plot();
  Form3->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
  StatusBar1->SimpleText=IntToStr(TrackBar1->Position);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
  Form4->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
  int i;
  String S;
  Memo2->Lines->Clear();
  for(i=0;i<StringGrid1->RowCount;i++)
  {
       S = StringGrid1->Cells[0][i]+ " \t ";//=IntToStr(StringGrid1->RowCount-1);
       S+= StringGrid1->Cells[1][i]+ " \t ";//=FloatToStr(w);
       S+= StringGrid1->Cells[2][i]+ " \t ";//=FloatToStr(a);
       S+= StringGrid1->Cells[3][i]+ " \t ";//=FloatToStr(b);
       S+= StringGrid1->Cells[4][i]+ " \t ";//=FloatToStr(c);
       S+= StringGrid1->Cells[5][i];//=FloatToStr(am[2]);
       Memo2->Lines->Append(S);
  }
  Memo2->SelectAll();
  Memo2->CopyToClipboard();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button6Click(TObject *Sender)
{
  Button7->Enabled = true;
  int i;
  String S;
  W1.clear();
  for(i=1;i<StringGrid1->RowCount;i++)
  {
       W1.push_back(StrToFloat(StringGrid1->Cells[1][i]));
  }
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::Button7Click(TObject *Sender)
{
  double B = 0.0;
  double K = 0.0;
  int i,j,k,N,best_i,best_j;
  double w1,w2;
  W2.clear();
  for(i=1;i<StringGrid1->RowCount;i++)
  {
       W2.push_back(StrToFloat(StringGrid1->Cells[1][i]));
  }
  N = W1.size();
  if(N>W2.size()) N=W2.size();
  std::vector<double> tmpW(W1.begin(),W1.begin()+N);
  //W1 потрібно залишити недоторканим для наступних порівнянь
  //Для всіх частот...
  while(tmpW.size()>0)
  {
       //Кожну частоту першої послідовності порівнюємо з частотою другої послідовності
       best_i = 0;
       best_j = 0;
       K=1e100;
       for(i=0;i<tmpW.size();i++)
       {
         for(j=0;j<W2.size();j++)
         {
       w1 = tmpW.at(i);
       w2 = W2.at(j);
       if( fabs(w1-w2)<K )
       {
              best_i = i;
              best_j = j;
              K = fabs(w1-w2);
       }
         }
       }
       w1 = tmpW.at(best_i);
       w2 = W2.at(best_j);
       B += 2.0*w1*w2/(w1*w1+w2*w2);
       tmpW.erase(tmpW.begin()+best_i);
       W2.erase(W2.begin()+best_j);
  }
  B /= N;
  Memo2->Clear();
  Memo2->Lines->Append(FloatToStr(B));
  Memo2->SelectAll();
  Memo2->CopyToClipboard();
  StatusBar1->SimpleText="B = "+FloatToStr(B);
}
//---------------------------------------------------------------------------
```

Исходный код программного обеспечения. Мониторинговый прогноз на средние сроки.

```
unit Analitic;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils;

type
  TExtrapol = class
  private
    DataY     : array of Double;
    DataX     : array of Double;
    CountData : Integer;
    maxW      : Double;
    A         : array [0..25] of Double;
    B         : array [0..25] of Double;
    C         : array [0..25] of Double;
    W         : array [0..25] of Double;
    function  FindAB(Wch:Double; var cA:Double; var cB:Double; var cC:Double):Double;
    function  FindW(garm : Integer):Double;
  public
    deviation : Double;
    dispertion: Double;
    procedure SetData(var Xarray, Yarray: array of Double; lengthArray: Integer);
    procedure GetExtra(var Xarray, Yarray: array of Double; lengthArray: Integer);
  end;

implementation

function  TExtrapol.FindAB(Wch:Double; var cA:Double; var cB:Double; var cC:Double):Double;
var Yco, Ysi, YY : Double;
    coco, sisi, cosi : Double;
    d          : Double;
    i          : Integer;
    si, co     : Double;
    Y, _s, _c    : Double;
    N          : Integer;
begin
  Yco  := 0.0;
  Ysi  := 0.0;
  coco := 0.0;
  sisi := 0.0;
  cosi := 0.0;
  YY := 0.0;
  _s := 0.0;
  _c := 0.0;
  i := 0;
  N := CountData;
  Y := 0.0;
  while CountData-1>=i do
  begin  // Шукаэмо суми для системи рівнянь методу
    // найменших квадратів
    si := sin(Wch*DataX[i]);
    co := cos(Wch*DataX[i]);
    YY := YY + DataY[i]*DataY[i];
    Y  := Y + DataY[i];
    Yco := Yco + DataY[i]*co;
    Ysi := Ysi + DataY[i]*si;
    coco := coco + co*co;
    sisi := sisi + si*si;
    cosi := cosi + co*si;
    _c  := _c + co;
    _s  := _s + si;
    i := i + 1;// + ((CountData-1)div 500);
    //N := N + 1;
  end;
  Yco  := Yco / N;
  Ysi  := Ysi / N;
  coco := coco / N;
  sisi := sisi / N;
  cosi := cosi / N;
  YY := YY / N;
```

```
  _s := _s / N;
  _c := _c / N;
  Y := Y / N;

  //    cC   cA    cB
  //  | 1    _c    _s   |  Y
  //  | _c  coco  cosi  |  Yco
  //  | _s  cosi  sisi  |  Ysi
  //
  d  :=  sisi*coco+_c*cosi*_s*2.0-_s*_s*coco-cosi*cosi-_c*_c*sisi;  //  Розв'язуємо  систему  методом
Крамара
  if abs(d)<1e-30 then
  begin  // Якщо коливанння немає, то попадемо сюди
    cB := 0.0;
    cA := 0.0;
    cC := Y;
  end else
  begin  // Ось коефіцієнти
  //
  //  |  y    co    si   |
  //  | yco  coco  cosi  |
  //  | ysi  cosi  sisi  |
  //
    cC := (Y*coco*sisi+_c*cosi*Ysi+_s*cosi*Yco-_s*coco*Ysi-cosi*cosi*Y-Yco*_c*sisi)/d;
    //
    //  | 1    y    si   |
    //  | co  yco  cosi  |
    //  | si  ysi  sisi  |
    //
    cA := (sisi*Yco+_c*Ysi*_s+Y*cosi*_s-_s*_s*Yco-cosi*Ysi-_c*Y*sisi)/d;
  //
  //  | 1    co    y   |
  //  | co  coco  yco  |
  //  | si  cosi  ysi  |
  //
    cB := (Ysi*coco+_c*Yco*_s+_c*cosi*Y-Y*coco*_s-_c*_c*Ysi-cosi*Yco)/d;
  end;
  // повернемо середнє квадратичне відхилення:
  Result := (YY-2.0*Y*cC-2.0*cA*Yco-2.0*cB*Ysi+cC*cC+2.0*cC*cA*_c+2.0*cC*cB*_s+
             cA*cA*coco+2.0*cA*cB*cosi+cB*cB*sisi);
end;

function  TExtrapol.FindW(garm : Integer):Double;
var
    cA, cB, cC: Double;
    mW        : Double;
    Sk, Sm    : Double;
    dW        : Double;
begin
  MaxW := 0.5*Pi*CountData/(DataX[CountData-1]-DataX[0]); // За теоремою Котельникова-Найквіста
  mW   := 0.0;//0.25*Pi/(DataX[CountData-1]-DataX[0]);
  dW   := 0.001*MaxW;
  Sm   := 1e200;
  cA := 0.0;
  cB := 0.0;
  cC := 0.0;
  //WriteLn('Finded min W=',mW:4:4);
  while mW<MaxW do
  begin
    Sk := FindAB(mW,cA,cB,cC);
    if Sk<Sm then
    begin
      //WriteLn('Finded min W=',mW:4:4,'  S=',Sk:4:4);
      Sm := Sk;
      A[garm] := cA;
      B[garm] := cB;
      C[garm] := cC;
      W[garm] := mW;
    end;
    mW := mW + dW;
  end;
  Result := Sm;//W[garm];
end;

procedure TExtrapol.SetData(var Xarray, Yarray: array of Double; lengthArray: Integer);
var i: Integer;
begin
  SetLength(DataX,lengthArray);
  SetLength(DataY,lengthArray);
```

```
  for i:=0 to lengthArray-1 do
  begin
    DataX[i] := Xarray[i];
    DataY[i] := Yarray[i];
    //WriteLn(DataX[i],' ::: ',DataY[i]);
  end;
  CountData := lengthArray;
end;

{**
 Xarray повинен бути заздалегіть підготовлений
**}
procedure TExtrapol.GetExtra(var Xarray, Yarray: array of Double; lengthArray: Integer);
var garm : Integer;
    i    : Integer;
begin
  //writeln(1);
  //FindAB(0.0,A[0],B[0],C[0]);  //Пошук сталоъ компоненти (частота=0)
  //writeln(2);
  //W[0] := 0.0;
  for garm:=0 to 25 do //Пошук інших гармонік
  begin
    FindW(garm); // Шукаємо наступну складову
    for i:=0 to CountData-1 do
    begin //Знайдену гармоніку вилучаємо з даних
      DataY[i] := DataY[i] -
          (C[garm]+A[garm]*cos(W[garm]*DataX[i])+B[garm]*sin(W[garm]*DataX[i]));
    end;
    //writeln(garm,':  ',W[garm],'  ',C[garm],'  ',A[garm],'  ',B[garm]);
  end;
  //За основу кроку в часі беремо останній крок
  deviation  := 0.0;
  dispertion := 0.0;
  for i:=0 to CountData-1 do
  begin
    deviation:=deviation+abs(DataY[i]);
    dispertion:=dispertion+sqr(DataY[i]);
  end;
  deviation:=deviation/CountData;
  dispertion:=dispertion/CountData;

  for i:=0 to lengthArray-1 do
  begin
    Yarray[i] := 0.0;
    for garm:=0 to 25 do
    begin
      Yarray[i]:=Yarray[i]                                                                +
(C[garm]+A[garm]*cos(W[garm]*Xarray[i])+B[garm]*sin(W[garm]*Xarray[i]));
    end;
  end;
end;

end.

program project1;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes, SysUtils, Analitic, CustApp
  { you can add units after this };


var
  inData: array [0..255] of Double;
  x     : array [0..255] of Double;
  xp    : array [0..2]   of Double;
  yp    : array [0..2]   of Double;
  n, i, k: Integer;
  prognozer: TExtrapol;
begin
  n := 0;
  while (not System.EOF(input))and( n<(255) ) do
  begin
    ReadLn(inData[n]);
    inc(n);
```

```
  end;
  if n=(255) then
  begin
    prognozer := TExtrapol.Create;
    writeln(' № ', '  ', ' Прогноз ', '  ', ' СреднееОтклонение', '  ', 'ДисперсияОтклонения');
    for i:=0 to 255 do
    begin
      x[i] := i;
    end;
    xp[0] := n;
    xp[1] := n+1;
    yp[0] := 0.0;
    yp[1] := 0.0;
    k := n;
    while not System.EOF(input) do
    begin
      for i:=1 to n-1 do
      begin
        inData[i-1] := inData[i];
      end;
      readln(inData[n-1]);
      prognozer.SetData(x, inData, n);
      prognozer.GetExtra(xp, yp, 2);
      writeln(k:5, '  ', yp[1]:3:2, '  ', prognozer.deviation, '  ', prognozer.dispertion);
      inc(k);
      if k>1500 then break;
    end;
    prognozer.Destroy;
  end;
end.
```

Листинг программного обеспечения сжатия и восстановления фотографического изображения

Исходный код программного обеспечения кодирования и восстановления фотографического изображения SPIHT алгоритмом с и без отложенной передачи значимых битов

```
Содержимое файла imagebylayer.h
#ifndef IMAGEBYLAYER_H
#define IMAGEBYLAYER_H

#include "colorlayer.h"

#define LayerRED   0
#define LayerGREEN 1
#define LayerBLUE  2

#define LayerCg    0
#define LayerY     1
#define LayerCo    2

class ImageByLayer
{
public:
    void ReadImageFromFile(const char *file_name);
    void WriteImageToFile(const char* file_name);
    void NewImage();
    short getR(int x, int y);
    short getG(int x, int y);
    short getB(int x, int y);
    short getY(int x, int y);
    short getCg(int x, int y);
    short getCo(int x, int y);
    short* getData(int layer_num);
    ImageByLayer();
    ~ImageByLayer();
    void RGB_To_Ygo();
    void Ygo_To_RGB();
private:
    ColorLayer* _R_Cg;
    ColorLayer* _G_Y;
    ColorLayer* _B_Co;

};

#endif // IMAGEBYLAYER_H

Содержимое файла imagebylayer.cpp
#include <fstream>
#include <memory.h>
#include "imagebylayer.h"

char BMP_Head[]={
      //0     1     2     3     4     5     6     7     8     9     A     B     C     D     E     F
/*0*/   0x42, 0x4D, 0x36, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x36, 0x00, 0x00, 0x00, 0x28,
0x00,
/*1*/   0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x01, 0x00, 0x18, 0x00, 0x00,
0x00,
/*2*/   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x0B, 0x00, 0x00, 0x12, 0x0B, 0x00, 0x00, 0x00,
0x00,
/*3*/  0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

void ImageByLayer::ReadImageFromFile(const char* file_name)
{
    std::ifstream infile;
    infile.open(file_name,std::ios_base::in | std::ios_base::binary);
    if(infile.is_open()==true)
    {
        unsigned char* data = new unsigned char[512*512*3];
        infile.read((char*)data,54);
        infile.read((char*)data,512*512*3);
        infile.close();
```

```
        if( _R_Cg==NULL ) _R_Cg = new ColorLayer();
        if( _G_Y==NULL )  _G_Y = new ColorLayer();
        if( _B_Co==NULL ) _B_Co = new ColorLayer();
        _R_Cg->import24RGB(data,0);
        _G_Y->import24RGB(data,1);
        _B_Co->import24RGB(data,2);
        delete data;
    }
}

void ImageByLayer::WriteImageToFile(const char *file_name)
{
    if( _R_Cg!=NULL && _G_Y!=NULL &&_B_Co!=NULL )
    {
        std::ofstream outfile;
        outfile.open(file_name);
        unsigned char* data = new unsigned char[512*512*3];
        _R_Cg->export24RGB(data,0);
        _G_Y->export24RGB(data,1);
        _B_Co->export24RGB(data,2);
        //_G_Y->export24RGB(data,0);
        //_G_Y->export24RGB(data,1);
        //_G_Y->export24RGB(data,2);
        outfile.write(BMP_Head,54);
        outfile.write((char*)data,512*512*3);
        outfile.close();
        delete data;
    }
}

void ImageByLayer::NewImage()
{
    if( _R_Cg==NULL ) _R_Cg = new ColorLayer();
    if( _G_Y==NULL )  _G_Y  = new ColorLayer();
    if( _B_Co==NULL ) _B_Co = new ColorLayer();
    memset(_R_Cg->getData(),0,512*512);
    memset(_B_Co->getData(),0,512*512);
    memset(_G_Y->getData(),0,512*512);
}

short ImageByLayer::getR(int x, int y)
{
    if( _R_Cg!=NULL ) return _R_Cg->getXY(x,y);
    return 0;
}

short ImageByLayer::getG(int x, int y)
{
    if( _G_Y!=NULL ) return _G_Y->getXY(x,y);
    return 0;
}

short ImageByLayer::getB(int x, int y)
{
    if( _B_Co!=NULL ) return _B_Co->getXY(x,y);
    return 0;
}

short ImageByLayer::getCg(int x, int y)
{
    if( _R_Cg!=NULL ) return _R_Cg->getXY(x,y);
    return 0;
}

short ImageByLayer::getY(int x, int y)
{
    if( _G_Y!=NULL ) return _G_Y->getXY(x,y);
    return 0;
}

short ImageByLayer::getCo(int x, int y)
{
    if( _B_Co!=NULL ) return _B_Co->getXY(x,y);
    return 0;
}

short *ImageByLayer::getData(int layer_num)
{
```

```
    switch (layer_num) {
    case 0:
        if(_R_Cg!=NULL) return _R_Cg->getData();
        break;
    case 1:
        if(_G_Y!=NULL) return _G_Y->getData();
        break;
    case 2:
        if(_B_Co!=NULL) return _B_Co->getData();
        break;
    }
    return NULL;
}

ImageByLayer::ImageByLayer()
{
    _R_Cg = NULL;
    _G_Y = NULL;
    _B_Co = NULL;
}

ImageByLayer::~ImageByLayer()
{
    if( _R_Cg != NULL) delete _R_Cg;
    if( _G_Y  != NULL) delete _G_Y;
    if( _B_Co != NULL) delete _B_Co;
}


/*
 G          R          B     Y          Cg         Co
 |          |          |     |          |          |
 |          |->-(-1)->(+)   (+)<-(-/2)<-|          |
 |          |          |     |          |          |
 |         (+)<-(/2)-<-|     |->-(+1)->(+)         |
 |          |          |     |          |          |
 |->-(-1)->(+)         |     |         (+)<-(-/2)<-|
 |          |          |     |          |          |
(+)<-(/2)-<-|          |     |          |->-(+1)->(+)
 |          |          |     |          |          |
 Y          Cg         Co    G          R          B
*/
void ImageByLayer::RGB_To_Ygo()
{
    int x,y;
    short R,G,B;
    short Y,Cg,Co;
    for(y=0;y<512;y++)
    {
        for(x=0;x<512;x++)
        {
            R = _R_Cg->getXY(x,y);
            G = _G_Y->getXY(x,y);
            B = _B_Co->getXY(x,y);
            Co = B - R;
            Cg = R + Co/2 - G;
            Y  = G + Cg/2;
            _R_Cg->setXY(x,y,Cg);
            _G_Y->setXY(x,y,Y);
            _B_Co->setXY(x,y,Co);
        }
    };
}

void ImageByLayer::Ygo_To_RGB()
{
    int x,y;
    short R,G,B;
    short Y,Cg,Co;
    for(y=0;y<512;y++)
    {
        for(x=0;x<512;x++)
        {
            Cg = _R_Cg->getXY(x,y);
            Y  = _G_Y->getXY(x,y);
            Co = _B_Co->getXY(x,y);
            G = Y - Cg/2;
            R = Cg - Co/2 + G;
            B = Co + R;
```

```
            if(G<0) G=0;
            if(G>255) G=255;
            if(R<0) R=0;
            if(R>255) R=255;
            if(B<0) B=0;
            if(B>255) B=255;
            _R_Cg->setXY(x,y,R);
            _G_Y->setXY(x,y,G);
            _B_Co->setXY(x,y,B);
        }
    };
}
```

Содержимое файла bitfile.h

```
#ifndef BITFILE_H
#define BITFILE_H

#include <fstream>
#include <stdlib.h>
#include <cstring>

class BitFile
{
public:
    BitFile(const char* file_name);
    BitFile();
    ~BitFile();
    void setNameFile(const char* file_name);
    bool openForReadOnly();
    bool openForWriteOnly();
    bool close();
    char readBit();
    void writeBit(const char bit);
    //void apply();
    bool isOpen();
    int  getSize();
    void useQM(const bool setUseQM);
    bool isUseQM();
private:
    bool usedQM;
    int  totalSize;
    bool modified;
    bool  forRead;
    char* FileName;
    std::fstream file;
    unsigned int currentBit;
    unsigned int currentByte;
    unsigned int buffSize;
    char* Buff;
    unsigned int puck(const unsigned char* inBuff, unsigned int length, unsigned char* outBuff);
    unsigned int unpuck(const unsigned char* inBuff, unsigned int &length, unsigned char* outBuff);
};

#endif // BITFILE_H
```

Содержимое файла bitfile.cpp

```
#include "bitfile.h"
#include <fstream>
#include <stdlib.h>
#include <cstring>

BitFile::BitFile(const char* file_name)
{
    FileName = new char[strlen(file_name)+1];
    strcpy(FileName,file_name);
    Buff = new char[1024];
    modified = false;
    forRead = false;
    currentBit = 0;
    currentByte = 0;
}

BitFile::BitFile()
{
    FileName = NULL;
    Buff = new char[1024];
    modified = false;
    forRead = false;
    currentBit = 0;
    currentByte = 0;
```

```
}

void BitFile::setNameFile(const char* file_name)
{
    if (FileName!=NULL) delete FileName;
    FileName = new char[strlen(file_name)+1];
    strcpy(FileName,file_name);
}

BitFile::~BitFile()
{
    delete FileName;
    delete Buff;
}

bool BitFile::openForReadOnly()
{
    file.open(FileName,std::ios_base::in | std::ios_base::binary);
    if( file.is_open()==true )
    {
        modified = false;
        forRead = true;
        currentBit = 0;
        currentByte = 0;
        file.read(Buff,1024);
        buffSize = file.gcount();
        totalSize = 0;
    }
    return file.is_open();
}

bool BitFile::openForWriteOnly()
{
    file.open(FileName,std::ios_base::out | std::ios_base::binary | std::ios_base::trunc );
    if( file.is_open()==true )
    {
        modified = true;
        forRead = false;
        currentBit = 0;
        currentByte = 0;
        int i;
        for(i=0;i<1024;i++) Buff[i]=0;
        buffSize = 1024;
        totalSize = 0;
    }
    return file.is_open();
}

bool BitFile::close()
{
    if(modified==true && file.is_open()==true && forRead==false)
    {
        if(currentBit!=0) currentByte ++;
        file.write(Buff, currentByte);
    }
    if(file.is_open()==true)
    {
        file.close();
    }
    totalSize = 0;
    return file.is_open();
}

char BitFile::readBit()
{
    unsigned char inBuff[1024];
    int puckedLength;
    if( file.is_open()!=true || buffSize==0 ) return 0;
    char result = (Buff[currentByte] & (1<<currentBit))>>currentBit;
    if( forRead==true )
    {
        if( currentBit==7 )
        {
            currentBit = 0;
            currentByte++;
            if( currentByte==buffSize )
            {
                currentByte = 0;
                if( forRead==true )
```

```
                {
                    file.read(Buff,512);
                    buffSize = file.gcount();
                    //file.read(inBuff,1024);
                    //puckedLength = file.gcount();
                    //buffSize = unpuck(inBuff,puckedLength,Buff);
                    //file.seekg(buffSize-puckedLength,std::ios_base::cur);
                    //buffSize = 512;
                }
            }
        }else
        {
            currentBit++;
        }
    }
    totalSize++;
    return result;
}

void BitFile::writeBit(const char bit)
{
    unsigned char outBuff[1024];
    int puckedLength;
    if( file.is_open()!=true || forRead==true) return;
    Buff[currentByte] |= bit<<currentBit;
    currentBit++;
    if(currentBit==8)
    {
        currentBit = 0;
        currentByte++;
        if(currentByte==512)
        {
            //puckedLength = puck(Buff,512,outBuff);
            //file.write(outBuff,puckedLength);
            //totalSize += puckedLength*8;
            file.write(Buff,512);
            for(int i=0;i<1024;i++) Buff[i]=0;
            currentByte = 0;
        }
        buffSize = currentByte+1;
    }
    totalSize++;
}

bool BitFile::isOpen()
{
    return file.is_open();
}

int BitFile::getSize()
{
    return (totalSize+7)/8;
}

void BitFile::useQM(const bool setUseQM)
{

}

bool BitFile::isUseQM()
{
    return false;
}

unsigned int BitFile::puck(const unsigned char* inBuff, unsigned int length, unsigned char* outBuff)
{

  if( (length == 0)||(inBuff == NULL)||(outBuff == NULL) )
  {
    return 0;
  }
  unsigned char v = 0; //Доля в блоці одиничних бітів
  //Рахування одиничних бітів.
  unsigned long bits1 = 0;
  unsigned long i = 0;
  unsigned char b = 0;
  unsigned long upsLength = 0;
  while( i < length )
  {
```

```
  b = inBuff[i];
  while( b!=0 )
  {
    if( (b & 0x01)!=0 )
    {
      bits1++;
    }
    b = b / 2;
  }
  i++;
}
v = bits1 * 255 / (length*8);

//
unsigned long hiNum = 0xFFFFFFFF; //Початкове значення верхньої межі
unsigned long loNum = 0x00000000; //Початкове значення нижньої межі
int outLength = 1;
outBuff[0] = v;
//По всьому буфері пройдемося побітно.
for(i = 0;i<length*8;i++)
{
   //Отримати біт:
   unsigned char b = inBuff[i/8];
   b = 0x01 & (b >> (7-(i % 8)));

   // [loNum...a) bit=1      [a..hiNum) bit=0       [hiNum] end
   unsigned long a = loNum + (hiNum - loNum)*v/256;
   if( b == 0 )
   {
       loNum = a;
       hiNum = hiNum - 1;
   }else
   {
       hiNum = a-1;
   }

   while( (hiNum & 0xFF000000) == (loNum & 0xFF000000) )
   {
       outBuff[outLength] = hiNum >> 24;
       hiNum = (hiNum << 8) | 0x000000FF;
       loNum = (loNum << 8);
       hiNum &= 0xFFFFFFFF;
       loNum &= 0xFFFFFFFF;
       outLength++;
       if( upsLength>0 )
       {
           while(upsLength>0)
           {
               if( b == 0)
               {
                  outBuff[outLength] = 0x00;
               }else
               {
                  outBuff[outLength] = 0xFF;
               }
               upsLength--;
               outLength++;
           }
       }
   }

   while( ((hiNum & 0x00FF0000)==0x00000000) && ((loNum & 0x00FF0000)==0x00FF0000) )
   {
      upsLength++;
      hiNum = (hiNum & 0xFF000000) | ((hiNum << 8)&0x00FFFF00) | 0x000000FF;
      loNum = (loNum & 0xFF000000) | ((loNum << 8)&0x00FFFF00);
   }

}
//hiNum как Признак конца блока

b = (hiNum >> 24) & 0xFF;
outBuff[outLength] = b;
hiNum = hiNum << 8;
outLength++;
while(upsLength>0)
{
    outBuff[outLength] = 0x00;
    upsLength--;
```

```
      outLength++;
  }

  b = (hiNum >> 24) & 0xFF;
  outBuff[outLength] = b;
  hiNum = hiNum << 8;
  outLength++;
  b = (hiNum >> 24) & 0xFF;
  outBuff[outLength] = b;
  hiNum = hiNum << 8;
  outLength++;
  b = (hiNum >> 24) & 0xFF;
  outBuff[outLength] = b;
  outLength++;
  return outLength;
}

unsigned int BitFile::unpuck(const unsigned char* inBuff, unsigned int &length, unsigned char*
outBuff)
{
  if( (length == 0)||(inBuff == NULL)||(outBuff == NULL) )
  {
    return 0;
  }
  unsigned char v = 0; //Доля в блоці одиничних бітів
  unsigned int loaded = 0;
  unsigned long i = 0;
  for(i=0;i<1024;i++) outBuff[i] = 0;
  i = 0;
  v = inBuff[0];
  loaded++;

  //
  unsigned long hiNum = 0xFFFFFFFF; //Початкове значення верхньої межі
  unsigned long loNum = 0x00000000; //Початкове значення нижньої межі
  unsigned long currentNum = 0;
  unsigned long a = 0;
  currentNum |= inBuff[1]; currentNum = currentNum << 8;
  currentNum |= inBuff[2]; currentNum = currentNum << 8;
  currentNum |= inBuff[3]; currentNum = currentNum << 8;
  currentNum |= inBuff[4];
  loaded += 4;
  unsigned int outLength = 0;

  while( outLength < 1024*8 )//
  {
     char bit = 0;

     //cout << currentNum << endl;
     // [loNum...a) bit=1      [a..hiNum) bit=0       [hiNum] end
     a = loNum + (hiNum - loNum)*v/256;
     //cout << loNum << "  " << a << "  " << hiNum << endl;
     if( currentNum < a )
     {
         bit = 1;
         hiNum = a-1;
     }else
     if( currentNum < hiNum )
     {
         bit = 0;
         loNum = a;
         hiNum = hiNum - 1;
     }else
     {
         //cout << "Find block end.\n ";
         length = loaded;
         return (outLength+7) / 8;
     }

     while( (hiNum & 0xFF000000) == (loNum & 0xFF000000) )
     {
         hiNum = (hiNum << 8) | 0x000000FF;
         loNum = (loNum << 8);
         hiNum &= 0xFFFFFFFF;
         loNum &= 0xFFFFFFFF;
         if( loaded < length )
         {
            currentNum = (currentNum << 8) | inBuff[loaded];
         }else
```

```
            {
               currentNum = (currentNum << 8);
            }
            currentNum &= 0xFFFFFFFF;
            loaded++;
        }
        while( ((hiNum & 0x00FF0000) == 0x00000000) && ((loNum & 0x00FF0000)==0x00FF0000) )
        {
            //cout << "ОППА!\n";
            hiNum = (hiNum & 0xFF000000) | ((hiNum << 8)&0x00FFFF00) | 0x000000FF;
            loNum = (loNum & 0xFF000000) | ((loNum << 8)&0x00FFFF00);
            currentNum = (currentNum & 0xFF000000) | ((currentNum << 8)&0x00FFFF00);
            if( loaded < length )
            {
                currentNum |= inBuff[loaded];
                loaded++;
            }
        }
        if(loNum > hiNum)
        {
            //cout << "Error Num!" << loNum << "  " << a << "  " << hiNum << endl;
        }
        outBuff[outLength/8] |= (1 << (7-(outLength%8)))*bit;
        outLength++;
    }
    if( hiNum != currentNum)
    {
      //cout << "Not take end of block! outLength =" << outLength/8 << "   " << outLength%8 <<"\n";
      //cout << loNum << " " << a << " " << hiNum << " " << currentNum << endl;
    }
    length = loaded;
    return (outLength+7)/8;
}
```

Содержимое файла colorlyaer.cpp

```
#ifndef COLORLAYER_H
#define COLORLAYER_H

class ColorLayer
{
public:
    ColorLayer();
    ~ColorLayer();
    void import24RGB(const unsigned char* bmp_data, int chanel);
    void import32RGBA(const unsigned char *bmp_data, int chanel);
    void export24RGB(unsigned char *bmp_data, int chanel);
    void export32RGBA(unsigned char *bmp_data, int chanel);
    void setXY(int x, int y, short color);
    short getXY(int x, int y);
    short *getData();
private:
    short* data;
};
#endif // COLORLAYER_H
```

Содержимое файла colorlyaer.cpp

