УДК 004.627

О.В. Тарасов, Є.В. Онопко

Харківський національний економічний університет, Харків

ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ СТИСНЕННЯ ІНФОРМАЦІЇ

У статті розглянуті найбільш поширені алгоритми стиснення інформації, що використовуються для обробки текстових, графічних та відеоданих. Проведено їх порівняльний аналіз та виділені сильні та слабкі сторони. Стисло розповідається про ключові моменти функціонування кожного алгоритму, наведені приклади програмного забезпечення та технології, що базуються на них. Робиться висновок про доцільність використання модифікованого методу Хаффмана при стисненні текстових значень у базах даних.

Ключові слова: *стиснення даних, LZ77, Хаффман, Шеннон–Фано, PPM , RLE, архів, кодування.*

Вступ

Характерною особливістю більшості типів даних є їх надлишковість. Ступінь надлишковості даних залежить від типу даних. Наприклад, для відеоданих ступінь надлишковості в декілька разів більша ніж для графічних даних, а ступінь надлишковості графічних даних, у свою чергу, більша за ступінь надлишковості текстових даних. Іншим фактором, що впливає на ступінь надлишковості є прийнята система кодування.

Прикладом систем кодування можуть бути звичайні мови спілкування, які є ні чим іншим, як системами кодування понять та ідей для висловлення думок. Так, встановлено, що кодування текстових даних за допомогою засобів української мови дає в середньому надлишковість на 20 – 25% більшу ніж кодування аналогічних даних засобами англійської мови [1].

Для людини надлишковість даних часто пов'язана з якістю інформації, оскільки надлишковість, як правило, покращує зрозумілість та сприйняття інформації. Однак, коли мова йде про зберігання та передачу інформації засобами комп'ютерної техніки, то надлишковість відіграє негативну роль, оскільки приводить до зростання вартості зберігання та передачі інформації. Особливо актуальною ця проблема стає у випадку необхідності обробки величезних обсягів інформації при незначних об'ємах носіїв даних і її вирішення базується на постійному позбавленні від надлишковості або стисненні даних. Коли методи стиснення даних застосовуються до готових файлів, то часто замість терміну "стиснення даних" вживають термін "архівування даних", стиснений варіант даних називають архівом, а програмні засоби, що реалізують методи стиснення називаються архіваторами.

Існує багато практичних алгоритмів стиснення даних, але всі вони базуються на трьох теоретичних способах зменшення надлишковості даних. Перший спосіб полягає в зміні вмісту даних, другий - у зміні структури даних, а третій – в одночасній зміні як структури, так і вмісту даних.

Якщо при стисненні даних відбувається зміна їх вмісту, то метод стиснення є незворотнім, тобто при відновленні (розархівуванні) даних з архіву не відбувається повне відновлення інформації. Такі методи часто називаються методами стиснення з регульованими втратами інформації. Зрозуміло, що ці методи можна застосовувати тільки для таких типів даних, для яких втрата частини вмісту не приводить до суттєвого спотворення інформації. До таких типів даних можна віднести відео- та аудіодані, а також графічні дані. Методи стиснення з регульованими втратами інформації забезпечують значно більший ступінь стиснення, але їх не можна застосовувати до текстових даних. Прикладами форматів стиснення з втратами інформації можуть бути: JPEG (Joint Photographic Experts Group) для графічних даних, MPG - для відеоданих та MP3 - для аудіо даних [2].

Якщо при стисненні даних відбувається тільки зміна структури даних, то метод стиснення є зворотнім. У цьому випадку з архіву можна відновити інформацію повністю.

Зворотні методи стиснення можна застосовувати до будь-яких типів даних, але вони дають менший ступінь стиснення у порівнянні з незворотними методами стиснення.

Приклади форматів стиснення без втрати інформації: GIF (Graphics Interchange Format), TIFF (Tagged Image File Format) – для графічних даних; ZIP, ARJ, RAR, CAB, LH – для довільних типів даних. Існує багато різних практичних методів стиснення без втрати інформації, які, як правило, мають різну ефективність для різних типів даних та різних обсягів [3].

Мета статті – проаналізувати існуючі алгоритми стиснення даних та зробити висновок про доцільність їх використання при стисненні текстової інформації у базах даних.

Основна частина

Алгоритм LZ77. Цей словниковий алгоритм стиснення є найстарішим серед методів LZ. Опис було опубліковано в 1977 р., але сам алгоритм розроблений не пізніше 1975.

Алгоритм LZ77 є родоначальником цілого сімейства словникових схем - так званих алгоритмів з ковзаючим словником, або ковзаючим вікном. Дійсно, в LZ77 як словника використовується блок уже закодованою послідовності. Як правило, у міру виконання обробки положення цього блоку щодо початку послідовності постійно змінюється, словник "ковзає" по вхідному потоку даних [4].

