МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра комп'ютерних наук

Пояснювальна записка

до кваліфікаційної роботи
другого (магістерського) рівня

на тему АНАЛІЗ МЕТОДІВ НЕРІВНОМІРНОГО КОДУВАННЯ

Виконав: студент 2 курсу, групи ІКК 2.1
спеціальності
122 Комп'ютерні науки

Безменов Дмитро Володимирович

Керівник Розенвассер Д.М.

Рецензент Григор'єва Т.І.

Одеса – 2023

ДОВІДКА

кафедри КН про виконану магістерську роботу

студента 2 курсу ФКПІ та КН групи ІКК 2.1

Безменова Дмитра Володимировича

на тему Аналіз методів нерівномірного кодування

Висновок нормоконтролера пояснювальна записка до кваліфікаційної роботи виконана у повній відповідності до вимог ДСТУ. Оформлення роботи відповідає вимогам.

Нормоконтролер викл. каф. ІПІ (науковий ступінь, вчене звання, посада) 15.11.2023 (підпис, дата) Кеміщенко І.В (і. б. прізвище)

Висновок відповідального за наявність плагіату згідно з сертифікатом ІД 1015677889 унікальність роботи підтверджено

Відповідальна особа викл. каф. ІПІ (науковий ступінь, вчене звання, посада) 15.11.2023 (підпис, дата) Кеміщенко І.В (і. б. прізвище)

Попередня експертиза (захист) магістерської роботи

(бакалаврської роботи чи магістерської роботи)

студ. Безменов Д.В. (прізвище і.б.) проведена "15" 12 2023р.

Висновки кваліфікаційна робота виконана у повному обсязі. В роботі проведено аналіз методів нерівномірного кодування. Кваліфікаційна робота відповідає вимогам до випускних кваліфікаційних робіт зі спеціальності 122 Комп'ютерні науки та рекомендована до захисту.

Члени комісії (підпис) к.т.н, доц. Соловєвська І.М. (науковий ступінь, вчене звання, посада, прізвище і.б.)

(підпис) к.т.н, доц. Русу О.П. (науковий ступінь, вчене звання, посада, прізвище і.б.)

(підпис) к.т.н, доц. Розенвассер О.М. (науковий ступінь, вчене звання, посада, прізвище і.б.)

МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра комп'ютерних наук
Освітній ступінь магістр
Галузь знань 12 Інформаційні технології
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
к.т.н., доц.
_____ І.М.Соловська
"25" 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ

Безменова Дмитра Володимировича

1. Тема роботи: Аналіз методів нерівномірного кодування
керівник роботи Розенвассер Денис Михайлович
затверджені наказом закладу вищої освіти від 25.09.2023 р. № 1959

2. Строк подання студентом роботи 11.12.2023

3. Вихідні дані до роботи: Задано різноманітні типи даних, такі як текст, аудіо, відео. Необхідно проаналізувати особливості кожного з типів даних, методи їх стиснення, варіанти використання нерівномірних кодів для стиснення, надати рекомендації щодо можливостей їхнього використання у сучасних кодеках та архіваторах

4. Зміст розрахунково-пояснювальної записки
Розділ 1: Використання нерівномірного кодування для стиснення даних без втрат
Розділ 2: Використання нерівномірного кодування для стиснення даних з втратами
Розділ 3: Аналіз практичної реалізації алгоритмів стиснення даних в сучасних системах та мережах

5. Перелік графічного матеріалу (з зазначенням обов'язкових креслень)
Слайд 1 – Префіксне кодування

Слайд 2 – Арифметичне кодування

Слайд 3 – Словникове кодування

Слайд 4 – Сучасні методи нерівномірного кодування

Слайд 5 – Висновки та рекомендації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.09.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Вступ	25.09.2023-5.10.2023	Вик
2	Нерівномірне кодування як метод стиснення	6.10.2023-19.10.2023	Вик
3	Використання нерівномірного кодування для стиснення даних без втрат	20.10.2023-31.10.2023	Вик
4	Використання нерівномірного кодування для стиснення даних з втратами	1.11.2023-9.11.2023	Вик
5	Аналіз практичної реалізації алгоритмів стиснення даних в сучасних системах та мережах	10.11.2023-20.11.2023	Вик
6	Висновки та рекомендації	21.11.2023-28.11.2023	Вик
7	Перелік посилань	29.11.2023-5.12.2023	Вик
8	Оформлення презентації	6.12.2023-11.12.2023	Вик

Студент _____ Д.В. Безменов
(підпис)

Керівник роботи _____ Д.М. Розенвассер
(підпис)

ВІДГУК КЕРІВНИКА

магістерської роботи студента Безменова Д.В.
на тему: «Аналіз методів нерівномірного кодування»

Завдання використання нерівномірних кодів для стиснення інформації є актуальною темою для нових та існуючих інформаційних систем та мереж.

У роботі розглядаються методи стиснення інформації зі втратами та без втрат за допомогою нерівномірних кодів.

Студент Безменов Д.В. добре розібрався з проблемами даної тематики, приділив увагу докладному аналізу методів стиснення інформації в Україні та світі.

Робота проводилася значною мірою самостійно. Графік консультацій не порушувався.

Завдання на ВКР виконано. Необхідні для цього розрахунки проведені. При оформленні пояснювальної записки та демонстраційних слайдів використовувались комп'ютерні технології.

Під час виконання магістерської роботи студент Безменов Д.В. глибоко вивчив класифікацію нерівномірних кодів, їхні алгоритми, переваги та недоліки, принципи стиснення інформації, принципи роботи програм-архіваторів даних для стиснення даних різного походження, показав уміння користуватись навчальною та технічною літературою, ставити та розв'язувати інженерні задачі.

Магістерська робота відповідає вимогам до випускних робіт та заслуговує оцінки «добре».

Студент Безменов Д.В. заслуговує присвоєння кваліфікації магістр з комп'ютерних наук за спеціальністю 122 Комп'ютерні науки.

Керівник
к.т.н., доцент кафедри КН Д.М. Розенвассер

РЕЦЕНЗІЯ

магістерської роботи студента Безменова Д.В.
на тему: «Аналіз методів нерівномірного кодування»

Магістерська робота містить 3 розділи текстової частини, демонстраційні слайди, виконана згідно з завданням на магістерську роботу.

У роботі розглядаються нерівномірні коди, методи стиснення даних, програми-архіватори у яких вони застосовуються.

Актуальність теми полягає в тому, що на сьогоднішній день існує досить багато методів стиснення даних, в кожному з яких використовуються нерівномірні коди. Для оцінки ефективності стиснення потрібно врахувати тип мережі, тип даних, якість даних, потреби користувачів та інші фактори. У даній роботі для кожного з цих варіантів зроблено висновки та надано рекомендації щодо використання нерівномірних кодів.

Магістерська робота виконана відповідно до завдання. Демонстраційні матеріали й пояснювальна записка виконані охайно й відповідно до вимог ЄСКД. Прийняті рішення обґрунтовано, розрахунки виконано правильно.

Автором показана достатня теоретична підготовка. Робота виконана грамотно, текст її послідовний та зрозумілий, оформлення роботи та демонстраційних слайдів якісне.

До недоліків роботи варто віднести те, що у роботі:

- недостатньо докладно розглянуто гібридний адаптивний метод нерівномірного кодування;
- недостатньо уваги приділено застосування нерівномірних кодів для шифрування інформації.

Зазначені недоліки суттєво не знижують якості виконаної роботи.

Магістерська робота відповідає вимогам до випускних робіт та заслуговує оцінки «добре».

Студент Безменов Д.В. заслуговує присвоєння кваліфікації магістр з комп'ютерних наук за спеціальністю 122 Комп'ютерні науки.

Рецензент
к.т.н., доцент, заф. кафедри ІТ Григорєва Т.І.

UNICHECK
by Turnitin

Ім'я користувача:
Анна Серединко

Дата перевірки:
11.12.2023 23:53:34 MSK

Дата звіту:
11.12.2023 23:57:34 MSK

ID перевірки:
1015995241

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100001433

Назва документа: Безменов_Аналіз_методів_нерівномірного_кодування

Кількість сторінок: 56 Кількість слів: 10820 Кількість символів: 78779 Розмір файлу: 964.41 KB ID файлу: 1015677889

12.4%
Схожість

Найбільша схожість: 1.99% з джерелом з Бібліотеки (ID файлу: 1015327943)

11.3% Джерела з Інтернету	235	Сторінка 58
1.99% Джерела з Бібліотеки	1	Сторінка 61

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	4

РЕФЕРАТ

Текстова частина магістерської роботи: 52 с., 15 рис., 2 табл., 10 джерел.

НЕРІВНОМІРНІ КОДИ, СТИСНЕННЯ, ПРЕФІКСНИЙ КОД, АРИФМЕТИЧНИЙ КОД, СЛОВНИКОВІ КОДИ, LZW, RLE

Об'єкт дослідження – нерівномірні коди в комп'ютерних системах.

Мета роботи – дослідження нерівномірного кодування в комп'ютерних системах як засобу стиснення даних, огляд існуючих алгоритмів стиснення для різних типів даних та аналіз їх реалізації в сучасних комп'ютерних системах і мережах відповідно до специфіки цих мереж і вимог до їх адекватної роботи.

Метод дослідження – аналітичний з використанням комп'ютерних технологій.

У магістерській роботі зроблено огляд нерівномірних кодів, історії алгоритмів стиснення даних; розглянуто методології стиснення різних типів даних; обґрунтовано необхідність використання методів компресії в сучасних комп'ютерних мережах; проаналізовано конкретні технології стиснення даних, що використовуються сьогодні в сучасних комп'ютерних системах і мережах.

Зроблено висновки та рекомендації щодо розвитку теорії та практики нерівномірних кодів та методів стиснення даних.

ABSTRACT

The text part of the master paper: 52 pp., 15 figures, 2 tables, 10 references.

NON-UNIFORM CODES, COMPRESSION, PREFIX CODE, ARITHMETIC CODE, DICTIONARY CODES, LZW, RLE

Object of research are non-uniform codes in computer systems.

The purpose of the work is to research of non-uniform coding in computer systems as a means of data compression, review of existing compression algorithms for various types of data and analysis of their implementation in modern computer systems and networks in accordance with the specifics of these networks and requirements for their adequate operation.

The research method is analytical with the use of computer technologies.

In the master's paper, an overview of non-uniform codes, the history of data compression algorithms is made; the methodologies of compression of various types of data are considered; the necessity of using compression methods in modern computer networks is substantiated; specific data compression technologies used today in modern computer systems and networks are analyzed.

Conclusions and recommendations have been made regarding the development of the theory and practice of non-uniform codes and data compression methods.

ЗМІСТ

1 ВИКОРИСТАННЯ НЕРІВНОМІРНОГО КОДУВАННЯ ДЛЯ СТИСНЕННЯ ДАНИХ БЕЗ ВТРАТ

1.1 Нерівномірне кодування як метод стиснення

Методи стиснення даних мають досить довгу історію розвитку, яка почалася задовго до винаходу першого комп'ютера.

Стиснення інформації – це проблема, яка має досить давню історію, набагато більшу за історію розвитку комп’ютерної техніки, яка зазвичай йшла разом з історією розвитку проблеми кодування та шифрування інформації. Усі алгоритми стиснення працюють із вхідним потоком інформації, мінімальною одиницею якого є біт. Метою процесу стиснення, як правило, є отримання більш компактного вихідного потоку одиниць інформації з деякого початкового некомпактного вхідного потоку за допомогою певного перетворення.

Таким чином, стиснення даних – це алгоритмічне перетворення даних, яке виконується з метою зменшення обсягу, зайнятого цими даними. Використовується для більш раціонального використання пристроїв зберігання та передачі даних.

Нерівномірне кодування - це метод кодування, при якому символам алфавіту присвоюються коди різної довжини. Цей метод дозволяє більш ефективно кодувати символи, які зустрічаються в алфавіті з різною частотою.

Нерівномірні коди використовуються для перетворення даних таким чином, щоб зменшити їх розмір (стиснення) або забезпечити додаткову безпеку. Вони використовуються для стиснення даних або шифрування інформації.

Статистичне нерівномірне кодування використовує статистичну інформацію про частоту зустрічання символів в алфавіті для визначення довжини кодів.

Перевагами нерівномірного кодування є більш ефективне кодування символів, які зустрічаються в алфавіті з різною частотою та може бути використано для підвищення ефективності обробки інформації. Це може призвести до зменшення довжини кодованого повідомлення, що, в свою чергу, може призвести до зменшення витрат на передачу та зберігання інформації.

Недоліками нерівномірного кодування є те, що воно може бути складним для реалізації, оскільки вимагає знань про частоту зустрічання символів в алфавіті, а також може призвести до збільшення витрат на обробку інформації, оскільки вимагає додаткових операцій для визначення довжини кодів.

Стиснення даних засноване на усуненні надмірності, що міститься у вихідних даних [1]:

– часове - в даних, одновимірних сигналах, аудіо тощо;

– просторове - кореляція між сусідніми пікселями або елементами даних;

– спектральне – кореляція між компонентами кольору або люмінесценції (тут використовується частотна область для використання зв’язків між частотою зміни даних);

– психовізуальне - використовують перцептивні властивості зорової системи людини.

Найпростішим прикладом надмірності є повторення фрагментів у тексті (наприклад, слів природної або машинної мови). Така надмірність зазвичай усувається шляхом заміни повторюваної послідовності посиланням на вже закодований фрагмент із зазначенням його довжини. Інший тип надмірності пов'язаний з тим, що деякі значення в стислих даних

зустрічаються частіше, ніж інші. Зменшення обсягу даних досягається шляхом заміни часто зустрічаються даних короткими кодовими словами, а рідкісних - довгими (ентропійне кодування). Стиснення без втрат даних, які не володіють властивістю надмірності (наприклад, випадковий сигнал або білий шум, зашифровані повідомлення), принципово неможливе.

Основними технічними характеристиками процесів стиснення та результатів їх роботи є:

– ступінь стиснення об'ємів вхідного і вихідного потоків;

– швидкість стиснення – час, необхідний для стиснення певної кількості інформації вхідного потоку до отримання з нього еквівалентного вихідного потоку;

– якість стиснення – значення, яке показує, наскільки вихідний потік упакований шляхом застосування до нього повторного стиснення за тим самим або іншим алгоритмом.

Основною характеристикою алгоритму стиснення даних є коефіцієнт стиснення, тобто величина, яка визначає різницю між обсягом початкових даних і стислих даних і може бути розрахована за такою формулою:

$$k = \frac{S_0}{S_c} \tag{1.1}$$

де k - ступінь стиснення, S_0 – обсяг вихідних даних, S_c – обсяг стислих даних.