```
#include "colorlayer.h"

ColorLayer::ColorLayer()
{
    data = new short[512*512];
}

ColorLayer::~ColorLayer()
{
    delete data;
}

void ColorLayer::import24RGB(const unsigned char *bmp_data, int chanel)
{
    int i, j;
    for(i=0;i<512;i++)
    {
        for(j=0;j<512;j++)
        {
            data[i*512+j] = bmp_data[(512*(511-i)+j)*3+chanel];
        }
```

```
    }
}

void ColorLayer::import32RGBA(const unsigned char* bmp_data, int chanel)
{
    int i, j;
    for(i=0;i<512;i++)
    {
        for(j=0;j<512;j++)
        {
            data[i*512+j] = bmp_data[(512*(511-i)+j)*4+chanel];
        }
    }
}

void ColorLayer::export24RGB(unsigned char *bmp_data, int chanel)
{
    int i, j;
    for(i=0;i<512;i++)
    {
        for(j=0;j<512;j++)
        {
            if(data[i*512+j]<0) bmp_data[(512*(511-i)+j)*3+chanel]=0; else
            if(data[i*512+j]>255) bmp_data[(512*(511-i)+j)*3+chanel] =255; else
                bmp_data[(512*(511-i)+j)*3+chanel] = data[i*512+j];
        }
    }
}

void ColorLayer::export32RGBA(unsigned char *bmp_data, int chanel)
{
    int i, j;
    for(i=0;i<512;i++)
    {
        for(j=0;j<512;j++)
        {
            if(data[i*512+j]<0) bmp_data[(512*(511-i)+j)*4+chanel]=0; else
            if(data[i*512+j]>255) bmp_data[(512*(511-i)+j)*4+chanel] =255; else
                bmp_data[(512*(511-i)+j)*4+chanel] = data[i*512+j];
        }
    }
}

void ColorLayer::setXY(int x, int y, short color)
{
    data[y*512+x] = color;
}

short ColorLayer::getXY(int x, int y)
{
    return data[y*512+x];
}

short *ColorLayer::getData()
{
    return data;
}

Содержимое файла intwavelet.h
#ifndef INTWAVELET_H
#define INTWAVELET_H

#define WAVELET_HAAR 0
#define WAVELET_INT5 1
#define WAVELET_DOB8 2

class IntWavelet
{
public:
    IntWavelet();
    ~IntWavelet();
    short QuantL;
    short QuantH;
    void  setImageData(short* imageData);
    short *getImageData();
    void  setwaveletData(short *waveletData);
    short *getwaveletData();
    bool  isTrasformed();
    void  transform(int wawe_type);
```

```
    void  untransform(int wawe_type);
    short getWaveletOne(int x, int y);
    void  setWaveletOne(int x, int y, short w);
    bool  isMainSet(int xRoot, int yRoot, short mainBit);
    void  prepareMainValues();
    void  saveWaveletToFile(const char* FileName);
    void  saveMainValuesToFile(const char* FileName);
private:
    bool   is_wavelet;
    short*  data;
    short* wavelet;
    short* mainValues;
    short isMainSetRecursive(int xRoot, int yRoot);
    void  haarTransform();
    void  haarUntransform();
    void  int5Transform();
    void  int5Untransform();
    void  D8Transform();
    void  D8Untransform();
};

#endif // INTWAVELET_H
```

Содержимое файла intwavelet.cpp

```
#include "intwavelet.h"
#include <stdlib.h>
#include <fstream>
#include <cmath>

float DOB8[]={
     0.230377813309,
     0.714846570553,
     0.630880767930,
    -0.027983769417,
    -0.187034811719,
     0.030841381836,
     0.032883011667,
    -0.010597401785
};

IntWavelet::IntWavelet()
{
    is_wavelet = false;
    data = NULL;
    wavelet = NULL;
    mainValues = NULL;
    QuantL = 1;
    QuantH = 1;
}

IntWavelet::~IntWavelet()
{
    //if(data!=NULL) delete data;
    //if(wavelet!=NULL) delete wavelet;
    if( mainValues!=NULL ) delete mainValues;
}

void  IntWavelet::setImageData(short *imageData)
{
    data = imageData;
}

short* IntWavelet::getImageData()
{
    return data;
}

void  IntWavelet::setwaveletData(short int* waveletData)
{
    wavelet = waveletData;
}

short*  IntWavelet::getwaveletData()
{
    return wavelet;
}

bool  IntWavelet::isTrasformed()
```

```
{
    return (wavelet != NULL);
}

void  IntWavelet::transform(int wawe_type)
{
    switch (wawe_type) {
    case WAVELET_HAAR:
        haarTransform();
        break;
    case WAVELET_INT5:
        int5Transform();
        break;
    case WAVELET_DOB8:
        D8Transform();
        break;
    default:
        break;
    }
}

void  IntWavelet::untransform(int wawe_type)
{
    switch (wawe_type) {
    case WAVELET_HAAR:
        haarUntransform();
        break;
    case WAVELET_INT5:
        int5Untransform();
        break;
    case WAVELET_DOB8:
        D8Untransform();
        break;
    default:
        break;
    }
}

void  IntWavelet::haarTransform()
{
    int p = 512;
    int i = 0;
    int j = 0;
    short int L[256];
    short int H[256];
    //copy data to wavelet
    for(i=0;i<512*512;i++)
    {
        wavelet[i] = data[i];
    }

    while(p>1)
    {
        //horizontal transform each line
        for(i=0;i<p*512;i+=512)
        {
            //horizontal
            for(j=0;j<p/2;j++)
            {
                H[j] = wavelet[i+2*j+1]-wavelet[i+2*j];
                L[j] = wavelet[i+2*j] + H[j]/2;
            }
            for(j=0;j<p/2;j++)
            {
                wavelet[i+j] = L[j];// / QuantL;
                wavelet[i+p/2+j] = H[j];// / QuantH;
            }
        }
        //vertical transform each collum
        for(i=0;i<p;i++)
        {
            for(j=0;j<p/2;j++)
            {
                H[j] = wavelet[i+(2*j+1)*512]-wavelet[i+(2*j)*512];
                L[j] = wavelet[i+(2*j)*512] + H[j]/2;
            }
            //lazy to haar
            for(j=0;j<p/2;j++)
            {
```

```
                wavelet[i+j*512] = L[j];// / QuantL;
                wavelet[i+(p/2+j)*512] = H[j];// / QuantH;
            }
        }
        p/=2;
    }
}

void  IntWavelet::haarUntransform()
{
    int p = 2;
    int i = 0;
    int j = 0;
    short L[256];
    short H[256];

    while(p<=512)
    {
        //vertical untransform each collum
        for(i=0;i<p;i++)
        {
            for(j=0;j<p/2;j++)
            {
                L[j] = wavelet[i+j*512];//*QuantL;
                H[j] = wavelet[i+(p/2+j)*512];//*QuantH;
            }
            //lazy to haar
            for(j=0;j<p/2;j++)
            {
                wavelet[i+(2*j)*512] = L[j] - H[j]/2;
                wavelet[i+(2*j+1)*512] = H[j] + wavelet[i+(2*j)*512];

            }
        }
        //horizontal untransform each line

        for(i=0;i<p*512;i+=512)
        {
            //horizontal
            for(j=0;j<p/2;j++)
            {
                L[j] = wavelet[i+j];//*QuantL;
                H[j] = wavelet[i+p/2+j];//*QuantH;
            }
            for(j=0;j<p/2;j++)
            {
                wavelet[i+2*j] = L[j] - H[j]/2;
                wavelet[i+2*j+1] = H[j] + wavelet[i+2*j];
            }
        }

        p*=2;
    }
    //copy to data
    for(i=0;i<512*512;i++)
    {   // Need only use for discrete coefficients
        //if(wavelet[i]<0){ data[i] = 0; }else
          //if(wavelet[i]>255){ data[i] = 255; }else
            {data[i] = wavelet[i];}
    }

}

void  IntWavelet::int5Transform()
{
    /*
     (  0 -1  2 -1  0 )/2 Hi
     ( -1  2  6  2 -1 )/8 Lo
    */
    int p = 512;
    int i = 0;
    int j = 0;
    short L[256];
    short H[256];
    //copy data to wavelet
    for(i=0;i<512*512;i++)
    {
        wavelet[i] = data[i];
    }
```

```
    while(p>2)
    {
        //horizontal transform each line
        for(i=0;i<p*512;i+=512)
        {
            //horizontal
            //Hi(i) = x(2i+1)-(x(2i)+(x(2i+2))/2
            for(j=0;j<p/2-1;j++)
            {
                H[j] = wavelet[i+2*j+1]-(wavelet[i+2*j]+wavelet[i+2*j+2])/2;
            }
            H[j] = wavelet[i+2*j+1]-wavelet[i+2*j];
            //Lo(i) = x(2i) + (Hi(i-1)+Hi(i))/4
            L[0] = wavelet[i] + H[0]/2;
            for(j=1;j<p/2;j++)
            {
                L[j] = wavelet[i+2*j] + (H[j-1]+H[j])/4;
            }


            for(j=0;j<p/2;j++)
            {
                wavelet[i+j] = L[j];//QuantL;
                wavelet[i+p/2+j] = H[j];//QuantH;
            }
        }
        //vertical transform each collum
        for(i=0;i<p;i++)
        {
            //Hi(i) = x(2i+1)-(x(2i)+(x(2i+2))/2
            for(j=0;j<p/2-1;j++)
            {
                H[j] = wavelet[i+(2*j+1)*512]-(wavelet[i+(2*j)*512]+wavelet[i+(2*j+2)*512])/2;
            }
            H[j] = wavelet[i+(2*j+1)*512]-wavelet[i+(2*j)*512];

            L[0] = wavelet[i] + H[0]/2;
            for(j=1;j<p/2;j++)
            {
                L[j] = wavelet[i+(2*j)*512] + (H[j-1]+H[j])/4;
            }

            for(j=0;j<p/2;j++)
            {
                wavelet[i+j*512] = L[j];//QuantL;
                wavelet[i+(p/2+j)*512] = H[j];//QuantH;
            }
        }
        p/=2;
    }
}

void  IntWavelet::int5Untransform()
{
    int p = 4;
    int i = 0;
    int j = 0;
    short L[256];
    short H[256];

    while(p<=512)
    {
        //vertical untransform each collum
        for(i=0;i<p;i++)
        {
            for(j=0;j<p/2;j++)
            {
                L[j] = wavelet[i+j*512];//*QuantL;
                H[j] = wavelet[i+(p/2+j)*512];//*QuantH;
            }
            wavelet[i] = L[0] - H[0]/2;
            for(j=1;j<p/2;j++)
            {
                wavelet[i+(2*j)*512] = L[j] - (H[j-1]+H[j])/4;
            }
            for(j=0;j<p/2-1;j++)
            {
                wavelet[i+(2*j+1)*512] = H[j] + (wavelet[i+(2*j)*512]+wavelet[i+(2*j+2)*512])/2;
```

```
            }
            wavelet[i+(2*j+1)*512] = H[j] + wavelet[i+(2*j)*512];
        }
        //horizontal untransform each line

        for(i=0;i<p*512;i+=512)
        {
            //horizontal
            for(j=0;j<p/2;j++)
            {
                L[j] = wavelet[i+j];//*QuantL;
                H[j] = wavelet[i+p/2+j];//*QuantH;
            }
            //Lo(i) = x(2i) + (Hi(i-1)+Hi(i))/4
            wavelet[i] = L[0] - H[0]/2;
            for(j=1;j<p/2;j++)
            {
                wavelet[i+2*j] = L[j] - (H[j-1]+H[j])/4;
            }
            for(j=0;j<p/2-1;j++)
            {
                wavelet[i+2*j+1] = H[j] + (wavelet[i+2*j]+wavelet[i+2*j+2])/2;
            }
            wavelet[i+2*j+1] = H[j] + wavelet[i+2*j];
        }

        p*=2;
    }
    //copy to data
    for(i=0;i<512*512;i++)
    {   // Need only use for discrete coefficients
        //if(wavelet[i]<0){ data[i] = 0; }else
          //if(wavelet[i]>255){ data[i] = 255; }else
            {data[i] = wavelet[i];}
    }

}

void  IntWavelet::D8Transform()
{
    int p = 512;
    int i = 0;
    int j = 0;
    short L[256];
    short H[256];
    //copy data to wavelet
    for(i=0;i<512*512;i++)
    {
        wavelet[i] = data[i];
    }

    while(p>8)
    {
        //horizontal transform each line
        for(i=0;i<p*512;i+=512)
        {
            //horizontal
            for(j=0;j<p/2;j++)
            {
                L[j] = round( DOB8[0]*wavelet[i+(2*j+0)%p]+DOB8[1]*wavelet[i+(2*j+1)%p]+
                              DOB8[2]*wavelet[i+(2*j+2)%p]+DOB8[3]*wavelet[i+(2*j+3)%p]+
                              DOB8[4]*wavelet[i+(2*j+4)%p]+DOB8[5]*wavelet[i+(2*j+5)%p]+
                              DOB8[6]*wavelet[i+(2*j+6)%p]+DOB8[7]*wavelet[i+(2*j+7)%p]);
                H[j] = round( DOB8[7]*wavelet[i+(2*j+0)%p]-DOB8[6]*wavelet[i+(2*j+1)%p]+
                              DOB8[5]*wavelet[i+(2*j+2)%p]-DOB8[4]*wavelet[i+(2*j+3)%p]+
                              DOB8[3]*wavelet[i+(2*j+4)%p]-DOB8[2]*wavelet[i+(2*j+5)%p]+
                              DOB8[1]*wavelet[i+(2*j+6)%p]-DOB8[0]*wavelet[i+(2*j+7)%p]);
            }
            for(j=0;j<p/2;j++)
            {
                wavelet[i+j] = L[j];//QuantL;
                wavelet[i+p/2+j] = H[j];//QuantH;
            }
        }
        //vertical transform each collum
        for(i=0;i<p;i++)
        {
            for(j=0;j<p/2;j++)
            {
```

```
                L[j] = round( DOB8[0]*wavelet[i+512*((2*j+0)%p)]+DOB8[1]*wavelet[i+512*((2*j+1)%p)]+
                              DOB8[2]*wavelet[i+512*((2*j+2)%p)]+DOB8[3]*wavelet[i+512*((2*j+3)%p)]+
                              DOB8[4]*wavelet[i+512*((2*j+4)%p)]+DOB8[5]*wavelet[i+512*((2*j+5)%p)]+

DOB8[6]*wavelet[i+512*((2*j+6)%p)]+DOB8[7]*wavelet[i+512*((2*j+7)%p)]);
                H[j] = round( DOB8[7]*wavelet[i+512*((2*j+0)%p)]-DOB8[6]*wavelet[i+512*((2*j+1)%p)]+
                              DOB8[5]*wavelet[i+512*((2*j+2)%p)]-DOB8[4]*wavelet[i+512*((2*j+3)%p)]+
                              DOB8[3]*wavelet[i+512*((2*j+4)%p)]-DOB8[2]*wavelet[i+512*((2*j+5)%p)]+
                              DOB8[1]*wavelet[i+512*((2*j+6)%p)]-
DOB8[0]*wavelet[i+512*((2*j+7)%p)]);
            }
            for(j=0;j<p/2;j++)
            {
                wavelet[i+j*512] = L[j];//QuantL;
                wavelet[i+(p/2+j)*512] = H[j];//QuantH;
            }
        }
        p/=2;
    }
}

void  IntWavelet::D8Untransform()
{
    int p = 16;
    int i = 0;
    int j = 0;
    short L[256];
    short H[256];

    while(p<=512)
    {
        //vertical untransform each collum
        //A0 = h6*L-3+h1*H-3+h4*L-2+h3*H-2+h2*L-1+h5*H-1+h0*L0+h7*H0
        //A1 =
        //A2 = h6*L-2+h1*H-2+h4*L-1+h3*H-1+h2*L0+h5*H0+h0*L1+h7*H1
        //A3 =
        //A4 = h6*L-1+h1*H-1+h4*L0+h3*H0+h2*L1+h5*H1+h0*L2+h7*H2
        //A5 =
        //A6 = h6*L0+h1*H0+h4*L1+h3*H1+h2*L2+h5*H2+h0*L3+h7*H3
        //A7 = h7*L0-h0*H0+h5*L1-h2*H1+h3*L2-h4*H2+h1*L3-h6*H3

        for(i=0;i<p;i++)
        {
            for(j=0;j<p/2;j++)
            {
                L[j]=wavelet[i+j*512];//QuantL;
                H[j]=wavelet[i+(p/2+j)*512];//QuantH;
            }
            for(j=0;j<p/2;j++)
            {
                wavelet[i+(2*j)*512] =
                              round( DOB8[6]*L[(p/2+j-3)%(p/2)]+DOB8[1]*H[(p/2+j-3)%(p/2)]+
                                     DOB8[4]*L[(p/2+j-2)%(p/2)]+DOB8[3]*H[(p/2+j-2)%(p/2)]+
                                     DOB8[2]*L[(p/2+j-1)%(p/2)]+DOB8[5]*H[(p/2+j-1)%(p/2)]+
                                     DOB8[0]*L[(p/2+j  )%(p/2)]+DOB8[7]*H[(p/2+j  )%(p/2)]);
                wavelet[i+(2*j+1)*512]=
                              round( DOB8[7]*L[(p/2+j-3)%(p/2)]-DOB8[0]*H[(p/2+j-3)%(p/2)]+
                                     DOB8[5]*L[(p/2+j-2)%(p/2)]-DOB8[2]*H[(p/2+j-2)%(p/2)]+
                                     DOB8[3]*L[(p/2+j-1)%(p/2)]-DOB8[4]*H[(p/2+j-1)%(p/2)]+
                                     DOB8[1]*L[(p/2+j  )%(p/2)]-DOB8[6]*H[(p/2+j  )%(p/2)]);
            }

        }
        //horizontal untransform each line
        for(i=0;i<p*512;i+=512)
        {
            //horizontal
            for(j=0;j<p/2;j++)
            {
                L[j] = wavelet[i+j];//QuantL;
                H[j] = wavelet[i+p/2+j];//QuantH;
            }
            for(j=0;j<p/2;j++)
            {
                wavelet[i+(2*j)] =
                              round( DOB8[6]*L[(p/2+j-3)%(p/2)]+DOB8[1]*H[(p/2+j-3)%(p/2)]+
                                     DOB8[4]*L[(p/2+j-2)%(p/2)]+DOB8[3]*H[(p/2+j-2)%(p/2)]+
                                     DOB8[2]*L[(p/2+j-1)%(p/2)]+DOB8[5]*H[(p/2+j-1)%(p/2)]+
                                     DOB8[0]*L[(p/2+j  )%(p/2)]+DOB8[7]*H[(p/2+j  )%(p/2)]);
```

```
                wavelet[i+(2*j+1)] =
                                round( DOB8[7]*L[(p/2+j-3)%(p/2)]-DOB8[0]*H[(p/2+j-3)%(p/2)]+
                                       DOB8[5]*L[(p/2+j-2)%(p/2)]-DOB8[2]*H[(p/2+j-2)%(p/2)]+
                                       DOB8[3]*L[(p/2+j-1)%(p/2)]-DOB8[4]*H[(p/2+j-1)%(p/2)]+
                                       DOB8[1]*L[(p/2+j  )%(p/2)]-DOB8[6]*H[(p/2+j  )%(p/2)]);
            }
        }
        p*=2;
    }
    //copy to data
    for(i=0;i<512*512;i++)
    {   // Need only use for rounded coefficients
        //if(wavelet[i]<0){ data[i] = 0; }else
          //if(wavelet[i]>255){ data[i] = 255; }else
            {data[i] = wavelet[i];}
    }
}

short  IntWavelet::getWaveletOne(int x, int y)
{
    if( wavelet==NULL ) return 0;
    return wavelet[y*512+x];
}

void  IntWavelet::setWaveletOne(int x, int y, short w)
{
    if( wavelet==NULL ) return;
    wavelet[y*512+x]=w;
}

bool IntWavelet::isMainSet(int xRoot, int yRoot, short mainBit)
{
    if( wavelet==NULL ) return false;
    if( mainValues==NULL ) prepareMainValues();
    return (mainValues[yRoot*512+xRoot]>=mainBit);
}

short IntWavelet::isMainSetRecursive(int xRoot, int yRoot)
{
    if( xRoot==0 && yRoot==0 ) return mainValues[0];
    if( xRoot>255 || yRoot>255 )
    {
        return mainValues[yRoot*512+xRoot];
    }
    mainValues[xRoot+512*yRoot]           |=           isMainSetRecursive(2*xRoot,2*yRoot)           |
isMainSetRecursive(2*xRoot+1,2*yRoot) |
            isMainSetRecursive(2*xRoot,2*yRoot+1) | isMainSetRecursive(2*xRoot+1,2*yRoot+1);
    return mainValues[xRoot+512*yRoot];
}

void  IntWavelet::prepareMainValues()
{
    if( wavelet==NULL ) return;
    int x,y;
    if( mainValues==NULL ) mainValues = new short[512*512];
    for(y=0;y<512*512;y+=512)
    {
        for(x=0;x<512;x++)
        {
            mainValues[x+y] = abs(wavelet[x+y]);
        }
    }
    //int p=256;
    isMainSetRecursive(0,1);
    isMainSetRecursive(1,1);
    isMainSetRecursive(1,0);
}

void  IntWavelet::saveWaveletToFile(const char* FileName)
{
    if( wavelet==NULL ) return;
    int x,y;
    std::ofstream outFile;
    outFile.open(FileName,std::ios_base::out | std::ios_base::trunc);
    if( outFile.is_open()==true )
    {
        outFile << "Wavelet coefficients.\n";
        for(y=0;y<512*512;y+=512)
        {
```

```
            for(x=0;x<512;x++)
            {
                outFile << wavelet[x+y] << " \t ";
            }
            outFile << "\n";
        }
        outFile.close();
    }
}

void  IntWavelet::saveMainValuesToFile(const char* FileName)
{
    if( mainValues==NULL ) return;
    int x,y;
    std::ofstream outFile;
    outFile.open(FileName,std::ios_base::out | std::ios_base::trunc);
    if( outFile.is_open()==true )
    {
        outFile << "mainValues coefficients.\n";
        for(y=0;y<512*512;y+=512)
        {
            for(x=0;x<512;x++)
            {
                outFile << mainValues[x+y] << " \t ";
            }
            outFile << "\n";
        }
        outFile.close();
    }
}
```

Содержимое файла spihtcode.h

```
#ifndef SPIHTCODE_H
#define SPIHTCODE_H

#include "intwavelet.h"
#include "imagebylayer.h"
#include "bitfile.h"

class SpihtCode
{
public:
    SpihtCode();
    ~SpihtCode();
    void setBMPname(const char* bmpFileName);
    void setSPIHTname(const char* spihtFileName);
    void transformToSPIHT(double compressRatio, int WaveletType, const char* newSPIHTname);
    void transformToBMP(double compressRatio, int WaveletType, const char* newBMPname);
    void transformToSPIHT(double compressRatio, int WaveletType);
    void transformToBMP(double compressRatio, int WaveletType);
    void saveBMPas(const char* newBMPname);
    void saveSPIHTas(const char* newSpihtName);
    void transformToSPIHT_O(double compressRatio, int WaveletType);
    void transformToBMP_O(double compressRatio, int WaveletType);
private:
    char* bmpName;
    char* spihtName;
    int   waveletType;
    IntWavelet WaveletCoderY;
    IntWavelet WaveletCoderCg;
    IntWavelet WaveletCoderCo;
    ImageByLayer BMPimage;
    BitFile SPIHTfile;
};

#endif // SPIHTCODE_H
```

Содержимое файла spihtcode.cpp

```
#include <cstring>
#include <stdlib.h>
#include <list>
#include "spihtcode.h"

/*
    char* bmpName;
    char* spihtName;
    int   waveletType;
```

```
    IntWavelet WaveletCoder;
    ImageByLayer BMPimage;
    BitFile SPIHTfile;
*/

struct PointRecord
{
    int x,y;
    //short Layer;
};

SpihtCode::SpihtCode()
{
    bmpName = NULL;
    spihtName = NULL;
}

SpihtCode::~SpihtCode()
{
    if(bmpName!=NULL) delete bmpName;
    if(spihtName!=NULL) delete spihtName;
}

void SpihtCode::setBMPname(const char* bmpFileName)
{
    if(bmpName!=NULL) delete bmpName;
    bmpName = new char[strlen(bmpFileName)+1];
    strcpy(bmpName,bmpFileName);
}

void SpihtCode::setSPIHTname(const char* spihtFileName)
{
    if(spihtName!=NULL) delete spihtName;
    spihtName = new char[strlen(spihtFileName)+1];
    strcpy(spihtName,spihtFileName);
}

void SpihtCode::transformToSPIHT(double compressRatio, int WaveletType, const char* newSPIHTname)
{
    setSPIHTname(newSPIHTname);
    transformToSPIHT(compressRatio,WaveletType);
}

void SpihtCode::transformToBMP(double compressRatio, int WaveletType, const char* newBMPname)
{
    setBMPname(newBMPname);
    transformToBMP(compressRatio, WaveletType);
}

void SpihtCode::transformToSPIHT(double compressRatio, int WaveletType)
{
    if(bmpName==NULL || spihtName==NULL) return;
    BMPimage.ReadImageFromFile(bmpName);
    BMPimage.RGB_To_Ygo();

    //BMPimage.WriteImageToFile(spihtName);
    //return;

    short* waveletY  = new short[512*512];
    short* waveletCg = new short[512*512];
    short* waveletCo = new short[512*512];

    WaveletCoderCg.setImageData(BMPimage.getData(0));
    WaveletCoderY.setImageData(BMPimage.getData(1));
    WaveletCoderCo.setImageData(BMPimage.getData(2));

    WaveletCoderCg.setwaveletData(waveletCg);
    WaveletCoderY.setwaveletData(waveletY);
    WaveletCoderCo.setwaveletData(waveletCo);
    WaveletCoderY.transform(WaveletType);
    WaveletCoderY.prepareMainValues();
    WaveletCoderCg.transform(WaveletType);
    WaveletCoderCg.prepareMainValues();
    WaveletCoderCo.transform(WaveletType);
    WaveletCoderCo.prepareMainValues();

    if(compressRatio>0.5)
    {
        WaveletCoderY.QuantL = 1;
```

```
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }else
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }

    //WaveletCoderY.saveWaveletToFile("Wavelet.txt");
    //WaveletCoderY.saveMainValuesToFile("ManValues.txt");


    /*unsigned char* R = (unsigned char*)BMPimage.getData(0);
    unsigned char* G = (unsigned char*)BMPimage.getData(1);
    unsigned char* B = (unsigned char*)BMPimage.getData(2);
    for(int y=0;y<512;y++)
    {
        for(int x=0;x<512;x++)
        {
            R[y*512+x] = (WaveletCoderCg.getWaveletOne(x,y)+256)%256;
            G[y*512+x] = (WaveletCoderY.getWaveletOne(x,y)+256)%256;
            B[y*512+x] = (WaveletCoderCo.getWaveletOne(x,y)+256)%256;
        }
    }
    BMPimage.WriteImageToFile(spihtName);
    return;*/

    if(SPIHTfile.isOpen()==true) SPIHTfile.close();
    SPIHTfile.setNameFile(spihtName);
    SPIHTfile.openForWriteOnly();
    int WantFileSize = (int)(compressRatio * (512*512*3)); //want size in bits, set compressRatio>1
for lossless compress
    short BitLayer = 0x4000;
    std::list<PointRecord> listIsetsY;    //insignificant sets
    std::list<PointRecord> listIpointsY;  //insignificant points
    std::list<PointRecord> listSpointsY;  //significant points

    std::list<PointRecord> listIsetsCg;    //insignificant sets
    std::list<PointRecord> listIpointsCg;  //insignificant points
    std::list<PointRecord> listSpointsCg;  //significant points

    std::list<PointRecord> listIsetsCo;    //insignificant sets
    std::list<PointRecord> listIpointsCo;  //insignificant points
    std::list<PointRecord> listSpointsCo;  //significant points

    PointRecord P;
    P.x = 0;
    P.y = 0;
    listIpointsY.push_back(P);
    listIpointsCg.push_back(P);
    listIpointsCo.push_back(P);
    P.x = 1;
    P.y = 0;
    listIsetsY.push_back(P);
    listIsetsCg.push_back(P);
    listIsetsCo.push_back(P);
    P.x = 0;
    P.y = 1;
    listIsetsY.push_back(P);
    listIsetsCg.push_back(P);
    listIsetsCo.push_back(P);
    P.x = 1;
    P.y = 1;
    listIsetsY.push_back(P);
    listIsetsCg.push_back(P);
    listIsetsCo.push_back(P);

    std::list<PointRecord>::iterator it;
    while( SPIHTfile.getSize()<WantFileSize && BitLayer>1)
    {
        BitLayer = BitLayer>>1;
        // send bits of man pixels
```

```
for(it=listSpointsY.begin(); it!=listSpointsY.end(); it++)
{
    if( (abs(WaveletCoderY.getWaveletOne((*it).x,(*it).y)) & BitLayer)!=0 )
    {
        SPIHTfile.writeBit(1);
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listSpointsCg.begin(); it!=listSpointsCg.end(); it++)
{
    if( (abs(WaveletCoderCg.getWaveletOne((*it).x,(*it).y)) & BitLayer)!=0 )
    {
        SPIHTfile.writeBit(1);
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listSpointsCo.begin(); it!=listSpointsCo.end(); it++)
{
    if( (abs(WaveletCoderCo.getWaveletOne((*it).x,(*it).y)) & BitLayer)!=0 )
    {
        SPIHTfile.writeBit(1);
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
//Check puuu sets
for(it=listIsetsY.begin(); it!=listIsetsY.end(); it++)
{
    if( WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        SPIHTfile.writeBit(1);
        listIpointsY.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsY.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsY.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsY.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsY.push_back(P);
        }
        it = listIsetsY.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
```