В 1977 два дослідники з Ізраїлю запропонували абсолютно інший підхід до цієї проблеми. Абрам Лемпель і Якоб Зів висунули ідею формування "словника" загальних послідовностей даних.

Завдяки цьому принципу алгоритми LZ іноді називаються методами стиснення з використанням ковзного вікна. Ковзне вікно можна представити у вигляді буфера (або більш складної динамічної структури даних), який організований таким чином, щоб запам'ятовувати вже відому раніше інформацію і надавати до неї доступ. Таким чином, сам процес стиснення (або кодування) згідно LZ77 нагадує написання програми, команди якої дозволяють звертатися до елементів «ковзаючого вікна», і замість значень послідовності, що стискається, вставляти посилання на ці значення в «ковзному вікні». Розмір ковзаючого вікна може динамічно змінюватися і складати 2, 4 або 32 кілобайт. Слід також зазначити, що розмір вікна кодувальника може бути менше або дорівнювати розмірам вікна декодувальника, але не навпаки [8].

Метод кодування згідно з принципом ковзаючого вікна враховує інформацію, яка вже раніше зустрічалася, тобто інформацію, яка вже відома для кодувальника і декодувальника (друге і наступні входження деякого рядка символів в повідомленні замінюються посиланнями на її перше входження).

До недоліків даного методу можна віднести:

– довжина підрядка, яку можна закодувати, обмежена розміром буфера;

– мала ефективність при кодуванні незначного обсягу даних.

Алгоритм Хаффмана. Алгоритм Хаффмана – є адаптивним алгоритмом оптимального префіксного кодування алфавіту з мінімальною надмірністю. Був розроблений в 1952 році аспірантом Массачусетського технологічного інституту Девідом Хаффманом при написанні їм курсової роботи. В даний час використовується в багатьох програмах стиснення даних.

Цей метод кодування складається з двох основних етапів:

1). Побудова оптимального кодового дерева.

2). Побудова відображення код-символ на основі побудованого дерева [4].

Ідея алгоритму полягає в наступному: знаючи ймовірності символів у повідомленні, можна описати процедуру побудови кодів змінної довжини, що складаються з цілої кількості бітів. Символам з більшою ймовірністю ставляться у відповідність коротші коди. Коди Хаффмана мають властивість префіксності (тобто жодне кодове слово не є префіксом іншого), що дозволяє однозначно їх декодувати.

Класичний алгоритм Хаффмана на вході отримує таблицю частот символів у повідомленні. Далі на підставі цієї таблиці будується дерево кодування Хаффмана.

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

2). Вибираються два вільних вузла дерева з найменшими вагами.

3). Створюється їх батько з вагою, рівною їх сумарній вазі.

4). Батько додається в список вільних вузлів, а два його нащадка видаляються з цього списку.

5). Одній дузі, котра виходить з батьківського вузла, ставиться у відповідність біт 1, інший - біт 0.

6). Кроки, починаючи з другого, повторюються до тих пір, поки в списку вільних вузлів не залишиться тільки один вільний вузол. Він і буде вважатися коренем дерева [3].

Класичний алгоритм Хаффмана має ряд істотних недоліків. По-перше, для відновлення вмісту стиснутого повідомлення декодер повинен знати таблицю частот, якою користувався кодер. Отже, довжина стиснутого повідомлення збільшується на довжину таблиці частот, яка повинна надсилатися попереду даних, що може перекреслити всі зусилля зі стиснення повідомлення. Крім того, необхідність наявності повної частотної статистики перед початком власне кодування, потребує двох проходів за повідомленням: одного для побудови моделі повідомлення (таблиці частот і Н-дерева), іншого для кодування [6].

Алгоритм Шеннона–Фано. Алгоритм Шеннона–Фано – один з перших алгоритмів стиснення, який сформулювали американські вчені Шеннон і Фано. Даний метод стиснення має велику схожість з алгоритмом Хаффмана та використовує коди змінної довжини: символи, що часто зустрічаються, замінюються кодом меншої довжини, рідко зустрічається – кодом більшої довжини. Коди Шеннона–Фано, як і Хаффмана, префіксні. Ця властивість дозволяє однозначно декодувати послідовність кодових слів.

Подібно алгоритму Хаффмана, алгоритм Шеннона–Фано використовує надмірність повідомлення, яка базується на неоднорідному розподілі частот символів його алфавіту, тобто замінює коди частіших символів короткими двійковими послідовностями, а коди більш рідкісних символів - довшими двійковими послідовностями [4].