Очевидно, що вищий ступінь стиснення означає більшу ефективність алгоритму; k=1 означає, що алгоритм не стискає, а k<1 означає, що алгоритм шкідливий, тобто обсяг результуючого повідомлення більший, ніж початковий.

Існує кілька різних підходів до проблеми стиснення інформації. Одні мають дуже складну теоретичну математичну базу, інші базуються на

властивостях інформаційного потоку і досить прості алгоритмічно. Будь-який метод і алгоритм, який реалізує стиснення даних, призначений для зменшення кількості інформації у вихідному потоці в бітах за допомогою її оборотного або необоротного перетворення. Тому, перш за все, за критерієм, пов'язаним з природою або форматом даних, усі методи стиснення можна розділити на дві категорії: стиснення з втратами та стиснення без втрат.

Основний принцип, на якому працюють алгоритми стиснення без втрат, полягає в тому, що будь-який невипадковий файл міститиме надлишкову інформацію, яку можна стиснути за допомогою методів статистичного моделювання, які визначають ймовірність появи символу чи фрази. Потім ці статистичні моделі можна використовувати для створення кодів для конкретних символів або фраз на основі ймовірності їх появи та призначення найкоротших кодів найпоширенішим даним. Такі методи включають ентропійне кодування, кодування довжини серії та стиснення за допомогою словника. Використовуючи ці методи та інші, 8-бітний символ або рядок таких символів можна представити лише кількома бітами, що призводить до видалення великої кількості зайвих даних.

Для кожного виду цифрової інформації, як правило, існують оптимальні алгоритми стиснення без втрат. Стиснення даних без втрат використовується в багатьох програмах. Наприклад, він використовується у всіх файлових архіваторах. Він також використовується як компонент стиснення з втратами. Стиснення без втрат використовується, коли важлива тотожність між стиснутими даними та джерелом. Типовим прикладом є виконувані файли та вихідний код.

У загальному вигляді сенс стиснення без втрат полягає в наступному: у вихідних даних виявляється будь-яка закономірність і з урахуванням цієї закономірності генерується друга послідовність, яка повністю описує вхідну.

1.2 Історичне походження. Префіксне кодування

Найпростіші алгоритми стиснення, також звані оптимальними алгоритмами кодування, є статистичними і засновані на розподілі ймовірностей елементів вхідного повідомлення (текст, зображення, файл). На практиці частота появи елементів у вхідному повідомленні використовується як наближення до ймовірностей. Імовірність — це абстрактне математичне поняття, пов’язане з нескінченною експериментальною вибіркою даних, а частота появи — це значення, яке можна обчислити для кінцевих наборів даних. При достатньо великій кількості елементів у сукупності експериментальних даних можна сказати, що частота появи елемента близька (з деякою точністю) до його ймовірності [2].

Якщо згадані ймовірності відрізняються, то можна зіставити коротші кодові слова для елементів, які, швидше за все, зустрічаються (звичайні), і, навпаки, замінити елементи, які малоймовірні, довшими кодовими словами. Таким чином можна зменшити середню довжину кодового слова. Оптимальний алгоритм кодування робить це так, що середня довжина кодового слова є мінімальною, тобто якщо довжина кодування коротша, вона стає незворотною. Це робиться за допомогою префіксних алгоритмів Шеннона-Фано та Хаффмана [3].

Алгоритм побудови коду стиснення Шеннона-Фано виглядає наступним чином.

1. Усі m символів дискретного джерела розташовані в порядку зменшення ймовірності їх появи.

2. Сформований стовпець символів розділений на дві групи так, щоб сумарні ймовірності кожної групи дещо відрізнялися одна від одної.

3. Верхня група кодується символом «1», а нижня – «0».

4. Кожна група ділиться на дві підгрупи з близькими сумарними ймовірностями; верхня підгрупа кодується символом «1», а нижня – «0».

5. Процес поділу та кодування триває до тих пір, поки кожна підгрупа не міститиме один символ вихідного повідомлення.

6. Для кожного символу джерела записується код; читання коду здійснюється зліва направо.

Класичний алгоритм Хаффмана на вхід отримує таблицю частот появи символів у повідомленні. Далі на основі цієї таблиці будується дерево кодування Хаффмана (Н-дерево).

1. Символи вхідного алфавіту утворюють список вільних вузлів. Кожен аркуш має вагу, яка може дорівнювати або ймовірності, або кількості входжень символу в стискається повідомлення.

2. Вибираються два вільні вузли дерева з найменшими вагами.

3. Їхній батько створюється з вагою, що дорівнює їхній загальній вазі.

4. Батьківський вузол додається до списку вільних вузлів, а два його нащадки видаляються з цього списку.

5. Одна дуга, яка виходить з батьківської лінії, відображається в біті 1, інша – в біті 0. Значення бітів гілок, що походять від кореня, не залежать від ваги нащадків.

6. Кроки, починаючи з другого, повторюються до тих пір, поки в списку вільних вузлів не залишиться один вільний вузол. Вважається, що це корінь дерева.

Щоб визначити код для кожного з символів, включених у повідомлення, ми повинні пройти від листа дерева, що відповідає поточному символу, до його кореня, накопичуючи біти при русі по гілках дерева (перша гілка на шляху відповідає молодший значущий біт). Отримана таким чином послідовність бітів є кодом цього символу, записаним у зворотному порядку.

Недоліком обох методів є те, що вони не здатні економно кодувати повідомлення, ніж один біт на елемент повідомлення (літеру).

Як для кодування Шеннона-Фано, так і для кодування Хаффмана середню кількість бітів, які використовуються для кодування одного символу (літери), можна обчислити за такою формулою:

$$n_{av} = \sum_{i=1}^{k} p(x_i) \cdot n_i \qquad (1.2)$$

де n_{av} – середня кількість бітів, що використовуються для кодування одного символу, $p(x_i)$ – ймовірність виникнення і-го символу та n_i – кількість бітів, що використовуються для кодування і-го символу.

Шеннона-Фано є досить старим методом стиснення, і на сьогоднішній день він не має практичного інтересу. У більшості випадків довжина стислої послідовності цим методом дорівнює довжині стислої послідовності з використанням кодування Хаффмана. Але є послідовності, в яких методика Шеннона-Фано не дає однозначного кодування, тому метод стиснення Хаффмана вважається більш ефективним.

Під час побудови коду Шеннона-Фано розбиття набору елементів може бути здійснено декількома способами. Вибір n-рівневого розділу може погіршити параметри наступного рівня розділу (n + 1) і викликати неоптимальний код в цілому. Іншими словами, оптимальна поведінка на кожному кроці ще не гарантує оптимальності всієї сукупності дій. Тому код Шеннона-Фано в цілому не є оптимальним, хоча він дає оптимальні результати з деякими розподілами ймовірностей. Іншими словами, для одного і того ж розподілу ймовірностей можна побудувати кілька кодів Шеннона-Фано, і всі вони можуть давати різні результати. Якщо ми побудуємо всі можливі коди Шеннона-Фано для заданого розподілу

ймовірностей, то всі коди Хаффмана, тобто оптимальні коди, виявляться серед них.

Майже всі існуючі алгоритми стиснення даних можуть бути реалізовані в двох варіантах – статичному та динамічному. Кожен з них займає певну площу, використовується тоді, коли це зручно. Алгоритм Хаффмана не є винятком – використовується в багатьох архіваторах і форматах, іноді він використовується в статичному (JPEG, ARJ) або динамічному («компактна» утиліта для UNIX, ICE) варіантах.

Алгоритм Хаффмана стискає дані за рахунок ймовірності появи символів у джерелі, тому для успішного стиснення та розпакування даних вам потрібно знати ці ймовірності для кожного символу. Статичні алгоритми справляються з цим випадком просто – перед тим, як почати стискати файл, програма сама прокручує файл і обчислює, який символ знайдено і скільки разів. Потім, відповідно до ймовірності появи символів, будується бінарне дерево Хаффмана, з якого витягуються коди різної довжини, що відповідають кожному символу. І на третьому етапі оригінальний файл знову прокручується, коли кожен символ замінюється власним кодом з дерева. Таким чином, статичний алгоритм вимагає двох проходів через вихідний файл для кодування даних [4].

Динамічний алгоритм дозволяє реалізувати однопрохідну модель стиснення. Не знаючи реальної ймовірності появи символів у вихідному файлі, програма поступово змінює бінарне дерево з кожним символом, збільшуючи частоту його появи в дереві та відповідно перебудовуючи дерево. Однак стає очевидним, що при виграші в кількості проходів через вихідний файл втрачається ступінь стиснення, оскільки в статичному алгоритмі частота появи символів була відома з самого початку, а довжини кодів цих символів були ближче до оптимальної, тоді як динамічна модель, вивчаючи джерело, поступово приходить до своїх реальних частотних характеристик і формує їх лише після повного проходження вихідного

файлу. Цей недолік можна компенсувати іншою перевагою динамічного алгоритму. Оскільки динамічне бінарне дерево постійно модифікується новими символами, немає необхідності запам'ятовувати їх частоти заздалегідь. При розархівуванні програма, отримавши код символу з архіву, відновить дерево так само, як і при стисненні, і збільшить частоту його появи на одиницю. Крім того, не потрібно пам'ятати, які символи в двійковому дереві не зустрічаються. Усі символи, які будуть додані до дерева, нам знадобляться для відновлення вихідних даних. Статичний алгоритм для цього випадку трохи складніший – на самому початку стисненого файлу потрібно зберегти інформацію про символи, знайдені у вихідному файлі, та їх частоти. Це пов'язано з тим, що перед початком розархівації необхідно знати, які символи будуть виникати і який у них код.

1.3 Розвиток стиснення без втрат. Арифметичне кодування

Надалі метою було винайти схему кодування, щоб кодувати декілька елементів в менше ніж один біт. У 1977 р. Й. Рейсенен запропонував один з найкращих методів, який називається арифметичне кодування і було запатентовано компанією IBM.

Арифметичне кодування є одним з ентропійних алгоритмів стиснення. На відміну від алгоритму Хаффмана, цей код ні має жорсткого постійного зв’язку між введеними символами і групами бітів в вихідному потоці. Це дає алгоритму чудову гнучкість представляючи дробові частоти за вхідними символами. Як правило, цей код перевершує алгоритм Хаффмана за умовою ефективності стиснення і дозволяє стискання даних з ентропією менше ніж 1 біт на закодований символ.

Це алгоритм забезпечує майже оптимальне стиснення за умовою Шеннона. Майже H біт вимагається для кожного символу, де H є

інформаційною ентропією джерела. На відміну від алгоритму Хаффмана, в арифметичному кодування метод шоу висока ефективність для дробовий нерівний інтервали з в ймовірність розподіл з закодований символи. Проте в для рівномірного розподілу символів, для наприклад, рядка 010101...0101 з довжиною s, метод арифметичного кодування підходить гірше ніж префіксний код Хаффмана і може навіть дати на один біт більше.

Принцип роботи алгоритму такий: нехай є певний алфавіт, а також дані про частоту використання символів (за бажанням). Потім розглянемо відрізок від 0 до 1 на координатній прямій. Ми називаємо це робочим сегментом. Розставимо точки на цьому відрізку таким чином, щоб довжини утворених відрізків дорівнювали частоті використання символу, а кожному такому відрізку відповідав один символ. Тепер ми беремо символ з потоку і знаходимо для нього сегмент серед новоутворених, тепер сегмент для цього символу став робочим. Ми розділимо його так само, як розділяємо інтервал від 0 до 1. Виконайте цю операцію для кількох послідовних символів. Потім виберіть будь-яке число з робочого відрізка. Біти цього числа разом з довжиною його бітового запису є результатом арифметичного кодування використаних символів потоку.

Алгоритми стиснення даних, що використовують у своїй роботі метод арифметичного кодування, перед безпосереднім кодуванням формують модель вхідних даних на основі кількісних або статистичних характеристик, а також повторів або шаблонів, виявлених у початковій послідовності – будь-яка додаткова інформація, яка дозволяє уточнити ймовірність появи символу P у процесі кодування. Очевидно, що чим точніше визначена або передбачена ймовірність символу, тим вище ефективність стиснення.

Приклад коду на мові Python для арифметичного кодування:

```
from decimal import Decimal, getcontext

def arithmetic_encode(data, probabilities):
```

```
    getcontext().prec = 100  # точність обчислень

    low = Decimal(0)
    high = Decimal(1)
    result = Decimal(0)
    for symbol in data:
        range_width = high - low
        high = low + range_width * probabilities[symbol][1]
        low = low + range_width * probabilities[symbol][0]

# Обчислення результату
    result = (low + high) / 2
    return result
def arithmetic_decode(encoded_data, probabilities, length):
    getcontext().prec = 100  # точність обчислень
    low = Decimal(0)
    high = Decimal(1)
    result = []
    for _ in range(length):
        range_width = high - low
        # Пошук символа
        for symbol, (low_range, high_range) in probabilities.items():
            if low_range <= encoded_data < high_range:
                result.append(symbol)
                high = low + range_width * high_range
```

```
                low = low + range_width * low_range
                break
        return result


# Приклад використання
data = "ABABAC"
probabilities = {
    "A": (Decimal(0), Decimal(0.4)),
    "B": (Decimal(0.4), Decimal(0.8)),
    "C": (Decimal(0.8), Decimal(1))
}
encoded_data = arithmetic_encode(data, probabilities)
print("Encoded data:", encoded_data)
decoded_data = arithmetic_decode(encoded_data, probabilities, len(data))
print("Decoded data:", "".join(decoded_data))
```

У цьому прикладі використано дві функції: arithmetic_encode для кодування даних та arithmetic_decode для декодування даних. Ми передаємо дані, ймовірності появи символів та довжину даних. Функція arithmetic_encode обчислює закодоване значення, а функція arithmetic_decode відновлює оригінальні дані.

Ще одним добре відомим алгоритмом в цій групі методів є Run Length Encoding (RLE).

Припустимо, що розмір алфавіту становить N = 256, і ми стискаємо простий текстовий файл (ASCII). Швидше за все, ми не знайдемо в такому файлі всі N символів нашого алфавіту. Потім ми встановлюємо нульову довжину коду пропущеного символу. У цьому випадку список довжин кодів

міститиме велику кількість нулів (пропущених довжин кодів), згрупованих разом. Кожну групу можна стиснути за допомогою кодування довжини серії. Цей алгоритм надзвичайно простий. Замість послідовності з M однакових послідовних елементів ми будемо зберігати перший елемент цієї послідовності та кількість його повторень, тобто (M-1), наприклад: RLE ("AAABBCCCDDDDDD") = A2 B1 C2 D5.