```
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIsetsCg.begin(); it!=listIsetsCg.end(); it++)
{
    if( WaveletCoderCg.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        SPIHTfile.writeBit(1);
        listIpointsCg.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
        }
        it = listIsetsCg.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIsetsCo.begin(); it!=listIsetsCo.end(); it++)
{
    if( WaveletCoderCo.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        SPIHTfile.writeBit(1);
        listIpointsCo.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCo.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
```

```
            listIpointsCo.push_back(P);
        }
        it = listIsetsCo.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
// check puuu pixels
for(it=listIpointsY.begin(); it!=listIpointsY.end(); it++)
{
    if( abs(WaveletCoderY.getWaveletOne((*it).x,(*it).y))>=BitLayer )
    {
        SPIHTfile.writeBit(1);
        if(WaveletCoderY.getWaveletOne((*it).x,(*it).y)>0)
        {
            SPIHTfile.writeBit(1);
        }else
        {
            SPIHTfile.writeBit(0);
        }
        listSpointsY.push_back(*it);
        it = listIpointsY.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIpointsCg.begin(); it!=listIpointsCg.end(); it++)
{
    if( abs(WaveletCoderCg.getWaveletOne((*it).x,(*it).y))>=BitLayer )
    {
        SPIHTfile.writeBit(1);
        if(WaveletCoderCg.getWaveletOne((*it).x,(*it).y)>0)
        {
            SPIHTfile.writeBit(1);
        }else
        {
            SPIHTfile.writeBit(0);
        }
        listSpointsCg.push_back(*it);
        it = listIpointsCg.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIpointsCo.begin(); it!=listIpointsCo.end(); it++)
{
    if( abs(WaveletCoderCo.getWaveletOne((*it).x,(*it).y))>=BitLayer )
    {
        SPIHTfile.writeBit(1);
        if(WaveletCoderCo.getWaveletOne((*it).x,(*it).y)>0)
        {
            SPIHTfile.writeBit(1);
        }else
        {
            SPIHTfile.writeBit(0);
        }
        listSpointsCo.push_back(*it);
        it = listIpointsCo.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
```

```
    }

    listIpointsY.clear();
    listIsetsY.clear();
    listSpointsY.clear();
    listIpointsCg.clear();
    listIsetsCg.clear();
    listSpointsCg.clear();
    listIpointsCo.clear();
    listIsetsCo.clear();
    listSpointsCo.clear();

    SPIHTfile.close();

    WaveletCoderY.setImageData(NULL);
    WaveletCoderCg.setImageData(NULL);
    WaveletCoderCo.setImageData(NULL);
    WaveletCoderY.setwaveletData(NULL);
    WaveletCoderCg.setwaveletData(NULL);
    WaveletCoderCo.setwaveletData(NULL);

    delete waveletY;
    delete waveletCg;
    delete waveletCo;
}

void SpihtCode::transformToBMP(double compressRatio, int WaveletType)
{
    if(bmpName==NULL || spihtName==NULL) return;

    BMPimage.NewImage();
    short* waveletY  = new short[512*512];
    short* waveletCg = new short[512*512];
    short* waveletCo = new short[512*512];

    // 0 0 1 1 2 2
    // 1 2 0 2 0 1
    // 2 1 2 0 1 0
    WaveletCoderCg.setImageData(BMPimage.getData(0));
    WaveletCoderY.setImageData(BMPimage.getData(1));
    WaveletCoderCo.setImageData(BMPimage.getData(2));
    WaveletCoderY.setwaveletData(waveletY);
    WaveletCoderCg.setwaveletData(waveletCg);
    WaveletCoderCo.setwaveletData(waveletCo);

    if(compressRatio>0.5)
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }else
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }

    if(SPIHTfile.isOpen()==true) SPIHTfile.close();
    SPIHTfile.setNameFile(spihtName);
    SPIHTfile.openForReadOnly();
    //int  WantFileSize  =  (int)(compressRatio  *  (512*512*3*8));  //want  size  in  bits,  set
compressRatio>1 for lossless compress
    //int FileSize = 0;
    short BitLayer = 0x4000;
    std::list<PointRecord> listIsetsY;    //insignificant sets
    std::list<PointRecord> listIpointsY;  //insignificant points
    std::list<PointRecord> listSpointsY;  //significant points

    std::list<PointRecord> listIsetsCg;    //insignificant sets
    std::list<PointRecord> listIpointsCg;  //insignificant points
    std::list<PointRecord> listSpointsCg;  //significant points
```

```
std::list<PointRecord> listIsetsCo;    //insignificant sets
std::list<PointRecord> listIpointsCo;  //insignificant points
std::list<PointRecord> listSpointsCo;  //significant points

PointRecord P;
P.x = 0;
P.y = 0;
listIpointsY.push_back(P);
listIpointsCg.push_back(P);
listIpointsCo.push_back(P);
P.x = 1;
P.y = 0;
listIsetsY.push_back(P);
listIsetsCg.push_back(P);
listIsetsCo.push_back(P);
P.x = 0;
P.y = 1;
listIsetsY.push_back(P);
listIsetsCg.push_back(P);
listIsetsCo.push_back(P);
P.x = 1;
P.y = 1;
listIsetsY.push_back(P);
listIsetsCg.push_back(P);
listIsetsCo.push_back(P);

std::list<PointRecord>::iterator it;
short tmp;
while( BitLayer>1)
{
    BitLayer = BitLayer>>1;
    // send bits of man pixels
    for(it=listSpointsY.begin(); it!=listSpointsY.end(); it++)
    {
        if( SPIHTfile.readBit()==1 )
        {
            tmp = WaveletCoderY.getWaveletOne((*it).x,(*it).y);
            if(tmp>0)
            {
                tmp += BitLayer;
            }else
            {
                tmp -= BitLayer;
            }
            WaveletCoderY.setWaveletOne((*it).x,(*it).y,tmp);
        }
    }
    for(it=listSpointsCg.begin(); it!=listSpointsCg.end(); it++)
    {
        if( SPIHTfile.readBit()==1 )
        {
            tmp = WaveletCoderCg.getWaveletOne((*it).x,(*it).y);
            if(tmp>0)
            {
                tmp += BitLayer;
            }else
            {
                tmp -= BitLayer;
            }
            WaveletCoderCg.setWaveletOne((*it).x,(*it).y,tmp);
        }
    }
    for(it=listSpointsCo.begin(); it!=listSpointsCo.end(); it++)
    {
        if( SPIHTfile.readBit()==1 )
        {
            tmp = WaveletCoderCo.getWaveletOne((*it).x,(*it).y);
            if(tmp>0)
            {
                tmp += BitLayer;
            }else
            {
                tmp -= BitLayer;
            }
            WaveletCoderCo.setWaveletOne((*it).x,(*it).y,tmp);
        }
    }
    //Check puuu sets
    for(it=listIsetsY.begin(); it!=listIsetsY.end(); it++)
```

```
{
    if( SPIHTfile.readBit()==1 )// WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        listIpointsY.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsY.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsY.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsY.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsY.push_back(P);
        }
        it = listIsetsY.erase(it);
        it--;
    }
}
for(it=listIsetsCg.begin(); it!=listIsetsCg.end(); it++)
{
    if( SPIHTfile.readBit()==1 )// WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        listIpointsCg.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
        }
        it = listIsetsCg.erase(it);
        it--;
    }
}
for(it=listIsetsCo.begin(); it!=listIsetsCo.end(); it++)
{
    if( SPIHTfile.readBit()==1 )// WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
```

```
            listIpointsCo.push_back(*it);
            if( (*it).x*2<256 && (*it).y*2<256 )
            {
                P.x = (*it).x*2;
                P.y = (*it).y*2;
                listIsetsCo.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2;
                listIsetsCo.push_back(P);
                P.x = (*it).x*2;
                P.y = (*it).y*2+1;
                listIsetsCo.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2+1;
                listIsetsCo.push_back(P);
            }else
            {
                P.x = (*it).x*2;
                P.y = (*it).y*2;
                listIpointsCo.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2;
                listIpointsCo.push_back(P);
                P.x = (*it).x*2;
                P.y = (*it).y*2+1;
                listIpointsCo.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2+1;
                listIpointsCo.push_back(P);
            }
            it = listIsetsCo.erase(it);
            it--;
        }
    }

    // check puuu pixels
    for(it=listIpointsY.begin(); it!=listIpointsY.end(); it++)
    {
        if( SPIHTfile.readBit()==1 )
        {
            listSpointsY.push_back(*it);
            if( SPIHTfile.readBit()==1)
            {
                WaveletCoderY.setWaveletOne((*it).x,(*it).y,BitLayer);
            }else
            {
                WaveletCoderY.setWaveletOne((*it).x,(*it).y,-BitLayer);
            }
            it = listIpointsY.erase(it);
            it--;
        }
    }
    for(it=listIpointsCg.begin(); it!=listIpointsCg.end(); it++)
    {
        if( SPIHTfile.readBit()==1 )
        {
            listSpointsCg.push_back(*it);
            if( SPIHTfile.readBit()==1)
            {
                WaveletCoderCg.setWaveletOne((*it).x,(*it).y,BitLayer);
            }else
            {
                WaveletCoderCg.setWaveletOne((*it).x,(*it).y,-BitLayer);
            }
            it = listIpointsCg.erase(it);
            it--;
        }
    }
    for(it=listIpointsCo.begin(); it!=listIpointsCo.end(); it++)
    {
        if( SPIHTfile.readBit()==1 )
        {
            listSpointsCo.push_back(*it);
            if( SPIHTfile.readBit()==1)
            {
                WaveletCoderCo.setWaveletOne((*it).x,(*it).y,BitLayer);
            }else
            {
                WaveletCoderCo.setWaveletOne((*it).x,(*it).y,-BitLayer);
```

```
                }
                it = listIpointsCo.erase(it);
                it--;
            }
        }
    }

    listIpointsY.clear();
    listIsetsY.clear();
    listSpointsY.clear();
    listIpointsCg.clear();
    listIsetsCg.clear();
    listSpointsCg.clear();
    listIpointsCo.clear();
    listIsetsCo.clear();
    listSpointsCo.clear();

    SPIHTfile.close();

     /*unsigned char* R = (unsigned char*)BMPimage.getData(0);
     unsigned char* G = (unsigned char*)BMPimage.getData(1);
     unsigned char* B = (unsigned char*)BMPimage.getData(2);
     for(int y=0;y<512;y++)
     {
         for(int x=0;x<512;x++)
         {
             R[y*512+x] = (WaveletCoderCg.getWaveletOne(x,y)+256)%256;
             G[y*512+x] = (WaveletCoderY.getWaveletOne(x,y)+256)%256;
             B[y*512+x] = (WaveletCoderCo.getWaveletOne(x,y)+256)%256;
         }
     }
     BMPimage.WriteImageToFile(bmpName);
     return;*/

    //WaveletCoderY.saveWaveletToFile("_Wavelet.txt");
    //WaveletCoderY.prepareMainValues();
    //WaveletCoderY.saveMainValuesToFile("_ManValues.txt");

    WaveletCoderY.untransform(WaveletType);
    WaveletCoderCg.untransform(WaveletType);
    WaveletCoderCo.untransform(WaveletType);


    BMPimage.Ygo_To_RGB();
    BMPimage.WriteImageToFile(bmpName);

    WaveletCoderY.setImageData(NULL);
    WaveletCoderCg.setImageData(NULL);
    WaveletCoderCo.setImageData(NULL);
    WaveletCoderY.setwaveletData(NULL);
    WaveletCoderCg.setwaveletData(NULL);
    WaveletCoderCo.setwaveletData(NULL);
    delete waveletY;
    delete waveletCg;
    delete waveletCo;
}

void SpihtCode::saveBMPas(const char* newBMPname)
{
    BMPimage.WriteImageToFile(newBMPname);
}

void SpihtCode::saveSPIHTas(const char* newSpihtName)
{
    if( spihtName!=NULL ) delete spihtName;
    spihtName = new char[strlen(newSpihtName)+1];
    strcpy(spihtName,newSpihtName);
}

void SpihtCode::transformToSPIHT_O(double compressRatio, int WaveletType)
{
    if(bmpName==NULL || spihtName==NULL) return;
    BMPimage.ReadImageFromFile(bmpName);
    BMPimage.RGB_To_Ygo();

    //BMPimage.WriteImageToFile(spihtName);
    //return;

    short* waveletY  = new short[512*512];
```

```
    short* waveletCg = new short[512*512];
    short* waveletCo = new short[512*512];

    WaveletCoderCg.setImageData(BMPimage.getData(0));
    WaveletCoderY.setImageData(BMPimage.getData(1));
    WaveletCoderCo.setImageData(BMPimage.getData(2));

    WaveletCoderCg.setwaveletData(waveletCg);
    WaveletCoderY.setwaveletData(waveletY);
    WaveletCoderCo.setwaveletData(waveletCo);
    WaveletCoderY.transform(WaveletType);
    WaveletCoderY.prepareMainValues();
    WaveletCoderCg.transform(WaveletType);
    WaveletCoderCg.prepareMainValues();
    WaveletCoderCo.transform(WaveletType);
    WaveletCoderCo.prepareMainValues();

    if(compressRatio>0.5)
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }else
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }

    //WaveletCoderY.saveWaveletToFile("Wavelet.txt");
    //WaveletCoderY.saveMainValuesToFile("ManValues.txt");


    /*unsigned char* R = (unsigned char*)BMPimage.getData(0);
    unsigned char* G = (unsigned char*)BMPimage.getData(1);
    unsigned char* B = (unsigned char*)BMPimage.getData(2);
    for(int y=0;y<512;y++)
    {
        for(int x=0;x<512;x++)
        {
            R[y*512+x] = (WaveletCoderCg.getWaveletOne(x,y)+256)%256;
            G[y*512+x] = (WaveletCoderY.getWaveletOne(x,y)+256)%256;
            B[y*512+x] = (WaveletCoderCo.getWaveletOne(x,y)+256)%256;
        }
    }
    BMPimage.WriteImageToFile(spihtName);
    return;*/

    if(SPIHTfile.isOpen()==true) SPIHTfile.close();
    SPIHTfile.setNameFile(spihtName);
    SPIHTfile.openForWriteOnly();
    int WantFileSize = (int)(compressRatio * (512*512*3)); //want size in bits, set compressRatio>1
for lossless compress
    short BitLayer = 0x4000;
    std::list<PointRecord> listIsetsY;    //insignificant sets
    std::list<PointRecord> listIpointsY;  //insignificant points
    std::list<PointRecord> listSpointsY;  //significant points

    std::list<PointRecord> listIsetsCg;    //insignificant sets
    std::list<PointRecord> listIpointsCg;  //insignificant points
    std::list<PointRecord> listSpointsCg;  //significant points

    std::list<PointRecord> listIsetsCo;    //insignificant sets
    std::list<PointRecord> listIpointsCo;  //insignificant points
    std::list<PointRecord> listSpointsCo;  //significant points

    std::list<PointRecord> LayersY; // x - Layer, y - LayerCount
    std::list<PointRecord> LayersCg; // x - Layer, y - LayerCount
    std::list<PointRecord> LayersCo; // x - Layer, y - LayerCount

    PointRecord P, Layer;
    P.x = 0;
    P.y = 0;
```

```
listIpointsY.push_back(P);
listIpointsCg.push_back(P);
listIpointsCo.push_back(P);
P.x = 1;
P.y = 0;
listIsetsY.push_back(P);
listIsetsCg.push_back(P);
listIsetsCo.push_back(P);
P.x = 0;
P.y = 1;
listIsetsY.push_back(P);
listIsetsCg.push_back(P);
listIsetsCo.push_back(P);
P.x = 1;
P.y = 1;
listIsetsY.push_back(P);
listIsetsCg.push_back(P);
listIsetsCo.push_back(P);

std::list<PointRecord>::iterator it;
while( SPIHTfile.getSize()<WantFileSize && BitLayer>1)
{
    BitLayer = BitLayer>>1;

    // send bits of man pixels

    //Check puuu sets
    for(it=listIsetsY.begin(); it!=listIsetsY.end(); it++)
    {
        if( WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
        {
            SPIHTfile.writeBit(1);
            listIpointsY.push_back(*it);
            if( (*it).x*2<256 && (*it).y*2<256 )
            {
                P.x = (*it).x*2;
                P.y = (*it).y*2;
                listIsetsY.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2;
                listIsetsY.push_back(P);
                P.x = (*it).x*2;
                P.y = (*it).y*2+1;
                listIsetsY.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2+1;
                listIsetsY.push_back(P);
            }else
            {
                P.x = (*it).x*2;
                P.y = (*it).y*2;
                listIpointsY.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2;
                listIpointsY.push_back(P);
                P.x = (*it).x*2;
                P.y = (*it).y*2+1;
                listIpointsY.push_back(P);
                P.x = (*it).x*2+1;
                P.y = (*it).y*2+1;
                listIpointsY.push_back(P);
            }
            it = listIsetsY.erase(it);
            it--;
        }else
        {
            SPIHTfile.writeBit(0);
        }
        if( SPIHTfile.getSize()>=WantFileSize ) break;
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
    for(it=listIsetsCg.begin(); it!=listIsetsCg.end(); it++)
    {
        if( WaveletCoderCg.isMainSet((*it).x,(*it).y,BitLayer)==true )
        {
            SPIHTfile.writeBit(1);
            listIpointsCg.push_back(*it);
            if( (*it).x*2<256 && (*it).y*2<256 )
            {
```

```
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
        }
        it = listIsetsCg.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIsetsCo.begin(); it!=listIsetsCo.end(); it++)
{
    if( WaveletCoderCo.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        SPIHTfile.writeBit(1);
        listIpointsCo.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCo.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsCo.push_back(P);
        }
        it = listIsetsCo.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
```

```
if( SPIHTfile.getSize()>=WantFileSize ) break;

// check puuu pixels
Layer.x = BitLayer;
Layer.y = 0;
for(it=listIpointsY.begin(); it!=listIpointsY.end(); it++)
{
    if( abs(WaveletCoderY.getWaveletOne((*it).x,(*it).y))>=BitLayer )
    {
        SPIHTfile.writeBit(1);
        Layer.y++;
        if(WaveletCoderY.getWaveletOne((*it).x,(*it).y)>0)
        {
            SPIHTfile.writeBit(1);
        }else
        {
            SPIHTfile.writeBit(0);
        }
        listSpointsY.push_back(*it);
        it = listIpointsY.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
LayersY.push_back(Layer);

Layer.x = BitLayer;
Layer.y = 0;
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIpointsCg.begin(); it!=listIpointsCg.end(); it++)
{
    if( abs(WaveletCoderCg.getWaveletOne((*it).x,(*it).y))>=BitLayer )
    {
        SPIHTfile.writeBit(1);
        Layer.y++;
        if(WaveletCoderCg.getWaveletOne((*it).x,(*it).y)>0)
        {
            SPIHTfile.writeBit(1);
        }else
        {
            SPIHTfile.writeBit(0);
        }
        listSpointsCg.push_back(*it);
        it = listIpointsCg.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
    if( SPIHTfile.getSize()>=WantFileSize ) break;
}
LayersCg.push_back(Layer);

Layer.x = BitLayer;
Layer.y = 0;
if( SPIHTfile.getSize()>=WantFileSize ) break;
for(it=listIpointsCo.begin(); it!=listIpointsCo.end(); it++)
{
    if( abs(WaveletCoderCo.getWaveletOne((*it).x,(*it).y))>=BitLayer )
    {
        SPIHTfile.writeBit(1);
        Layer.y++;
        if(WaveletCoderCo.getWaveletOne((*it).x,(*it).y)>0)
        {
            SPIHTfile.writeBit(1);
        }else
        {
            SPIHTfile.writeBit(0);
        }
        listSpointsCo.push_back(*it);
        it = listIpointsCo.erase(it);
        it--;
    }else
    {
        SPIHTfile.writeBit(0);
    }
```

```
            if( SPIHTfile.getSize()>=WantFileSize ) break;
        }
        LayersCo.push_back(Layer);

        if( SPIHTfile.getSize()>=WantFileSize ) break;
    }

/*  BitLayer = 1;
    int K = 0;
    int col = 0;
    int Z = 0;
    std::list<PointRecord>::iterator pit;
    while( (SPIHTfile.getSize()<WantFileSize)&&(BitLayer<15) )
    {
        Z = 0;
        col = 0;
        pit = LayersY.begin();
        K = (*pit).x >> BitLayer;
        for(it=listSpointsY.begin();it!=listSpointsY.end();it++)
        {
            while( (*pit).y==col && pit!=LayersY.end() )
            {
                col = 0;
                pit++;
                K = (*pit).x >> BitLayer;
            }
            if(K==0) break;
            if( (abs(WaveletCoderY.getWaveletOne((*it).x,(*it).y)) & K)!=0 )
            {
                SPIHTfile.writeBit(1);
            }else
            {
                SPIHTfile.writeBit(0);
            }
            Z++;
            if( SPIHTfile.getSize()>=WantFileSize ) break;
            col++;
        }
        if( SPIHTfile.getSize()>=WantFileSize ) break;

        col = 0;
        pit = LayersCg.begin();
        K = (*pit).x >> BitLayer;
        for(it=listSpointsCg.begin();it!=listSpointsCg.end();it++)
        {
            while( (*pit).y==col && pit!=LayersCg.end() )
            {
                col = 0;
                pit++;
                K = (*pit).x >> BitLayer;
            }
            if(K==0) break;
            if( (abs(WaveletCoderCg.getWaveletOne((*it).x,(*it).y)) & K)!=0 )
            {
                SPIHTfile.writeBit(1);
            }else
            {
                SPIHTfile.writeBit(0);
            }
            if( SPIHTfile.getSize()>=WantFileSize ) break;
            col++;
        }
        if( SPIHTfile.getSize()>=WantFileSize ) break;

        col = 0;
        pit = LayersCo.begin();
        K = (*pit).x >> BitLayer;
        for(it=listSpointsCo.begin();it!=listSpointsCo.end();it++)
        {
            while( (*pit).y==col && pit!=LayersCo.end() )
            {
                col = 0;
                pit++;
                K = (*pit).x >> BitLayer;
            }
            if(K==0) break;
            if( (abs(WaveletCoderCo.getWaveletOne((*it).x,(*it).y)) & K)!=0 )
            {
                SPIHTfile.writeBit(1);
```

```
            }else
            {
                SPIHTfile.writeBit(0);
            }
            if( SPIHTfile.getSize()>=WantFileSize ) break;
            col++;
        }
        if( SPIHTfile.getSize()>=WantFileSize ) break;

        BitLayer++;
    }
*/
    LayersCg.clear();
    LayersCo.clear();
    LayersY.clear();

    listIpointsY.clear();
    listIsetsY.clear();
    listSpointsY.clear();
    listIpointsCg.clear();
    listIsetsCg.clear();
    listSpointsCg.clear();
    listIpointsCo.clear();
    listIsetsCo.clear();
    listSpointsCo.clear();

    SPIHTfile.close();

    WaveletCoderY.setImageData(NULL);
    WaveletCoderCg.setImageData(NULL);
    WaveletCoderCo.setImageData(NULL);
    WaveletCoderY.setwaveletData(NULL);
    WaveletCoderCg.setwaveletData(NULL);
    WaveletCoderCo.setwaveletData(NULL);

    delete waveletY;
    delete waveletCg;
    delete waveletCo;
}

void SpihtCode::transformToBMP_O(double compressRatio, int WaveletType)
{
    if(bmpName==NULL || spihtName==NULL) return;

    BMPimage.NewImage();
    short* waveletY  = new short[512*512];
    short* waveletCg = new short[512*512];
    short* waveletCo = new short[512*512];

    // 0 0 1 1 2 2
    // 1 2 0 2 0 1
    // 2 1 2 0 1 0
    WaveletCoderCg.setImageData(BMPimage.getData(0));
    WaveletCoderY.setImageData(BMPimage.getData(1));
    WaveletCoderCo.setImageData(BMPimage.getData(2));
    WaveletCoderY.setwaveletData(waveletY);
    WaveletCoderCg.setwaveletData(waveletCg);
    WaveletCoderCo.setwaveletData(waveletCo);

    if(compressRatio>0.5)
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }else
    {
        WaveletCoderY.QuantL = 1;
        WaveletCoderY.QuantH = 1;
        WaveletCoderCg.QuantL = 1;
        WaveletCoderCg.QuantH = 1;
        WaveletCoderCo.QuantL = 1;
        WaveletCoderCo.QuantH = 1;
    }

    if(SPIHTfile.isOpen()==true) SPIHTfile.close();
    SPIHTfile.setNameFile(spihtName);
```

```
    SPIHTfile.openForReadOnly();
    //int  WantFileSize  =  (int)(compressRatio  *  (512*512*3*8));  //want  size  in  bits,  set
compressRatio>1 for lossless compress
    //int FileSize = 0;
    short BitLayer = 0x4000;
    std::list<PointRecord> listIsetsY;    //insignificant sets
    std::list<PointRecord> listIpointsY;  //insignificant points
    std::list<PointRecord> listSpointsY;  //significant points

    std::list<PointRecord> listIsetsCg;    //insignificant sets
    std::list<PointRecord> listIpointsCg;  //insignificant points
    std::list<PointRecord> listSpointsCg;  //significant points

    std::list<PointRecord> listIsetsCo;    //insignificant sets
    std::list<PointRecord> listIpointsCo;  //insignificant points
    std::list<PointRecord> listSpointsCo;  //significant points

    std::list<PointRecord> LayersY; // x - Layer, y - LayerCount
    std::list<PointRecord> LayersCg; // x - Layer, y - LayerCount
    std::list<PointRecord> LayersCo; // x - Layer, y - LayerCount

    PointRecord P;
    PointRecord Layer;
    P.x = 0;
    P.y = 0;
    listIpointsY.push_back(P);
    listIpointsCg.push_back(P);
    listIpointsCo.push_back(P);
    P.x = 1;
    P.y = 0;
    listIsetsY.push_back(P);
    listIsetsCg.push_back(P);
    listIsetsCo.push_back(P);
    P.x = 0;
    P.y = 1;
    listIsetsY.push_back(P);
    listIsetsCg.push_back(P);
    listIsetsCo.push_back(P);
    P.x = 1;
    P.y = 1;
    listIsetsY.push_back(P);
    listIsetsCg.push_back(P);
    listIsetsCo.push_back(P);

    std::list<PointRecord>::iterator it;
    //short tmp;
    while( BitLayer>1)
    {
        BitLayer = BitLayer>>1;
        // send bits of man pixels

        //Check puuu sets
        for(it=listIsetsY.begin(); it!=listIsetsY.end(); it++)
        {
            if( SPIHTfile.readBit()==1 )// WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
            {
                listIpointsY.push_back(*it);
                if( (*it).x*2<256 && (*it).y*2<256 )
                {
                    P.x = (*it).x*2;
                    P.y = (*it).y*2;
                    listIsetsY.push_back(P);
                    P.x = (*it).x*2+1;
                    P.y = (*it).y*2;
                    listIsetsY.push_back(P);
                    P.x = (*it).x*2;
                    P.y = (*it).y*2+1;
                    listIsetsY.push_back(P);
                    P.x = (*it).x*2+1;
                    P.y = (*it).y*2+1;
                    listIsetsY.push_back(P);
                }else
                {
                    P.x = (*it).x*2;
                    P.y = (*it).y*2;
                    listIpointsY.push_back(P);
                    P.x = (*it).x*2+1;
                    P.y = (*it).y*2;
                    listIpointsY.push_back(P);
```

```
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsY.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsY.push_back(P);
        }
        it = listIsetsY.erase(it);
        it--;
    }
}
for(it=listIsetsCg.begin(); it!=listIsetsCg.end(); it++)
{
    if( SPIHTfile.readBit()==1 )// WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        listIpointsCg.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCg.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIpointsCg.push_back(P);
        }
        it = listIsetsCg.erase(it);
        it--;
    }
}
for(it=listIsetsCo.begin(); it!=listIsetsCo.end(); it++)
{
    if( SPIHTfile.readBit()==1 )// WaveletCoderY.isMainSet((*it).x,(*it).y,BitLayer)==true )
    {
        listIpointsCo.push_back(*it);
        if( (*it).x*2<256 && (*it).y*2<256 )
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIsetsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2+1;
            listIsetsCo.push_back(P);
        }else
        {
            P.x = (*it).x*2;
            P.y = (*it).y*2;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2+1;
            P.y = (*it).y*2;
            listIpointsCo.push_back(P);
            P.x = (*it).x*2;
            P.y = (*it).y*2+1;
            listIpointsCo.push_back(P);
```

```
                    P.x = (*it).x*2+1;
                    P.y = (*it).y*2+1;
                    listIpointsCo.push_back(P);
                }
                it = listIsetsCo.erase(it);
                it--;
            }
        }

        // check puuu pixels
        Layer.x = BitLayer;
        Layer.y = 0;
        for(it=listIpointsY.begin(); it!=listIpointsY.end(); it++)
        {
            if( SPIHTfile.readBit()==1 )
            {
                listSpointsY.push_back(*it);
                Layer.y++;
                if( SPIHTfile.readBit()==1)
                {
                    WaveletCoderY.setWaveletOne((*it).x,(*it).y,BitLayer+(BitLayer >> 2));
                }else
                {
                    WaveletCoderY.setWaveletOne((*it).x,(*it).y,-BitLayer-(BitLayer >> 2));
                }
                it = listIpointsY.erase(it);
                it--;
            }
        }
        LayersY.push_back(Layer);

        Layer.x = BitLayer;
        Layer.y = 0;
        for(it=listIpointsCg.begin(); it!=listIpointsCg.end(); it++)
        {
            if( SPIHTfile.readBit()==1 )
            {
                listSpointsCg.push_back(*it);
                Layer.y++;
                if( SPIHTfile.readBit()==1)
                {
                    WaveletCoderCg.setWaveletOne((*it).x,(*it).y,BitLayer+(BitLayer >> 2));
                }else
                {
                    WaveletCoderCg.setWaveletOne((*it).x,(*it).y,-BitLayer-(BitLayer >> 2));
                }
                it = listIpointsCg.erase(it);
                it--;
            }
        }
        LayersCg.push_back(Layer);