Код Шеннона–Фано будується за допомогою дерева, як і в алгоритмі Хаффмана. Побудова цього дерева починається від кореня. Вся множина кодованих елементів відповідає кореню дерева (вершині першого рівня). Вона розбивається на дві підмножини з приблизно однаковими сумарними ймовірностями. Ці підмножини відповідають двом вершинам другого рівня, які з'єднуються з коренем. Далі кожна з цих підмножин розбивається на дві підмножини з приблизно однаковими сумарними ймовірностями. Їм відповідають вершини третього рівня. Якщо підмножина містить єдиний елемент, то йому відповідає кінцева вершина кодового дерева. Така підмножина розбиттю не підлягає. Подібним чином поступаємо до тих пір, поки не отримаємо всі кінцеві вершини. Гілки кодового дерева розмічаються символами 1 і 0, як і у випадку коду Хаффмана.

На відміну від алгоритму Хаффмана, алгоритм Шеннона–Фано не залишається завжди оптимальним для вторинних алфавітів з більш ніж двома символами [5].

Кодування Шеннона–Фано є досить старим методом стиснення, і на сьогоднішній день не представляє особливого практичного інтересу. У більшості випадків, довжина послідовності, отримана за цим методом, дорівнює довжині послідовності з використанням кодування Хаффмана. Але на деяких послідовностях можуть сформуватися неоптимальні коди Шеннона–Фано, тому більш ефективним вважається стиснення методом Хаффмана.

Алгоритм стиснення PPM. PPM (Prediction by Partial Matching – пророкування по частковому співпадінню) – адаптивний статистичний алгоритм стиснення даних без втрат, заснований на контекстному моделюванні і пророкуванні. Модель PPM використовує контекст – безліч символів в стислому потоці, що передують даному, щоб передбачати значення символу на основі статистичних даних. Сама модель PPM лише пророкує значення символу, безпосереднє стиснення здійснюється алгоритмом Хаффмана і арифметичним кодуванням [6].

Довжина контексту, який використовується при прогнозі зазвичай сильно обмежена. Ця довжина позначається n і визначає порядок моделі PPM, що позначається як PPM(n). Необмежені моделі так само існують і позначаються просто PPM*. Якщо пророцтво символу по контексту з n символів не може бути зроблено, то відбувається спроба передбачити його за допомогою n – 1 символів. Рекурсивний перехід до моделей з меншим порядком відбувається поки пророкування не відбудеться в одній з моделей, або коли контекст стане нульової довжини (n = 0).

Велике значення для алгоритму PPM має проблема обробки нових символів, які ще не зустрічалися у вхідному потоці. Це проблема носить назву «нульової частоти». Деякі варіанти реалізацій PPM вважають лічильник нового символу рівним фіксованій величині, наприклад, одиниці. Опубліковані дослідження алгоритмів сімейства PPM з'явилися в середині 1980-х років. Програмні реалізації не були популярні до 1990-х років, тому що моделі PPM вимагають значний об’єм оперативної пам'яті. Сучасні реалізації PPM є кращими серед алгоритмів стиснення без втрат для текстів [2].

Варіанти алгоритму PPM на даний момент широко використовуються, головним чином для компресії надлишкової інформації і текстових даних. Відомі реалізації алгоритму PPM: RAR, 7-Zip, WinZip.

Алгоритм RLE. Алгоритм RLE (Run-length encoding – кодування повторів або довжин серій) – простий алгоритм стиснення даних, який оперує серіями даних, тобто послідовностями, в яких один і той же символ зустрічається кілька разів підряд. При кодуванні рядок однакових символів, що складають серію, замінюється рядком, який містить сам символ, що повторюється, і кількість його повторів. Алгоритм RLE буде давати кращий ефект стиснення при більшій довжині послідовності даних, що повторюється.

Програмні реалізації алгоритмів RLE відрізняються простотою, високою швидкістю роботи, але в середньому забезпечують недостатнє стиснення. Найкращими об'єктами для даного алгоритму є прості графічні файли, в яких великі одноколірні ділянки зображення кодуються довгими послідовностями однакових байтів. Однак це кодування погано підходить для зображень з плавним переходом тонів, таких як фотографії. Цей метод також може давати помітний виграш на деяких типах файлів баз даних, що мають таблиці з фіксованою довжиною полів. Для текстових даних методи RLE, як правило, неефективні [7].

Методом кодування довжин серій можуть бути стиснуті довільні файли з двійковими даними, оскільки специфікації на формати файлів часто включають в себе повторювані байти в області вирівнювання даних. Проте, сучасні системи стиснення частіше використовують алгоритми на основі LZ77, які є узагальненням методу кодування довгих серій.

Звукові дані, які мають довгі послідовні серії

байт, такі як низькоякісні звукові семпли, також можуть бути стиснуті з допомогою RLE.