Тут також необхідно враховувати перетворення Берроуза-Вілера (BWT), яке часто використовується разом з RLE. Його також історично називають стисненням сортування блоків, хоча насправді це не алгоритм стиснення. Це алгоритм, який використовується в техніках стиснення даних для перетворення вихідних даних. Термін BWT також називається алгоритмом повного стиснення, який використовує BWT як один із кроків.

Цей алгоритм змінює порядок символів у вхідному рядку таким чином, що повторювані підрядки утворюють послідовні послідовності ідентичних символів на виході. Таким чином, комбінація BWT і RLE виконує завдання стиснення усунення повторюваних підрядків, тобто завдання, подібне до алгоритмів Лемпеля-Зіва.

Крім того, майже точно повторювані (з невеликими відмінностями) підрядки вхідного тексту призводять до послідовностей ідентичних символів, які рідко перемежовуються іншими символами. Якщо після цього ми зробимо крок заміни кожного символу на відстань до його попередньої зустрічі (так званий алгоритм «переміщення вперед», MTF), то отриманий набір чисел матиме надзвичайно вдалий статистичний розподіл для застосування коду Хаффмана або арифметичне стиснення.

На практиці алгоритм стиснення у вигляді BWT → MTF/RLE → Хаффман, який використовується в архіваторі bzip2, трохи перевершує найкращі реалізації LZH за якістю стиснення при аналогічній швидкості.

1.4 Альтернативний підхід. Словникове кодування

Досі ми розглядали статистичний підхід до стиснення файлів невідомого формату. Існує ще один фундаментальний підхід, словникове кодування, коли на кожному кроці роботи алгоритму стиснення наступний символ розміщується як є (зі спеціальним прапорцем без стиснення) або межі слова з попереднього тексту, які збігаються з наступними символами тексту. Видобування файлів, стиснутих таким чином, відбувається дуже швидко, тому ці алгоритми використовуються для створення програм, що саморозпаковуються. Алгоритми словника менш математично обґрунтовані, але більш практичні.

Серед словникових алгоритмів, першим був алгоритм LZ77, розроблений ізраїльськими математиками Яковом Зівом і Авраамом Лемпелем, який було опубліковано у 1977 р. Багато програм для стиснення використовували LZ77 або його модифікації. Основна ідея з LZ77 полягає в тому, що в другому і наступних випадках появи повідомлення його буде замінено за посилання до його перше виникнення [5].

LZ77 використовує попередньо переглянуту частину повідомлення як словник. Щоб зменшити стиснення, він намагається замінити наступний фрагмент вказівником на вміст словника. Тут використовується ковзаюче вікно повідомлення, розділене на дві нерівні частини. Перший, великий за розміром, включає вже переглянуту частину повідомлення. Другий, набагато менший, - це буфер, який містить незашифровані символи вхідного потоку. Як правило, розмір вікна становить кілька кілобайт, а розмір буфера - не більше ста байт. Алгоритм намагається знайти у словнику (більша частина вікна) фрагмент, який відповідає вмісту буфера.

На відміну від LZ77, який працює з уже отриманими даними, LZ78, запропонований у 1978 році, зосереджений на даних, які тільки будуть отримані (LZ78 не використовує «ковзне» вікно, він зберігає словник уже

переглянутих фраз). Алгоритм зчитує символи повідомлення, поки накопичений підрядок не буде повністю включений в одну з фраз словника. Як тільки цей рядок більше не відповідає хоча б одній фразі словника, алгоритм генерує код, що складається з індексу рядка в словнику, який містив рядок введення до останнього введеного символу, і символу, який порушив відповідність. Потім введений підрядок додається до словника. Якщо словник вже заповнений, то фраза, яка рідко вживалася в порівняннях, раніше видаляється.

У 1984 році Велч опублікував свій алгоритм як вдосконалену реалізацію алгоритму LZ78. Цей алгоритм отримав назву LZW (Lempel-Ziv-Welch).

Під час кодування (стиснення) повідомлення цей алгоритм динамічно створює словник фраз: групи бітів (кодів) фіксованої довжини (наприклад, 12-біт, як запропоновано в оригінальній статті Велча) призначаються певним послідовностям символи (словосполучення). Словник ініціалізується всіма односимвольними фразами (у випадку 8-бітних символів це 256 фраз). Під час кодування алгоритм сканує текст, символ за символом зліва направо. Коли алгоритм читає наступний символ у цій позиції, є рядок W максимальної довжини, який відповідає якійсь фразі зі словника. Потім код цієї фрази надсилається на вихід, а рядок WK, де K — символ після W у вхідному повідомленні, вводиться до словника як нова фраза з деяким кодом, призначеним для неї (WK ще не міститься в словнику). Символ K використовується як початок наступної фрази. Більш формально цей алгоритм можна описати такою послідовністю кроків:

1. Ініціалізація словника всіма можливими односимвольними фразами. Ініціалізація вхідної фрази W першим символом повідомлення.

2. Якщо END_OF_MESSAGE, надішліть код для W на вихід і завершіть алгоритм.

3. Прочитайте наступний символ K з початкового повідомлення.

4. Якщо фраза WK вже є в словнику, тоді встановіть для вхідної фрази W значення WK і перейдіть до кроку 2.

5. В іншому випадку надішліть код для W на вихід, додайте WK до словника, встановіть для вхідної фрази W значення K і перейдіть до кроку 2.

Іншим широко використовуваним вдосконаленням алгоритму LZ був LZSS, який використовується в широко популярному форматі Deflate у поєднанні з кодуванням Хаффмана.

2 ВИКОРИСТАННЯ НЕРІВНОМІРНОГО КОДУВАННЯ ДЛЯ СТИСНЕННЯ ДАНИХ З ВТРАТАМИ

Стиснення із втратами передбачає зменшення розміру файлу, як правило, шляхом видалення дрібних деталей, для повного збереження яких потрібен великий обсяг даних. При стисненні з втратами неможливо відновити вихідний файл через видалення важливих даних. Стиснення з втратами даних найчастіше використовується для зберігання зображень і аудіоданих, і хоча воно може досягти дуже високих коефіцієнтів стиснення шляхом видалення даних, воно не розглядається в цій статті. Стиснення даних без втрат — це зменшення розміру файлу таким чином, що функція декомпресії може точно відновити вихідний файл без втрати даних. Стиснення даних без втрат повсюдно використовується в обчислювальній техніці, від економії місця на вашому персональному комп’ютері до надсилання даних через Інтернет, обмін даними через захищену оболонку або перегляд зображень PNG або GIF.

Серед алгоритмів стиснення даних із втратами можна виділити ті, які використовуються для: стиснення аудіо та мови (алгоритм A-law, μ -law алгоритм, лінійне передбачливе кодування (LPC)); стиснення зображення (блокове кодування, алгоритми дискретного косинусного перетворення, фрактальне стиснення, вейвлет-стиснення), стиснення відео.

2.1 Аналіз алгоритмів стиснення звуку та мови

Звук — це коливальний процес, що поширюється в пружному середовищі, наприклад у повітрі. Основним засобом передачі інформації від людини до людини є звукове повідомлення. Існує дві причини необхідності стиснення/декомпресії аудіоданих: економія пам'яті при зберіганні

аудіоінформації та низька пропускна здатність каналів для передачі цифрової інформації на відстань. Використання компресії/декомпресії ефективно вирішує обидві вищезазначені проблеми.

Будь-яке стиснення інформації призводить до погіршення її якості. Проте в процесі еволюції людський слух навчився пристосовуватися до певних видів перешкод, не помічаючи їх присутності в отриманій звуковій інформації. Перш за все, слід зазначити, що слух має логарифмічну чутливість, тобто рівень сприйманого шуму залежить від загального рівня сигналу. Слух людини також є нелінійною системою, дослідженням якої займався М. А. Сапожков. Він ввів поняття критичної смуги мови, визначивши її як «таку смугу частотного діапазону мови, яка сприймається як єдине ціле і за допомогою слухового відчуття може бути замінена еквівалентним тоном». Тому Міжнародний консультативний комітет з телеграфу та телефонії (CCITT) при розробці рекомендації G.711 використовував підхід стиснення звукових зразків (цифрових кодів), заснований на властивостях людського слуху.

Рекомендації CCITT G.711 пропонують два алгоритми для перетворення звукових сигналів: μ-law і A-law. μКодування за законом широко використовується в Сполучених Штатах і Японії, а кодування за законом використовується в Європі. Обидва ці алгоритми перетворюють оригінальні зразки вихідної послідовності сигналу PCM у зразки байтів. Кожен зразок вихідної послідовності перетворюється в один байт.

У рекомендаціях CCITT μзакон і закон наведені в табличній формі. Для представлення числа виділено 3 поля: поле знака, поле мантиси та поле порядку (рис. 1.1). Поле порядку містить ступінь, до якого слід підвести число 2, щоб помножити результат цієї операції на мантису, щоб отримати справжнє абсолютне значення цього числа у форматі з фіксованою комою.

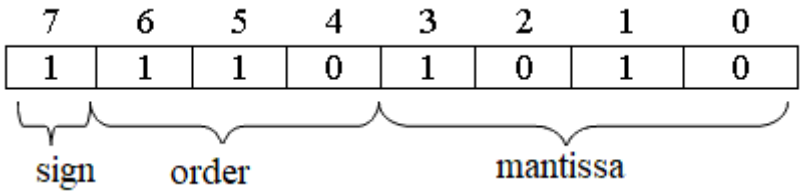

Рисунок 2.1 – Приклад байтової структури для μ-закону та A-закону

A-закон — це алгоритм стиснення з втратами, який використовується для стиснення аудіоданих [8].

Його можна виразити наступними формулами:

$$y = \frac{Ax}{1 + \ln A}, \qquad \text{for } x \leq \frac{1}{A} \tag{2.1}$$

$$y = \frac{1 + \ln Ax}{1 + \ln A}, \qquad \text{for } \frac{1}{A} \leq x \leq 1 \tag{2.2}$$

де A – параметр стиснення. В Європі A=87,6.

Давайте подивимося на алгоритм роботи A-закону:

Крок 1: якщо число від’ємне, воно змінюється, і припускається, що s = 0, інакше s = 1.

Крок 2: 16-бітове число перетворюється на 8-бітне згідно з наведеною нижче таблицею (s — знаковий біт; зірочки вказують на біти, які втрачаються під час стиснення):

Таблиця 2.1 – Перетворення чисел A-закону

Початковий номер	Стиснуте число
s000 0000 wxyz ****	S000 wxyz
s000 0001 wxyz ****	S001 wxyz
s000 001w xyz* ****	S010 wxyz

s000 01wx yz ** ****	S011 wxyz
s000 1wxy z*** ****	s100 wxyz
s001 wxyz **** ****	s101 wxyz
s01w xyz* **** ****	s110 wxyz
s1wx yz ** **** ****	s111 wxyz

Крок 3: Кожен другий біт слід інвертувати, починаючи з правого боку.

μ-закон — це аналоговий алгоритм стиснення, який використовується в системах цифрового зв'язку в Північній Америці та Японії для зміни динамічного діапазону аналогового мовного сигналу перед оцифруванням.

$$y = \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)}, \qquad \text{де } \mu = 255 \tag{2.3}$$

Таблиця 2.2 – Перетворення чисел μ-закону

Початковий номер	Стиснуте число
від +8158 до +4063 через 16 інтервалів з 256	0x80 + інтервал номер
від +4062 до +2015 через 16 інтервалів з 128	0x90 + інтервал номер
від +2014 до +991 через 16 інтервалів з 64	0xA0 + інтервал номер
від +990 до +479 через 16 інтервалів з 32	0xB0 + інтервал номер
від +478 до +223 через 16 інтервалів з 16	0xC0 + інтервал номер
від +222 до +95 через 16 інтервалів з 8	0xD0 + інтервал номер
від +94 до +31 через 16 інтервалів з 4	0xE0 + інтервал номер
від +30 до +1 через 15 інтервалів з 2	0xF0 + інтервал номер
0	0xFF
−1	0x7F
від −31 до −2 через 15 інтервалів з 2	0x70 + інтервал номер
від −95 до −32 через 16 інтервалів з 4	0x60 + інтервал номер

від −223 до −96 через 16 інтервалів з 8	0x50 + інтервал номер
від −479 до −224 через 16 інтервалів з 16	0x40 + інтервал номер
від −991 до −480 через 16 інтервалів з 32	0x30 + інтервал номер
від −2015 до −992 через 16 інтервалів з 64	0x20 + інтервал номер
від −4063 до −2016 через 16 інтервалів з 128	0x10 + інтервал номер
від −8159 до −4064 у 16 інтервалах по 256	0x00 + номер інтервалу

При кодуванні звукової інформації за μ-законом або A-законом всі семпли вхідної послідовності кодуються незалежно, а стиснення, що дозволяє зменшити розмір повідомлення в 2 рази, досягається за рахунок властивостей слуху. Це кодування можна застосувати до будь-якого сигналу, включаючи музику, без помітної втрати якості. Однак таке кодування не може забезпечити значного стиснення сигналу, оскільки воно не враховує властивості сигналу. Тому значного стиснення сигналу можна досягти, передаючи не сам сигнал, а його приріст від відліку до відліку.

На цьому принципі заснована дельта-модуляція. Існують більш складні алгоритми, наприклад, алгоритм адаптивної диференціальної імпульсно-кодової модуляції (ADPCM). Цей алгоритм використовує адаптивний передбачувач значення вхідної вибірки і кодує не сам вхідний сигнал, а помилку його передбачення. Це дозволило зменшити кількість біт, виділених для кодування однієї вибірки, до 4, що відповідає швидкості передачі 32 кбіт/с, без шкоди для якості інформації, що передається [6].

Тепер розглянемо лінійне прогнозуюче кодування (LPC).