        Layer.x = BitLayer;
        Layer.y = 0;
        for(it=listIpointsCo.begin(); it!=listIpointsCo.end(); it++)
        {
            if( SPIHTfile.readBit()==1 )
            {
                listSpointsCo.push_back(*it);
                Layer.y++;
                if( SPIHTfile.readBit()==1)
                {
                    WaveletCoderCo.setWaveletOne((*it).x,(*it).y,BitLayer+(BitLayer >> 2));
                }else
                {
                    WaveletCoderCo.setWaveletOne((*it).x,(*it).y,-BitLayer-(BitLayer >> 2));
                }
                it = listIpointsCo.erase(it);
                it--;
            }
        }
        LayersCo.push_back(Layer);
    }
/*
    BitLayer = 1;
    int K = 0;
    int col = 0;
    int Z;
    std::list<PointRecord>::iterator pit;
```

```
while( BitLayer<15 )
{
    Z = 0;
    col = 0;
    pit = LayersY.begin();
    K = (*pit).x >> BitLayer;
    for(it=listSpointsY.begin();it!=listSpointsY.end();it++)
    {
        while( (*pit).y==col && pit!=LayersY.end() )
        {
            col = 0;
            pit++;
            K = (*pit).x >> BitLayer;
        }
        if(K==0) break;
        Z++;
        if( SPIHTfile.readBit()==1 )
        {
            tmp = WaveletCoderY.getWaveletOne((*it).x,(*it).y);
            if( tmp>0 )
            {
                tmp+=K;
            }else
            {
                tmp-=K;
            }
            WaveletCoderY.setWaveletOne((*it).x,(*it).y,tmp);
        }
        col++;
    }

    col = 0;
    pit = LayersCg.begin();
    K = (*pit).x >> BitLayer;
    for(it=listSpointsCg.begin();it!=listSpointsCg.end();it++)
    {
        while( (*pit).y==col && pit!=LayersCg.end() )
        {
            col = 0;
            pit++;
            K = (*pit).x >> BitLayer;
        }
        if(K==0) break;
        if( SPIHTfile.readBit()==1 )
        {
            tmp = WaveletCoderCg.getWaveletOne((*it).x,(*it).y);
            if( tmp>0 )
            {
                tmp+=K;
            }else
            {
                tmp-=K;
            }
            WaveletCoderCg.setWaveletOne((*it).x,(*it).y,tmp);
        }
        col++;
    }

    col = 0;
    pit = LayersCo.begin();
    K = (*pit).x >> BitLayer;
    for(it=listSpointsCo.begin();it!=listSpointsCo.end();it++)
    {
        while( (*pit).y==col && pit!=LayersCo.end() )
        {
            col = 0;
            pit++;
            K = (*pit).x >> BitLayer;
        }
        if(K==0) break;
        if( SPIHTfile.readBit()==1 )
        {
            tmp = WaveletCoderCo.getWaveletOne((*it).x,(*it).y);
            if( tmp>0 )
            {
                tmp+=K;
            }else
            {
                tmp-=K;
```

```
                }
                WaveletCoderCo.setWaveletOne((*it).x,(*it).y,tmp);
            }
            col++;
        }

        BitLayer++;
    }
*/
    LayersCg.clear();
    LayersCo.clear();
    LayersY.clear();


    listIpointsY.clear();
    listIsetsY.clear();
    listSpointsY.clear();
    listIpointsCg.clear();
    listIsetsCg.clear();
    listSpointsCg.clear();
    listIpointsCo.clear();
    listIsetsCo.clear();
    listSpointsCo.clear();

    SPIHTfile.close();

     /*unsigned char* R = (unsigned char*)BMPimage.getData(0);
     unsigned char* G = (unsigned char*)BMPimage.getData(1);
     unsigned char* B = (unsigned char*)BMPimage.getData(2);
     for(int y=0;y<512;y++)
     {
         for(int x=0;x<512;x++)
         {
             R[y*512+x] = (WaveletCoderCg.getWaveletOne(x,y)+256)%256;
             G[y*512+x] = (WaveletCoderY.getWaveletOne(x,y)+256)%256;
             B[y*512+x] = (WaveletCoderCo.getWaveletOne(x,y)+256)%256;
         }
     }
     BMPimage.WriteImageToFile(bmpName);
     return;*/

    //WaveletCoderY.saveWaveletToFile("_Wavelet.txt");
    //WaveletCoderY.prepareMainValues();
    //WaveletCoderY.saveMainValuesToFile("_ManValues.txt");

    WaveletCoderY.untransform(WaveletType);
    WaveletCoderCg.untransform(WaveletType);
    WaveletCoderCo.untransform(WaveletType);


    BMPimage.Ygo_To_RGB();
    BMPimage.WriteImageToFile(bmpName);

    WaveletCoderY.setImageData(NULL);
    WaveletCoderCg.setImageData(NULL);
    WaveletCoderCo.setImageData(NULL);
    WaveletCoderY.setwaveletData(NULL);
    WaveletCoderCg.setwaveletData(NULL);
    WaveletCoderCo.setwaveletData(NULL);
    delete waveletY;
    delete waveletCg;
    delete waveletCo;
}
```

Содержимое файла main.cpp

```
#include <cstring>
#include <iostream>
#include <QString>
#include "spihtcode.h"
//#include "intwavelet.h"

int main(int argc, char *argv[])
{
    SpihtCode coder;
    double CompressRatio = 1.0;
    if(argc < 4)
    {
        std::cout << "\n";
        std::cout << "Use: coder -c file.bmp file.spiht 0.1\n";
```

```
        std::cout << "to compress size(bmp)/10\n\n";
        std::cout << "\n";
        std::cout << "Use: coder -d file.bmp file.spiht\n";
        std::cout << "to decompress\n\n";
        return 0;
    }
    if(argc>4) CompressRatio = QString(argv[4]).toDouble();
    int waveletType = WAVELET_INT5;
    if(CompressRatio<0.99) waveletType = WAVELET_DOB8;
    if(strcmp(argv[1],"-c")==0) {
        coder.setBMPname(argv[2]);
        coder.setSPIHTname(argv[3]);
        coder.transformToSPIHT(CompressRatio,waveletType);
    }else
    if(strcmp(argv[1],"-d")==0) {
        coder.setBMPname(argv[2]);
        coder.setSPIHTname(argv[3]);
        coder.transformToBMP(CompressRatio,waveletType);
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ Д

Сравнение сохранения контуров основными методами сжатия фотографических изображений

Метод сравнения сохранения контурной информации.

Пускай имеется сортированные вейвлет коэффициенты преобразованного изображения w_i, которые сортированы в виде:

$$2^n \le |w_{i,n}| < 2^{n+1},$$

для $n=n_{max},...,2,1,0$. В соответствии сортировки определённо распределение: M_n – количество элементов множества при заданном n и $M = \sum_{n=0}^{n_{max}} M_n$ есть общее количество вейвлет коэффициентов и для их записи необходимо использование $N = \sum_{n=0}^{n_{max}} n_{max} \cdot M_n$ битов.

При сохранении методом SPIHT без потери информации, благодаря групповой передачи нулевых битов для старших разрядов, можно оценить общее количество информации записи вейвлет коэффициентов:

$$\hat{N} = \sum_{n=0}^{n_{max}} n \cdot M_n . \quad (1)$$

Соответственно, результирующим коэффициентом сжатия будет:

$$g = N / \left(\hat{N} + \tilde{N}\right),$$

где $\tilde{N}$ – количество информации для записи дерева сортировки. Эту величину можно принять пропорциональной количеству переданной информации, тогда используя коэффициент доли дополнительной информации d на сортировку вейвлет коэффициентов получим:

$$g = \frac{N}{\hat{N}(1+d)} .$$

Для сжатия с потерями информации соответственно $\hat{g} > g$ и вейвлет коэффициенты передаются с приоритетом сохранения абсолютной погрешности. Пускай при сохранении с потерями было передано $k<n_{max}$ битовых плоскостей вейвлет коэффициентов. Тогда при заданной степени сжатия $\hat{g}$ для записи информации будет использовано $N = g\hat{N} / \hat{g}$ бит информации. Соответственно, по множественной записи это число битов составит

$$N = \sum_{n=n_{max}-k}^{n_{max}} (n-k+1)\cdot M_n .$$

Благодаря тому, что MSE для пикселей изображения и для его вейвлет преобразования совпадают, можно оценить потерю качества отображения:

$$MSE \le \frac{\sum M_n \cdot \left(2^{n_{max}-k-1}\right)^2}{M}, \quad MSE \le \left(2^{n_{max}-k-1}\right)^2 .$$

А количество переданных коэффициентов определяется соотношением:

$$M(k) = \sum_{n=n_{max}-k}^{n_{max}} M_n .$$

Алгоритм с использованием отложенной передачи значимых битов позволяет передать информацию про наличие большего количества коэффициентов за счёт понижения точности значения уже полученных коэффициентов. Для отложенной передачи значимых битов, количество переданной информации оценивается так:

$$\hat{N} = \sum_{i=1}^{n_{max}} \sum_{n=n_{max}}^{i} M_n .$$

Соответственно, для получения начальной информации про наличие всех коэффициентов достаточно передать $\sum_{n=n_{max}}^{1} M_n$ бит информации. На этом

$$\hat{N} = \sum_{n=n_{max}}^{1} M_n + \tilde{N}$$

этапе, передача информации дерева сортировки вейвлет коэффициентов окончена полностью, и далее происходит только уточнение значения коэффициентов из списка значимых коэффициентов.

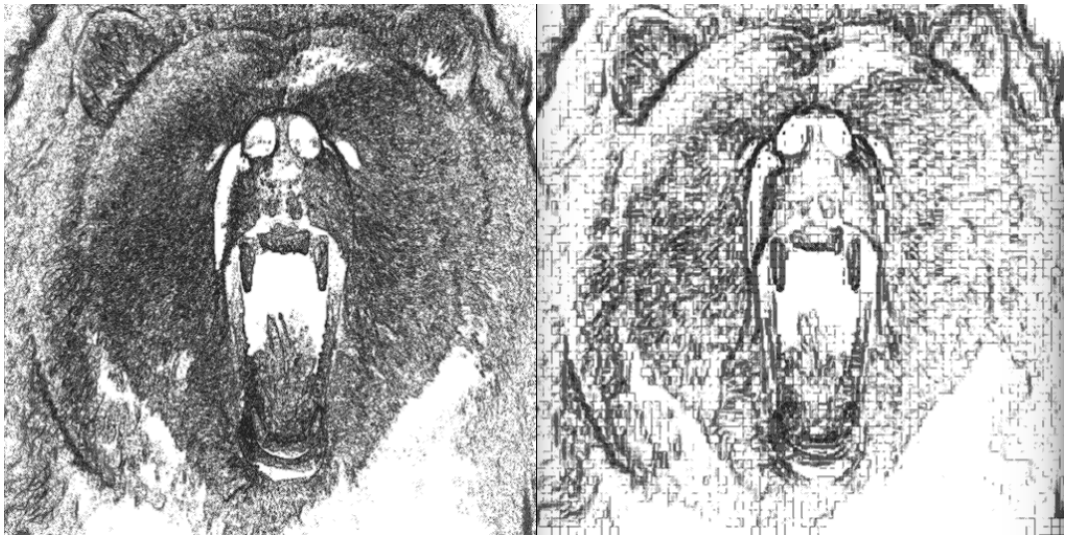


Рис. Д.1. Контур оригинального и сжатого в 50 раз изображения прогрессируещим jpeg кодеком

Оценка целостности контура была произведена статистическим исследованием. Для сжатых с потерями и оригинальных изображений были получены контуры (в работе использован SPIHT алгоритм). Далее для контурных матриц был произведён подсчёт неправильно отображённых и правильно отображённых пикселей. Правильно отображённые фоновые пиксели не подсчитывались. В результате получена оценка совпадения контуров: $PSNR = 10lg(N/L)$, где L — количество неправильно отображённых пикселей, N — общее количество сравнённых пикселей.

Для изображений показанных на рис. 1, показатель соответствия контуров составляет PSNR=10,8866.

Показатель качества соответствия сжатого изображения оригиналу рассчитывалось по формулам:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\overline{p}_i - p_i)^2, \quad PSNR = 10lg\left(255^2 / MSE\right),$$

где N – общее количество пикселей; $\overline{p}_i$ — пиксель сжатого изображения; p_i – пиксель оригинального изображения.

Таблица Д.1

Сравнение совпадения контурной информации по 1322 изображениям

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
0	19,2351	14,1354	56,0848	13,1539	30,6004	18,4163	52,2482	13,7694
1	2,76167	39,3074	56,7989	13,044	5,13102	33,9267	11,5049	26,9131
2	3,76568	36,6139	36,4575	16,8951	3,37402	37,5679	20,1592	22,0414
3	2,07385	41,7953	45,0711	15,0528	3,73917	36,6753	22,3362	21,1506
4	20,7101	21,8072	57,954	12,8691	25,1382	20,1241	43,5559	15,3499
5	18,9938	22,5586	64,2526	11,973	22,752	20,9904	66,1929	11,7146
6	9,05948	28,9887	38,0526	16,5231	8,83272	29,2089	23,503	20,7083
7	4,0775	35,9229	25,8223	19,8909	5,2428	33,7395	13,1533	25,7501
8	9,1086	28,9418	46,9087	14,7057	7,41318	30,7307	15,9968	24,0502
9	8,94716	29,0971	46,3011	14,819	8,18398	29,8715	23,8296	20,5885
10	15,1463	24,5247	59,4514	12,6476	17,1772	23,4318	40,9628	15,883
11	3,33302	37,6741	53,4351	13,5743	15,986	24,056	32,4113	17,9169
12	9,38799	28,6794	40,2375	16,0382	10,2821	27,8892	22,2475	21,1852
13	8,92697	29,1167	36,1139	16,9773	8,91445	29,1289	25,779	19,9055
14	10,7465	27,5054	39,9592	16,0985	11,7152	26,7558	26,6049	19,6316
15	17,3156	23,3621	46,2911	14,8209	17,3502	23,3447	37,1708	16,7268
16	13,7779	25,3472	62,2667	12,2457	20,0748	22,0778	54,8596	13,3457
17	3,33683	37,6641	37,681	16,6083	5,05423	34,0577	25,1311	20,1266
18	17,0487	23,497	75,1421	10,6131	16,7609	23,6449	30,6991	18,3883

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
19	6,80523	31,474	26,0618	19,8107	7,17367	31,016	30,7091	18,3855
20	7,15338	31,0406	42,5251	15,5579	7,02785	31,1943	18,4246	22,8229
21	11,1934	27,1516	43,5034	15,3603	12,7552	26,017	27,4738	19,3524
22	14,0712	25,1642	63,0137	12,1421	14,737	24,7626	38,4354	16,4362
23	9,65414	28,4365	33,0385	17,7504	9,04124	29,0062	37,5282	16,6437
24	3,65485	36,8734	36,8277	16,8073	4,08889	35,8987	11,9448	26,5872
25	14,893	24,6712	58,7153	12,7558	15,421	24,3686	41,2504	15,8222
26	9,33072	28,7325	32,5694	17,8746	8,72143	29,3191	38,9713	16,3159
27	11,1992	27,1471	47,4046	14,6144	25,8197	19,8918	31,2569	18,2319
28	14,8177	24,7152	41,5179	15,7661	15,4515	24,3514	41,3456	15,8022
29	18,9302	22,5877	51,1678	13,9509	17,5951	23,223	30,7718	18,3678
30	13,2296	25,6999	50,6335	14,042	13,7745	25,3493	27,3259	19,3993
31	7,23763	30,9389	69,2192	11,3263	9,3023	28,759	26,275	19,7399
32	9,80665	28,3004	42,1821	15,6282	12,9791	25,8659	29,2411	18,8109
33	7,1332	31,0651	42,2932	15,6054	10,3233	27,8544	29,9279	18,6093
34	8,76949	29,2713	26,1742	19,7733	7,84445	30,2396	29,0276	18,8746
35	14,9664	24,6284	68,4615	11,4219	12,6201	26,1096	47,5731	14,5836
36	4,68987	34,7076	28,7842	18,9477	5,34275	33,5755	25,6843	19,9374
37	6,90343	31,3495	68,5723	11,4078	9,46582	28,6076	20,9736	21,6974
38	5,3263	33,6023	85,3079	9,51102	8,44002	29,6039	43,6662	15,3279
39	6,58211	31,7635	40,9973	15,8757	6,4936	31,8811	20,576	21,8636
40	16,2045	23,9381	58,3076	12,8163	14,9397	24,644	33,9263	17,5201
41	7,20519	30,9779	38,941	16,3227	7,1377	31,0596	25,5094	19,9968
42	6,72988	31,5707	51,3344	13,9226	9,98828	28,141	40,5266	15,976
43	12,1555	26,4353	60,3062	12,5236	12,4386	26,2354	29,884	18,622
44	10,8163	27,4492	59,1815	12,6871	11,9891	26,555	22,982	20,903
45	11,5264	26,8969	45,0826	15,0506	13,4138	25,5798	29,2375	18,812
46	2,23858	41,1314	21,6841	21,408	3,36577	37,5891	19,2155	22,4578
47	10,4141	27,7783	57,3857	12,9547	11,9914	26,5534	49,9867	14,1537
48	11,4722	26,9379	41,3169	15,8082	13,2993	25,6542	24,9258	20,1978
49	9,3502	28,7144	51,7845	13,8468	19,6538	22,2619	42,2821	15,6077
50	9,23888	28,8184	40,4511	15,9922	28,6513	18,9879	31,1244	18,2688
51	24,0382	20,5128	71,5982	11,0328	27,695	19,2828	45,6915	14,9341
52	7,6266	30,4842	30,8001	18,3598	9,21891	28,8372	49,5251	14,2343
53	5,65451	33,0829	20,2247	22,0132	6,50538	31,8654	13,604	25,4575
54	14,6719	24,8011	43,9964	15,2625	15,5048	24,3215	35,422	17,1453
55	8,27652	29,7738	44,1982	15,2227	7,65692	30,4497	38,9938	16,3109
56	13,2419	25,6918	54,3641	13,4246	14,2989	25,0248	23,8179	20,5927
57	14,971	24,6258	60,4123	12,5083	15,3166	24,4276	27,6363	19,3012
58	10,5667	27,652	47,8363	14,5357	10,5568	27,6601	37,7748	16,5868

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
59	4,54245	34,985	32,1634	17,9836	5,57103	33,2121	14,8539	24,694
60	5,51501	33,2999	42,966	15,4683	18,3331	22,8661	35,0055	17,2481
61	9,0977	28,9522	45,5018	14,9702	7,51581	30,6113	20,844	21,7512
62	3,28241	37,8069	16,0509	24,0208	3,51452	37,2135	29,3185	18,788
63	5,65115	33,0881	56,7142	13,057	8,82261	29,2189	34,1261	17,4691
64	10,1614	27,9917	42,9191	15,4778	12,2251	26,3858	29,0884	18,8564
65	19,4023	22,3738	53,046	13,6377	16,1229	23,9819	26,9643	19,515
66	3,19033	38,0541	66,5638	11,666	4,47503	35,1149	34,7074	17,3224
67	2,69336	39,5249	37,144	16,733	3,16008	38,1368	12,0115	26,5388
68	14,5897	24,8499	39,6691	16,1618	14,6997	24,7846	24,7613	20,2553
69	11,385	27,0042	58,0604	12,8532	9,38725	28,68	30,7015	18,3876
70	3,97822	36,137	31,3275	18,2123	4,38021	35,3009	20,808	21,7662
71	15,8172	24,1482	51,4779	13,8984	15,5647	24,288	33,181	17,713
72	13,6893	25,4032	51,5574	13,885	30,6198	18,4108	54,3296	13,4301
73	6,12082	32,3946	43,6084	15,3394	6,36018	32,0614	25,6454	19,9506
74	15,3106	24,431	56,3292	13,1161	15,8853	24,1109	30,0568	18,572
75	15,0349	24,5888	43,9576	15,2701	14,6146	24,835	31,3806	18,1976
76	18,6766	22,7048	68,1075	11,4669	19,7288	22,2288	37,3291	16,6899
77	6,74748	31,548	38,3621	16,4528	6,273	32,1813	17,0135	23,515
78	6,7595	31,5325	39,2085	16,2632	10,3623	27,8217	22,8634	20,948
79	9,04747	29,0003	35,768	17,0609	8,28952	29,7602	26,1085	19,7952
80	17,4482	23,2958	63,5022	12,075	15,309	24,4318	27,3259	19,3993
81	6,51236	31,856	27,0567	19,4853	8,41788	29,6267	21,0158	21,6799
82	17,1727	23,434	64,1809	11,9827	27,9603	19,2	37,2374	16,7112
83	8,24434	29,8077	68,3009	11,4423	10,8801	27,3981	26,6161	19,6279
84	15,436	24,3601	42,9142	15,4788	16,2718	23,9021	26,6645	19,6121
85	8,27815	29,7721	47,9378	14,5172	7,03952	31,1799	19,2907	22,4238
86	25,0783	20,1448	15,4278	24,3647	25,4227	20,0264	44,843	15,0969
87	2,25418	41,071	57,0405	13,0071	2,96409	38,693	34,0899	17,4783
88	5,01163	34,1312	46,362	14,8076	10,6487	27,5849	21,3993	21,5228
89	9,64018	28,4491	38,0166	16,5313	9,7251	28,3729	48,2319	14,4641
90	14,7047	24,7817	36,4423	16,8987	15,4921	24,3286	28,1952	19,1273
91	11,3306	27,0457	51,5886	13,8797	13,7224	25,3822	24,1362	20,4774
92	14,4575	24,929	45,7234	14,928	14,5132	24,8955	25,3815	20,0405
93	7,72169	30,3766	49,535	14,2326	8,95351	29,0909	21,0955	21,647
94	12,3695	26,2837	42,7426	15,5136	20,1679	22,0376	23,1301	20,8473
95	11,9594	26,5766	46,6602	14,7519	14,2004	25,0848	28,8124	18,9392
96	10,1222	28,0253	42,4151	15,5804	12,6858	26,0644	30,5748	18,4235
97	8,9417	29,1024	41,3458	15,8022	8,17625	29,8797	29,298	18,794
98	20,9526	21,706	54,5505	13,3948	20,1537	22,0437	54,5118	13,401

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
99	9,60417	28,4816	51,7213	13,8574	10,7136	27,5321	40,8499	15,907
100	7,00218	31,2261	42,4135	15,5807	8,41618	29,6285	24,6357	20,2995
101	18,7524	22,6697	58,5114	12,786	21,4413	21,5058	33,7423	17,5673
102	15,5232	24,3112	37,3632	16,6819	18,758	22,6671	38,6338	16,3914
103	11,1157	27,2121	66,0391	11,7348	12,1144	26,4648	31,9906	18,0303
104	18,4234	22,8234	40,3633	16,0111	21,8027	21,3606	31,8769	18,0613
105	3,55264	37,1198	52,8644	13,6675	5,02356	34,1106	43,0372	15,4539
106	2,1822	41,3529	30,093	18,5615	2,44805	40,3544	20,4452	21,919
107	10,3865	27,8014	48,6548	14,3883	12,9023	25,9174	30,8998	18,3317
108	18,5115	22,782	28,6599	18,9853	22,7159	21,0042	49,1323	14,3035
109	8,54424	29,4973	23,6313	20,6611	10,1696	27,9847	33,9034	17,5259
110	6,95242	31,2881	44,5704	15,1499	11,4692	26,9401	26,1729	19,7738
111	16,9083	23,5688	67,3435	11,5649	19,7937	22,2002	44,957	15,0749
112	24,6881	20,2811	52,2514	13,7688	27,9588	19,2004	47,551	14,5876
113	24,8956	20,2083	58,9391	12,7227	28,6063	19,0016	67,332	11,5664
114	12,9523	25,8839	33,1141	17,7306	14,1234	25,132	38,1222	16,5072
115	6,80159	31,4786	30,6532	18,4013	7,13643	31,0612	24,3492	20,4011
116	3,80477	36,5242	42,3965	15,5842	4,4275	35,2076	18,0252	23,0132
117	20,654	21,8307	49,9898	14,1532	20,1877	22,0291	39,5684	16,1838
118	17,3447	23,3475	43,8899	15,2835	17,6514	23,1952	43,5912	15,3428
119	12,4649	26,217	37,6948	16,6052	12,9083	25,9135	29,6502	18,6902
120	4,52356	35,0212	43,9716	15,2674	5,74628	32,9431	21,3767	21,532
121	18,514	22,7808	79,6603	10,106	18,3493	22,8584	37,3365	16,6881
122	12,3424	26,3028	32,5785	17,8722	21,2071	21,6012	38,1258	16,5064
123	4,72184	34,6486	40,229	16,04	5,81756	32,836	19,2086	22,4609
124	4,86847	34,3829	23,596	20,674	5,49501	33,3314	17,451	23,2944
125	5,91629	32,6898	38,5852	16,4024	6,79448	31,4877	25,7879	19,9025
126	4,5005	35,0656	43,1644	15,4283	5,27533	33,6858	20,1296	22,0541
127	9,20066	28,8544	50,5315	14,0596	8,6415	29,399	30,8506	18,3455
128	7,36587	30,7863	69,9708	11,2325	7,74393	30,3516	15,1078	24,5468
129	9,9599	28,1657	25,0965	20,1385	11,6067	26,8366	29,996	18,5895
130	4,38638	35,2887	38,3524	16,4549	5,82349	32,8271	28,4916	19,0365
131	2,45542	40,3283	39,7164	16,1514	16,2513	23,9131	37,5666	16,6348
132	10,5916	27,6316	58,7519	12,7504	13,2221	25,7048	33,6631	17,5877
133	11,5437	26,8839	70,9165	11,1159	12,4028	26,2604	31,6969	18,1105
134	8,13816	29,9203	62,342	12,2352	17,5021	23,269	31,3829	18,1969
135	13,253	25,6845	48,7768	14,3665	13,8344	25,3116	28,6144	18,9991
136	8,3863	29,6594	67,701	11,5189	14,7455	24,7576	22,9628	20,9103
137	10,9798	27,3189	40,3082	16,0229	13,436	25,5654	33,7974	17,5531
138	7,70721	30,3929	24,5446	20,3317	28,6023	19,0028	28,7006	18,973

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
139	6,84338	31,4254	48,2473	14,4613	5,39849	33,4854	27,0129	19,4994
140	6,96798	31,2687	28,5105	19,0307	8,77125	29,2696	42,9597	15,4696
141	3,03084	38,4995	33,9519	17,5135	6,60374	31,735	11,7539	26,7271
142	4,89737	34,3315	69,2445	11,3231	4,94937	34,2398	14,7075	24,78
143	22,2491	21,1846	26,5623	19,6455	25,1547	20,1184	43,3903	15,383
144	9,02567	29,0212	35,5517	17,1136	11,6081	26,8356	44,3796	15,1871
145	10,0824	28,0595	42,5611	15,5505	12,65	26,089	29,437	18,7529
146	16,5387	23,7608	23,6659	20,6484	17,2167	23,4118	35,8115	17,0503
147	6,51539	31,852	41,2585	15,8205	7,19035	30,9958	36,3453	16,9218
148	20,5616	21,8696	107,248	7,52303	18,7777	22,658	35,4366	17,1418
149	108,649	7,41027	28,7518	18,9575	7,55118	30,5705	16,6122	23,7222
150	1,39563	45,2354	44,077	15,2466	1,24735	46,211	6,06761	32,4705
151	13,433	25,5673	54,9919	13,3248	21,8291	21,3501	37,2539	16,7074
152	17,4869	23,2765	13,0063	25,8477	20,8156	21,763	39,8663	16,1187
153	1,47044	44,7819	36,9022	16,7898	2,51352	40,1252	38,7036	16,3758
154	12,9825	25,8636	39,1143	16,2841	14,446	24,9358	38,0697	16,5192
155	15,593	24,2722	44,9389	15,0784	15,3542	24,4062	32,6359	17,8569
156	9,93175	28,1903	52,8231	13,6743	12,2515	26,367	25,8374	19,8858
157	24,3688	20,3941	25,9289	19,8551	22,8049	20,9702	40,014	16,0866
158	6,24471	32,2206	40,9888	15,8775	6,50382	31,8674	24,4064	20,3807
159	7,90676	30,1708	27,6538	19,2957	7,84315	30,241	37,014	16,7635
160	7,20326	30,9802	42,6849	15,5253	9,3272	28,7358	20,3963	21,9398
161	9,82342	28,2855	45,0592	15,0551	7,82126	30,2653	19,9994	22,1105
162	20,47	21,9084	36,0972	16,9813	19,2036	22,4631	35,412	17,1478
163	13,3333	25,6321	46,0449	14,8672	14,2168	25,0748	33,7365	17,5688
164	16,2208	23,9294	34,571	17,3566	17,4708	23,2846	32,9307	17,7788
165	14,9096	24,6615	67,5168	11,5426	16,2409	23,9186	29,8595	18,6291
166	20,4595	21,9129	54,7221	13,3676	29,1855	18,8275	33,2796	17,6873
167	7,48966	30,6416	54,8502	13,3472	10,2342	27,9297	40,2622	16,0329
168	15,899	24,1034	59,8716	12,5864	22,7013	21,0098	38,9676	16,3167
169	13,145	25,7556	56,8498	13,0362	26,2936	19,7338	34,0727	17,4827
170	8,97422	29,0709	59,001	12,7136	11,0169	27,2896	31,7135	18,1059
171	25,3271	20,0591	54,4353	13,4132	28,2827	19,1004	49,0612	14,316
172	21,0718	21,6568	29,1582	18,8356	26,183	19,7704	41,7779	15,7119
173	3,64525	36,8962	36,3899	16,9112	16,402	23,8328	22,2148	21,198
174	10,5332	27,6796	49,3144	14,2713	12,7432	26,0252	29,0193	18,8771
175	12,2169	26,3916	57,7925	12,8934	13,883	25,2811	33,9721	17,5084
176	18,1006	22,9769	50,4305	14,0769	15,7877	24,1644	41,775	15,7125
177	17,3527	23,3435	61,7499	12,3181	16,0965	23,9962	31,1486	18,262
178	16,9279	23,5587	55,9976	13,1674	16,7201	23,666	31,1441	18,2633