До позитивних сторін алгоритму, можна віднести те, що він не вимагає додаткової пам'яті при роботі, і швидко виконується.

Алгоритм застосовується в графічних форматах PCX, TIFF [3].

Висновки

Стиснення скорочує об'єм пам'яті, необхідного для зберігання файлів на зовнішніх пристроях, і кількість часу, необхідного для передачі інформації по каналу встановленої смуги пропускання. Це є своєрідна форма кодування. Іншими цілями кодування є пошук і виправлення помилок, а також шифрування.

Процес пошуку і виправлення помилок протилежний стисненню – він збільшує надмірність даних, коли їх не потрібно представляти в зручній для сприйняття людиною формі. Видаляючи з тексту надмірність, стиснення сприяє шифруванню, що ускладнює пошук шифру статистичним методом.

Існує багато різних практичних методів стиснення без втрати інформації, які, як правило, мають різну ефективність для різних типів даних та різних обсягів. Однак, в основі цих методів лежать два алгоритми:

- алгоритм RLE (Run Length Encoding);
- алгоритм Хаффмана.

Найбільш ефективними для стискання текстової інформації є метод PPM, графічних та відеоданих – LZW та RLE. Алгоритм Хаффмана, с свою чергу, є досить універсальним і його можна застосовувати для стиснення будь-яких типів даних. Аналіз методів стиснення показує, що алгоритми, які з успіхом використовуються для обробки файлів великих розмірів, особливо текстових, не є ефективними для стиснення текстів невеликої довжини, що часто зустрічаються в базах даних, як окремі поля.

Тому для вирішення цієї проблеми рекомендується використовувати або "чистий" метод Хаффмана [9], або необхідно розробити новий модифікований метод, що базується на алгоритмі Хаффмана, для його використання в базах даних.

Список літератури

1. Артюшенко В.М. Цифровое сжатие видеоинформации и звука / В.М. Артюшенко, О.И. Шелухин. – М.: Дашков и Ко, 2004. – 426 с.

2. Сэломон Д. Сжатие данных, изображений и звука. Data Compression Methods / Д. Сэломон – М.: Техносфера, 2004. – 368 с.

3. Ватолин Д. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с.

4. Седжвик Р. Фундаментальные алгоритмы на C++. Части 1-4. Анализ. Структуры данных. Сортировка. Поиск / Р. Седжвик. – М.: ДиаСофт, 2002. – 688 с.

5. Кохманюк Д. Сжатие информации: как это делается / Д. Кохманюк. – К.: Index PRO, 1993. – № 1-2.

6. Кричевский Р.Е. Сжатие и поиск информации / Р.Е. Кричевский. – М.: Радио и связь, 1989. – 424 с.

7. Рябко Б.Я. Сжатие информации с помощью стопки книг. Проблемы передачи информации / Б.Я. Рябко. – М., 1988. – Т. 16, № 4. – С. 16-21.

8. Ziv J. Compression of individual sequences via variable-rate coding / J. Ziv, A. Lempel // IEEE Transactions, IT-24. – No. 5 (Sept.1978). – P. 530-536.

9. Мартин Дж. Организация баз данных в вычислительных системах / Дж. Мартин. – М.: Мир, 1980. – 662 с.

Надійшла до редколегії 4.10.2011

Рецензент: канд. екон. наук, проф. І.О. Золотарьова, Харківський національний економічний університет, Харків.

ОБЗОР И СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ СЖАТИЯ ИНФОРМАЦИИ

О.В. Тарасов, Е.В. Онопко

В статье рассмотрены наиболее распространенные алгоритмы сжатия информации, которые используются для обработки текстовых, графических и видеоданных. Проведен сравнительный анализ и выделены их сильные и слабые стороны. Кратко рассказывается о ключевых моментах функционирования каждого алгоритма, приведенные примеры программного обеспечения и технологий, которые базируются на них. Делается вывод о целесообразности использования модифицированного метода Хаффмана при сжатии текстовых значений в базах данных.

Ключевые слова: *сжатие данных, LZ77, Хаффман, Шеннон–Фано, PPM , RLE, архив, кодирование.*

REVIEW AND COMPARATIVE ANALYSIS OF DATA COMPRESSION METHODS

A.V. Tarasov, E.V. Onopko

This article discusses the most common data compression algorithms , compares them and stresses the main advantages and weaknesses. Briefly describes the main principles of each algorithm. Examples of software and technologies which are based on them are given. The conclusion about the expedience of using the modified method of Huffman for compression of text values in the database are drawn.

Keywords: *data compression, LZ77, Huffman, Shannon–Fano, PPM, RLE, archives, coding.*