Кодування мови на основі методу лінійного передбачення означає, що в лінію передачі надходять не параметри мовного сигналу, а параметри деякого фільтра, в певному сенсі еквівалентного голосовому тракту, і параметри сигналу збудження цього фільтра (рис. 2.2). В якості такого фільтра використовується фільтр лінійного прогнозування (LPF), який раніше називався фільтром аналізатора з передавальною функцією A(z). При

кодуванні (на передачі) оцінюються параметри ФНЧ і параметри сигналу збудження, а при декодуванні (на прийомі) сигнал збудження пропускається через синтезатор фільтра, на виході якого виходить відновлений мовний сигнал. Різні варіанти алгоритмів кодування відрізняються набором параметрів фільтра, що передається, способом генерації сигналу збудження і рядом інших деталей, а процедура кодування мови виглядає наступним чином:

– оцифрований мовний сигнал «розрізається» на сегменти тривалістю 20 мс;

– для кожного сегмента оцінюються параметри ФНЧ і параметри сигналу збудження; сигнал збудження в найпростішому випадку може бути залишком прогнозу, отриманого пропусканням мовного сегмента через фільтр A(z) з параметрами, отриманими з оцінки для цього сегмента;

– фільтра і параметри сигналу збудження кодуються за певним законом і передаються в канал зв'язку.

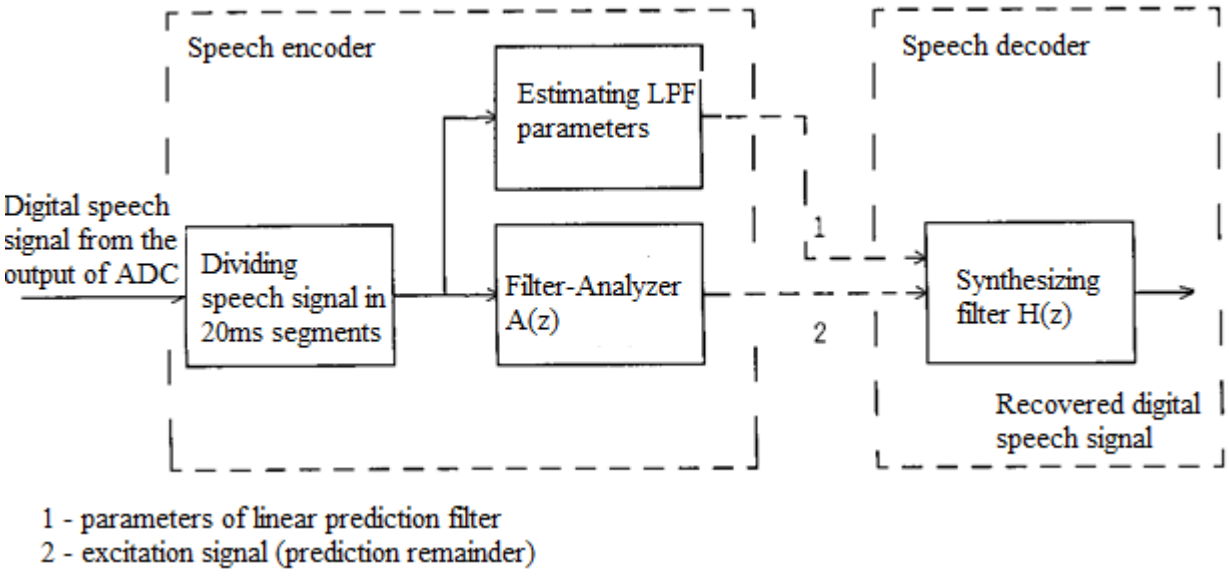

Рисунок 2.2 – Кодування мовлення на основі методу лінійного передбачення

Процес декодування мови полягає в проходженні прийнятого сигналу збудження через синтезуючий фільтр відомої структури, параметри якого

передаються одночасно з сигналом збудження. На вхід аналізуючого фільтра сигнал надходить безпосередньо з виходу АЦП, а вихідний сигнал синтезуючого фільтра надходить на вхід ЦАП. Наведений вище опис процесів кодування та декодування мови пояснює основний принцип роботи кодека. Практичні схеми набагато складніші, і пов'язано це в основному з наступними двома моментами.

По-перше, мовний сигнал має два типи внутрішніх кореляцій, короткочасну і тривалу надмірність, тому переважна більшість сучасних мовних кодеків використовують два передбачники: короткочасний і довгостроковий. Перший передбачник (STP), що враховує короткочасну надлишковість мовного сигналу, пов'язаний з кореляціями між близько розташованими відліками сигналу і визначає огинаючу спектра. Другий, довготривалий передбачник (ДПП) визначає точну структуру мовного сигналу і пов'язаний із співвідношенням між собою двох сегментів сигналу, власне, двох суміжних періодів основного тону. Період основного тону мови коливається в широких межах.

Поєднання двох передбачників з різними характеристиками може значно усунути залишкову надмірність і привести решту прогнозу в його статистичних характеристиках до білого шуму. У цьому випадку залишок передбачення та коефіцієнти прогнозів STP і LTP передаються на приймальну сторону.

По-друге, використання залишку передбачення як сигналу збудження є недостатньо ефективним, оскільки для операції кодування потрібно занадто багато бітів. Тому більш економічні (за завантаженням каналу зв'язку, а не за обчислювальними витратами) методи генерації сигналу збудження знаходять краще практичне застосування.

2.2 Дослідження алгоритмів стиснення зображень

Кодування з усіканням блоків є одним із найпростіших методів кодування, який вимагає незначних обчислювальних витрат для стиснення зображень. Спочатку представлений Делпом і Мітчеллом. Ключова ідея BTC полягає в тому, щоб виконувати квантування блоків пікселів із збереженням моменту, щоб якість зображення залишалася прийнятною і в той же час зменшувався попит на простір для зберігання. Навіть якщо коефіцієнт стиснення алгоритму поступається стандартному алгоритму стиснення JPEG, BTC набув популярності завдяки своїй практичній корисності [7].

Зображення розміром 4x4 пікселя, яке має 8 біт (0-255) на піксель, потребує передачі 128 біт інформації, щоб надіслати це зображення. Але ми можемо покращити це, якщо будемо готові трохи пожертвувати. Щоб використовувати BTC, спочатку ми розбиваємо наше зображення на блоки 4x4. Потім ми кодуємо кожен блок із 16 пікселів окремо. Кожен блок досить малий порівняно із зображенням 512x512. Спочатку ми обчислюємо середнє значення $\bar{x}$та стандартне відхилення σ піксельних значень:

$$\bar{x} = \frac{1}{16}\sum_{i=0}^{15} x_i \tag{2.4}$$

$$\bar{\sigma} = \sqrt{\frac{1}{16}\sum_{i=0}^{15}(x_i - \bar{x})^2} \tag{2.5}$$

Потім ми вирішуємо, наскільки точно ми хочемо надіслати наше середнє значення та стандартне відхилення (зазвичай 8 біт для кожного). Далі ми також надсилаємо двійковий блок 4x4 з 1, якщо вихідний піксель більший або дорівнює середньому, і 0, якщо він менший за середнє. Отже, ми

використовуємо 32 біти для надсилання блоку 4x4 у порівнянні зі 128 бітами початкового зображення.

Дискретне косинусне перетворення (DCT) є найпопулярнішим інструментом обробки сигналів для стиснення зображень і звуків, який можна знайти в таких стандартах, як JPEG.

Щоб виконати DCT-перетворення на зображенні, спочатку ми маємо отримати інформацію про файл зображення (піксельне значення у вигляді цілого числа з діапазоном 0–255), яке ми ділимо на блоки матриці NxN, а потім застосовуємо дискретне косинусне перетворення до цього блоку даних.

Швидке дискретне косинусне перетворення (FDCT)

$$t(u,v) = c(u)c(v)\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} I(k,l)\cos\frac{(2k+1)u\pi}{2N}\cos\frac{(2l+1)v\pi}{2N} \tag{2.5}$$

де i, j = 0, …, N-1, c(i)= $\begin{cases}\sqrt{\frac{1}{N}}, & i = 0\\ \sqrt{\frac{2}{N}}, & i \neq 0\end{cases}$.

Зворотне дискретне косинусне перетворення (IDCT)

$$I(k,l) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c(u)c(v)t(u,v)\cos\frac{(2k+1)u\pi}{2N}\cos\frac{(2l+1)v\pi}{2N} \tag{2.6}$$

де i, j = 0, …, N-1, c(i)= $\begin{cases}\sqrt{\frac{1}{N}}, & i = 0\\ \sqrt{\frac{2}{N}}, & i \neq 0\end{cases}$.

Дискретне перетворення має такі властивості:

– некорельовані коефіцієнти. Коефіцієнти незалежні один від одного, тобто точність одного коефіцієнта не залежить від іншого;

– енергетичне «ущільнення». Перетворення зберігає основну інформацію в невеликій кількості коефіцієнтів. Ця властивість найбільш помітна на фотореалістичних зображеннях.

Коефіцієнти t(u,v) – амплітуди просторових частот зображення. У випадку зображень з плавними переходами більша частина інформації міститься в низькочастотному спектрі.

Після застосування дискретного косинусного перетворення ми побачимо, що його понад 90% даних будуть у нижчій частотній складовій. Для спрощення ми взяли матрицю розміром 8 X 8 із усіма значеннями 255 (вважаючи, що зображення повністю біле), і ми збираємося виконати двовимірне дискретне косинусне перетворення для спостереження за виходом.

Давайте поглянемо на фрактальне стиснення. Значною мірою недоліки якості відновленого зображення у стисненому форматі за JPEG зумовлені тим, що під час стиснення не враховуються особливості зображення, тобто не виявляється його структура, характерні ділянки тощо (рис. 2.3). Саме врахування специфіки зображення лежить в основі фрактального методу, запропонованого Мандельбротом у 1988 році. За Мандельбротом, фрактал — це структура, яка виділяється під час аналізу зображення і має однакову форму незалежно від його розміру. Наприклад, у зображенні крони дерева фрактал — це зображення листка. Отже, зображення можна зібрати з фракталів, тобто зображення з точки зору фрактального підходу є суперпозицією фракталів. У свою чергу, окремий фрактал можна описати якимось стандартним способом.

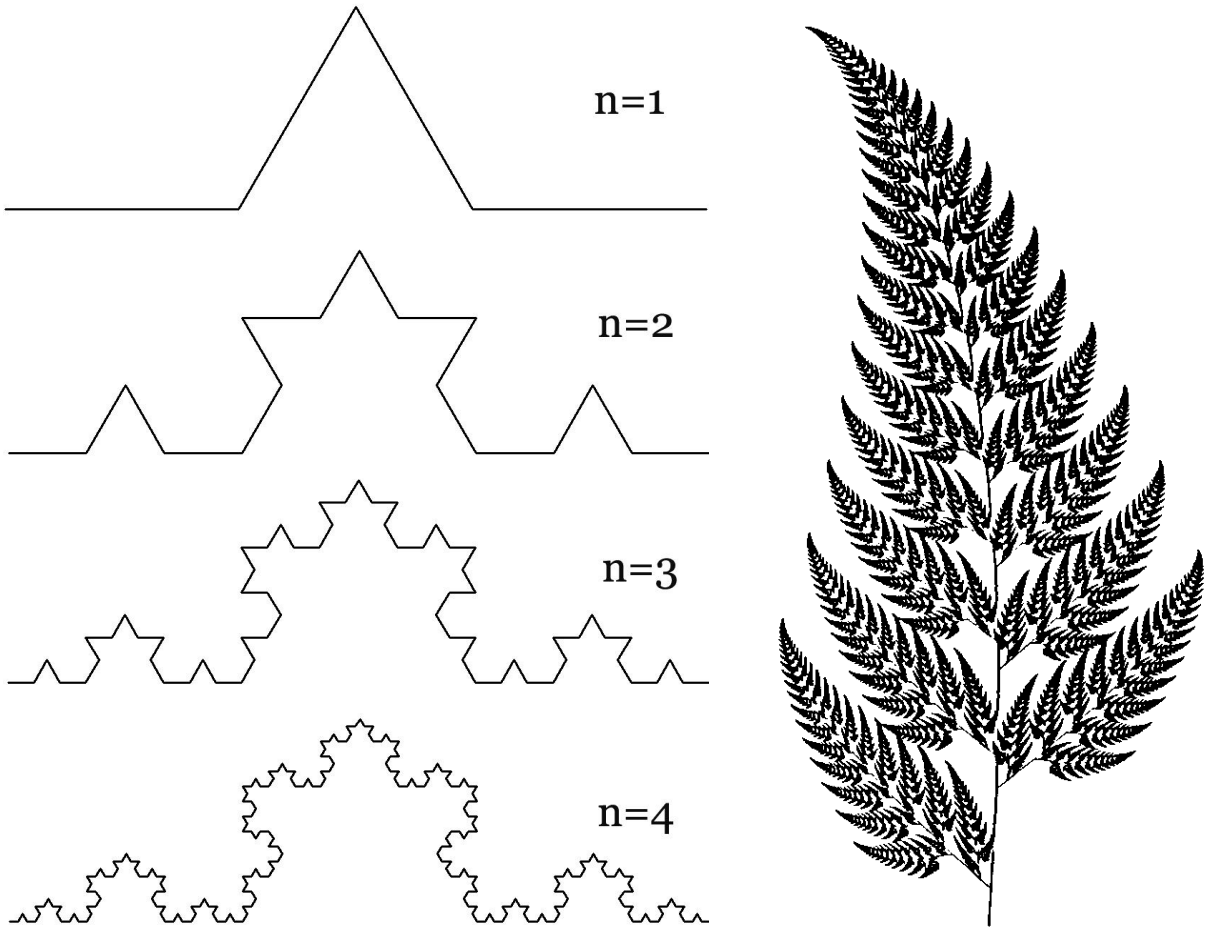

Рисунок 2.3 – Прості приклади фракталів

У фрактальному кодуванні зображень ми намагаємося знайти багато стислих відображень, які відображають блоки домену в набір блоків рангу.

Основний алгоритм кодування виглядає наступним чином:

1. Розбийте оригінальне зображення на блоки рангу, що не перетинаються.

2. Ми покриваємо зображення послідовністю блоків доменів, які можуть перетинатися (домени можуть бути різного розміру і зазвичай їх тисячі).

3. Для кожного рангового блоку ми знаходимо домен і відповідне перетворення, яке охоплює ранговий блок найкращим чином.

4. Якщо ми не отримуємо достатньо точного збігу, тоді ми ділимо рангові блоки на менші. Ми продовжуємо цей процес, доки не досягнемо прийнятного збігу або розмір блоків рангу не досягне певної попередньо визначеної межі.

У процесі перетворення реальних растрових даних у фрактальні коди реалізуються 2 великі переваги:

– можливість масштабувати фрактальні зображення без введення артефактів і втрати деталей. Процес фрактальної панарамізації не залежить від роздільної здатності растрового зображення. Масштаб обмежений лише обсягом вільної пам'яті комп'ютера;

– розмір фізичних даних, які використовуються для запису фрактальних кодів, набагато менший за розмір вихідних растрових даних. Коефіцієнт стиснення реального зображення за допомогою фрактального кодування досягає 200:1.