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
179	14,6411	24,8193	41,4306	15,7844	31,3046	18,2186	29,8885	18,6207
180	4,17087	35,7263	33,5166	17,6256	5,10297	33,9743	18,2293	22,9154
181	5,27826	33,681	33,2212	17,7025	6,77248	31,5158	18,9266	22,5894
182	12,1945	26,4075	56,9879	13,0152	13,6053	25,4566	27,9233	19,2115
183	19,7621	22,2141	53,9186	13,496	17,3633	23,3382	37,7338	16,5962
184	14,2261	25,0691	28,4653	19,0445	14,4182	24,9526	28,2063	19,1239
185	5,77898	32,8938	49,7185	14,2004	6,36581	32,0537	21,1315	21,6322
186	14,5921	24,8484	55,4201	13,2575	15,3654	24,4	28,7332	18,9631
187	12,7313	26,0333	77,7727	10,3143	13,5276	25,5064	44,8409	15,0973
188	17,8818	23,0826	58,9754	12,7174	16,072	24,0094	39,7199	16,1506
189	8,21409	29,8396	60,043	12,5616	7,36361	30,789	22,184	21,21
190	16,3	23,887	25,6968	19,9332	13,5266	25,507	28,3572	19,0775
191	4,17498	35,7177	49,9752	14,1557	39,5595	16,1858	14,9303	24,6495
192	12,0964	26,4777	42,0895	15,6473	13,2233	25,704	31,253	18,233
193	12,5459	26,1608	69,0053	11,3532	13,5148	25,5146	25,044	20,1567
194	22,4946	21,0892	58,8535	12,7354	26,4457	19,6837	55,6629	13,2195
195	12,1341	26,4506	52,167	13,7829	18,6012	22,74	41,175	15,8381
196	18,3359	22,8648	43,4291	15,3752	17,4001	23,3198	40,7363	15,9312
197	11,8739	26,6389	62,7009	12,1853	13,5293	25,5053	46,3977	14,8009
198	13,8636	25,2933	50,2765	14,1035	14,0013	25,2074	40,5926	15,9619
199	7,50154	30,6278	26,112	19,794	7,13442	31,0636	18,1343	22,9608
200	3,12999	38,2199	13,9071	25,2661	32,015	18,0237	22,3379	21,15
201	1,38814	45,2821	28,7979	18,9436	1,43674	44,9833	31,8449	18,07
202	6,11882	32,3975	55,5224	13,2414	6,40925	31,9947	24,5161	20,3418
203	17,2252	23,4075	34,7791	17,3044	17,5235	23,2584	33,8469	17,5404
204	5,93314	32,6651	45,3708	14,9953	7,83189	30,2535	50,9123	13,9943
205	14,9354	24,6465	42,2471	15,6149	16,6407	23,7074	34,9588	17,2597
206	9,52765	28,5511	51,8634	13,8336	10,8699	27,4063	31,3923	18,1943
207	26,3786	19,7058	27,4133	19,3716	26,9817	19,5094	46,5391	14,7745
208	4,14797	35,7741	17,5043	23,2679	5,29525	33,6531	23,7403	20,6211
209	2,09605	41,7028	43,3157	15,3979	1,75064	43,2669	31,1418	18,2639
210	16,1062	23,9909	13,613	25,4517	19,9066	22,1509	38,254	16,4773
211	1,86513	42,7166	48,141	14,4805	2,1097	41,6464	47,7305	14,5549
212	6,39354	32,016	46,6892	14,7465	29,4644	18,7448	25,9991	19,8316
213	14,8034	24,7236	51,2731	13,933	15,8147	24,1496	42,2527	15,6137
214	11,0832	27,2375	33,2795	17,6873	8,80332	29,2379	19,5065	22,3272
215	3,63514	36,9204	33,3946	17,6573	5,30126	33,6432	29,1126	18,8492
216	11,6598	26,797	30,5079	18,4426	11,951	26,5827	56,3825	13,1079
217	5,13021	33,9281	39,0368	16,3013	6,07706	32,4569	29,2691	18,8026
218	7,17492	31,0145	73,9263	10,7548	7,40402	30,7415	18,0128	23,0192

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
219	13,8329	25,3126	46,2384	14,8307	37,936	16,5498	27,9621	19,1994
220	7,93799	30,1366	44,6506	15,1343	7,25142	30,9223	21,6505	21,4214
221	16,1726	23,9552	30,738	18,3773	14,9181	24,6565	27,9686	19,1974
222	5,98915	32,5835	44,4452	15,1743	6,61434	31,7211	21,0695	21,6577
223	4,44724	35,169	79,5986	10,1127	5,55631	33,2351	21,4928	21,4849
224	27,1465	19,4565	73,1571	10,8457	28,8301	18,9339	47,676	14,5648
225	14,3307	25,0054	65,958	11,7455	13,273	25,6714	31,6612	18,1203
226	12,988	25,86	45,667	14,9387	19,4066	22,3718	44,2785	15,207
227	8,85742	29,1847	33,7073	17,5763	11,5178	26,9034	28,945	18,8993
228	3,78688	36,5652	42,2755	15,609	5,18828	33,8303	12,6771	26,0704
229	3,07797	38,3655	48,2912	14,4534	4,03236	36,0196	7,15076	31,0438
230	7,1991	30,9852	46,9196	14,7037	18,543	22,7672	29,5617	18,7162
231	8,06854	29,9949	35,9221	17,0236	7,13828	31,0589	41,6857	15,7311
232	11,5669	26,8665	67,5149	11,5428	11,8559	26,6521	23,0263	20,8863
233	17,6697	23,1862	57,7147	12,9051	23,1162	20,8525	35,4245	17,1447
234	9,55972	28,5219	30,9316	18,3227	7,88728	30,1923	32,9063	17,7852
235	6,36355	32,0568	66,2689	11,7046	7,8455	30,2384	20,5829	21,8607
236	8,21228	29,8415	77,6006	10,3335	9,10387	28,9463	28,718	18,9677
237	8,82154	29,2199	30,7882	18,3631	9,8852	28,2311	24,1088	20,4873
238	10,6678	27,5693	48,3808	14,4373	9,99617	28,1341	48,968	14,3326
239	8,69113	29,3493	39,2258	16,2594	7,16453	31,0271	17,7993	23,1227
240	6,61668	31,718	61,9034	12,2965	6,41049	31,993	41,6735	15,7336
241	4,14585	35,7785	44,126	15,2369	10,4055	27,7855	44,6403	15,1363
242	6,96181	31,2764	51,3489	13,9202	7,35985	30,7934	32,2064	17,972
243	9,72836	28,37	73,6451	10,7879	11,7822	26,7063	31,4502	18,1783
244	3,91192	36,283	61,1055	12,4092	2,13379	41,5478	8,89135	29,1514
245	6,2693	32,1864	46,6055	14,7621	4,96473	34,2129	38,2095	16,4874
246	3,84897	36,4239	18,8912	22,6056	3,48976	37,2749	10,1053	28,0398
247	3,14592	38,1758	53,7886	13,517	15,6605	24,2347	32,4704	17,9011
248	13,2604	25,6797	64,8356	11,8945	13,2285	25,7006	25,7303	19,9219
249	9,59131	28,4932	51,374	13,9159	9,60486	28,481	15,8745	24,1168
250	8,47128	29,5718	62,0182	12,2804	19,1631	22,4815	34,8677	17,2823
251	30,9348	18,3219	34,0524	17,4879	28,926	18,905	39,2708	16,2494
252	13,2153	25,7092	51,4689	13,8999	11,2476	27,1096	30,8855	18,3357
253	4,47095	35,1228	47,6886	14,5625	4,79785	34,5099	17,5712	23,2348
254	8,02869	30,0379	35,8893	17,0315	6,63015	31,7003	27,9929	19,1899
255	0,935613	48,7089	55,7365	13,208	1,8261	42,9003	23,5662	20,685
256	12,5311	26,171	30,9527	18,3168	13,1589	25,7464	51,9806	13,814
257	10,6175	27,6103	33,6895	17,5809	9,87759	28,2378	34,3349	17,4161
258	3,39211	37,5214	40,9465	15,8865	5,25313	33,7224	22,7833	20,9785

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
259	2,26724	41,0208	62,9377	12,1526	22,4802	21,0948	25,0789	20,1446
260	11,0019	27,3014	43,3157	15,3979	7,71351	30,3858	13,425	25,5725
261	2,95986	38,7054	28,659	18,9856	9,83116	28,2787	30,6918	18,3903
262	16,1062	23,9909	14,5068	24,8994	19,9066	22,1509	38,254	16,4773
263	3,05836	38,421	42,8833	15,485	2,87758	38,9503	9,52637	28,5523
264	2,6184	39,7701	61,0709	12,4141	2,27051	41,0083	5,69314	33,0238
265	3,70832	36,7473	55,0631	13,3136	5,64573	33,0964	32,9367	17,7772
266	7,27372	30,8957	48,3738	14,4386	9,98941	28,14	20,0607	22,0839
267	6,47756	31,9026	37,0775	16,7486	54,9534	13,3309	19,2302	22,4511
268	3,99379	36,1031	33,1218	17,7285	6,74556	31,5504	20,7766	21,7793
269	1,22109	46,3958	40,7248	15,9336	2,08382	41,7536	20,2058	22,0213
270	5,31796	33,6159	38,3039	16,4659	8,85946	29,1827	26,1789	19,7718
271	5,57723	33,2024	76,2477	10,4863	13,3461	25,6237	27,0111	19,5
272	2,24852	41,0929	43,6079	15,3395	43,9146	15,2786	59,9094	12,5809
273	18,6332	22,7251	66,2375	11,7087	24,3775	20,391	34,4498	17,3871
274	7,5244	30,6014	33,3889	17,6588	6,55848	31,7947	13,7732	25,3501
275	3,49784	37,2548	40,9418	15,8875	4,73387	34,6265	23,2397	20,8062
276	11,7938	26,6978	60,7124	12,4653	10,8103	27,454	32,6679	17,8484
277	2,77577	39,2631	52,8174	13,6753	4,23384	35,5961	26,3581	19,7125
278	22,9863	20,9014	25,0783	20,1449	18,6808	22,7029	34,1522	17,4624
279	11,2053	27,1423	35,0241	17,2435	11,1721	27,1681	29,8787	18,6236
280	4,28489	35,492	44,8519	15,0952	5,29151	33,6592	26,5673	19,6439
281	7,65375	30,4533	43,8169	15,298	9,62804	28,46	23,8299	20,5884
282	7,37463	30,776	60,3227	12,5212	8,17856	29,8773	14,8196	24,7141
283	10,3129	27,8632	22,1692	21,2158	10,8375	27,4322	22,018	21,2752
284	17,345	23,3473	56,7026	13,0587	12,6108	26,1159	37,7659	16,5888
285	1,76739	43,1842	43,173	15,4265	2,26227	41,0399	44,7983	15,1056
286	7,81015	30,2776	54,4651	13,4084	7,01063	31,2157	21,4321	21,5095
287	2,84545	39,0478	66,2957	11,7011	7,8149	30,2723	14,7918	24,7304
288	3,93777	36,2258	53,1034	13,6284	4,94213	34,2525	14,0799	25,1588
289	3,67162	36,8337	39,4201	16,2165	4,3638	35,3335	28,503	19,033
290	21,6675	21,4146	59,3073	12,6686	16,5261	23,7674	29,6488	18,6907
291	7,81227	30,2753	43,2469	15,4117	8,33394	29,7138	44,7603	15,1129
292	6,9807	31,2528	38,8739	16,3376	17,2743	23,3828	22,2338	21,1905
293	7,96156	30,1108	20,3707	21,9507	7,40356	30,742	13,8827	25,2813
294	5,81012	32,8471	55,6277	13,225	6,90011	31,3537	30,8991	18,3319
295	3,03688	38,4822	22,1456	21,2251	2,96493	38,6905	8,35449	29,6924
296	11,6241	26,8236	15,8206	24,1463	11,6528	26,8022	27,4649	19,3552
297	5,23166	33,758	50,1574	14,1241	5,36854	33,5337	38,0075	16,5334
298	2,13146	41,5573	26,3658	19,71	2,03446	41,9618	33,9428	17,5158

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
299	8,25715	29,7942	32,7375	17,8299	10,6089	27,6174	24,6703	20,2873
300	4,02925	36,0263	26,5266	19,6572	2,93093	38,7907	9,16979	28,8836
301	10,427	27,7676	15,6599	24,235	12,4813	26,2056	26,8066	19,566
302	3,33864	37,6594	16,7092	23,6717	4,39634	35,269	36,6868	16,8406
303	1,00571	48,0813	31,5024	18,1639	1,18393	46,6643	16,8151	23,6168
304	0,693794	51,3062	13,747	25,3666	0,987859	48,2369	24,7574	20,2567
305	6,29432	32,1518	61,4307	12,3631	7,34225	30,8142	21,9728	21,2931
306	6,4299	31,9667	14,1511	25,115	4,44549	35,1724	9,85289	28,2595
307	12,7257	26,0372	41,1058	15,8527	29,5171	18,7293	25,7676	19,9093
308	1,08511	47,4213	26,5595	19,6464	1,20829	46,4874	24,9443	20,1914
309	5,90764	32,7025	32,286	17,9505	7,35168	30,8031	19,2112	22,4597
310	2,88696	38,922	53,1272	13,6245	15,824	24,1445	25,5697	19,9763
311	5,57012	33,2135	16,7696	23,6403	39,5439	16,1892	20,7644	21,7844
312	9,05481	28,9932	29,647	18,6912	7,67513	30,4291	29,1042	18,8517
313	4,46228	35,1397	59,0485	12,7066	4,5431	34,9837	27,4556	19,3582
314	1,99123	42,1484	57,8046	12,8916	2,84231	39,0574	5,01793	34,1203
315	6,25641	32,2043	28,9086	18,9103	50,0088	14,1499	17,4235	23,3081
316	12,4214	26,2474	29,1985	18,8236	11,5411	26,8859	56,7699	13,0484
317	18,5472	22,7652	69,3811	11,306	29,3862	18,7679	23,3084	20,7806
318	3,29202	37,7816	49,3982	14,2566	2,49177	40,2006	6,43057	31,9658
319	4,35529	35,3505	54,2519	13,4425	5,31358	33,6231	19,8059	22,1949
320	10,8219	27,4447	48,9355	14,3383	11,9563	26,5789	34,0806	17,4807
321	7,1666	31,0245	34,1657	17,459	10,0976	28,0464	26,3483	19,7157
322	3,99936	36,091	58,8415	12,7371	59,3792	12,6581	19,5103	22,3255
323	4,9888	34,1709	88,9181	9,151	7,04497	31,1732	17,0851	23,4785
324	5,71306	32,9934	90,3167	9,01545	5,76024	32,922	24,7806	20,2486
325	9,14733	28,9049	23,1532	20,8386	9,0274	29,0195	17,4085	23,3156
326	4,80431	34,4982	39,6915	16,1569	63,7134	12,0462	28,8668	18,9228
327	7,58672	30,5297	50,2758	14,1036	42,6999	15,5223	32,1339	17,9915
328	2,65799	39,6397	63,6823	12,0504	2,2129	41,2316	7,86307	30,219
329	2,70499	39,4875	52,0711	13,7989	3,49472	37,2626	15,5853	24,2765
330	4,26877	35,5247	46,0165	14,8725	5,82103	32,8308	23,1083	20,8554
331	19,5286	22,3174	52,9438	13,6545	20,9151	21,7216	51,641	13,8709
332	13,8179	25,322	29,5831	18,7099	14,8558	24,6929	36,91	16,7879
333	11,5998	26,8418	26,5329	19,6551	13,1722	25,7376	29,9245	18,6103
334	13,3686	25,6091	40,1171	16,0642	21,7488	21,3821	32,3859	17,9237
335	6,68813	31,6247	29,6924	18,6779	7,10328	31,1016	25,8508	19,8813
336	6,06133	32,4794	31,3305	18,2115	6,32397	32,111	16,6955	23,6788
337	5,45687	33,3919	19,6125	22,2801	8,84785	29,194	23,2315	20,8093
338	3,52906	37,1776	46,6656	14,7509	4,93389	34,267	20,6344	21,839

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
339	3,39555	37,5126	21,1823	21,6113	5,18212	33,8407	24,3593	20,3975
340	9,27013	28,7891	45,6012	14,9513	7,7775	30,314	10,2179	27,9436
341	14,6548	24,8112	49,3938	14,2574	15,7802	24,1685	34,9083	17,2722
342	5,40049	33,4821	35,6934	17,0791	5,64208	33,102	14,1827	25,0956
343	7,2959	30,8692	49,3837	14,2591	7,20616	30,9767	36,2532	16,9439
344	16,0883	24,0006	39,2215	16,2603	15,2154	24,4852	29,4204	18,7578
345	6,54687	31,8101	42,6135	15,5399	6,55345	31,8014	14,3938	24,9673
346	4,17059	35,7269	38,7927	16,3558	4,58558	34,9029	15,8181	24,1477
347	15,2406	24,4707	46,659	14,7521	15,6153	24,2598	35,6707	17,0846
348	12,7925	25,9917	31,5662	18,1463	14,3357	25,0024	29,7748	18,6538
349	5,33162	33,5936	38,7734	16,3601	5,65539	33,0816	19,1458	22,4893
350	3,59997	37,0048	64,1651	11,9848	3,49057	37,2729	29,0717	18,8614
351	8,3096	29,7392	30,2333	18,5211	8,08805	29,9739	13,9802	25,2206
352	16,2632	23,9067	52,6252	13,7069	16,5018	23,7802	32,8446	17,8015
353	8,40067	29,6445	54,8644	13,345	23,7623	20,613	32,6191	17,8614
354	1,47499	44,755	39,0229	16,3044	9,30766	28,754	20,7497	21,7906
355	4,59942	34,8767	28,1778	19,1327	5,12305	33,9402	31,4696	18,173
356	4,62217	34,8339	24,6457	20,296	5,11124	33,9603	31,1692	18,2563
357	9,94623	28,1776	37,2413	16,7103	9,03517	29,0121	19,854	22,1738
358	6,95876	31,2802	46,0506	14,8661	7,72983	30,3674	53,6377	13,5414
359	6,59387	31,748	35,012	17,2465	7,0264	31,1961	19,1548	22,4852
360	5,21771	33,7812	36,0873	16,9837	6,29983	32,1442	11,5479	26,8807
361	14,1241	25,1316	52,6743	13,6988	14,6021	24,8425	25,7402	19,9186
362	9,60688	28,4792	56,4829	13,0925	8,80624	29,235	21,7047	21,3997
363	6,23384	32,2357	31,7193	18,1043	7,09468	31,1122	18,8014	22,647
364	11,2706	27,0919	32,6116	17,8634	20,0056	22,1078	24,9898	20,1756
365	4,9853	34,177	41,1737	15,8384	7,41523	30,7283	21,2972	21,5644
366	5,65383	33,084	47,3813	14,6187	6,05507	32,4884	35,5416	17,1161
367	6,39018	32,0205	50,2884	14,1015	6,68402	31,63	22,769	20,9839
368	13,0719	25,8041	39,9195	16,1071	15,1693	24,5115	31,0666	18,2849
369	5,95182	32,6378	22,9748	20,9058	6,79154	31,4914	29,6035	18,7039
370	10,4437	27,7537	28,698	18,9738	9,55899	28,5226	48,9361	14,3382
371	10,7862	27,4734	37,3508	16,6848	12,4248	26,245	35,651	17,0894
372	6,79316	31,4894	42,1102	15,6431	7,15785	31,0351	17,4873	23,2764
373	2,54967	40,0011	44,3874	15,1856	2,79957	39,189	47,6658	14,5667
374	12,9876	25,8602	31,0895	18,2785	12,5985	26,1244	25,1482	20,1207
375	2,59493	39,8483	50,357	14,0896	3,03643	38,4835	9,02477	29,0221
376	14,9233	24,6535	73,505	10,8045	15,4675	24,3424	26,2915	19,7345
377	9,24294	28,8146	67,1506	11,5898	12,3701	26,2833	26,8207	19,5614
378	4,95885	34,2232	31,7606	18,093	2,39552	40,5428	13,1906	25,7255

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
379	11,7055	26,763	38,3025	16,4663	11,4401	26,9622	31,575	18,1439
380	4,44896	35,1656	50,9399	13,9896	5,65112	33,0881	18,6704	22,7077
381	14,3722	24,9804	25,818	19,8923	15,6008	24,2679	32,9808	17,7656
382	4,38101	35,2993	53,427	13,5756	6,53136	31,8307	48,1544	14,4781
383	15,8547	24,1276	35,8123	17,0501	16,5133	23,7741	31,5418	18,1531
384	8,31585	29,7327	54,3343	13,4293	9,45304	28,6194	45,4664	14,977
385	26,8683	19,546	33,6431	17,5929	30,7826	18,3647	46,2468	14,8292
386	6,34454	32,0828	64,5726	11,9298	6,87118	31,3902	32,5223	17,8872
387	14,5978	24,845	38,6416	16,3897	14,7856	24,734	32,643	17,855
388	9,07572	28,9732	55,7354	13,2082	11,1425	27,1912	35,3003	17,1752
389	15,1212	24,5391	48,4879	14,4181	14,893	24,6712	53,9478	13,4913
390	4,32751	35,406	31,6785	18,1155	7,56524	30,5544	39,7678	16,1402
391	21,9848	21,2883	65,3728	11,8229	24,0827	20,4967	54,0595	13,4734
392	15,4629	24,345	43,0246	15,4565	16,1462	23,9694	29,0662	18,863
393	2,84832	39,039	46,7081	14,743	2,93533	38,7777	23,4813	20,7164
394	19,3104	22,415	55,367	13,2658	17,6838	23,1793	62,4171	12,2247
395	14,1755	25,1001	38,4721	16,4279	14,5138	24,8952	25,5021	19,9993
396	22,8927	20,9369	44,5401	15,1558	23,7849	20,6048	55,4704	13,2496
397	6,96914	31,2672	28,5327	19,024	16,9995	23,5221	17,7849	23,1298
398	6,00563	32,5596	64,5858	11,9281	6,07533	32,4594	16,8176	23,6155
399	8,40363	29,6415	35,0588	17,2349	7,97725	30,0937	20,5652	21,8682
400	4,92984	34,2741	53,8587	13,5057	5,0185	34,1193	18,7866	22,6538
401	15,955	24,0728	22,3512	21,1448	14,5413	24,8787	25,0567	20,1523
402	6,57446	31,7736	36,9798	16,7715	5,60448	33,1601	23,6941	20,638
403	8,52914	29,5127	42,6992	15,5224	10,4271	27,7676	33,5923	17,606
404	7,3054	30,8579	65,979	11,7427	8,01011	30,058	29,2336	18,8132
405	7,87158	30,2096	45,5829	14,9548	7,13584	31,0619	23,4702	20,7205
406	10,0176	28,1155	33,0194	17,7554	10,8277	27,4401	20,7306	21,7986
407	12,6129	26,1145	32,7579	17,8245	14,8262	24,7102	33,9965	17,5021
408	8,64766	29,3928	26,1458	19,7828	11,0393	27,272	28,3667	19,0746
409	3,09179	38,3266	46,0916	14,8584	3,1839	38,0716	26,7897	19,5714
410	2,79195	39,2126	42,4034	15,5828	2,86657	38,9835	10,5298	27,6824
411	4,5226	35,023	69,2558	11,3217	5,92703	32,6741	30,2448	18,5178
412	5,55079	33,2437	35,5237	17,1204	7,95679	30,116	23,1659	20,8338
413	8,00457	30,064	28,4331	19,0543	7,78831	30,3019	20,2038	22,0222
414	9,59245	28,4922	31,4452	18,1797	18,5531	22,7625	36,7788	16,8188
415	4,48234	35,1007	35,2864	17,1787	5,42055	33,4499	19,5332	22,3153
416	4,2516	35,5598	34,5852	17,353	5,2838	33,6719	47,8735	14,5289
417	8,1128	29,9474	56,5731	13,0786	8,55384	29,4876	19,8267	22,1858
418	7,00967	31,2169	34,8252	17,2929	7,75904	30,3346	48,425	14,4294

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
419	7,7816	30,3094	38,0253	16,5293	10,8521	27,4205	34,3874	17,4028
420	15,2026	24,4924	32,0339	18,0186	13,8043	25,3305	24,2721	20,4287
421	11,2328	27,121	46,5899	14,765	11,2619	27,0986	31,1939	18,2494
422	11,0182	27,2886	52,8645	13,6675	12,956	25,8814	24,0099	20,523
423	5,89595	32,7197	22,3161	21,1584	7,54818	30,574	19,5879	22,291
424	21,7881	21,3664	20,055	22,0863	20,7041	21,8097	40,1839	16,0498
425	21,7749	21,3717	33,5614	17,614	19,7148	22,2349	31,986	18,0316
426	3,24452	37,9078	37,1172	16,7393	4,71816	34,6554	27,7063	19,2792
427	2,59239	39,8568	78,1766	10,2693	3,97354	36,1473	16,4674	23,7983
428	7,45962	30,6765	54,9937	13,3245	8,46109	29,5823	24,6724	20,2866
429	14,3105	25,0177	44,2561	15,2113	13,7401	25,371	25,2922	20,0711
430	4,58168	34,9103	30,0283	18,5802	5,68641	33,034	20,834	21,7554
431	6,47611	31,9045	38,5487	16,4106	28,3307	19,0856	36,1029	16,98
432	2,58068	39,8961	72,6662	10,9042	2,76786	39,2879	18,8885	22,6069
433	3,96236	36,1717	25,499	20,0003	5,17526	33,8522	14,9017	24,6661
434	8,61299	29,4277	56,2669	13,1257	10,7816	27,4771	21,7325	21,3886
435	26,6509	19,6166	30,4609	18,4559	29,6208	18,6989	59,9778	12,571
436	5,15819	33,8809	49,5067	14,2375	6,05773	32,4846	11,126	27,204
437	3,12828	38,2247	41,9082	15,6848	3,62129	36,9535	21,4528	21,5011
438	2,67333	39,5898	31,8431	18,0705	4,34958	35,3619	9,77437	28,329
439	13,8065	25,3291	51,9281	13,8228	13,3844	25,5988	28,3554	19,0781
440	17,0759	23,4831	35,2655	17,1838	17,2274	23,4064	36,3669	16,9167
441	9,72987	28,3687	50,1799	14,1202	9,8075	28,2996	28,3583	19,0772
442	19,141	22,4915	52,0681	13,7994	18,5182	22,7788	42,5682	15,5491
443	9,68578	28,4081	51,8569	13,8347	11,7328	26,7428	29,5548	18,7182
444	5,45118	33,401	80,0362	10,0651	6,82195	31,4526	41,6358	15,7415
445	4,09811	35,8791	57,0704	13,0026	16,6839	23,6848	22,8346	20,9589
446	4,90307	34,3214	59,4841	12,6428	17,0469	23,4979	25,2603	20,082
447	7,63836	30,4708	29,2026	18,8224	52,2148	13,7749	35,2036	17,1991
448	14,3414	24,999	55,5262	13,2408	16,7301	23,6608	38,0803	16,5168
449	24,9746	20,1808	71,7579	11,0134	28,3068	19,093	51,1978	13,9458
450	3,01675	38,54	52,5809	13,7142	3,89044	36,3308	27,8425	19,2366
451	15,575	24,2823	51,231	13,9401	16,4892	23,7868	35,1926	17,2018
452	13,6845	25,4063	40,6931	15,9404	19,6112	22,2807	49,1166	14,3062
453	9,22341	28,833	59,5923	12,627	18,3127	22,8758	20,8518	21,7479
454	9,19648	28,8584	71,7536	11,0139	9,17545	28,8783	21,8775	21,3309
455	9,4046	28,664	80,1675	10,0508	9,24829	28,8096	44,0677	15,2484
456	13,5826	25,4712	32,4055	17,9184	21,4425	21,5053	25,2226	20,095
457	10,9572	27,3368	34,8255	17,2929	9,33101	28,7322	17,202	23,4192
458	23,9171	20,5566	66,1818	11,716	27,3005	19,4074	62,3069	12,2401