Вейвлет-стиск. Основна ідея вейвлет-аналізу полягає в представленні даних у формі грубого наближення та детальної інформації. Чим детальніша інформація, тим вищий рівень деталізації (частота).

Вейвлет це математична функція, яка дозволяє аналізувати різні частотні компоненти даних. Графік функції має вигляд хвилеподібних коливань з амплітудою, яка зменшується до нуля при переміщенні графіка від початку координат. Вейвлет-спектрограми принципово відрізняються від звичайних спектрів Фур'є тим, що вони дають чітку часову прив'язку спектра різних характеристик сигналу.

Вейвлет-перетворення є дискретними та неперервними. Для стиснення сигналу використовується дискретне вейвлет-перетворення.

Найбільш відомі приклади вейвлетів показані на рис. 2.4 і рис. 2.5.

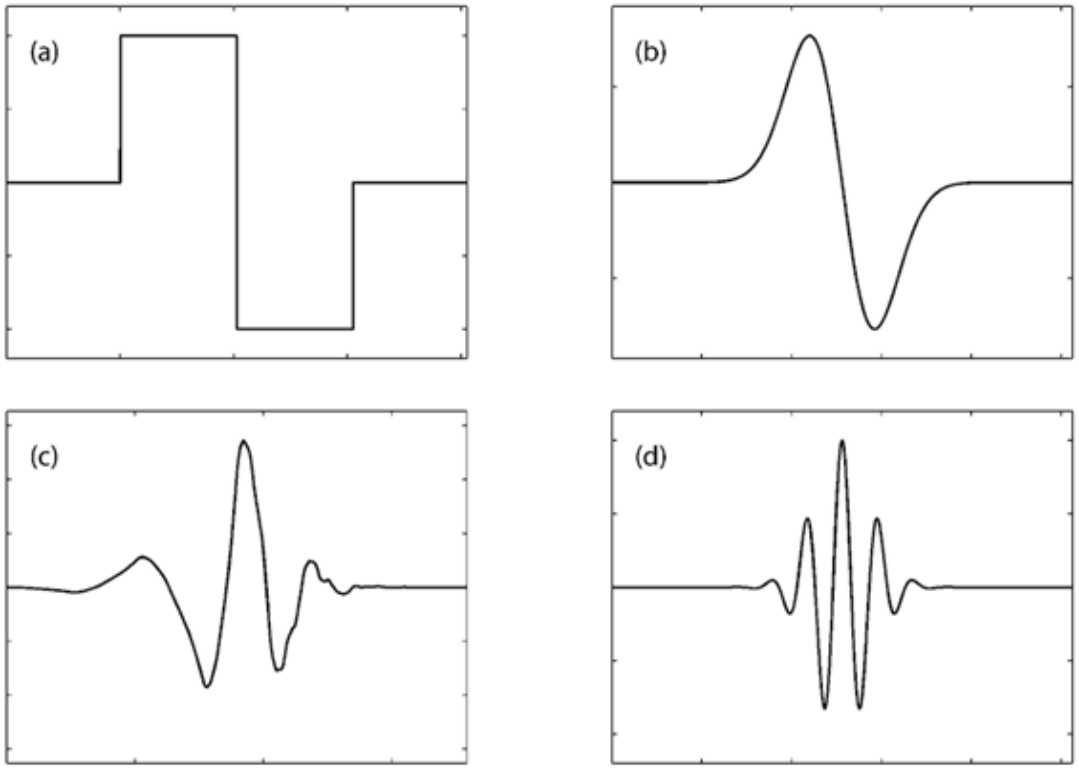

Рисунок 2.4 – Загальні материнські вейвлети, що використовуються для вейвлет-аналізу: (a) вейвлет Хаара, (b) вейвлет Гаусса 1-го порядку, (c) вейвлет Добеші 4-го порядку та (d) вейвлет Морле.

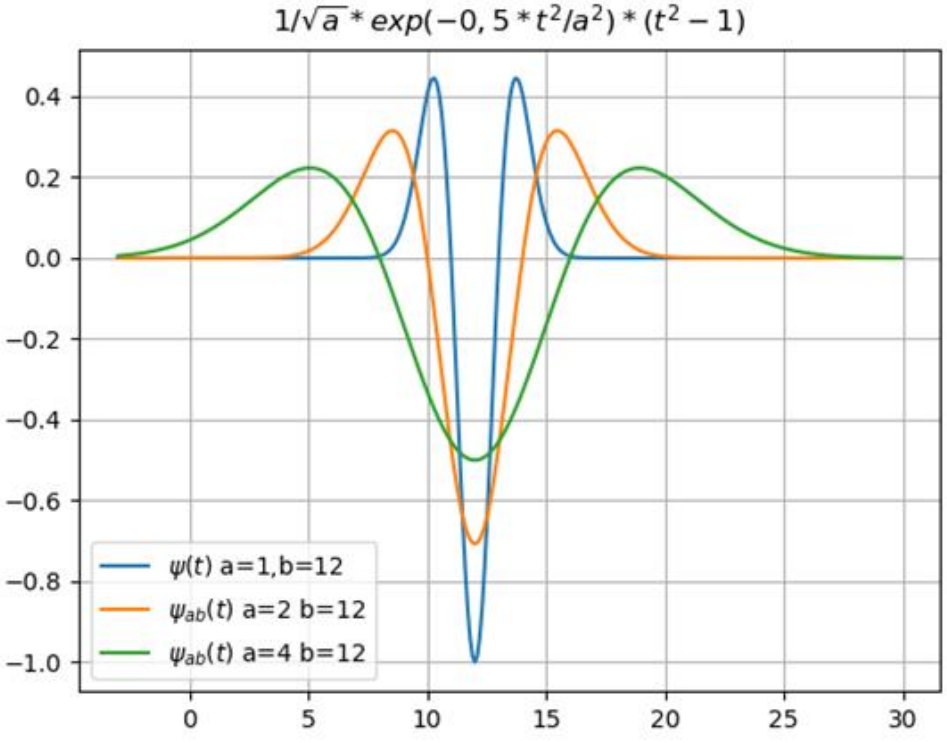

Рисунок 2.5 – «Мексиканський капелюх»

Хаара — один із перших і найпростіших вейвлетів. Перетворення Хаара використовується для стиснення вхідних сигналів, стиснення зображень, переважно кольорових і чорно-білих з плавними переходами. Ідеально підходить для знімків рентгенівського типу. Цей вид архівації відомий давно і безпосередньо виходить з ідеї використання когерентності областей. Коефіцієнт стиснення встановлений і варіюється від 5 до 100. При спробі встановити більший коефіцієнт на різких межах, особливо тих, що йдуть по діагоналі, виникає «ефект сходів» — сходинки різної яскравості розміром у кілька пікселів.

Нехай є одновимірний дискретний вхідний сигнал S. Кожній парі сусідніх елементів ставляться два числа: їх напівсума і напіврізниця. Після повторення цієї операції для кожного елемента вихідного сигналу на виході надходять два сигнали, один з яких є грубою версією вхідного сигналу (напівсумами), а другий містить детальну інформацію, необхідну для відновлення вихідного сигналу. (піврізниці). Аналогічно, перетворення Хаара може бути застосоване до отриманого сигналу тощо.

На прикладі перетворення Хаара добре видно структуру дискретного вейвлет-перетворення сигналу. На кожному кроці перетворення сигнал розбивається на дві складові: наближення з нижчою роздільною здатністю та детальну інформацію.

Вейвлет-перетворення з нульовими коефіцієнтами краще стискається відомими методами стиснення без втрат. Існують також способи змінити порядок коефіцієнтів певним чином для забезпечення кращого стиснення.

Переваги з вейвлет стиснення:

– адаптація до особливостей конкретного сигналу;

– можливість прогресивного (з поступовим збільшенням деталізації) і локального розпакування.

2.3 Огляд алгоритмів стиснення відео

По суті, відео — це тривимірний масив кольорових пікселів. Два виміри означають вертикальну і горизонтальну роздільну здатність кадру, а третій вимір-час. Кадр — це масив усіх пікселів, видимих камерою в певний момент часу, або просто зображення.

Стиснення було б неможливим, якби кожен кадр був унікальним, а розташування пікселів було абсолютно випадковим, але це не так. Тому можна стиснути, по-перше, саму картинку – наприклад, фотографія блакитного неба без сонця фактично зводиться до опису граничних точок і градієнта заливки. По-друге, ви можете стискати схожі суміжні кадри. Зрештою, алгоритми стиснення зображення та відео схожі, якщо розглядати відео як тривимірне зображення з часом як третьою координатою.

Однією з найпотужніших технологій для підвищення ступеня стиснення відеоданих є компенсація руху. У будь-якій сучасній системі стиснення відео наступні кадри в потоці використовують подібність областей у попередніх кадрах для збільшення коефіцієнта стиснення.

Однак через рух будь-яких об'єктів в кадрі (або самої камери) використання подібності сусідніх кадрів було неповним. Технологія компенсації руху дозволяє знаходити схожі ділянки, навіть якщо вони зсунуті відносно попереднього кадру.

На даний момент майже всі алгоритми стиснення відео використовують дискретне косинусне перетворення (DCT) або його модифікації для усунення просторової надмірності. Інші методи, такі як фрактальне стиснення та дискретне вейвлет-перетворення, також вивчалися, але зараз вони зазвичай використовуються лише для стиснення нерухомих зображень.

Використання більшості методів стиснення (таких як дискретне косинусне перетворення та вейвлет-перетворення) також передбачає використання процесу квантування. Квантування може бути як скалярним, так і векторним, однак більшість схем стиснення на практиці використовують скалярне квантування через його простоту.

Двома найуспішнішими кодеками всіх часів були MPEG-2 і H.264, обидва стандартизовані кодеки, розроблені в основному для ринку мовлення, причому потокове передавання не має значення для першого та запізнілою думкою для другого. Стандарти мають вирішальне значення для мовлення, щоб об'єднати разом безліч постачальників у сферах кодування, передачі та декодування. Незважаючи на те, що кожен кодек несе гонорар, він був розумним, а єдиний патентний пул був добре керованим і прозорим.

Сучасне цифрове телевізійне мовлення стало доступним саме завдяки стисненню відео. Телевізійні станції можуть транслювати не тільки відео високої чіткості (HDTV), а й кілька телевізійних каналів в одному фізичному телевізійному каналі (6 МГц).

Незважаючи на те, що більшість відеоконтенту сьогодні транслюється за стандартом стиснення відео MPEG-2, тим не менш, нові та більш ефективні стандарти стиснення відео вже використовуються в мовленні, наприклад, H.264 і VC-1.

Існує кілька основних технологій, реалізованих у різних кодеках AVI. Наприклад, Indeo 3.2 і Cinepak використовують векторне квантування, міжнародні стандарти MPEG-1, MPEG-2, MPEG-4, H.261 і H.263 використовують комбінацію дискретного косинусного перетворення та компенсації руху. Деякі кодеки останнього покоління засновані на дискретному вейвлет-перетворенні. Інші технології включають рекурсивний алгоритм стиснення зображень, розроблений Iterated Systems.

Стандарт MPEG-2 навмисно не визначає, як стискати зображення (звук), він лише вказує, як має бути розроблено стиснене зображення (звук).

Стандарт не визначає, як має бути реалізований кодер або декодер MPEG-2; він лише визначає структуру даних. Це дозволяє учасникам ринку конкурувати один з одним за створення кращих пристроїв і алгоритмів.

Формати стиснення сімейства MPEG зменшують обсяг інформації таким чином:

– усувається тимчасова надмірність відео (враховується лише диференційована інформація);

– усувається просторова надмірність зображень шляхом придушення дрібних деталей сцени;

– частина кольорової інформації виключається;

– інформації результуючого цифрового потоку підвищується шляхом вибору оптимального математичного коду для його опису.

MPEG-2 використовується для загального стиснення рухомих зображень і звуку і визначає формат відеопотоку, який може бути представлений у вигляді трьох типів кадрів:

– незалежно стислі кадри (I-кадри, інтра-кадри);

– кадри, стиснуті з використанням передбачення на основі попередніх кадрів (P-кадри, прогнозовані кадри);

– кадри, стиснуті за допомогою передбачення на основі попередніх і наступних кадрів (B-кадри, двонаправлені кадри).

Відповідні групи кадрів від одного I-кадру до іншого утворюють GOP (Group Of Pictures) – набір кадрів.

У порівнянні з MPEG-1 формат стиснення MPEG-2 має наступні переваги:

– MPEG-2 забезпечує масштабованість різних рівнів якості зображення в одному відеопотоці;

– у форматі стиснення MPEG-2 точність векторів руху підвищена до 1/2 пікселя;

– користувач може вибрати довільну точність дискретного косинусного перетворення;

– Формат стиснення MPEG-2 включає додаткові режими передбачення.

Кодек Діксона використовує метод стиснення, який називається "розподілом пікселів". Цей метод стискає відео, розподіляючи пікселі в просторі. Відкритий кодек Діксона підтримує широкий спектр форматів відео, включаючи MPEG-4, H.264 та HEVC. Він також підтримує широкий спектр операційних систем, включаючи Windows, macOS, Linux та Android.

MPEG-4 використовує технологію так званого фрактального стиснення зображень. Фрактальне (контурне) стиснення передбачає виділення із зображення контурів і текстур об’єктів. Контури представлені у вигляді т.зв. сплайни (поліноміальні функції) і кодуються контрольними точками. Текстури можуть бути представлені як коефіцієнти просторового частотного перетворення (наприклад, дискретне косинусне або вейвлет-перетворення).

MPEG-4 включає прогресивні та черезрядкові методи сканування та підтримує довільну просторову роздільну здатність і швидкість передачі даних у діапазоні від 5 кбіт/с до 10 Мбіт/с. MPEG-4 покращив алгоритм стиснення, якість і ефективність якого покращено для всіх підтримуваних швидкостей передачі даних.

3 АНАЛІЗ ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМІВ СТИСНЕННЯ ДАНИХ В СУЧАСНИХ СИСТЕМАХ ТА МЕРЕЖАХ

У сучасних комп'ютерних системах і програмуванні, нерівномірні коди використовуються для стиснення даних, зменшення їх розміру і покращення ефективності передачі. Найвідомішими сучасними нерівномірними кодами є:

1. Адаптивне кодування з втратами (Adaptive Lossy Coding, ALC): Цей код використовується для стиснення зображень та відео з втратами. Він використовує модель прогнозування для видалення зайвої інформації з даних і використовує нерівномірні коди для ефективного кодування залишкової інформації.

2. Адаптивне бінарне кодування (Adaptive Binary Coding, ABC): Цей код використовується для стиснення текстових даних. Він використовує нерівномірні коди для кодування символів залежно від їх відносної ймовірності в тексті. Часті символи отримують короткі коди, тоді як рідкі символи отримують довші коди.

3. Відкритий кодек Діксона (Dixon's Open Codec, DOC): Цей кодек використовується для стиснення аудіоданих. Він використовує нерівномірні коди для кодування амплітуд звукових сигналів. Він використовує адаптивну модель ймовірностей для ефективного кодування великого динамічного діапазону амплітуд.

4. Ймовірнісне адаптивне кодування (Probabilistic Adaptive Coding, PAC): Цей код використовується для стиснення даних з великим обсягом повторюваної інформації, таких як зображення з великою кількістю пікселів однакового кольору. Він використовує нерівномірні коди для ефективного кодування повторюваних символів.

5. Гібридне адаптивне кодування (Hybrid Adaptive Coding, HAC): Цей код використовується для стиснення даних, що містять різні типи інформації,

такі як текст, зображення та відео. Він використовує комбінацію нерівномірних кодів для ефективного кодування різних типів даних.

Ці сучасні нерівномірні коди використовуються в різних сферах, включаючи зображення, відео, аудіо, текстові дані та інші типи даних. Вони дозволяють досягти високого ступеня стиснення без втрати якості інформації.

3.1 Стиснення даних без втрат

Великі корпорації використовували алгоритми стиснення для зберігання постійно зростаючих обсягів даних, але справжнє поширення алгоритмів відбулося з народженням Інтернету наприкінці 80-х. Пропускна здатність каналу була надзвичайно вузька. Для стиснення даних, що передаються по мережі, були придумані файлові архіватори.

Архіватор файлів — це програмне забезпечення, призначене для упаковки без втрат одного або кількох файлів в один архівний файл або в серію архівів для легкого перенесення та/або зберігання даних. Розпакування архівів виконується за допомогою того ж архіватора або за допомогою сторонніх сумісних утиліт. Більшість сучасних архіваторів також стискають архівні дані.

Серед стандартних і найбільш корисних властивостей програм-архіваторів на даний момент слід відзначити наступні:

– створення багатотомних архівів з можливістю вказати довільний розмір тому;

– створення архівів SFX, що саморозпаковуються;

– створення багатотомних SFX-архівів;

– автоматичне видалення файлів після архівування;

– повне архівування каталогів і дисків із збереженням атрибутів файлів;

– розміщення авторських коментарів внизу в архіві;

– пароль доступу до архіву;

– підтримка захищеного режиму (DPMI, VCPI), розширення і розширення пам'яті;

– введення в архів циклічних кодів помилок, що дозволяють відновити пошкоджені архіви;

– надання детальної інформації в кінці процесу архівування та на вимогу (коефіцієнт стиснення, приблизний час стиснення/розпакування, розміри файлів тощо);

– наявність довідкової системи;

– відносно невеликий розмір програмного модуля архіватора.

Більшість програм-архіваторів стискають кожен файл окремо, але деякі стискають файли в загальний потік, що збільшує ступінь стиснення, але в той же час ускладнює роботу з отриманим архівом. Наприклад, заміна файлу в такому архіві на новішу версію може вимагати перекодування всього архіву. Прикладом програми, яка може стискати файли в спільному потоці, є RAR. Архіватори Unix (gzip, bzip2) майже завжди стискають файли в загальний потік.

Розглянемо два найпоширеніших представника файлових архіваторів: формати ZIP і RAR.

Формат ZIP. Філ Кац випустив PKZIP в 1989 році. У зв'язку з патентом на алгоритм LZW він був змушений змінити базовий алгоритм на новий - так з'явився Deflate. Deflate був запатентований Katz, але використовувався ширше, наприклад, у форматі Portable Network Graphics (PNG), який з'явився в 1994 році.

Deflate це алгоритм стиснення без втрат, який використовує комбінацію алгоритмів LZ77 і Хаффмана. Потік Deflate містить серію блоків. Перед кожним блоком є трибітовий заголовок:

Один біт: позначка останнього блоку.

– 1: блок останній;

– 0: блок не останній.

Два біти: метод кодування даних.

– 00: дані не кодуються (блок представляє безпосередньо вихідні дані);

– 01: дані, закодовані за допомогою статичного алгоритму Хаффмана;

– 10: дані, закодовані за допомогою динамічного алгоритму Хаффмана;

– 11: зарезервоване значення (помилка).

Більшість блоків кодуються за допомогою методу 10 (динамічного Хаффмана), який забезпечує оптимізоване дерево кодів Хаффмана для кожного нового блоку. Інструкції щодо створення дерева коду Хаффмана слідують безпосередньо за заголовком блоку.

Стиснення проводиться в два етапи:

– заміна повторюваних рядків покажчиками (алгоритм LZ77);

– заміна символів на нові залежно від частоти їх появи (алгоритм Хаффмана).

ZIP використовувався виключно до середини 90-х, але в 1993 році інженер Євген Рошаль придумав власний формат і алгоритм RAR. Останні версії базуються на PPM (Prediction by Partial Matching — адаптивний статистичний алгоритм стиснення даних без втрат) і LZSS (Lempel-Ziv- Storer

-Szymanski — алгоритм стиснення даних без втрат, отриманий від методу LZ77).

Основні відмінності між стисненням ZIP і RAR:

1. Стиснення у формат ZIP відбувається трохи швидше, однак розмір кінцевого архіву трохи більший у порівнянні з RAR.

2. У RAR є можливість створення «Безперервного архіву», за рахунок чого досягається ще більше стиснення, у ZIP такої можливості немає.

3. ZIP так і не навчився створювати багатотомний (архів, розділений на кілька файлів однакового розміру) типу архіву, а RAR знав, як це зробити з самого початку.

4. Відновити пошкоджений ZIP-архів дуже важко; у RAR ця функція передбачена спочатку в алгоритмі.

5. До цього дня будь-який ZIP-архіватор/архів може мати проблеми зі стисненням/розпакуванням файлів розміром понад 2 гігабайти.

6. Щоб розпакувати/створити RAR-архів, необхідно встановити архіватор, а ZIP-архів можна розпакувати/створити в популярній операційній системі стандартним способом.

І формати RAR, і ZIP у певний момент почали використовувати техніку PPM (передбачення шляхом часткового збігу). Це адаптивний статистичний алгоритм стиснення даних без втрат, заснований на контекстному моделюванні та передбаченні. Модель PPM використовує контекст, який є набором символів у нестисненому потоці, що передує цьому, щоб передбачити значення символу на основі статистики. Сама модель PPM лише передбачає значення символу, пряме стиснення здійснюється за допомогою алгоритмів ентропійного кодування, таких як алгоритм Хаффмана, арифметичне кодування.

Довжина контексту, яка використовується в передбаченні, зазвичай дуже обмежена. Ця довжина позначається n і визначає порядок моделі PPM, який позначається як PPM(n). Також існують необмежені моделі, які просто позначаються PPM. Якщо символ неможливо передбачити з контексту з n символів, тоді робиться спроба передбачити його за допомогою n-1 символів. Рекурсивний перехід до моделей нижчого порядку триває до тих пір, поки передбачення не відбудеться в одній із моделей або коли контекст стане нульовою довжиною (n = 0). Моделі ступеня 0 і −1 слід описати окремо. Модель нульового порядку еквівалентна випадку контекстно-вільного моделювання, коли ймовірність символу визначається виключно частотою його появи в стиснутому потоці даних. Подібна модель зазвичай використовується з кодуванням Хаффмана. Модель порядку −1 — це статична модель, яка присвоює певне фіксоване значення ймовірностям символу; зазвичай усі символи, які можуть зустрічатися в стиснутому потоці даних, вважаються рівноімовірними. Щоб отримати хорошу оцінку ймовірності символу, необхідно враховувати контексти різної довжини. PPM є різновидом стратегії змішування, коли оцінки ймовірності на основі контекстів різної довжини об’єднуються в одну загальну ймовірність. Отримана оцінка кодується будь-яким ентропійним кодувальником (EE), зазвичай різновидом арифметичного кодера. На етапі ентропійного кодування відбувається саме стиснення.

Для конкретної довжини контексту була використана наступна формула для визначення ймовірності символу *x*:

$$p(x) = \frac{c}{a} \times \left(\frac{a}{a+b}\right)^{param1} \tag{3.1}$$

де a — це кількість контекстних збігів, b — потужність набору символів, які зустрічаються після збігів, а c — кількість разів, коли x трапляється після кожного збігу. І param1 є параметром, який можна налаштувати.

Розподіли ймовірностей для різних довжин контексту були об'єднані разом за допомогою зваженого середнього (групове голосування). Вага w для певної довжини контексту n була розрахована за допомогою наступної рекурсивної функції:

$$w(n) = \begin{cases} 1, if\ n = maxLength \\ param2,\ if\ w(n) < param2 \\ param3 \times w(n+1) + (1 - param3) \times \frac{w(n+1) \times b}{a+b},\ otherwise \end{cases} \quad (3.2)$$

де param2 і param3 — параметри, які можна налаштувати (в діапазоні від нуля до одиниці), а maxLength — це довжина максимального збігу контексту. Нарешті, отриманий розподіл ймовірностей за символами було нормалізовано до суми до одиниці.

Середній коефіцієнт помилок передбачення показаний на рис. 3.1.

Одним із способів вимірювання ефективності стиснення є використання розміру файлу стиснутих даних. Однак розмір файлу залежить від певного типу схеми кодування (наприклад, арифметичного кодування або кодування Хаффмана). Оскільки PPM займається створенням розподілу ймовірностей для передбачення символів, є способи вимірювати його ефективність стиснення безпосередньо з цих розподілів. Одним із них є перехресна ентропія (рис. 3.2), яка вимірює середню кількість бітів, необхідних для ідентифікації події, взятої з набору, якщо схема кодування, що використовується для набору, оптимізована для оціненого розподілу ймовірностей, а не для справжнього розподілу.

Для послідовності з N символів x_i та ймовірності $p(x_i)$, призначеної кожному символу алгоритмом передбачення, перехресну ентропію можна визначити як:

$$H = -\sum_{i=1}^{N} \frac{1}{N} \log_2 p(x_i) \tag{3.3}$$

Важливою проблемою для алгоритму PPM є обробка нових символів, які ще не були помічені у вхідному потоці. Ця проблема називається проблемою нульової частоти. Деякі реалізації PPM припускають, що лічильник нового символу дорівнює фіксованому значенню, наприклад, одиниці. Інші реалізації, такі як PPM-D, збільшують псевдолічильник нового символу кожного разу, коли новий символ з'являється в потоці (іншими словами, PPM-D оцінює ймовірність появи нового символу як відношення кількості символів унікальних символів до загальної кількості використаних символів).

Слід зазначити, що хоча PPM добре працює в тестах стиснення тексту, існують інші сучасні алгоритми, які перевершують його. Одним із прикладів тесту стиснення є премія Хаттера. Це конкурс на стиснення перших 100 МБ Вікіпедії. Алгоритм під назвою PAQ зараз домінує в конкурсі. PAQ тісно пов'язаний із PPM, покращуючи його шляхом поєднання контекстів, які є довільними функціями історії введення.