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
459	5,9282	32,6723	51,7201	13,8576	6,98801	31,2437	42,7053	15,5212
460	3,23035	37,9458	37,5	16,6502	16,3804	23,8443	21,5351	21,4679
461	8,01306	30,0548	44,3378	15,1953	10,2737	27,8963	17,4853	23,2774
462	4,1625	35,7437	25,6284	19,9564	16,7759	23,6371	17,2003	23,4201
463	6,82505	31,4487	56,1193	13,1486	10,5148	27,6948	20,7202	21,8029
464	1,30402	45,8251	59,728	12,6072	1,97793	42,2066	12,0724	26,4949
465	8,79395	29,2471	54,0684	13,4719	11,412	26,9836	22,252	21,1834
466	19,6601	22,2591	41,8636	15,6941	24,2729	20,4284	41,2854	15,8149
467	19,421	22,3654	52,429	13,7394	24,8784	20,2144	39,846	16,1231
468	19,8437	22,1784	74,1562	10,7279	19,8772	22,1637	40,9651	15,8825
469	11,8342	26,668	59,9339	12,5773	13,165	25,7424	53,0133	13,6431
470	22,5756	21,058	43,8367	15,2941	22,1382	21,2279	50,1956	14,1175
471	9,79557	28,3102	56,4095	13,1038	11,6601	26,7968	39,1256	16,2816
472	32,0596	18,0116	83,6165	9,68496	34,1938	17,4518	60,1353	12,5482
473	5,29208	33,6583	50,791	14,0151	6,15207	32,3504	60,9848	12,4264
474	4,07743	35,9231	39,1125	16,2845	5,20321	33,8054	32,9947	17,7619
475	21,4127	21,5174	56,9427	13,022	25,1625	20,1157	45,9144	14,8918
476	4,57651	34,9201	45,0229	15,0621	5,32321	33,6073	9,99153	28,1382
477	1,11244	47,2053	33,2774	17,6878	1,96847	42,2482	10,1619	27,9913
478	16,4276	23,8193	31,8455	18,0698	17,7975	23,1236	48,4704	14,4213
479	10,6607	27,5751	64,5736	11,9297	11,7285	26,7459	29,5405	18,7225
480	9,05213	28,9958	40,7809	15,9217	11,1605	27,1771	20,8555	21,7464
481	4,94891	34,2406	44,8558	15,0944	6,17043	32,3245	42,9095	15,4797
482	12,9604	25,8784	42,6461	15,5332	12,9628	25,8768	24,8379	20,2285
483	8,48759	29,5551	50,3134	14,0971	10,9072	27,3765	22,7476	20,9921
484	8,04606	30,0191	68,9332	11,3622	9,07598	28,9729	44,1893	15,2245
485	7,95844	30,1142	46,1495	14,8475	8,33498	29,7127	37,4684	16,6575
486	12,5314	26,1708	49,7685	14,1917	12,8748	25,936	36,0159	17,0009
487	20,5209	21,8869	47,0399	14,6815	24,7789	20,2492	64,6731	11,9163
488	10,5478	27,6676	57,6093	12,9209	12,3228	26,3166	24,6098	20,3086
489	9,85928	28,2539	57,3753	12,9563	11,2521	27,1061	32,6101	17,8638
490	4,46376	35,1368	39,4472	16,2105	7,35121	30,8036	16,1245	23,9811
491	9,44608	28,6258	47,5493	14,5879	12,1374	26,4483	31,5969	18,1379
492	19,936	22,1381	44,5091	15,1618	24,5628	20,3253	48,2587	14,4593
493	5,99845	32,57	56,053	13,1588	8,53542	29,5063	15,1291	24,5345
494	7,38615	30,7624	65,6213	11,7899	10,8224	27,4443	37,8655	16,5659
495	5,8205	32,8316	46,689	14,7465	8,68509	29,3553	18,3231	22,8708
496	16,0671	24,0121	70,6711	11,146	14,178	25,0985	26,0138	19,8267
497	16,2197	23,93	47,3861	14,6178	20,2433	22,0052	35,8561	17,0395
498	8,45186	29,5918	54,5389	13,3967	8,60183	29,439	23,8871	20,5675

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
499	14,3002	25,0239	71,7144	11,0187	17,1851	23,4277	41,3602	15,7991
500	9,94838	28,1758	58,4877	12,7895	12,3061	26,3284	28,3493	19,0799
501	9,94838	28,1758	46,741	14,7369	12,2981	26,334	26,0413	19,8175
502	20,5376	21,8798	50,6927	14,0319	24,252	20,4358	60,0803	12,5562
503	13,987	25,2163	35,6567	17,088	14,8157	24,7163	31,1846	18,252
504	13,2583	25,6811	58,9089	12,7272	15,5491	24,2967	38,403	16,4435
505	10,8845	27,3946	40,7869	15,9204	12,708	26,0492	34,5741	17,3558
506	9,95643	28,1687	35,807	17,0514	11,0794	27,2405	24,0326	20,5148
507	11,0364	27,2742	29,1413	18,8406	11,8298	26,6713	31,6686	18,1182
508	11,1279	27,2025	38,7885	16,3567	11,958	26,5776	31,3481	18,2066
509	7,73383	30,3629	44,2545	15,2117	8,49201	29,5506	24,5596	20,3264
510	3,45821	37,3538	57,3038	12,9671	10,4431	27,7542	19,7786	22,2069
511	7,32149	30,8388	39,9817	16,0936	10,6808	27,5587	33,2141	17,7043
512	12,5214	26,1777	36,0536	16,9918	16,7524	23,6492	40,668	15,9457
513	10,9914	27,3098	49,3144	14,2713	11,2759	27,0878	37,6328	16,6195
514	8,95211	29,0923	56,5707	13,079	11,319	27,0546	28,7011	18,9728
515	8,10036	29,9607	48,2431	14,4621	10,685	27,5553	18,3242	22,8703
516	6,40559	31,9996	49,6772	14,2077	9,61214	28,4744	21,7241	21,392
517	12,0701	26,4966	59,9375	12,5768	13,7937	25,3372	35,3122	17,1723
518	6,28305	32,1674	33,8991	17,527	7,8	30,2889	32,6838	17,8442
519	12,3928	26,2674	25,7799	19,9052	12,6727	26,0735	25,9945	19,8332
520	13,6146	25,4507	39,3475	16,2325	19,4016	22,3741	39,8315	16,1263
521	8,52848	29,5134	39,3475	16,2325	8,027	30,0397	26,4136	19,6942
522	6,98057	31,253	43,5399	15,353	5,95213	32,6374	21,7748	21,3717
523	11,7986	26,6942	15,2421	24,4699	11,5611	26,8708	24,1659	20,4667
524	11,7986	26,6942	65,4179	11,8169	11,5611	26,8708	24,1659	20,4667
525	10,5443	27,6704	47,1271	14,6654	12,0164	26,5353	22,3682	21,1382
526	5,66067	33,0735	58,7454	12,7513	5,11538	33,9532	12,4074	26,2572
527	20,5379	21,8797	71,4041	11,0563	16,6516	23,7017	43,7739	15,3065
528	16,5906	23,7336	48,7554	14,3703	15,2343	24,4744	42,6221	15,5381
529	17,4903	23,2749	44,8163	15,1021	16,0856	24,0021	28,3544	19,0784
530	18,8043	22,6457	47,0095	14,6871	13,9502	25,2392	21,3172	21,5562
531	14,2045	25,0823	31,087	18,2792	17,0361	23,5034	57,0396	13,0073
532	11,012	27,2935	26,685	19,6055	11,5329	26,8921	22,6626	21,0246
533	16,8533	23,5971	44,6616	15,1321	15,544	24,2996	23,0113	20,892
534	8,05724	30,0071	30,6002	18,4163	7,8245	30,2617	25,3816	20,0404
535	7,37832	30,7717	28,8683	18,9224	7,84649	30,2373	16,3287	23,8718
536	8,93237	29,1115	34,6125	17,3461	10,043	28,0935	20,5842	21,8601
537	7,52474	30,601	47,9854	14,5086	11,0669	27,2503	14,3041	25,0216
538	7,93347	30,1415	45,781	14,9171	7,93673	30,138	12,7141	26,0451

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
539	9,31383	28,7482	55,7619	13,2041	6,60057	31,7392	12,3996	26,2627
540	9,51433	28,5632	26,9082	19,5331	11,4276	26,9717	21,4194	21,5147
541	12,3293	26,312	57,3992	12,9527	13,618	25,4485	46,4195	14,7968
542	12,0192	26,5333	26,3525	19,7144	12,4915	26,1985	20,8005	21,7693
543	5,05306	34,0597	31,7793	18,0879	6,03578	32,5161	36,9635	16,7753
544	2,84929	39,0361	38,593	16,4006	3,83159	36,4632	14,35	24,9938
545	5,94222	32,6518	35,987	17,0079	6,60402	31,7346	32,1987	17,974
546	8,83514	29,2065	69,1399	11,3362	7,72509	30,3727	17,3129	23,3634
547	7,32397	30,8359	44,9243	15,0812	7,49713	30,6329	23,5077	20,7066
548	5,27639	33,6841	48,7639	14,3688	6,13465	32,375	14,677	24,7981
549	16,6725	23,6908	33,0604	17,7446	16,4559	23,8044	35,9234	17,0232
550	3,32567	37,6932	36,9167	16,7863	3,42723	37,4319	12,5398	26,165
551	15,4492	24,3527	58,7534	12,7501	15,5417	24,3008	34,2441	17,4391
552	10,2658	27,9029	77,1243	10,387	12,0694	26,4971	25,9454	19,8496
553	3,78851	36,5614	36,6022	16,8607	3,51489	37,2126	15,0508	24,5796
554	8,57936	29,4617	42,7966	15,5026	10,7417	27,5094	50,3667	14,0879
555	14,1808	25,0968	64,3037	11,9661	13,3457	25,624	21,8971	21,3231
556	9,25111	28,8069	58,8875	12,7303	10,8512	27,4212	21,8808	21,3295
557	15,3621	24,4018	26,9773	19,5108	15,2401	24,4711	56,47	13,0945
558	3,13918	38,1945	36,6416	16,8513	27,949	19,2035	17,434	23,3029
559	3,58668	37,0369	27,1044	19,47	28,0892	19,16	27,5009	19,3439
560	2,36857	40,6411	51,4891	13,8965	2,78183	39,2442	10,1401	28,01
561	12,1899	26,4108	57,6855	12,9095	13,3601	25,6146	23,8485	20,5816
562	0,737388	50,7769	44,9825	15,0699	1,66291	43,7134	17,8213	23,112
563	10,9779	27,3204	52,8934	13,6628	11,0716	27,2466	20,9384	21,7119
564	10,5048	27,7031	49,5657	14,2272	11,843	26,6615	30,2817	18,5072
565	2,87322	38,9634	30,7571	18,3719	3,05759	38,4232	18,9936	22,5587
566	6,16944	32,3259	58,7955	12,7439	7,24092	30,9349	26,5043	19,6645
567	19,6174	22,278	59,6725	12,6153	19,1995	22,465	39,8482	16,1226
568	4,34535	35,3703	58,5381	12,782	5,41888	33,4526	16,2584	23,9093
569	10,0629	28,0763	36,9461	16,7794	19,9173	22,1462	21,817	21,3549
570	12,2353	26,3785	51,6851	13,8635	12,8365	25,9619	25,0141	20,1671
571	9,05664	28,9915	44,9869	15,0691	9,78889	28,3161	27,1886	19,4431
572	11,0714	27,2468	58,1148	12,8451	10,278	27,8927	21,3115	21,5585
573	15,1322	24,5328	33,4147	17,652	14,8187	24,7146	22,5958	21,0503
574	10,1804	27,9755	75,9125	10,5245	11,8164	26,6811	27,3338	19,3968
575	13,2831	25,6648	39,0345	16,3018	17,0747	23,4837	49,1232	14,3051
576	6,48987	31,8861	46,7994	14,726	6,18019	32,3108	11,3735	27,0129
577	16,3107	23,8813	45,1948	15,029	14,6334	24,8239	33,3181	17,6772
578	9,44546	28,6263	45,054	15,0561	10,189	27,9681	28,4165	19,0594

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
579	2,73833	39,3811	32,1402	17,9898	4,48343	35,0986	12,2252	26,3857
580	10,3948	27,7945	62,1804	12,2577	12,2489	26,3689	24,4472	20,3662
581	6,81703	31,4589	25,3468	20,0523	6,44838	31,9418	23,2265	20,8111
582	13,7479	25,3661	37,3802	16,678	12,1731	26,4228	20,2857	21,987
583	18,0713	22,991	51,0297	13,9743	22,1837	21,2101	45,7252	14,9277
584	6,21221	32,2659	50,5217	14,0612	6,92053	31,328	14,1911	25,0905
585	10,7839	27,4753	45,1244	15,0426	9,82428	28,2848	30,0991	18,5597
586	24,1469	20,4736	47,5915	14,5802	24,6173	20,306	56,8291	13,0394
587	11,0242	27,2839	42,0288	15,6599	12,3887	26,2703	33,0997	17,7343
588	12,671	26,0746	45,0853	15,0501	16,7278	23,662	32,3286	17,9391
589	3,45304	37,3668	53,603	13,547	4,76935	34,5616	13,6335	25,4387
590	8,82928	29,2123	51,0645	13,9684	9,64992	28,4403	46,0911	14,8585
591	9,45044	28,6218	39,6676	16,1621	10,6738	27,5644	28,3206	19,0888
592	10,8642	27,4109	44,0334	15,2552	43,5128	15,3585	23,2052	20,8191
593	19,5272	22,318	41,3572	15,7998	19,7257	22,2302	39,3954	16,2219
594	5,14838	33,8974	39,272	16,2491	6,54675	31,8103	27,5408	19,3313
595	11,961	26,5755	39,1934	16,2665	12,124	26,4579	37,174	16,726
596	12,6887	26,0625	17,463	23,2884	14,0658	25,1675	23,6188	20,6656
597	8,21143	29,8424	53,8998	13,4991	10,0745	28,0664	48,5631	14,4047
598	4,0577	35,9652	61,0365	12,419	9,03824	29,0091	22,2099	21,1999
599	5,87792	32,7463	35,2657	17,1838	6,53963	31,8197	26,6614	19,6131
600	8,72867	29,3118	61,2648	12,3866	24,574	20,3213	41,0878	15,8566
601	16,637	23,7093	33,6826	17,5827	15,4828	24,3338	52,1489	13,7859
602	2,80819	39,1623	42,3665	15,5903	3,14355	38,1824	13,1403	25,7587
603	4,13244	35,8067	71,1962	11,0817	16,7066	23,673	47,6816	14,5638
604	3,43475	37,4129	60,9333	12,4337	3,34447	37,6443	17,8062	23,1194
605	13,5956	25,4629	61,6734	12,3288	14,8363	24,7043	61,8329	12,3064
606	22,5897	21,0526	47,9932	14,5072	20,6221	21,8441	34,3181	17,4203
607	14,117	25,136	38,2076	16,4878	12,9298	25,899	28,4315	19,0548
608	20,8026	21,7685	27,5962	19,3138	19,6848	22,2482	37,3439	16,6864
609	7,41707	30,7262	42,1726	15,6302	5,53963	33,2612	15,2849	24,4456
610	5,77964	32,8928	58,4258	12,7987	5,83484	32,8102	23,6092	20,6692
611	7,70943	30,3904	34,5645	17,3582	8,23902	29,8133	18,105	22,9748
612	15,3688	24,398	44,0064	15,2605	16,8151	23,6168	41,7222	15,7235
613	2,30573	40,8746	41,1117	15,8515	15,959	24,0707	11,4887	26,9254
614	15,0036	24,6069	40,7779	15,9223	14,3287	25,0067	35,3193	17,1706
615	8,07254	29,9906	42,8908	15,4835	9,46735	28,6062	24,4776	20,3554
616	7,99952	30,0695	23,6772	20,6442	8,4129	29,6319	17,319	23,3603
617	4,45156	35,1606	24,7357	20,2643	5,47858	33,3575	17,44	23,2999
618	4,09994	35,8752	41,8408	15,6988	5,32982	33,5965	23,401	20,7461

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
619	2,92819	38,7988	40,439	15,9948	2,92929	38,7956	9,63363	28,455
620	2,71582	39,4528	24,9663	20,1837	4,66426	34,7552	10,23	27,9333
621	4,90775	34,3132	30,4439	18,4608	12,879	25,9332	21,0399	21,6699
622	19,3623	22,3917	46,6366	14,7563	19,4821	22,3381	57,3499	12,9602
623	6,14048	32,3668	28,9829	18,888	6,78566	31,499	13,5894	25,4668
624	9,89219	28,225	61,4381	12,362	10,8079	27,456	22,3531	21,144
625	12,8438	25,957	41,6651	15,7353	18,7458	22,6727	40,8047	15,9166
626	11,9831	26,5595	25,8734	19,8737	10,9185	27,3675	27,2976	19,4083
627	25,5602	19,9795	19,6085	22,2819	27,4733	19,3526	47,3583	14,6229
628	12,462	26,219	44,7887	15,1074	13,1085	25,7797	26,7251	19,5924
629	2,73657	39,3867	44,5733	15,1493	3,98358	36,1253	27,5315	19,3342
630	2,32327	40,8088	37,5104	16,6478	2,8873	38,921	15,5192	24,3134
631	14,2214	25,0719	35,7465	17,0661	15,1405	24,528	31,7551	18,0945
632	15,1384	24,5292	35,416	17,1468	15,8797	24,114	39,6844	16,1584
633	7,24551	30,9294	73,4222	10,8142	6,82253	31,4519	18,586	22,7471
634	2,9919	38,6119	50,6054	14,0469	4,57753	34,9182	22,8524	20,9522
635	8,62666	29,414	50,2482	14,1084	7,74431	30,3511	23,4741	20,719
636	18,0617	22,9956	66,57	11,6652	22,59	21,0525	43,5784	15,3454
637	8,33413	29,7136	74,0505	10,7402	9,20949	28,8461	25,25	20,0856
638	3,62946	36,934	44,162	15,2298	4,55458	34,9618	16,6559	23,6995
639	5,96573	32,6175	65,4022	11,819	8,9625	29,0822	27,2286	19,4303
640	6,14346	32,3625	40,5845	15,9636	8,50654	29,5357	16,8559	23,5958
641	3,4454	37,386	58,7545	12,75	44,9609	15,0741	21,9615	21,2976
642	12,2433	26,3728	45,3439	15,0004	13,2198	25,7063	25,242	20,0883
643	6,20162	32,2807	41,4715	15,7758	6,14086	32,3662	30,8579	18,3435
644	19,3882	22,3801	46,8063	14,7247	17,7533	23,1452	30,6886	18,3913
645	12,9312	25,898	51,7185	13,8579	20,7389	21,7951	30,7847	18,3641
646	3,99272	36,1054	47,7049	14,5595	9,25721	28,8012	25,1861	20,1076
647	7,59974	30,5148	62,3526	12,2337	10,3302	27,8486	26,0438	19,8167
648	5,89399	32,7226	40,8548	15,9059	5,97374	32,6059	38,3314	16,4597
649	9,52434	28,5541	69,0752	11,3444	11,8386	26,6648	33,1104	17,7315
650	19,5869	22,2915	71,5864	11,0342	22,6408	21,033	27,3991	19,3761
651	3,38129	37,5491	60,0921	12,5545	16,2425	23,9177	26,911	19,5322
652	10,3404	27,8401	58,0279	12,8581	11,8824	26,6327	32,4037	17,9189
653	23,8006	20,599	59,4878	12,6422	26,8604	19,5486	53,946	13,4916
654	18,9752	22,5671	52,9321	13,6564	16,0564	24,0178	46,2081	14,8364
655	10,2888	27,8835	67,7078	11,518	25,3392	20,055	24,1996	20,4546
656	11,7703	26,715	28,082	19,1622	12,0316	26,5243	50,7186	14,0275
657	18,845	22,6269	55,1453	13,3006	16,7223	23,6649	56,3528	13,1125
658	8,4146	29,6301	37,4324	16,6658	10,8492	27,4229	35,4936	17,1278

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
659	6,00091	32,5665	46,834	14,7196	6,70753	31,5995	21,4619	21,4975
660	10,7804	27,4781	51,7606	13,8508	11,6448	26,8081	29,1166	18,848
661	12,3784	26,2775	58,1179	12,8446	14,429	24,9461	42,0882	15,6476
662	3,70711	36,7501	62,0201	12,2802	4,05742	35,9658	24,2679	20,4302
663	15,9302	24,0864	70,4818	11,1693	15,5358	24,3041	26,4129	19,6945
664	5,03842	34,0849	67,8525	11,4995	5,40518	33,4746	18,2325	22,9139
665	15,985	24,0566	46,7564	14,734	20,6359	21,8384	36,2194	16,952
666	29,7782	18,6528	44,2717	15,2083	31,6877	18,113	60,7191	12,4643
667	4,77913	34,5438	54,9127	13,3373	54,2658	13,4403	22,7738	20,9821
668	21,5061	21,4796	65,0596	11,8646	23,5005	20,7093	41,0001	15,8751
669	9,57829	28,505	75,747	10,5435	19,8411	22,1795	30,7789	18,3658
670	4,17507	35,7175	120,489	6,51186	4,67781	34,73	25,9149	19,8598
671	4,5376	34,9943	45,2465	15,0191	2,71021	39,4707	11,7655	26,7186
672	11,945	26,5871	54,6856	13,3733	12,4249	26,245	18,3352	22,8651
673	10,8785	27,3994	65,1199	11,8565	25,5183	19,9938	41,6187	15,745
674	10,2786	27,8922	80,4481	10,0205	11,9431	26,5884	44,8981	15,0862
675	8,68039	29,36	59,6436	12,6195	15,2663	24,4561	30,5912	18,4189
676	7,23045	30,9475	62,942	12,152	9,82479	28,2843	19,1698	22,4784
677	29,1333	18,843	42,005	15,6648	29,5823	18,7102	45,9642	14,8824
678	10,256	27,9112	37,2343	16,7119	19,0022	22,5547	21,5645	21,456
679	20,257	21,9993	58,5758	12,7764	24,828	20,232	59,3602	12,6609
680	8,85244	29,1895	35,255	17,1864	10,8257	27,4417	25,1421	20,1228
681	12,0005	26,5468	65,2609	11,8377	16,3654	23,8523	37,7896	16,5834
682	13,7684	25,3532	58,4262	12,7986	14,5924	24,8483	44,1407	15,234
683	5,85986	32,7731	55,6512	13,2213	7,04291	31,1758	31,2983	18,2204
684	8,54116	29,5005	58,9949	12,7145	9,24132	28,8161	26,9788	19,5104
685	7,16864	31,0221	36,3076	16,9309	7,36836	30,7834	15,8508	24,1298
686	16,2835	23,8958	40,0458	16,0797	23,0505	20,8772	43,6068	15,3397
687	18,6536	22,7155	61,6602	12,3307	18,8389	22,6297	59,3811	12,6578
688	17,135	23,4531	30,2627	18,5127	16,2746	23,9006	27,354	19,3904
689	15,5118	24,3176	23,5438	20,6933	15,1809	24,5048	27,9194	19,2127
690	9,90466	28,214	49,2733	14,2786	8,72071	29,3198	38,0356	16,527
691	9,55985	28,5218	45,6022	14,9511	9,794	28,3116	23,4004	20,7463
692	29,7418	18,6635	33,943	17,5158	31,7049	18,1083	60,0492	12,5607
693	4,28365	35,4945	75,9004	10,5259	16,5908	23,7335	14,5349	24,8826
694	3,29801	37,7658	54,5035	13,4023	3,11724	38,2554	18,7506	22,6705
695	9,60762	28,4785	42,7644	15,5092	8,16629	29,8903	39,703	16,1543
696	7,37299	30,7779	39,2534	16,2533	7,22108	30,9588	37,303	16,6959
697	7,08934	31,1187	27,12	19,465	5,97069	32,6103	20,0962	22,0685
698	19,2582	22,4385	50,0552	14,1418	15,7921	24,162	46,8644	14,7139

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
699	14,3769	24,9775	45,9958	14,8764	16,8179	23,6154	49,9728	14,1561
700	12,184	26,415	62,5716	12,2033	11,7478	26,7316	46,0493	14,8663
701	10,4225	27,7714	51,6867	13,8632	11,7017	26,7658	20,4343	21,9236
702	4,10413	35,8664	47,3138	14,631	5,318	33,6158	24,0368	20,5133
703	18,4356	22,8176	51,9817	13,8138	17,1158	23,4628	35,9784	17,01
704	12,214	26,3936	54,6394	13,3807	13,2426	25,6913	21,5899	21,4458
705	11,3488	27,0318	63,7094	12,0467	12,1294	26,454	33,2131	17,7046
706	9,71584	28,3812	42,902	15,4813	9,20351	28,8517	18,8311	22,6333
707	11,6931	26,7722	48,9129	14,3423	30,0652	18,5695	22,9233	20,9253
708	9,04023	29,0072	35,5767	17,1075	10,3762	27,81	19,4584	22,3487
709	14,7004	24,7842	25,2761	20,0766	14,4002	24,9634	25,4794	20,007
710	16,2157	23,9321	78,575	10,2251	15,1865	24,5016	34,6251	17,343
711	9,95931	28,1662	32,0894	18,0036	11,5486	26,8802	20,7336	21,7973
712	11,9611	26,5754	46,3436	14,811	13,1728	25,7372	24,3888	20,387
713	14,5249	24,8886	67,7741	11,5095	14,684	24,7939	29,4631	18,7452
714	8,19842	29,8562	36,6783	16,8426	9,7052	28,3907	37,1997	16,72
715	10,5322	27,6804	50,1978	14,1171	15,1241	24,5374	29,383	18,7689
716	10,6833	27,5567	41,6887	15,7304	9,79641	28,3095	23,6585	20,651
717	4,23036	35,6033	51,738	13,8546	4,95513	34,2297	24,4829	20,3535
718	8,43545	29,6086	38,9254	16,3261	13,5025	25,5225	39,3237	16,2377
719	7,5063	30,6223	37,0915	16,7453	7,1595	31,0332	23,4209	20,7387
720	15,1897	24,4998	74,1991	10,7228	15,5273	24,3089	35,785	17,0568
721	9,84272	28,2685	48,7328	14,3744	8,28439	29,7656	18,9115	22,5963
722	5,53743	33,2646	27,4216	19,369	16,91	23,5679	24,6731	20,2863
723	4,45945	35,1452	34,3905	17,402	5,19388	33,821	14,1004	25,1462
724	7,0687	31,144	31,3236	18,2134	6,46437	31,9203	38,1154	16,5088
725	19,6557	22,2611	42,4093	15,5816	22,2523	21,1833	30,8413	18,3481
726	12,1915	26,4097	41,2387	15,8247	13,0911	25,7913	35,6274	17,0951
727	5,17709	33,8491	17,5675	23,2366	5,80015	32,862	26,0533	19,8136
728	4,47081	35,1231	59,5022	12,6401	5,29047	33,6609	16,4633	23,8005
729	3,22574	37,9582	59,7327	12,6066	3,45257	37,3679	36,6827	16,8416
730	6,18731	32,3008	10,833	27,4358	6,56127	31,791	32,6063	17,8648
731	16,7877	23,631	40,482	15,9856	16,8349	23,6066	33,0961	17,7353
732	2,72053	39,4377	40,7741	15,9231	3,0008	38,5861	12,8217	25,9719
733	9,6299	28,4584	29,2889	18,7968	11,2016	27,1452	23,7399	20,6212
734	14,3579	24,989	29,5045	18,733	35,176	17,2059	42,1898	15,6267
735	0,674617	51,5497	29,4139	18,7598	45,2497	15,0185	23,4831	20,7157
736	7,41276	30,7312	35,7749	17,0592	7,78212	30,3088	28,698	18,9738
737	6,74181	31,5553	60,2388	12,5333	35,7609	17,0626	26,8685	19,5459
738	4,73578	34,623	42,7966	15,5026	5,74618	32,9432	21,1033	21,6438

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
739	1,33692	45,6087	42,3665	15,5903	2,0992	41,6897	25,7741	19,9071
740	2,53081	40,0656	44,5733	15,1493	2,65163	39,6605	18,1552	22,9508
741	5,91063	32,6981	61,4381	12,362	7,08331	31,1261	27,3398	19,3949
742	26,5747	19,6414	50,0648	14,1401	29,188	18,8267	47,2857	14,6362
743	15,3621	24,4018	42,702	15,5218	15,2401	24,4711	56,47	13,0945
744	13,5956	25,4629	80,0438	10,0642	14,8363	24,7043	61,8329	12,3064
745	15,1384	24,5292	65,636	11,788	15,8797	24,114	39,6844	16,1584
746	25,5602	19,9795	41,2399	15,8244	27,4733	19,3526	47,3583	14,6229
747	19,5872	22,2913	73,8148	10,7679	17,665	23,1885	31,2311	18,2391
748	14,207	25,0808	62,2969	12,2415	14,2927	25,0285	26,8013	19,5677
749	28,4926	19,0362	61,0781	12,4131	29,4805	18,7401	55,0271	13,3193
750	23,9353	20,55	60,9081	12,4373	20,2151	22,0173	36,1866	16,9598
751	16,8529	23,5973	33,7936	17,5541	16,3376	23,8671	38,2338	16,4819
752	17,221	23,4096	37,8735	16,5641	27,6969	19,2822	29,949	18,6032
753	23,5151	20,7039	43,0736	15,4466	22,811	20,9679	40,9395	15,8879
754	27,1731	19,448	55,1723	13,2964	23,4534	20,7267	36,4285	16,902
755	19,364	22,3909	53,1456	13,6215	18,1628	22,9472	32,1334	17,9917
756	7,2532	30,9202	38,1722	16,4959	7,14268	31,0536	25,6794	19,9391
757	7,40049	30,7456	41,8194	15,7033	9,38516	28,682	20,3889	21,9429
758	8,70294	29,3375	31,486	18,1685	7,85291	30,2302	27,9353	19,2077
759	11,2418	27,114	35,57	17,1091	12,7034	26,0524	24,2042	20,453
760	14,9643	24,6297	28,6287	18,9948	31,4797	18,1702	35,3665	17,159
761	13,4222	25,5743	39,1863	16,2681	20,8953	21,7298	27,575	19,3205
762	7,89205	30,187	62,4004	12,227	10,6388	27,593	21,298	21,564
763	10,4193	27,7741	42,8167	15,4985	10,9325	27,3564	35,6576	17,0878
764	4,05644	35,9679	34,8633	17,2834	5,5169	33,2969	24,7963	20,2431
765	3,22708	37,9546	55,3139	13,2741	3,66429	36,851	23,8287	20,5888
766	5,11867	33,9477	35,5046	17,1251	10,1929	27,9648	25,7162	19,9267
767	3,6457	36,8952	52,7583	13,685	3,76993	36,6041	27,91	19,2156
768	19,3127	22,4139	82,4678	9,80512	15,6354	24,2486	41,4993	15,77
769	7,1541	31,0397	68,7526	11,385	6,46695	31,9168	30,1524	18,5444
770	8,88987	29,1529	42,641	15,5343	8,05299	30,0117	19,6206	22,2766
771	21,3864	21,5281	30,1903	18,5335	22,3808	21,1333	37,7747	16,5868
772	4,87556	34,3703	35,4217	17,1454	5,61346	33,1462	37,252	16,7078
773	22,0747	21,2529	24,5898	20,3157	18,6847	22,7011	46,0736	14,8618
774	25,7287	19,9224	57,8556	12,8839	29,2157	18,8185	47,3664	14,6214
775	9,2314	28,8254	44,3789	15,1873	8,4288	29,6155	31,8386	18,0717
776	2,61865	39,7693	36,7475	16,8262	2,92718	38,8018	12,9376	25,8938
777	7,10657	31,0976	37,2352	16,7117	9,7523	28,3487	46,3715	14,8058
778	6,94074	31,3027	44,3925	15,1846	5,24808	33,7308	26,6111	19,6295