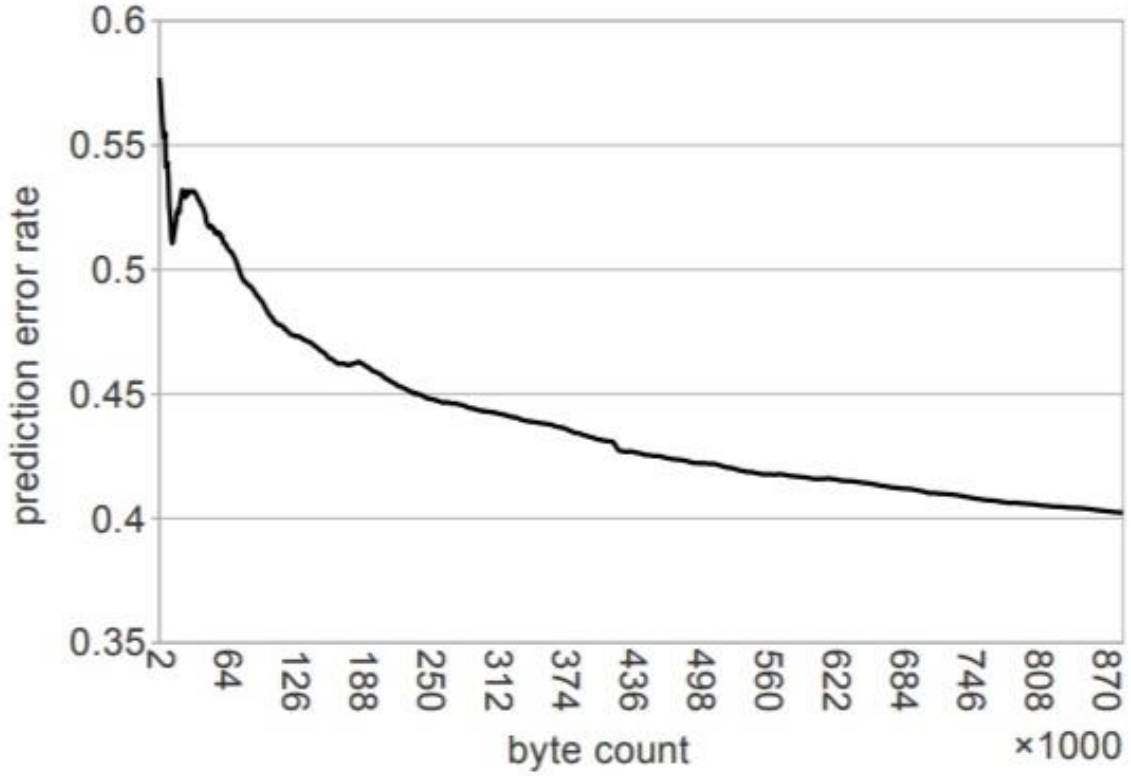

Рисунок 3.1 – Середня частота помилок прогнозування

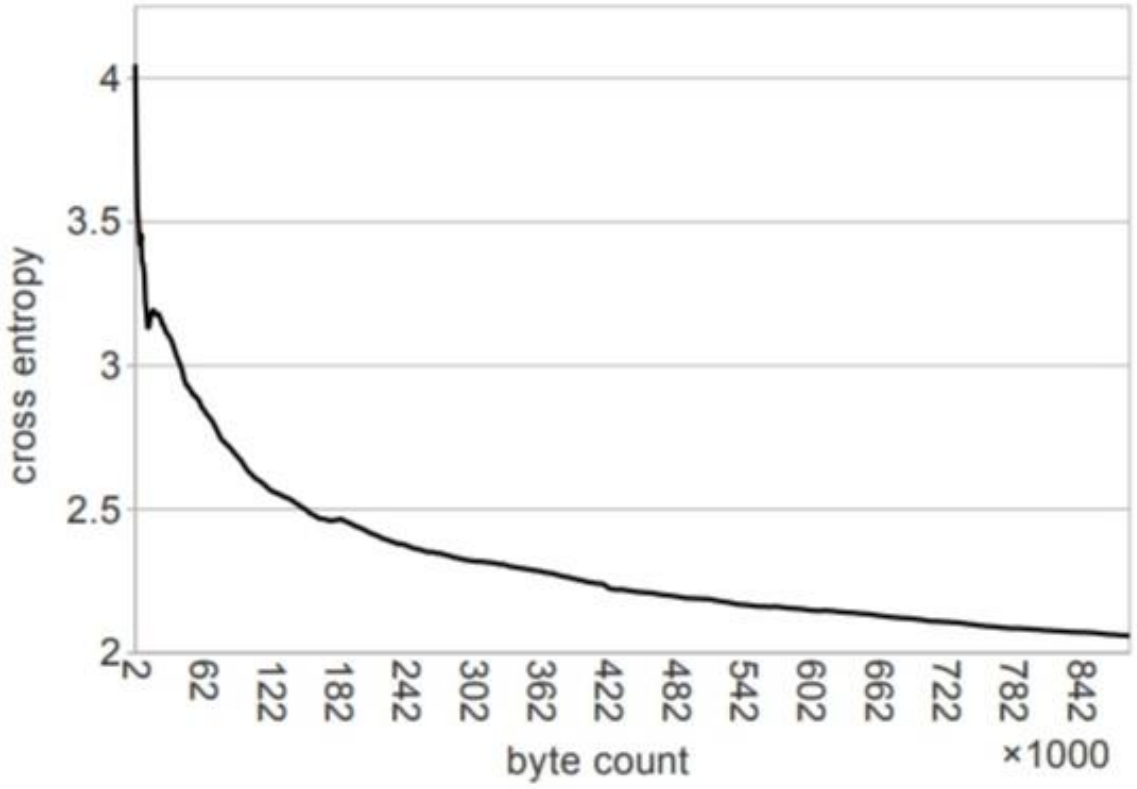

Рисунок 3.2 – Перехресна ентропія для кожного байта

PAQ з'явився в 2002 році. Автор Метт Махоні використав вдосконалену версію алгоритму PPM, використовуючи техніку під назвою «контекстне змішування». Це дозволяє використовувати більше однієї статистичної моделі для покращення прогнозу за частотою появи символів.

У сімействі PAQ наступні моделі в основному використовуються для збору статистики та прогнозування ймовірності наступного символу:

– n-grams – контекст; попередні n байт (як у PPM);

– словникові n-грами, які нечутливі до регістру літер (великих/малих) і неалфавітних символів (корисно в текстових даних);

– «розріджені» контексти, наприклад, другий і четвертий символи перед закодованим (корисно в деяких бінарних форматах);

– «аналогові» контексти, що складаються з верхньої половини двійкового представлення 8- або 16-бітних слів (корисно у форматах мультимедійних даних);

– двовимірні контексти (корисно для зображень, табличних даних). Довжина рядка визначається шляхом пошуку повторюваних шаблонів байтів;

– спеціалізовані моделі, такі як x86-виконувані файли або Windows Bitmap, TIFF, JPEG-зображення. Ці моделі активуються, коли виявляється цей тип файлу.

Усі версії PAQ передбачають і стискають по одному біту, але відрізняються деталями реалізації того, як передбачення об’єднуються та обробляються після цього. Як тільки прогнозується ймовірність появи наступного біта, біт стискається арифметичним кодувальником.

Зараз алгоритм PAQ набирає популярності завдяки дуже хорошому коефіцієнту стиснення (хоча працює він дуже повільно). Але завдяки збільшенню швидкодії комп'ютера, швидкість роботи стає менш критичною.

З іншого боку, алгоритм Лемпеля-Зіва-Маркова LZMA є компромісом між швидкістю та ступенем стиснення та може генерувати багато цікавих розгалужень.

Набір розробників з відкритим кодом LZMA, написаний на C++, використовує вдосконалений алгоритм стиснення LZ77, доповнений алгоритмом інтервального кодування, а також спеціальні процедури обробки бінарних файлів (рис. 3.3) [9]. LZMA підтримує різноманітні хеш-ланцюги, двійкові та префіксні дерева як основу для алгоритмів пошуку за словниками [10].

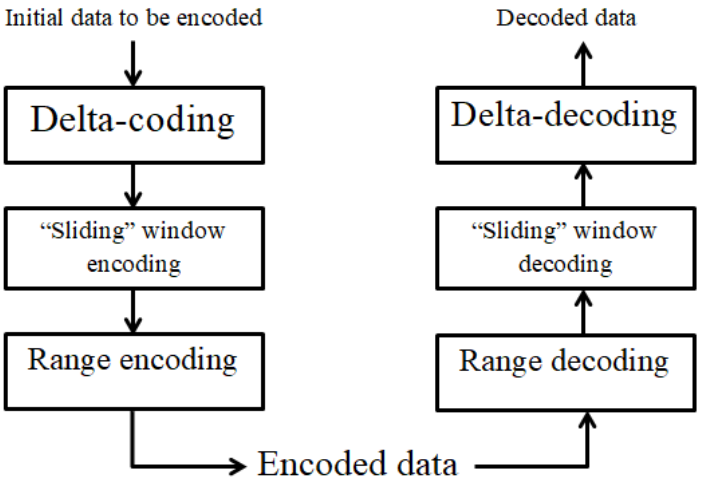

Рисунок 3.3 – Схема роботи алгоритму LZMA

3.2 Стиснення даних із втратами. Стандарти стиснення звуку

GSM (Глобальна система мобільного зв'язку) — стандарт системи стільникового зв'язку, популярний за межами США. GSM включає кодек, який часто називають просто GSM під час обговорення кодеків.

Вперше для GSM був винайдений Full Rate Standard для кодування мови. Оригінальний мовний кодек GSM «Full Rate» називається RPE-LTP (довгострокове прогнозування регулярного імпульсного збудження) з порядком передбачення 8. Цей кодек використовує інформацію з попередніх

зразків (ця інформація не змінюється дуже швидко), щоб передбачити поточний вибірку. Мовний сигнал розбивається на блоки по 20 мс. Потім ці блоки передаються до мовного кодека, який має швидкість 13 кбіт/с, щоб отримати блоки по 260 біт.

На початку 1990-х років, коли він розроблявся, кодек FR був хорошим компромісом між складністю реалізації та якістю звуку.

Регулярне імпульсне збудження – довгострокове прогнозування (RPE-LTP) у GSM – це схема, призначена для зменшення обсягу даних, що передаються між мобільною та базовою станціями. Принцип дії цієї схеми (рис. 3.4) полягає в наступному: для кожної вибірки мовного сигналу за допомогою внутрішньої логіки мобільної станції прогнозується наступна точка, після чого наступна вибірка передає не повне значення, а лише різниця між поточним значенням і попереднім, яке є значно меншим за обсягом.

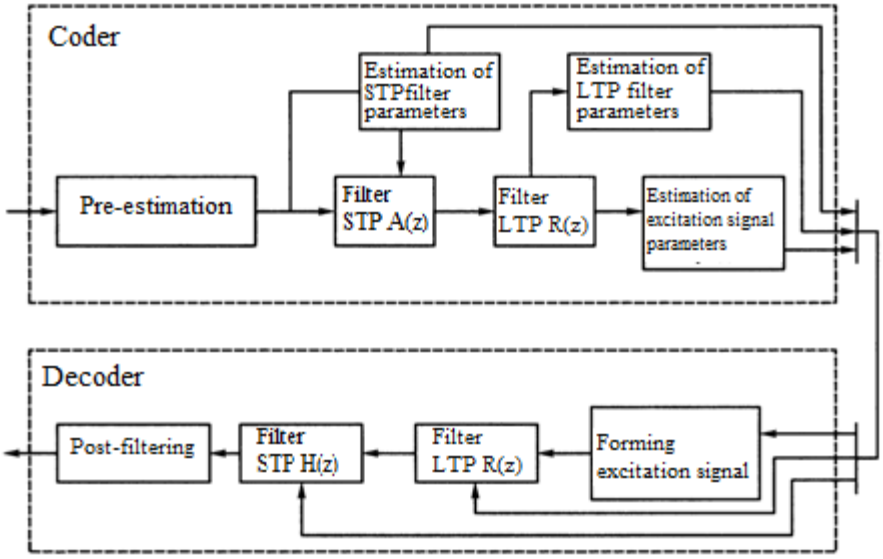

Рисунок 3.4 – Схема голосового кодека GSM

Модель вокодера складається з тонального генератора (який моделює голосові зв’язки) та фільтра, який змінює тон (який моделює форму ротової та носової порожнини). Короткочасний аналіз і фільтрація визначає

коефіцієнти фільтра та вимірювання похибок, довготривалий аналіз кількісно визначає гармоніки мови.

Оскільки математична модель генерації мовлення в повношвидкісному кодеку показує поступове зниження потужності для збільшення частоти, зразки подаються через фільтр попереднього акцентування, який покращує вищі частоти, що призводить до кращої ефективності передачі. Еквівалентний фільтр зменшення акценту на віддаленому кінці відновлює звук.

Короткостроковий аналіз (лінійне передбачення) виконує автокореляцію та рекурсію вхідного сигналу для визначення коефіцієнтів фільтра («відображення»). Коефіцієнти відбиття, які передаються по радіоканалу як вісім параметрів, що містять 36 біт інформації, перетворюються на коефіцієнти логарифмічної площі (LAR), оскільки вони пропонують більш вигідні характеристики компандування. Коефіцієнти відбиття потім використовуються для застосування короткочасної фільтрації до вхідного сигналу, що призводить до 160 вибірок залишкового сигналу.

Залишковий сигнал від короткочасної фільтрації сегментується на чотири підкадри по 40 вибірок кожен. Фільтр довгострокового прогнозування (LTP) моделює точні гармоніки мовлення, використовуючи комбінацію поточних і попередніх підкадрів. Параметри підсилення та затримки (затримки) для фільтра LTP визначаються крос-кореляцією поточного субкадру з попередніми залишковими субкадрами.

Пік крос-кореляції визначає затримку сигналу, а посилення обчислюється шляхом нормалізації коефіцієнтів крос-кореляції. Параметри застосовуються до довгострокового фільтра, і робиться прогноз поточного короткострокового залишку.

Замість FR були винайдені стандарти Enhanced Full Rate (EFR) і Adaptive Multi Rate (AMR), які поєднували кращу якість звуку та менший бітрейт.

Ефективне кодування (компресія) мови. Технологія EFR була розроблена компанією Nokia у 1995 році і згодом стала стандартною технологією кодування/декодування для GSM.

Enhanced Full Rate (EFR) — стандартизований алгоритм цифрового кодування голосу в широкосмуговому зв'язку GSM, а також реалізований на його основі кодек. Він є продовженням розробки розробленого на його основі алгоритму Full Rate (FR). Забезпечує хорошу якість зв'язку, відсутність сторонніх шумів. Хоча кодек Enhanced Full Rate (рис. 3.5) значно покращує якість зв’язку, він має більшу обчислювальну складність, що призводить до збільшення енергоспоживання мобільного пристрою приблизно на 5% порівняно зі «старим» кодеком Full Rate.

Частота дискретизації 8000 відліків/с, швидкість кодованого потоку 12,2 кбіт/с. Схема кодування називається кодер лінійного прогнозування з збудженням алгебраїчного коду (ACELP). Код представлений зразками з роздільною здатністю 13 рядків по 16 біт у кожному. Три молодших біта дорівнюють 0.

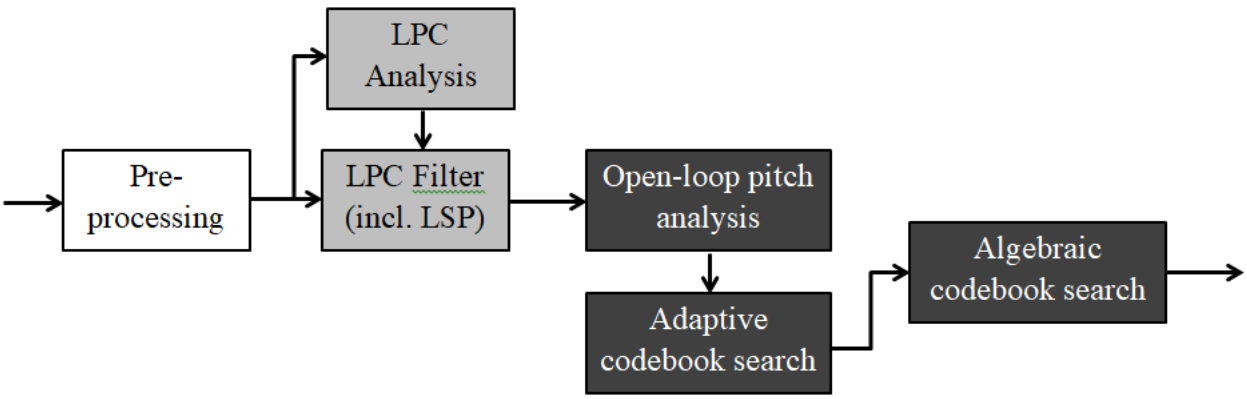

Рисунок 3.5 – Схема моделі вокодера EFR

Етап попередньої обробки для EFR складається з фільтра високих частот 80 Гц і деякого зменшення масштабу для зменшення складності впровадження. Короткостроковий аналіз, з іншого боку, відбувається двічі на

кадр і складається з автокореляції з двома різними асиметричними вікнами довжиною 30 мс, зосередженими навколо різних підкадрів. Результати перетворюються на коефіцієнти короткочасного фільтра, потім на спектральні пари ліній (для кращої ефективності передачі) і квантуються до 38 біт.

У кодеку EFR адаптивна кодова книга містить вектори збудження, які моделюють структуру довготривалої мови. Аналіз висоти тону з відкритим контуром виконується на половині кадру, і це дає дві оцінки затримки висоти тону для кожного кадру.