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
779	2,78571	39,2321	47,691	14,5621	3,03773	38,4798	37,1525	16,731
780	12,515	26,1822	45,0459	15,0577	20,3062	21,9782	26,2286	19,7553
781	2,65039	39,6646	46,1039	14,8561	22,6639	21,0241	15,7997	24,1578
782	6,39655	32,0119	35,2679	17,1832	6,50494	31,8659	26,9195	19,5295
783	8,86752	29,1748	35,2686	17,183	8,70487	29,3356	24,373	20,3926
784	13,7866	25,3416	38,3018	16,4664	34,9505	17,2617	29,5172	18,7293
785	12,4713	26,2126	55,3411	13,2698	13,7057	25,3928	34,0317	17,4931
786	11,3069	27,0639	39,8125	16,1304	9,47504	28,5992	16,0916	23,9988
787	13,2265	25,7019	66,4525	11,6806	14,529	24,8861	49,9694	14,1567
788	9,44308	28,6285	22,1086	21,2396	28,9421	18,9002	30,7262	18,3806
789	9,44776	28,6242	56,945	13,0217	28,9223	18,9062	30,7083	18,3857
790	3,97215	36,1503	43,3125	15,3985	5,04645	34,0711	25,5382	19,987
791	9,1798	28,8741	44,6979	15,1251	8,82037	29,2211	25,2651	20,0804
792	7,4535	30,6836	44,4267	15,1779	7,24628	30,9285	22,272	21,1756
793	8,98934	29,0562	72,813	10,8866	9,44237	28,6292	24,4814	20,3541
794	5,30549	33,6363	45,6487	14,9422	6,20637	32,274	19,0222	22,5456
795	21,3574	21,5398	59,2687	12,6743	20,1524	22,0443	42,4051	15,5824
796	14,5514	24,8727	36,4586	16,8948	14,8161	24,7161	42,0115	15,6634
797	15,6063	24,2648	52,8266	13,6737	15,2234	24,4806	30,1584	18,5426
798	14,1124	25,1388	60,5354	12,4906	14,6282	24,827	27,0979	19,4721
799	15,503	24,3225	78,6719	10,2144	14,3595	24,988	30,3476	18,4883
800	9,88429	28,2319	43,6638	15,3284	10,4926	27,7132	20,3812	21,9462
801	25,2955	20,0699	44,921	15,0818	28,2235	19,1186	55,4719	13,2493
802	8,23981	29,8125	31,5759	18,1437	10,4741	27,7285	15,6627	24,2334
803	8,83546	29,2062	43,2079	15,4195	10,7333	27,5161	23,1247	20,8493
804	18,1694	22,944	51,7162	13,8583	15,943	24,0794	41,793	15,7087
805	6,59961	31,7404	23,6426	20,6569	28,1188	19,1509	26,3093	19,7286
806	5,39609	33,4892	64,7371	11,9077	5,66961	33,0597	22,7017	21,0096
807	10,1548	27,9973	58,9898	12,7153	11,3388	27,0395	27,1163	19,4662
808	9,17167	28,8818	56,7486	13,0517	8,15674	29,9005	24,4623	20,3609
809	7,85692	30,2258	29,9006	18,6172	23,8947	20,5648	29,9244	18,6103
810	8,06428	29,9995	46,942	14,6996	7,09546	31,1112	33,0924	17,7362
811	7,86509	30,2167	27,4306	19,3661	7,76143	30,332	30,7517	18,3734
812	18,2303	22,9149	38,6187	16,3948	16,7571	23,6468	48,3204	14,4482
813	14,9492	24,6384	29,302	18,7929	15,3481	24,4097	37,0885	16,746
814	16,644	23,7057	12,6781	26,0697	15,9197	24,0921	26,3669	19,7096
815	8,40515	29,6399	36,0764	16,9863	7,71538	30,3837	27,679	19,2878
816	14,5468	24,8755	41,0085	15,8733	14,7217	24,7717	38,8961	16,3327
817	4,11137	35,8511	38,5893	16,4015	4,88395	34,3554	14,1395	25,1222
818	9,5329	28,5463	65,822	11,7634	10,1281	28,0203	31,2124	18,2443

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
819	5,92751	32,6734	28,7947	18,9445	6,54875	31,8076	19,6102	22,2811
820	2,7642	39,2994	61,6216	12,3361	3,00796	38,5654	47,8779	14,5281
821	9,30129	28,7599	54,2584	13,4415	10,8452	27,4261	28,3626	19,0759
822	5,23974	33,7446	43,4833	15,3644	5,13669	33,9171	13,8424	25,3065
823	6,98536	31,247	49,9623	14,158	6,52042	31,8453	24,3353	20,4061
824	8,025	30,0419	32,5817	17,8713	10,2752	27,895	35,7146	17,0739
825	3,48608	37,2841	34,9241	17,2683	5,83469	32,8104	17,8499	23,0981
826	20,3137	21,975	32,7335	17,8309	17,6944	23,1741	35,3689	17,1584
827	9,55695	28,5244	32,2646	17,9563	18,787	22,6536	36,5777	16,8665
828	6,37373	32,0429	35,0271	17,2427	6,4532	31,9353	19,3289	22,4066
829	13,9805	25,2203	43,1093	15,4394	26,8039	19,5669	26,6261	19,6247
830	7,4448	30,6937	25,6086	19,9631	7,48398	30,6482	40,9098	15,8943
831	5,83652	32,8077	26,8612	19,5483	5,9606	32,625	21,6863	21,4071
832	7,74205	30,3537	55,4538	13,2522	7,42529	30,7165	43,1419	15,4328
833	2,96418	38,6927	34,7275	17,3173	3,05737	38,4238	32,6518	17,8527
834	3,97246	36,1496	47,4364	14,6086	4,93582	34,2636	24,3034	20,4175
835	11,9698	26,5691	23,4703	20,7204	11,2954	27,0727	26,3746	19,7071
836	3,69798	36,7715	49,3217	14,2701	3,78316	36,5737	19,3668	22,3897
837	4,02242	36,0411	42,6615	15,5301	4,81315	34,4822	18,3039	22,8799
838	7,71053	30,3891	23,841	20,5843	6,39393	32,0154	40,8224	15,9128
839	6,93109	31,3148	26,7007	19,6004	6,60921	31,7278	22,9482	20,9158
840	12,0312	26,5246	27,7526	19,2647	11,5566	26,8742	29,5127	18,7306
841	4,66779	34,7486	42,8017	15,5016	5,58095	33,1966	20,8563	21,7461
842	4,45591	35,1521	27,7508	19,2653	23,2916	20,7868	23,8664	20,5751
843	6,61064	31,7259	27,0623	19,4835	6,26183	32,1968	23,5751	20,6817
844	2,37783	40,6072	36,8133	16,8107	2,64486	39,6827	33,1255	17,7276
845	4,16479	35,7389	27,76	19,2624	16,4212	23,8227	10,6327	27,5979
846	2,55106	39,9964	26,709	19,5976	2,79467	39,2042	36,0625	16,9897
847	7,28408	30,8833	28,1846	19,1306	6,59787	31,7427	33,0819	17,739
848	2,63273	39,7227	46,7835	14,729	2,72508	39,4232	30,2443	18,5179
849	3,54353	37,1421	30,8044	18,3586	3,98316	36,1262	22,6168	21,0422
850	4,23569	35,5923	31,8537	18,0676	8,25203	29,7996	15,5823	24,2782
851	2,1278	41,5722	36,1408	16,9709	2,41138	40,4855	6,82525	31,4484
852	3,77178	36,5999	39,9335	16,1041	5,12118	33,9434	14,5044	24,9008
853	5,72681	32,9726	50,0844	14,1368	5,96196	32,623	24,046	20,5099
854	12,7826	25,9984	32,2136	17,97	11,7479	26,7316	25,5097	19,9967
855	3,03873	38,477	51,5177	13,8917	3,32316	37,6998	18,2247	22,9176
856	6,50839	31,8613	21,5753	21,4517	17,1811	23,4298	16,2366	23,9209
857	4,65491	34,7726	22,8454	20,9548	4,40092	35,2599	27,4125	19,3718
858	4,51291	35,0417	30,0461	18,575	5,42356	33,4451	38,1584	16,499

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
859	8,07462	29,9884	46,4787	14,7857	23,9082	20,5599	25,4119	20,0301
860	8,78311	29,2578	27,815	19,2452	7,53671	30,5872	21,0627	21,6605
861	15,0112	24,6025	34,3733	17,4064	13,6569	25,4237	33,8347	17,5436
862	2,3311	40,7796	27,0943	19,4732	3,041	38,4705	14,3348	25,003
863	4,97346	34,1976	75,2288	10,6031	5,40099	33,4813	28,8375	18,9317
864	4,42781	35,207	32,9109	17,784	5,34613	33,57	27,5002	19,3441
865	6,11971	32,3962	67,6333	11,5276	6,35083	32,0742	20,8555	21,7464
866	6,81233	31,4649	52,3073	13,7596	6,77736	31,5096	35,0029	17,2487
867	3,69034	36,7895	39,646	16,1668	5,15569	33,8851	17,8351	23,1053
868	4,0156	36,0558	59,6231	12,6225	4,67571	34,7339	18,7742	22,6596
869	6,26154	32,1972	58,8202	12,7403	6,53839	31,8214	16,1347	23,9756
870	2,57703	39,9084	65,7364	11,7747	16,3463	23,8624	20,6949	21,8135
871	3,64358	36,9002	32,5773	17,8725	4,57883	34,9157	23,5331	20,6972
872	5,01108	34,1322	51,1479	13,9542	2,13957	41,5243	9,13115	28,9203
873	3,96662	36,1624	31,7034	18,1087	5,39689	33,4879	27,1632	19,4512
874	16,5175	23,7719	92,4376	8,81383	22,7967	20,9734	29,3404	18,7815
875	4,47012	35,1244	31,6762	18,1161	4,89063	34,3435	22,0885	21,2475
876	15,9922	24,0527	40,3283	16,0186	13,9413	25,2447	23,4249	20,7373
877	12,6203	26,1094	44,9461	15,077	12,2029	26,4016	20,8511	21,7482
878	8,96072	29,0839	73,265	10,8329	9,60955	28,4767	19,9083	22,1501
879	4,17951	35,7083	46,2197	14,8343	3,07476	38,3746	9,88789	28,2287
880	7,90647	30,1711	55,1909	13,2935	48,3643	14,4403	29,5008	18,7341
881	6,16487	32,3323	81,938	9,8611	5,31424	33,622	23,6089	20,6693
882	12,1485	26,4404	69,663	11,2708	13,5063	25,5201	34,919	17,2696
883	8,68881	29,3516	47,5317	14,5911	7,4522	30,6851	19,0879	22,5156
884	12,1778	26,4194	57,0451	13,0064	11,5094	26,9097	18,5902	22,7451
885	11,564	26,8687	58,4723	12,7918	11,9401	26,5907	45,8798	14,8984
886	13,1499	25,7524	76,3508	10,4745	53,8337	13,5097	29,4552	18,7476
887	11,5776	26,8584	56,9691	13,018	16,728	23,6619	32,1222	17,9947
888	13,741	25,3704	47,2655	14,6399	19,6674	22,2559	19,678	22,2512
889	10,0023	28,1288	35,1275	17,2179	9,96544	28,1609	18,0762	22,9887
890	8,58905	29,4519	46,2627	14,8262	6,56564	31,7853	27,6052	19,311
891	7,15289	31,0412	53,451	13,5717	6,72218	31,5806	43,0503	15,4513
892	11,0666	27,2505	57,8446	12,8855	10,1672	27,9867	26,0465	19,8158
893	13,2311	25,6989	66,8612	11,6273	13,8501	25,3017	30,9417	18,3199
894	10,9231	27,3639	58,0797	12,8503	12,0759	26,4924	23,2148	20,8155
895	3,499	37,2519	39,309	16,241	4,70459	34,6804	44,7881	15,1076
896	12,3887	26,2703	58,9694	12,7183	14,3791	24,9761	28,2316	19,1161
897	20,4139	21,9323	64,8491	11,8927	17,4438	23,298	61,9384	12,2916
898	24,7959	20,2432	44,7701	15,111	24,5272	20,3378	44,3429	15,1943

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
899	15,0739	24,5663	80,8408	9,97819	14,175	25,1003	36,657	16,8477
900	3,28629	37,7967	83,9832	9,64696	4,63742	34,8053	20,8707	21,7401
901	13,0668	25,8074	30,6949	18,3895	15,633	24,25	41,3875	15,7934
902	9,19863	28,8563	79,8443	10,0859	11,1238	27,2058	26,9974	19,5044
903	19,2575	22,4388	60,3448	12,518	22,2086	21,2004	35,1207	17,2195
904	4,42929	35,2041	59,2872	12,6716	5,18167	33,8414	24,2294	20,4439
905	30,0341	18,5785	33,8366	17,5431	29,5905	18,7078	50,5692	14,0531
906	11,8178	26,6801	33,5672	17,6125	11,748	26,7315	31,6324	18,1282
907	7,71984	30,3786	47,2256	14,6473	11,342	27,037	21,881	21,3294
908	23,1944	20,8231	67,7664	11,5105	26,9168	19,5303	48,8601	14,3517
909	17,0376	23,5027	44,1038	15,2413	22,8033	20,9708	45,7955	14,9143
910	4,51417	35,0392	61,5854	12,3413	5,2807	33,677	17,8125	23,1163
911	12,0983	26,4763	60,6936	12,4679	11,0141	27,2918	20,0298	22,0973
912	4,27763	35,5067	61,0605	12,4156	5,25064	33,7266	23,5233	20,7009
913	18,0397	23,0062	57,4894	12,939	17,1583	23,4413	30,5297	18,4364
914	25,344	20,0533	45,3027	15,0083	27,5501	19,3283	44,9586	15,0745
915	6,94541	31,2968	51,0767	13,9663	10,2168	27,9445	23,4358	20,7332
916	9,20288	28,8523	79,0572	10,172	11,3242	27,0507	30,7737	18,3672
917	18,1124	22,9713	34,95	17,2619	21,4691	21,4945	29,0773	18,8597
918	11,4497	26,9549	39,5771	16,1819	17,5697	23,2355	30,0159	18,5838
919	13,2503	25,6863	39,053	16,2977	16,4935	23,7845	30,9039	18,3305
920	4,72967	34,6342	36,704	16,8365	4,51162	35,0441	12,3601	26,2903
921	11,5481	26,8806	43,5888	15,3433	12,8614	25,945	38,789	16,3566
922	20,8612	21,744	40,5741	15,9658	25,3353	20,0563	46,7702	14,7314
923	11,4885	26,9256	51,6377	13,8715	12,2865	26,3422	21,9256	21,3118
924	13,7986	25,3341	49,5223	14,2348	21,42	21,5144	35,203	17,1992
925	9,78698	28,3178	64,3289	11,9627	11,8731	26,6395	35,0276	17,2426
926	11,8803	26,6343	54,5429	13,396	19,8561	22,1729	30,1297	18,5509
927	7,25413	30,9191	49,8912	14,1703	6,70805	31,5989	26,4308	19,6886
928	13,0795	25,799	68,1522	11,4612	13,575	25,476	33,0648	17,7435
929	14,5852	24,8526	66,4348	11,6829	15,7199	24,2018	35,4673	17,1342
930	12,5316	26,1707	58,6174	12,7703	26,5017	19,6653	46,2698	14,8248
931	18,6604	22,7124	36,5506	16,8729	17,1151	23,4632	30,93	18,3232
932	20,633	21,8396	48,2888	14,4539	28,6983	18,9737	41,3001	15,8118
933	16,9734	23,5354	56,9239	13,0249	38,9198	16,3274	59,9115	12,5806
934	14,1566	25,1116	46,9754	14,6934	21,401	21,5221	35,9629	17,0137
935	18,6305	22,7263	40,3835	16,0067	16,5546	23,7524	43,3762	15,3858
936	16,9997	23,522	43,7107	15,3191	16,8176	23,6155	44,0783	15,2463
937	16,789	23,6303	38,9314	16,3248	16,7198	23,6662	32,975	17,7671
938	13,0816	25,7976	45,5609	14,959	12,048	26,5125	32,8404	17,8026

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
939	11,4826	26,93	47,5989	14,5789	13,1331	25,7635	28,7943	18,9447
940	15,1939	24,4974	38,2962	16,4677	21,7126	21,3966	38,5233	16,4163
941	15,1198	24,5399	48,9799	14,3304	15,1928	24,498	28,88	18,9189
942	11,4258	26,973	41,5669	15,7558	25,9035	19,8636	39,7689	16,1399
943	9,96178	28,1641	56,5794	13,0776	19,9063	22,151	28,3533	19,0787
944	9,93948	28,1835	42,7991	15,5021	19,9902	22,1145	23,8409	20,5843
945	17,0592	23,4916	43,4707	15,3669	17,0584	23,4921	38,2471	16,4788
946	9,82052	28,2881	54,0877	13,4688	8,01866	30,0488	18,1906	22,9339
947	10,8858	27,3936	52,7485	13,6866	10,0052	28,1263	39,3717	16,2271
948	15,2683	24,455	44,3605	15,1909	14,9593	24,6326	32,463	17,903
949	13,4423	25,5613	47,9211	14,5203	14,5102	24,8973	40,6409	15,9515
950	16,3704	23,8496	50,7726	14,0182	16,0128	24,0414	29,0984	18,8534
951	9,6007	28,4847	45,73	14,9268	29,5612	18,7164	32,2775	17,9528
952	12,4785	26,2075	31,7622	18,0926	37,761	16,5899	27,9209	19,2122
953	13,0615	25,8109	39,4688	16,2057	14,2193	25,0732	33,973	17,5081
954	15,183	24,5036	48,2122	14,4677	26,7479	19,585	33,7673	17,5609
955	7,6142	30,4983	56,5307	13,0851	18,4535	22,8092	34,9719	17,2564
956	12,0983	26,4763	35,4413	17,1406	21,0185	21,6788	46,8242	14,7214
957	16,3803	23,8444	49,0908	14,3108	15,7528	24,1837	32,3847	17,924
958	12,1266	26,456	56,5879	13,0763	26,1799	19,7715	50,9346	13,9905
959	8,0347	30,0314	45,2448	15,0194	8,37022	29,6761	20,3443	21,962
960	11,0735	27,2451	47,2125	14,6497	12,9558	25,8815	31,9447	18,0428
961	18,5246	22,7758	39,1561	16,2748	18,5761	22,7517	43,2377	15,4135
962	14,2879	25,0315	58,3973	12,803	26,5661	19,6442	28,0686	19,1664
963	15,6681	24,2305	44,2072	15,221	38,6418	16,3897	31,2242	18,241
964	10,9376	27,3524	27,9751	19,1954	12,9244	25,9026	31,4071	18,1902
965	16,2706	23,9027	39,7588	16,1421	15,3704	24,3971	25,9316	19,8542
966	14,5586	24,8684	73,8147	10,7679	14,5568	24,8695	33,5195	17,6249
967	11,4077	26,9868	72,3719	10,9394	9,73378	28,3652	20,3619	21,9544
968	13,2434	25,6908	37,2571	16,7066	14,2075	25,0804	31,4875	18,168
969	4,92412	34,2842	39,5631	16,185	5,82242	32,8287	49,1562	14,2992
970	17,7153	23,1638	48,534	14,4099	15,6985	24,2137	29,8545	18,6306
971	13,0598	25,8121	44,6449	15,1354	14,4529	24,9317	34,2242	17,4441
972	2,60045	39,8298	49,6319	14,2156	2,7391	39,3786	11,8255	26,6744
973	6,77688	31,5102	45,0022	15,0661	7,42011	30,7226	27,8294	19,2407
974	27,4811	19,3501	32,8607	17,7973	30,1596	18,5423	51,7288	13,8562
975	24,9484	20,19	43,0307	15,4552	28,3126	19,0912	49,7613	14,193
976	7,04059	31,1786	35,8625	17,038	7,17288	31,0169	33,8513	17,5393
977	11,7221	26,7507	41,1149	15,8508	13,3047	25,6507	23,3299	20,7725
978	13,0384	25,8263	42,1102	15,6431	13,6921	25,4014	33,8657	17,5356

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
979	11,4166	26,9801	55,0632	13,3136	11,0238	27,2842	27,5264	19,3358
980	12,0561	26,5067	55,368	13,2656	13,5546	25,489	35,3225	17,1698
981	10,351	27,8312	26,6045	19,6317	11,9717	26,5677	31,384	18,1966
982	9,71126	28,3853	38,4231	16,439	11,6584	26,798	25,2639	20,0808
983	10,7376	27,5127	31,0442	18,2912	12,6198	26,1098	34,1012	17,4754
984	7,31604	30,8453	44,2443	15,2137	8,02221	30,0449	18,3527	22,8568
985	5,79978	32,8626	49,8221	14,1824	6,86045	31,4038	28,4758	19,0413
986	3,09105	38,3287	39,4426	16,2115	22,7884	20,9765	29,3084	18,7909
987	2,59493	39,8483	59,5589	12,6319	3,03643	38,4835	9,02477	29,0221
988	8,69446	29,346	58,1595	12,8384	7,89724	30,1813	18,2173	22,9211
989	4,26405	35,5343	21,0493	21,6661	4,6399	34,8006	12,7795	26,0006
990	9,21865	28,8375	42,0903	15,6472	10,4628	27,7378	36,8751	16,7961
991	6,31996	32,1165	42,983	15,4649	7,14294	31,0533	24,6761	20,2853
992	8,62127	29,4194	15,814	24,15	9,79697	28,309	54,2278	13,4464
993	11,7276	26,7467	38,0359	16,5269	13,8928	25,275	33,8451	17,5409
994	19,7844	22,2043	48,5235	14,4118	24,7634	20,2546	42,8504	15,4917
995	4,50354	35,0597	56,1803	13,1391	5,78576	32,8836	29,7457	18,6623
996	14,7646	24,7464	14,6677	24,8036	16,1655	23,959	31,4654	18,1741
997	18,2204	22,9197	40,1131	16,0651	17,5046	23,2677	49,708	14,2023
998	6,83461	31,4365	82,9656	9,75284	7,37716	30,773	15,5506	24,2959
999	5,09289	33,9915	27,7286	19,2722	17,0673	23,4875	24,2479	20,4373
1000	4,22221	35,62	47,2241	14,6475	5,89133	32,7265	35,3499	17,163
1001	1,57057	44,2097	23,6852	20,6413	1,81887	42,9348	15,2464	24,4675
1002	4,35384	35,3533	30,7764	18,3664	4,83458	34,4436	34,6048	17,3481
1003	8,86642	29,1758	69,1308	11,3374	10,9928	27,3087	30,2576	18,5141
1004	12,1636	26,4296	48,3337	14,4458	12,5846	26,134	34,6806	17,3291
1005	0,844918	49,5945	50,9637	13,9856	1,12401	47,1154	22,4707	21,0985
1006	12,372	26,282	43,374	15,3862	13,7766	25,348	28,1412	19,1439
1007	16,9958	23,524	45,352	14,9989	17,9462	23,0514	43,0183	15,4577
1008	6,68455	31,6294	55,7611	13,2042	6,5112	31,8576	25,7433	19,9175
1009	10,1228	28,0248	36,9036	16,7894	10,4437	27,7537	30,9545	18,3163
1010	6,36445	32,0556	62,5603	12,2048	6,94337	31,2994	14,3767	24,9776
1011	8,38779	29,6579	40,8297	15,9113	10,1616	27,9915	22,2664	21,1778
1012	11,6525	26,8024	54,3413	13,4282	25,6361	19,9537	24,5107	20,3437
1013	9,91342	28,2063	75,5702	10,5638	11,5161	26,9047	23,0854	20,8641
1014	3,25307	37,8849	67,8533	11,4994	4,03005	36,0246	38,4317	16,437
1015	6,94124	31,3021	31,959	18,0389	6,19442	32,2908	30,0079	18,5861
1016	10,8668	27,4087	34,2661	17,4335	12,5005	26,1923	28,1721	19,1344
1017	11,4224	26,9757	71,0802	11,0958	30,0689	18,5685	20,4031	21,9369
1018	5,78979	32,8775	52,5305	13,7226	23,3984	20,7471	23,4289	20,7358

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1019	13,7762	25,3482	49,1519	14,3	14,7128	24,7769	36,2014	16,9563
1020	13,0865	25,7944	44,7526	15,1144	13,1849	25,7293	33,1171	17,7298
1021	8,88279	29,1598	40,5711	15,9665	8,20419	29,8501	19,5092	22,326
1022	5,6979	33,0165	20,9424	21,7103	22,4147	21,1202	35,8913	17,031
1023	9,80284	28,3038	49,6629	14,2102	17,9884	23,031	21,2471	21,5848
1024	3,91588	36,2742	54,9327	13,3342	4,98077	34,1849	15,5424	24,3004
1025	7,30298	30,8608	37,9879	16,5379	7,64377	30,4646	21,2925	21,5663
1026	15,3949	24,3833	41,2607	15,8201	30,9035	18,3307	35,1125	17,2216
1027	8,80284	29,2383	33,0147	17,7567	10,8843	27,3948	29,3827	18,769
1028	2,02226	42,0141	41,6999	15,7281	3,88844	36,3353	14,8725	24,6831
1029	6,41221	31,9907	69,7475	11,2602	5,9919	32,5795	17,0727	23,4847
1030	5,58435	33,1914	36,1846	16,9603	6,48997	31,886	16,2289	23,925
1031	3,17693	38,0907	29,6943	18,6773	3,04183	38,4681	19,5137	22,324
1032	8,70495	29,3355	42,4883	15,5654	10,2519	27,9147	51,012	13,9774
1033	12,3232	26,3164	24,2612	20,4326	13,4608	25,5494	28,076	19,1641
1034	2,53018	40,0678	28,7825	18,9482	38,5816	16,4032	25,5752	19,9744
1035	10,4213	27,7724	48,4844	14,4188	10,7079	27,5367	23,1523	20,8389
1036	2,21608	41,2191	60,4178	12,5075	2,71139	39,467	11,9624	26,5744
1037	5,65408	33,0836	25,8424	19,8842	6,12656	32,3865	20,5344	21,8812
1038	14,214	25,0765	48,3126	14,4496	26,0626	19,8105	25,7196	19,9255
1039	3,29681	37,7689	37,0108	16,7642	16,6151	23,7208	21,0711	21,6571
1040	11,2322	27,1215	48,1631	14,4765	13,2934	25,6581	19,401	22,3743
1041	7,51058	30,6173	51,6865	13,8633	8,63778	29,4028	19,5388	22,3128
1042	5,84803	32,7906	67,5167	11,5426	5,89301	32,7241	15,9428	24,0795
1043	7,43976	30,6996	42,2843	15,6072	7,13765	31,0597	14,9459	24,6404
1044	2,39693	40,5377	63,3712	12,093	32,2063	17,972	19,1552	22,4851
1045	10,7702	27,4863	47,2798	14,6373	11,754	26,7271	26,5149	19,661
1046	3,59212	37,0238	70,5371	11,1625	3,90848	36,2906	28,7879	18,9466
1047	7,00524	31,2223	42,4635	15,5705	6,97419	31,2609	18,9521	22,5777
1048	10,0028	28,1284	53,8643	13,5048	7,73145	30,3656	13,5696	25,4794
1049	10,7725	27,4845	16,9823	23,5309	12,4434	26,232	30,1334	18,5498
1050	26,8997	19,5359	54,1833	13,4535	29,9077	18,6151	38,9363	16,3237
1051	14,6983	24,7854	40,0463	16,0795	13,942	25,2443	31,2704	18,2281
1052	17,7921	23,1262	38,5407	16,4124	14,8958	24,6695	25,8674	19,8757
1053	13,2386	25,6939	55,9092	13,1811	13,3797	25,6019	30,524	18,438
1054	17,5956	23,2227	43,067	15,4479	16,7404	23,6555	37,9332	16,5504
1055	8,05102	30,0138	20,5732	21,8648	8,06602	29,9976	21,5714	21,4532
1056	5,55769	33,2329	45,5163	14,9675	6,0868	32,443	23,8947	20,5648
1057	11,5996	26,8419	49,2688	14,2794	11,3032	27,0668	31,4351	18,1825
1058	0,896086	49,0838	71,8951	10,9968	1,21228	46,4587	22,5925	21,0515

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1059	12,7451	26,024	34,9271	17,2676	14,0066	25,2041	24,6159	20,3065
1060	9,80839	28,2988	57,2118	12,9811	10,3321	27,8471	31,2682	18,2287
1061	6,55578	31,7983	72,4217	10,9334	6,7786	31,508	24,1542	20,4709
1062	9,73946	28,3601	52,5334	13,7221	10,4355	27,7605	38,9159	16,3283
1063	5,32347	33,6069	45,5747	14,9563	4,81185	34,4846	21,3934	21,5252
1064	1,65715	43,7436	40,4131	16,0004	1,91572	42,4842	19,1133	22,5041
1065	7,383	30,7661	37,4854	16,6536	6,82151	31,4532	16,4025	23,8326
1066	8,12402	29,9354	61,278	12,3847	10,715	27,531	48,7266	14,3755
1067	8,802	29,2392	43,2548	15,4101	10,5419	27,6724	29,4025	18,7631
1068	9,41755	28,652	39,6003	16,1768	10,8466	27,4249	44,56	15,1519
1069	8,02594	30,0409	42,2827	15,6076	10,5115	27,6975	25,6258	19,9573
1070	25,9553	19,8463	68,2842	11,4444	30,3033	18,501	52,153	13,7852
1071	8,92848	29,1153	61,4437	12,3613	9,91958	28,2009	22,5139	21,0818
1072	4,94988	34,2389	45,4712	14,9761	5,33485	33,5884	39,3315	16,236
1073	9,58682	28,4973	46,6189	14,7596	8,91954	29,124	24,7622	20,255
1074	6,12426	32,3897	56,0278	13,1627	7,00263	31,2256	26,7549	19,5827
1075	15,8583	24,1257	77,0714	10,3929	17,1897	23,4254	32,7482	17,8271
1076	2,92517	38,8078	53,4932	13,5648	4,44485	35,1737	31,2923	18,2221
1077	13,0312	25,8311	20,4877	21,901	11,9287	26,5989	19,5537	22,3062
1078	3,05276	38,4369	56,7012	13,059	4,548	34,9744	23,3602	20,7613
1079	13,5351	25,5016	36,6658	16,8456	13,0454	25,8217	23,0013	20,8958
1080	5,40427	33,4761	32,2695	17,955	17,6954	23,1736	25,5969	19,9671
1081	6,4389	31,9546	33,1697	17,716	6,34665	32,0799	25,4546	20,0155
1082	9,05337	28,9946	60,5473	12,4889	8,40165	29,6435	22,7756	20,9814
1083	10,8637	27,4112	21,2839	21,5698	12,0267	26,5279	19,4356	22,3589
1084	17,9882	23,031	43,4223	15,3765	20,4384	21,9219	29,2876	18,7971
1085	10,4295	27,7656	44,3072	15,2013	8,37384	29,6723	49,7139	14,2013
1086	3,67836	36,8177	73,7463	10,776	3,07978	38,3604	38,3358	16,4587
1087	2,60837	39,8034	74,1943	10,7234	2,65837	39,6385	16,5071	23,7774
1088	3,79246	36,5524	65,3942	11,82	3,63113	36,93	30,6455	18,4035
1089	7,39514	30,7519	25,3183	20,0621	6,83345	31,438	27,951	19,2029
1090	10,1091	28,0365	59,8437	12,5904	12,1972	26,4056	63,4473	12,0825
1091	8,13112	29,9278	33,1784	17,7137	9,3489	28,7156	34,3118	17,4219
1092	6,89378	31,3617	28,4801	19,04	7,9552	30,1178	20,0586	22,0848
1093	12,8531	25,9506	51,9209	13,824	13,3259	25,6369	41,1351	15,8465
1094	4,88967	34,3452	67,9761	11,4837	5,59884	33,1688	17,7708	23,1367
1095	20,4546	21,915	26,8941	19,5376	20,2447	22,0046	43,3519	15,3906
1096	5,10089	33,9779	38,7313	16,3696	44,7097	15,1228	28,2745	19,1029
1097	10,7219	27,5254	57,5787	12,9256	11,7427	26,7355	27,686	19,2856
1098	2,57973	39,8993	33,4858	17,6336	2,6985	39,5084	15,2512	24,4647

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1099	11,4585	26,9483	47,3079	14,6321	12,2387	26,3761	27,0667	19,4821
1100	4,00813	36,072	63,7401	12,0426	5,14528	33,9026	11,8017	26,6919
1101	4,5101	35,0471	35,578	17,1072	5,66138	33,0724	14,3162	25,0142
1102	7,05866	31,1564	22,8358	20,9585	23,2023	20,8202	28,4538	19,048
1103	16,7184	23,6669	58,0853	12,8495	15,3897	24,3862	44,6089	15,1424
1104	6,42847	31,9687	28,6823	18,9785	6,35019	32,0751	24,7686	20,2528
1105	14,411	24,9569	51,5398	13,8879	15,8194	24,147	35,9572	17,0151
1106	13,4068	25,5843	29,7319	18,6664	13,4312	25,5685	35,6188	17,0972
1107	3,21978	37,9743	45,8778	14,8988	2,49042	40,2053	6,73977	31,5579
1108	15,2	24,4939	62,7558	12,1777	19,752	22,2186	31,9382	18,0446
1109	10,1733	27,9816	50,9889	13,9813	19,841	22,1795	30,9854	18,3077
1110	13,1601	25,7456	27,5731	19,3211	11,486	26,9274	17,9029	23,0723
1111	3,95485	36,1882	33,9224	17,5211	3,20208	38,0222	16,3181	23,8774
1112	11,6933	26,7721	22,714	21,0049	11,788	26,702	34,9166	17,2702
1113	9,14306	28,909	31,0372	18,2932	8,98475	29,0607	17,9635	23,043
1114	5,30194	33,6421	45,123	15,0429	7,64449	30,4638	10,1968	27,9615
1115	7,52328	30,6027	185,065	2,78431	6,77409	31,5138	23,1464	20,8411
1116	8,49783	29,5446	35,1331	17,2165	11,0187	27,2882	15,838	24,1368
1117	12,5394	26,1653	35,2238	17,1941	14,3024	25,0226	32,4114	17,9168
1118	10,975	27,3227	26,0561	19,8126	11,9321	26,5965	31,1051	18,2742
1119	11,9122	26,611	50,7867	14,0158	11,4779	26,9335	26,1189	19,7917
1120	3,62603	36,9422	61,8624	12,3023	2,55541	39,9816	7,21157	30,9702
1121	6,94926	31,292	48,4008	14,4338	6,53877	31,8209	21,8413	21,3452
1122	6,59546	31,7459	30,5429	18,4326	6,48574	31,8916	21,9023	21,321
1123	5,65709	33,0789	47,6593	14,5679	6,1841	32,3053	21,5231	21,4727
1124	10,8321	27,4365	40,6776	15,9437	10,9305	27,358	20,3077	21,9776
1125	185,121	2,78168	28,4413	19,0518	3,1875	38,0618	18,4975	22,7885
1126	9,35694	28,7081	25,8146	19,8935	9,34838	28,7161	37,6531	16,6148
1127	8,03123	30,0352	23,4432	20,7305	8,59136	29,4496	19,97	22,1232
1128	2,18617	41,3371	49,9813	14,1547	1,65876	43,7351	4,27219	35,5178
1129	5,95275	32,6364	62,8395	12,1661	5,95018	32,6402	23,4742	20,719
1130	10,0502	28,0873	55,5085	13,2436	12,5913	26,1294	26,7675	19,5786
1131	13,0169	25,8407	59,579	12,6289	13,1956	25,7222	20,6957	21,8132
1132	8,33426	29,7135	32,6985	17,8402	7,7313	30,3658	26,2239	19,7569
1133	14,1658	25,106	41,81	15,7052	14,7943	24,7289	28,7466	18,9591
1134	6,2951	32,1508	65,0606	11,8644	7,20472	30,9785	25,9837	19,8368
1135	7,0508	31,166	36,5087	16,8829	6,38474	32,0279	22,4375	21,1113
1136	3,46803	37,3291	44,7003	15,1246	3,35041	37,6289	9,10166	28,9484
1137	3,63978	36,9093	37,4584	16,6598	2,76953	39,2827	5,21628	33,7836
1138	8,88325	29,1594	43,2505	15,411	8,56547	29,4758	29,1348	18,8426

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1139	11,86	26,6491	38,6092	16,397	12,8681	25,9405	28,9343	18,9025
1140	10,5252	27,6862	49,01	14,3251	11,9081	26,6139	20,833	21,7558
1141	21,1895	21,6084	69,8441	11,2482	19,8114	22,1925	37,7948	16,5822
1142	5,2135	33,7882	51,3373	13,9222	4,10954	35,8549	5,89636	32,7191
1143	6,40226	32,0041	39,4273	16,2149	6,47051	31,912	17,6637	23,1892
1144	7,06231	31,1519	49,9886	14,1534	9,61579	28,4711	18,799	22,6481
1145	7,13606	31,0616	59,1829	12,6869	9,94026	28,1829	33,3577	17,6669
1146	5,93048	32,669	38,8002	16,3541	6,30007	32,1439	41,1149	15,8508
1147	4,36222	35,3367	41,3254	15,8065	2,87167	38,9681	10,0656	28,074
1148	9,9512	28,1733	30,3551	18,4862	10,6633	27,573	22,8902	20,9378
1149	10,654	27,5805	48,8077	14,361	11,0577	27,2575	23,8755	20,5718
1150	11,734	26,7419	44,3611	15,1908	12,9562	25,8813	31,7257	18,1026
1151	13,4385	25,5638	44,5257	15,1586	16,2937	23,8904	26,9208	19,529
1152	11,9584	26,5773	46,9497	14,6981	10,1197	28,0274	14,6151	24,8348
1153	11,5582	26,873	42,0745	15,6504	13,5558	25,4883	27,2118	19,4356
1154	4,72705	34,639	77,691	10,3234	16,8274	23,6104	35,5377	17,117
1155	12,0996	26,4754	59,3322	12,665	12,6974	26,0565	42,0437	15,6568
1156	7,6728	30,4317	37,0831	16,7473	10,4125	27,7797	21,4087	21,519
1157	4,99942	34,1524	56,3124	13,1187	5,94388	32,6494	35,1527	17,2116
1158	2,9469	38,7435	58,0603	12,8532	3,56026	37,1012	27,7791	19,2564
1159	10,6629	27,5733	57,831	12,8876	10,7091	27,5357	21,0692	21,6579
1160	11,7525	26,7282	31,2593	18,2312	13,3177	25,6422	35,0928	17,2264
1161	10,0388	28,0972	30,7756	18,3667	6,13099	32,3802	10,7685	27,4877
1162	18,0513	23,0006	27,2457	19,4249	17,522	23,2591	39,8034	16,1324
1163	3,64864	36,8882	56,1562	13,1429	4,46637	35,1317	22,4476	21,1074
1164	5,39623	33,489	66,2063	11,7128	39,1023	16,2868	29,1284	18,8445
1165	7,20181	30,982	53,6801	13,5345	7,47531	30,6582	17,5809	23,23
1166	10,5446	27,6702	27,2476	19,4243	12,6108	26,1159	33,0479	17,7479
1167	13,0162	25,8411	27,7335	19,2707	13,1699	25,7391	31,8453	18,0699
1168	13,2583	25,6811	49,5517	14,2296	21,0272	21,6752	37,087	16,7464
1169	17,7131	23,1649	41,3381	15,8038	16,2022	23,9393	29,1055	18,8513
1170	9,78599	28,3187	44,1859	15,2251	10,2003	27,9585	20,106	22,0643
1171	7,94461	30,1293	42,5613	15,5505	7,87488	30,2059	15,3344	24,4175
1172	5,00911	34,1356	32,7896	17,8161	4,74923	34,5983	37,7282	16,5975
1173	16,1634	23,9601	44,5588	15,1521	16,0766	24,0069	29,9282	18,6092
1174	10,34	27,8404	58,1785	12,8355	11,935	26,5944	25,9919	19,834
1175	12,2594	26,3614	32,8605	17,7973	13,5031	25,5221	37,813	16,578
1176	5,28808	33,6648	40,1296	16,0615	7,43002	30,711	12,6836	26,066
1177	5,08177	34,0105	56,2668	13,1258	5,74293	32,9481	20,9411	21,7108
1178	8,18698	29,8683	29,201	18,8229	9,27535	28,7842	24,6921	20,2796

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1179	12,4536	26,2249	46,3052	14,8182	14,3699	24,9817	27,6758	19,2888
1180	7,74659	30,3486	37,243	16,7099	5,28069	33,677	14,6432	24,8181
1181	8,24924	29,8025	37,9391	16,5491	8,08429	29,978	27,0725	19,4802
1182	11,537	26,8889	16,8709	23,5881	12,7184	26,0421	52,3294	13,7559
1183	5,98825	32,5848	81,4508	9,9129	9,16412	28,889	16,3923	23,838
1184	26,8669	19,5465	38,0247	16,5295	28,9764	18,8899	43,5012	15,3608
1185	4,56913	34,9341	31,1758	18,2544	4,4134	35,2353	12,5881	26,1316
1186	3,12964	38,2209	44,1421	15,2337	3,68335	36,8059	26,6221	19,626
1187	8,31193	29,7368	38,1444	16,5022	29,2895	18,7966	27,6989	19,2815
1188	3,86674	36,3839	26,8785	19,5427	36,0931	16,9823	15,5452	24,2989
1189	18,7657	22,6635	58,6358	12,7675	17,835	23,1053	57,0378	13,0076
1190	10,6333	27,5975	57,4793	12,9406	12,9977	25,8535	24,5303	20,3368
1191	4,7291	34,6352	22,9349	20,9209	6,35504	32,0684	37,9039	16,5571
1192	1,45451	44,8765	51,1403	13,9555	1,54832	44,3336	21,0437	21,6684
1193	17,8859	23,0806	38,9458	16,3216	27,7503	19,2654	33,2214	17,7024
1194	1,71372	43,452	31,7852	18,0863	2,29362	40,9204	14,647	24,8158
1195	5,4285	33,4372	55,1822	13,2948	6,74967	31,5452	27,3134	19,4033
1196	9,99851	28,1321	49,596	14,2219	11,9768	26,564	34,4492	17,3872
1197	3,33073	37,68	49,0695	14,3146	4,24469	35,5739	10,6221	27,6066
1198	7,43795	30,7017	27,1985	19,4399	8,99159	29,0541	17,2321	23,404
1199	8,7173	29,3232	45,2756	15,0135	10,535	27,6781	25,3573	20,0488
1200	7,39939	30,7469	52,3396	13,7542	29,8341	18,6365	25,6923	19,9347
1201	7,39382	30,7534	39,3519	16,2315	7,27185	30,8979	16,2395	23,9194
1202	15,6634	24,2331	57,3063	12,9668	16,5305	23,7651	33,3908	17,6583
1203	9,89587	28,2217	29,5927	18,7071	7,83389	30,2512	21,4133	21,5171
1204	3,22625	37,9568	35,1915	17,2021	2,29666	40,9089	26,3147	19,7268
1205	6,50164	31,8703	96,2373	8,46394	8,11416	29,9459	29,4783	18,7407
1206	7,05231	31,1642	50,0842	14,1368	6,49383	31,8808	32,1314	17,9922
1207	7,10846	31,0953	31,3855	18,1962	7,47864	30,6543	19,7247	22,2306
1208	4,10711	35,8601	36,3499	16,9207	2,27045	41,0086	4,1105	35,8529
1209	9,1838	28,8704	40,3935	16,0046	9,28614	28,7741	34,5279	17,3674
1210	9,92508	28,1961	46,1896	14,8399	11,7129	26,7575	38,303	16,4662
1211	8,39984	29,6454	36,7842	16,8176	6,8595	31,405	29,5426	18,7218
1212	17,6058	23,2177	49,6221	14,2173	16,0056	24,0454	29,1882	18,8267
1213	4,02347	36,0388	27,3285	19,3985	3,94868	36,2018	11,8973	26,6218
1214	4,12193	35,8288	32,416	17,9156	5,14407	33,9047	24,1637	20,4675
1215	13,7021	25,3951	31,3179	18,2149	11,9755	26,5649	26,6907	19,6036
1216	7,65949	30,4468	36,6811	16,842	6,83234	31,4394	34,563	17,3586
1217	4,3081	35,4451	48,1523	14,4785	4,35676	35,3475	44,5616	15,1516
1218	8,05071	30,0141	37,3842	16,677	6,04252	32,5064	9,24644	28,8113

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1219	10,0347	28,1007	70,5947	11,1554	10,9853	27,3146	55,0694	13,3126
1220	4,796	34,5132	56,6059	13,0736	5,61714	33,1405	22,1383	21,2279
1221	6,12264	32,392	30,2625	18,5127	6,37549	32,0405	23,05	20,8774
1222	10,7903	27,4701	40,8889	15,8987	12,4278	26,2429	22,2389	21,1885
1223	5,19312	33,8222	43,2858	15,4039	5,87543	32,75	15,3274	24,4215
1224	7,85974	30,2226	83,1457	9,734	7,03814	31,1816	31,3529	18,2053
1225	4,26109	35,5404	65,9423	11,7475	4,19922	35,6674	10,6969	27,5456
1226	13,9408	25,2451	26,7072	19,5982	13,7178	25,3851	22,1192	21,2354
1227	19,5834	22,293	44,1262	15,2369	17,8294	23,1081	37,0047	16,7657
1228	4,99088	34,1673	57,4526	12,9446	6,04975	32,4961	24,9101	20,2033
1229	6,8116	31,4658	60,3834	12,5124	11,4131	26,9827	35,4927	17,128
1230	22,3777	21,1345	36,2565	16,9431	39,5415	16,1897	38,937	16,3236
1231	18,9902	22,5602	27,4058	19,374	23,8562	20,5788	39,9514	16,1002
1232	3,43488	37,4126	39,35	16,2319	4,96097	34,2195	13,3944	25,5923
1233	6,47001	31,9127	28,5045	19,0325	6,22444	32,2488	15,5656	24,2875
1234	3,39973	37,5019	24,2535	20,4353	5,87674	32,7481	10,0377	28,0981
1235	10,9795	27,3192	30,0897	18,5624	19,589	22,2906	28,4475	19,0499
1236	16,8842	23,5812	31,0693	18,2842	14,2394	25,061	40,407	16,0017
1237	3,77382	36,5952	57,2344	12,9777	3,62394	36,9472	31,729	18,1017
1238	10,4941	27,7119	63,3871	12,0908	8,98608	29,0594	34,9076	17,2724
1239	12,928	25,9002	46,8569	14,7153	14,4023	24,9622	38,076	16,5178
1240	8,9624	29,0823	69,3429	11,3108	11,2251	27,127	24,663	20,2899
1241	3,9422	36,216	46,0476	14,8667	16,4492	23,8079	23,7592	20,6142
1242	3,41636	37,4595	63,3488	12,096	3,11948	38,2492	15,4572	24,3482
1243	12,0697	26,4969	33,2286	17,7006	13,5377	25,4999	27,64	19,3
1244	2,79707	39,1967	34,1585	17,4608	3,33031	37,6811	25,0248	20,1634
1245	5,50593	33,3142	31,7034	18,1087	4,63156	34,8163	7,77406	30,3178
1246	4,43583	35,1913	65,9948	11,7406	4,95303	34,2334	45,3268	15,0037
1247	3,08461	38,3468	106,055	7,62018	2,95314	38,7251	25,022	20,1644
1248	16,394	23,8371	57,9331	12,8723	16,5987	23,7293	44,1085	15,2404
1249	24,4269	20,3734	58,3068	12,8164	27,0759	19,4792	49,2876	14,2761
1250	13,0285	25,8329	57,0985	12,9983	9,23099	28,8258	15,9102	24,0973
1251	3,54796	37,1312	86,6543	9,375	54,6551	13,3782	25,291	20,0715
1252	5,07463	34,0227	39,0502	16,2983	4,27969	35,5026	37,4777	16,6553
1253	4,09801	35,8793	22,1638	21,2179	38,9467	16,3214	20,1937	22,0265
1254	2,07311	41,7984	40,6564	15,9482	2,3266	40,7964	14,5537	24,8713
1255	2,2047	41,2638	32,5008	17,8929	2,91208	38,8467	28,1503	19,1412
1256	4,17951	35,7083	33,1494	17,7213	3,07476	38,3746	9,88789	28,2287
1257	11,3269	27,0486	43,1	15,4413	11,697	26,7693	28,8422	18,9302
1258	30,8237	18,3531	45,7942	14,9146	28,2489	19,1108	62,9093	12,1565

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1259	6,9832	31,2497	56,8431	13,0372	57,1277	12,9939	30,3945	18,4749
1260	4,23781	35,588	37,3327	16,689	2,78102	39,2467	26,1183	19,7919
1261	7,1259	31,074	38,9029	16,3312	7,04778	31,1698	20,1553	22,043
1262	20,8038	21,7679	57,0643	13,0035	19,047	22,5343	43,0536	15,4506
1263	9,23404	28,823	49,4851	14,2413	8,59637	29,4445	31,2159	18,2433
1264	4,40092	35,2599	55,1795	13,2953	5,59813	33,1699	21,7832	21,3684
1265	10,7791	27,4792	45,6906	14,9343	12,5437	26,1623	34,0884	17,4787
1266	7,85034	30,233	51,2806	13,9317	7,38838	30,7598	13,832	25,3131
1267	8,66971	29,3707	50,3519	14,0905	6,42218	31,9772	9,98422	28,1445
1268	6,01487	32,5463	67,4125	11,556	6,06662	32,4719	18,8001	22,6476
1269	10,7407	27,5102	19,4282	22,3621	8,37368	29,6725	27,9901	19,1907
1270	9,89734	28,2204	37,9146	16,5547	11,6741	26,7864	32,839	17,803
1271	12,0545	26,5078	37,2293	16,7131	13,3967	25,5908	22,297	21,1659
1272	2,06044	41,8516	48,9935	14,328	2,32117	40,8167	17,554	23,2433
1273	6,41192	31,991	29,5929	18,707	7,66827	30,4369	19,361	22,3923
1274	6,63773	31,6904	44,8343	15,0986	6,06369	32,4761	14,8106	24,7194
1275	3,27759	37,8197	28,5886	19,007	4,14525	35,7798	26,6002	19,6331
1276	14,1524	25,1142	31,6162	18,1326	12,825	25,9697	19,5917	22,2894
1277	8,44661	29,5972	68,3602	11,4347	10,8273	27,4404	23,3625	20,7604
1278	4,25077	35,5615	53,1172	13,6261	17,6965	23,173	24,5282	20,3375
1279	6,45299	31,9356	47,8827	14,5272	18,2076	22,9258	20,7323	21,7979
1280	17,7741	23,1351	25,5048	19,9983	22,7182	21,0033	42,5066	15,5617
1281	9,26805	28,791	38,2244	16,484	9,24434	28,8133	26,4182	19,6927
1282	8,14637	29,9115	31,1943	18,2493	10,9418	27,349	33,0527	17,7467
1283	5,13094	33,9269	57,0569	13,0046	6,53017	31,8323	32,7685	17,8217
1284	9,66082	28,4305	36,6709	16,8444	19,9028	22,1525	24,8296	20,2314
1285	4,88508	34,3534	33,5005	17,6298	5,51277	33,3034	12,4196	26,2487
1286	14,6987	24,7852	46,7744	14,7306	16,0421	24,0256	33,3151	17,678
1287	7,00466	31,2231	42,7558	15,5109	5,72656	32,9729	13,2465	25,6888
1288	6,97481	31,2602	98,2969	8,28001	7,91909	30,1573	37,7072	16,6023
1289	4,39208	35,2774	38,1409	16,503	28,56	19,0156	27,5849	19,3174
1290	18,4215	22,8243	48,5854	14,4007	20,4215	21,9291	45,8703	14,9002
1291	6,09866	32,4261	64,6515	11,9192	17,2562	23,3919	26,5823	19,6389
1292	4,05484	35,9713	47,2002	14,6519	4,95894	34,223	14,8784	24,6797
1293	10,9815	27,3175	59,4566	12,6468	12,2678	26,3555	26,6053	19,6314
1294	5,56495	33,2216	64,9849	11,8746	6,3342	32,097	41,6398	15,7406
1295	11,2953	27,0729	52,9023	13,6613	12,3546	26,2943	27,5315	19,3342
1296	6,16412	32,3334	25,552	19,9823	17,5067	23,2667	27,6184	19,3068
1297	3,25282	37,8856	32,633	17,8577	3,72302	36,7129	21,9599	21,2982
1298	8,38432	29,6614	47,7753	14,5467	10,6363	27,595	33,7454	17,5665
1299	9,32763	28,7354	50,6293	14,0428	13,7615	25,3575	34,8617	17,2838
1300	10,0688	28,0712	51,1711	13,9503	24,81	20,2383	20,291	21,9847
1301	8,46834	29,5748	46,5793	14,7669	11,4652	26,9432	32,2788	17,9525

50x	JPEG2000		JPEG Progressive		SPIHT		OSPIHT	
	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR	dispersion	PSNR
1302	7,78989	30,3002	23,3663	20,759	11,8114	26,6848	33,5338	17,6211
1303	10,4149	27,7777	36,4368	16,9	11,0363	27,2743	19,0219	22,5457
1304	12,8801	25,9324	51,9095	13,8259	14,0608	25,1706	30,0856	18,5636
1305	18,8235	22,6368	19,3081	22,416	21,0013	21,6859	47,9597	14,5133
1306	16,0644	24,0135	31,6816	18,1146	18,3209	22,8719	33,7657	17,5613
1307	8,03555	30,0305	34,9415	17,264	11,2208	27,1303	31,6787	18,1154
1308	3,84874	36,4244	57,7084	12,906	3,79012	36,5578	21,4182	21,5152
1309	4,95471	34,2304	30,5562	18,4288	5,99089	32,581	26,0268	19,8224
1310	21,0998	21,6452	55,834	13,1928	24,3245	20,4099	55,834	13,1928
1311	11,852	26,655	28,334	19,0847	13,2494	25,6869	28,334	19,0847
1312	6,35261	32,0718	21,5498	21,4619	10,1258	28,0222	21,5498	21,4619
1313	7,13878	31,0583	45,1372	15,0401	7,24035	30,9356	45,1372	15,0401
1314	1,35883	45,4675	47,0909	14,6721	2,10039	41,6848	47,0909	14,6721
1315	5,87479	32,751	26,0129	19,827	6,41778	31,9831	26,0129	19,827
1316	4,84619	34,4228	29,3958	18,7651	7,35007	30,805	29,3958	18,7651
1317	4,38385	35,2937	37,4768	16,6556	5,68884	33,0303	37,4768	16,6556
1318	4,54572	34,9788	22,5535	21,0665	5,89441	32,722	22,5535	21,0665
1319	6,7779	31,5089	21,8719	21,3331	10,1171	28,0297	21,8719	21,3331
1320	4,14674	35,7767	24,8329	20,2302	4,19563	35,6749	24,8329	20,2302
1321	12,595	26,1269	39,6663	16,1624	23,0901	20,8623	39,6663	16,1624
1322	3,2524	37,8867	24,5969	20,3132	4,71826	34,6552	24,5969	20,3132

ПРИЛОЖЕНИЕ Е

Акты реализации результатов диссертационной работы

ФЛП Бурьянов Константин Владимирович
пер. Степной 31,г. Кировоград, Украина, 25011
тел : +38 (0522) 27-60-06
факс: +38 (0522) 27-60-91
e-mail : office@imperial.net.ua
http://www.imperial.net.ua

ЗАТВЕРДЖУЮ:
Директор ФОП Бур'янов К.В.
т.м. «ІСП Імперіал»
«21» жовтня 2014 р.

АКТ
про впровадження результатів дисертаційної роботи аспіранта кафедри «Програмне забезпечення» Кіровоградського національного технічного університету Дрєєва Олександра Миколайовича

Комісія у складі голови – Директора «ІСП Імперіал» Бур'янова К.В., членів комісії – провідного розробника «ІСП Імперіал» Король К.В., провідного розробника «ІСП Імперіал» Вельможко О.В., склала цей акт про те, що у діяльності «ІСП Імперіал» реалізовано результати наукових досліджень Дрєєва Олександра Миколайовича: при виконанні проекту «Proxy-Small», при розробці платформи віддаленого перегляду графічного контенту з метою модерації. Отриманий корисний ефект полягає в значному зменшені графічного трафіку, що в зменшує час очікування завантаження та грошові витрати на оплату мобільного доступу.

Голова комісії
Директор «ІСП Імперіал» _____ Бур'янов К.В.

Члени комісії:
провідний розробник _____ Король К.В.

провідний розробник _____ Вельможко О.В.

ЗАТВЕРДЖУЮ:
Проректор з наукової роботи
Кіровоградського національного
технічного університету
_____ О.М. Левченко
«_____»_____2015 р.

АКТ

реалізації результатів наукових досліджень дисертаційної роботи аспіранта кафедри «Програмне забезпечення» Дрєєва Олександра Миколайовича

Комісія у складі голови – заступника завідуючого кафедрою «Програмне забезпечення» Кіровоградського національного технічного університету кандидата фізико-математичних наук, доцента Якименко Н.М., членів комісії – доцента кафедри «Програмне забезпечення» кандидата технічних наук, Мелешко Є.В., доцента кафедри «Програмне забезпечення» кандидата технічних наук, доцента Коваленко О.В. склала цей акт про те, що при розробці лекційних, практичних та лабораторних занять з навчальних дисциплін «Комп'ютерні мережі» та «Комп'ютерна графіка» (лекція «Використання проксі-сервера з метою заощадження внутрішнього трафіку», практичне заняття «Реалізація прогресуючого алгоритму стиснення зображень» відповідно) у навчальному процесі Кіровоградського національного технічного університету були використані наступні результати наукових досліджень Дрєєва Олександра Миколайовича:

1. Математична модель передачі багатопакетних повідомлень з врахуванням використання додаткового сервера для додаткового стиснення графічної інформації; зокрема, сервер представлено як систему кешування та додаткового стиснення графічної інформації, математично формалізовані процедури передачі в мережі багатопакетних повідомлень, теоретично обґрунтовано лінійну залежність між розміром повідомлення та часом його доставки. Обґрунтовано зменшення часу відгуку серверу в разі зменшення трафіку. Знання узагальнюються до виділення умов, при яких використання серверу з додатковим стисненням інформації зменшує час отримання повідомлень.

2. Методи прогресуючого стиснення зображень широко використовувалися в часи модемного підключення до мережі Інтернет. Це дозволяло користувачам оцінити актуальність зображення до повного його завантаження, що в свою чергу значно пришвидшувало попередній перегляд та пошук потрібного зображення. В наш час прогресивне стиснення залишається актуальним, особливо для мобільних терміналів. Це пояснюється тим, що швидкісний доступ до глобальної мережі не є всеосяжним. Також й методи прогресивного стиснення зображень теж

набули значного розвитку. Розглянуто прогресивний алгоритм стиснення цифрового зображення SPIHT. Для алгоритму запропоновано модифікацію з відкладеною передачею уточнень значущих коефіцієнтів, що забезпечує більш високий ступінь збереження контурної інформації. Розроблено програмне забезпечення, яке реалізує даний метод. Проведено порівняльні досліди реалізацій SPIHT алгоритмів з визначенням втрат інформації та деформації контурів.

Застосування результатів дисертаційних досліджень Дрєєва Олександра Миколайовича дозволило підвищити рівень засвоєння навчального матеріалу з дисциплін «Комп'ютерні системи та мережі» та «Системне програмування» за рахунок більш поглибленого вивчення сучасних та перспективних методів передачі та перетворення інформації у телекомунікаційних мережах.

Голова комісії
Заступник завідуючого кафедри «Програмне забезпечення»
Кіровоградського національного
технічного університету
кандидат фізико-математичних наук, доцент Н.М. Якименко

Члени комісії:
доцент кафедри «Програмне забезпечення»
кандидат технічних наук, доцент Є.В. Мелешко

доцент кафедри «Програмне забезпечення»
кандидат технічних наук, доцент О.В. Коваленко