Результат відкритого циклу використовується для заповнення пошуку замкнутого циклу для швидкості та зниження вимог до обчислень. Затримка висоти тону застосовується до синтезатора, і результати порівнюються з несинтезованими вхідними сигналами (аналіз за синтезом), і визначається мінімальна перцептивно зважена помилка. Результати кодуються в 34 біти.

Залишковий сигнал, що залишається після квантування адаптивного пошуку в кодовій книзі, моделюється алгебраїчною (фіксованою) кодовою книгою знову з використанням підходу аналізу за синтезом. Результуюча затримка кодується як 35 біт на субкадр, а посилення як 5 біт на субкадр.

Останнім етапом для кодера є оновлення відповідної пам’яті, готової до наступного кадру.

Це все ще найкращий (вузькосмуговий) кодек, який використовується для мовного зв’язку через мобільні телефони в мережах 2G і 3G.

Для передачі голосу в 4G передбачена технологія VoLTE (Voice over Long-Term Evolution) - технологія передачі голосу через мережу LTE на основі IP Multimedia Subsystem (IMS).

VoLTE використовує режим кодування та декодування голосу Adaptive Multirate (AMR).

Адаптивний багатошвидкісний стандарт — це алгоритм кодування мовлення, який працює на восьми бітових швидкостях у діапазоні від 4,75 до 12,2 кбіт/с і спеціально розроблений для підвищення надійності з’єднання.

Цей стандарт був створений в процесі розробки алгоритмів стиснення звуку, що використовуються в стільникових мережах GSM, і об'єднує багато раніше існуючих форматів, включаючи GSM HR, FR, EFR, з типовою частотою дискретизації 8 кГц, 13 біт. Вибір конкретного формату відбувається автоматично, звідси і слово «адаптивний» у назві.

Режим кодування AMR включає адаптивний багатошвидкісний широкосмуговий (AMR-WB) і адаптивний багатошвидкісний вузькосмуговий (AMR-NB). Нижче показано вісім швидкостей кодування голосу AMR-NB і дев’ять швидкостей кодування голосу AMR-WB:

AMR-NB: 12,2 Кбіт /с, 10,2 Кбіт /с, 7,95 Кбіт /с, 7,4 Кбіт /с, 6,7 Кбіт /с, 5,9 Кбіт /с, 5,15 Кбіт /с і 4,75 Кбіт /с

AMR-WB: 23,85 Кбіт /с, 23,05 Кбіт /с, 19,85 Кбіт /с, 18,25 Кбіт /с, 15,85 Кбіт /с, 14,25 Кбіт /с, 12,65 Кбіт /с, 8,85 Кбіт /с і 6,6 Кбіт /с

Принцип кодека AMR полягає у використанні дуже подібних обчислень для набору кодеків для створення виходів із різною швидкістю. У GSM якість отриманого сигналу радіоінтерфейсу контролюється, і швидкість кодування мови може бути змінена. У такий спосіб забезпечується більший захист для областей із слабким сигналом за рахунок зменшення швидкості кодування та збільшення надмірності, а в зонах із хорошою якістю сигналу якість мовлення покращується.

З точки зору реалізації, використовується кодер ACELP. Насправді кодек AMR 12,2 кбіт/с обчислювально такий же, як і кодек EFR. Для швидкості нижче 12,2 кбіт/с короткочасний аналіз виконується лише один раз на кадр. Для 5,15 кбіт/с і нижче затримка висоти розімкненого контуру оцінюється лише один раз на кадр. Результатом цього є те, що при нижчій

швидкості вихідного біта, є менша кількість параметрів для передачі, і менше бітів використовується для їх представлення.

Специфікація повітряної передачі для GSM дозволяє розділити голосовий канал на два підканали, які можуть підтримувати окремі дзвінки. Голосовий кодер, який використовує половину пропускної здатності каналу, дозволить мережевим операторам подвоїти пропускну здатність стільника з дуже невеликими інвестиціями.

Кодек половинної швидкості — це кодек лінійного передбачення векторної суми збудження (VSELP), який працює на основі підходу аналізу шляхом синтезу, подібного до кодеків EFR і AMR. Результуючий вихід становить 5,7 кбіт/с, що включає 100 біт/с бітів індикатора режиму, які вказують, чи кадри містять голос чи ні. Індикатор режиму дозволяє кодеку працювати дещо інакше, щоб отримати найкращу якість.

Кодування мовлення з половинною швидкістю вперше було введено в середині 1990-х років, але суспільне сприйняття якості мовлення було настільки поганим, що воно зазвичай не використовується сьогодні. Однак завдяки вихідній швидкості передачі даних AMR чудово підходить для передачі по каналу половинної швидкості. Обмежуючи вихід до 6 найнижчих швидкостей кодування (4,75–7,95 кбіт/с), користувач усе ще може відчути переваги якості адаптивного кодування мови, а оператор мережі отримує переваги від збільшення пропускної здатності. Вважається, що з впровадженням AMR використання повітряного каналу з половинною швидкістю почне ставати набагато більш поширеним.

Під час розробки стандарту 5G Юкі Йошіда, старший науковий співробітник Національного інституту інформаційно-комунікаційних технологій, зазначив, що: «Мережі 5G дозволять нам передавати більше даних швидше, але змусять нас переглянути спосіб розробки та розгортання. наші транспортні мережі».

У стільниковому зв’язку цифрові дані перетворюються на аналогові радіочастотні сигнали, які передаються від базової станції до обладнання користувача і навпаки. У минулому весь процес модуляції/демодуляції виконувався на кожній стільниковій станції. Це вплинуло на вартість клітинки. У мережах 5G частина цього процесу переміщується з стільникового вузла в хмару. У результаті дані вже перетворюються на радіочастотні сигнали, коли вони досягають стільникового вузла. Поділ функціональних можливостей базової станції суттєво спрощує стільниковий вузол, але вимагає ширшої смуги пропускання в оптоволоконному з’єднанні між стільниковим вузлом і хмарою, а саме переднього переходу.

Ефективне передавання в 5G вимагає високої пропускної здатності, низької затримки, просторово-часового стиснення для радіочастотних сигналів 5G. Іншими словами, кількість даних, що транспортуються від кількох антен на стільниковому вузлі до хмари, має бути зменшено, і це скорочення має бути виконано швидко до зміни середовища бездротового каналу.

Одним із способів зробити це є стиснення об’єднаних радіоканалів. Скажімо, у вас є 100 сигналів антени, але лише 10 або 20 користувачів зараз на цих каналах. Ми можемо зайти, витягти лише дані з каналів, які використовуються, і стиснути їх. Це заощадить багато накладних витрат, не стискаючи потоки даних антен, які не передають активно.

Ці дані витягуються наосліп та адаптивно за допомогою алгоритмів, які називаються підпросторовим відстеженням. Потім використовуються методи стиснення аудіо з низькою затримкою – подібні до тих, що використовуються для обробки компакт-дисків і MP3, але на набагато вищих частотах – для зменшення розміру цих сигналів. Використовуючи ці інструменти, можна маршрутизувати сигнали LTE з 256 антенами, для яких потрібен оптичний інтерфейс 260 ГБод, через оптичний інтерфейс 10 ГБод.

3.3 Стиснення даних із втратами. Стандарти стиснення відеоданих

Одним із поширених, відкритих і безоплатних відеокодеків був VP9 – формат, розроблений Google. В основному він використовувався на платформі Google YouTube.

Через розрізнений ландшафт форматів відео та неузгоджену підтримку браузерів і пристроїв багато великих гравців у відеопросторі намагалися розробити стандарти, щоб спростити простір форматів і покращити якість відео. ISO/IEC Moving Picture Experts Group (MPEG) і ITU-T Video Coding Experts Group (VCEG) розробили HEVC, представили стандарт VP9. Кодек VP9 було розроблено з наміром зменшити бітрейт на 50% порівняно з VP8, забезпечуючи таку саму якість відео. Ще одна мета полягала в тому, щоб підвищити ефективність стиснення HEVC, залишаючись при цьому відкритим кодом і безоплатно.

VP9 налаштовано для відео з роздільною здатністю понад 1080p (наприклад, UHD), а також забезпечує стиснення без втрат.

У той час як відео 4K покращує якість зображення, зменшуючи окремі пікселі, кодек VP9 і HEVC збільшують їх, щоб зменшити бітрейт і розмір файлу. Хоча це може здатися суперечливим, механізм кодування бере більші пікселі та перетворює їх на вихід із вищою роздільною здатністю. Вихідне відео, що складається з відеокадрів, кодується або стискається для створення стисненого бітового потоку відео. Кожен окремий кадр спочатку розбивається на блоки пікселів. Потім блоки аналізуються на предмет просторової надлишковості та аналізуються часові зв’язки між кадрами, щоб скористатися перевагами областей, які не змінюються. Вони кодуються за допомогою векторів руху, які передбачають якості даного блоку в наступному кадрі. Залишкова інформація кодується за допомогою ефективного двійкового стиснення.

Як правило, для того, щоб клієнт міг переглядати відео в Інтернеті, мають бути виконані такі три умови:

– канал зв'язку між клієнтським пристроєм і постачальником послуг повинен мати достатню пропускну здатність для підтримки швидкості потоку стисненого відео;

– клієнтський пристрій повинен мати можливість декодувати отриманий стислий відеопотік бітів;

– клієнтський пристрій повинен мати можливість відображати декодоване відео.

Останнім форматом стиснення відео, який походить від VP9, був AV1.

Зараз AV1 широко використовується на таких платформах, як Netflix і Youtube, і в той час як Netflix закодував своє відео 1080p до 6,7 Мбіт/с, тоді як YouTube мав швидкість 5,1 Мбіт/с, тому жодна з компаній не просувалась за межі якості. Хоча Facebook стверджує, що AV1 економить 51% порівняно з x264 і 32,5% порівняно з VP9, компанія публічно не заявляла, що постачає відео, закодоване AV1. На даний момент, через рік після заморожування бітового потоку, ми не знаємо, чи досягає якась компанія економію бітрейту, про яку повідомляє Facebook, у будь-якому масштабі.

Але потім Intel і Netflix разом анонсували кодек SVT-AV1, який підтримує 10-бітне кодування 4K/60p у реальному часі під час роботи на потужних процесорах.

Якщо говорити про передісторію, Scalable Video Technology — це «технологія кодування відео на основі програмного забезпечення, яка дозволяє кодерам досягати на процесорах Intel Xeon Scalable найкращого компромісу між продуктивністю, затримкою та якістю зображення». Це технологія незалежності від кодеків, яку можна використовувати з різними кодеками, як-от H.264, HEVC і AV1, і різними реалізаціями кодеків (наприклад, x265 порівняно з кодувальником Main Concept HEVC).

Архітектура SVT дозволяє розділити ядро кодера на незалежно діючі потоки, як показано на рис. 3.6, кожен потік обробляє окремий сегмент вхідного зображення, які виконуються паралельно на різних ядрах процесора, без будь-яких втрат у точності. Ця архітектура SVT є стандартно-агностичною, тобто її можна застосовувати для розробки кодерів, сумісних з різними стандартами. SVT дозволяє будь-якому кодеру, сумісному зі стандартами, належним чином масштабувати свою продуктивність у відповідь на обмеження обчислень і пам’яті, зберігаючи при цьому плавне погіршення якості відео зі збільшенням продуктивності.

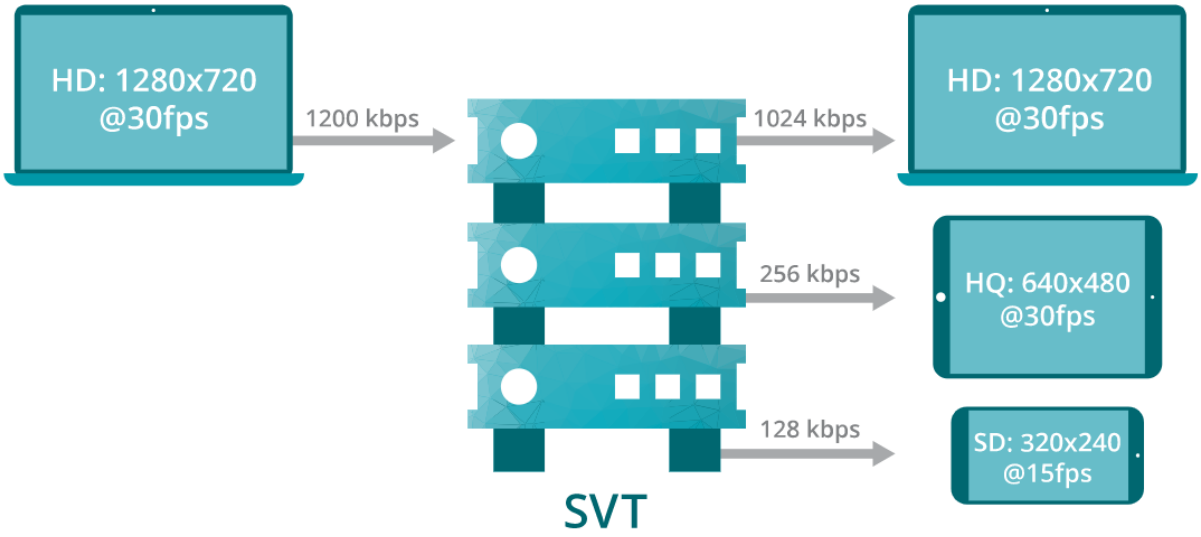

Рисунок 3.6 – Загальна схема роботи SVT

Без використання SVT усі учасники групової відеоконференції отримували б відео якості, яке б задовольнило термінал із найслабшими характеристиками. Тепер користувачі багато точкового відеоконференцзв'язку будуть бачити картинку в тій якості, в якій це дозволяє їх обладнання і канали зв'язку.

Адаптивне кодування з втратами - це метод стиснення даних, який використовує різні методи стиснення для різних частин даних. Це дозволяє досягти більшого ступеня стиснення, ніж статичне кодування, яке використовує один і той же метод стиснення для всіх частин даних.

В адаптивному кодуванні з втратами кодувальник аналізує дані і визначає, який метод стиснення буде найбільш ефективним для кожної частини даних. Наприклад, для областей даних, які містять багато повторюваних елементів, можна використовувати більш ефективний метод стиснення, наприклад, кодування з довжиною коду. Для областей даних, які містять багато змінних елементів, можна використовувати менш ефективний метод стиснення, наприклад, кодування з фіксованою довжиною коду.