УДК 621.391:519.726

Б.Ю. Жураковський

АНАЛІЗ КІЛЬКОСТІ АЛГОРИТМІВ СТИСНЕННЯ У КАСКАДІ ПРИ ВИКОРИСТАННІ КАСКАДНИХ МЕТОДІВ

В роботі розглядаються методи стиснення інформації в каналах передачі даних. Акцент робиться на каскадний метод стиснення. При реалізації каскадного методу стиснення виникає проблема використання кількості методів стиснення в каскаді. Розглядаються різні варіанти каскадів при побудові каскадних методів стиснення. Проаналізовано залежність ефективності стиснення інформації від кількості методів у каскаді для текстової та вимірювальної інформації. Наведено рекомендації щодо вибору кількості методів стиснення в каскаді.

Ключові слова: методи стиснення, алгоритми стиснення, середня довжина кодової комбінаціїї, коєфіцієнт стиснення.

Вступ

Проблема скорочення обсягів переданих даних виникає в системах управління інфокомунікаційними мережами при передачі великих масивів вимірювальної інформації та інформації контролю за станом об’єктів. При реєстрації процесів, що протікають із високою швидкістю, або при необхідності передачі значних обсягів даних у системах із великою кількістю абонентів (промислових мережах) апаратура передачі даних іноді не в змозі забезпечити інформаційний обмін у необхідному темпі, при цьому інформація, що надходить від датчиків, як правило має високу надмірність[1].

Стиснення даних – процедура зменшення надмірності повідомлення з метою зменшення його обсягу. У процесі стиснення усувається природна надмірність, властива практично будь-якому джерелу інформації [2]. При розгортанні, перед видачею інформації споживачу, відбувається відновлення первинного повідомлення зі стиснутого. Усунення надмірності досягається за рахунок застосування ефективного кодування[3].

При розв'язанні питання про використання того або іншого алгоритму стиснення даних у конкретній системі передачі даних необхідно оцінити сильні і слабкі сторони алгоритму з урахуванням умов, у яких він буде працювати[4]. Крім коефіцієнта стиснення важливе значення мають такі параметри, як швидкодія, необхідний обсяг пам'яті для роботи, здатність алгоритму адаптуватися до статистичних властивостей даних, що надходять, а при передачі даних у режимі реального часу – час затримки переданих даних [5]. Тому алгоритм, що показує кращі результати в одних умовах, може працювати значно гірше або взагалі не підходити для іншої системи. В данійроботі проведено аналіз існуючих методів стиснення з метою визначення найбільш перспективних для застосування при передачі даних.

Каскадний метод стиснення інформації

Поєднує у собі стиснення інформації двома або більшою кількістю методів. Його відмінність від інших методів стиснення в тому, що дані спочатку стискаються одним методом, а потім отриманий стиснутий масив обробляється іншим методом. Розгортання інформації робиться в зворотному порядку [6].

При реалізації каскадного методу стиснення може виникнути проблема неоднозначності трактування стиснутого масиву даних. Це може відбутися в тому випадку, якщо алгоритм стиснення першого каскаду в результаті роботи згенерує масив даних, що містить керуючі символи алгоритму другого каскаду.

При розгортанні (декомпресії) такого масиву процедура другого каскаду інтерпретує ці символи як ознаки стиснення, і інформація буде перекручена.

Для вирішення цієї проблеми повинен передбачатися чіткий поділ способу вказівки ознаки стиснення для кожного каскаду, що не дозволить використання однакових символів.

Розглянемо декілька вдомих алгоритмів стиснення.

Алгоритм стиснення Віттера

На початку 70 - х років були розроблені однопрохідні методи стиснення інформації, основані на класичні процедурі кодування Хаффмена. Всі ці методи незначно різнилися один від одного і сутність Їх полягає в тому, що передавач будує дерево Хаффмана в темпі отримання даних від джерела тобто "на літу". В процесі кодування проходить навчання "навчання" кодеру на статистичних характеристиках джерела повідомлень, в ході якого вираховуються оцінки початкових ймовірностей повідомлення і виконується відповідна модифікація кодового дерева Хаффмена. У зв'язку з неперервним змінюванням кодового дерева цей процес отримав назву динамічного кодування Хаффмена. Очевидно, що для вірного відновлення стиснутих даних, декодер також повинен безперервно "навчатись" поряд з кодером, виконуючи синхронне змінювання кодової таблиці на прийомній стороні. Для забезпечення синхронності процесів кодування і декодування кодер видає символ в нестиснутому вигляді, якщо він вперше з'явився на виході джерела, і відмічає його на кодовому дереві. При повторній появі символу на виході кодера він передається нерівномірною кодовою комбінацією, що визначається позицією символу на поточному кодовому дереві. Кодер коректує дерево Хаффмена збільшенням частоти передачі символів, котрі вже введені в дерево, або нарощую дерево, добавляючи в нього нові вузли.

Однією з головних умов, котра повинна виконуватись при модифікації кодового дерева, є збереження властивостей хаффменовського дерева. При статистичному кодуванні символи розташовуються в списку в не зростаючому порядку ваги (ймовірностей). Потім виконується об'єднання двох вузлів найменшої ваги W_i, W_j і заміна їх внутрішнім вузлом з вагою, рівною сумі початкової ваги W_i+W_j. Знову створений вузол розташовується в списку таким чином, щоб не руйнувався порядок розташування вузлів по вазі. Цей процес повторюється до тих пір, поки в списку не зостанеться один, так званий кореневий вузол.

Вперше алгоритм синтезу динамічного коду Хаффмена був запропонований Н. Феллером в 1973 році, а потім модифікований Р. Галлагером і Д. Книтом. В зв'язку з цим він отримав назву "Алгоритм FGK".

Подальше удосконалення алгоритму динамічного кодування даних нерівномірними кодами FGK було запропоновано Д. Віттером. Цей алгоритм отримав назву алгоритму *V*. При пошуку шляхів оптимізації процедури кодування даних кодом Хаффмена автор виходив з того, що в новому алгоритмі число обмінів вузлів в процесі модифікації кодового дерева повинно обмежуватись деяким малим числом (в найкращому випадку одиницею), а динамічне хаффменовське дерево повинно будуватись таким чином, щоб мінімізувати не тільки сумарну довжину загального шляху $\sum W_j l_j$, але і величини $\sum l_j$, $max\ \{l_j\}$. Мінімізація висоти дерева $h = max\ \{l\}$ дозволяє зарадити створенню довгих кодових комбінацій при кодуванні чергового символу в повідомленні. Віттеру в значній ступені вдалося вирішити поставлену задачу. Розроблений ним алгоритм має в порівнянні з алгоритмом FGKдві переваги [7].

Кількість обмінів вузлами, при яких черговий вузол переміщається вгору по кодовому дереву в процесі його модифікації, обмежується одиницею. В алгоритмі FGK верхня границя кількості обмінів складає $l_j/2$, де l_j- довжина кодового слова для $Z_j(k+1)$ - го символу до початку процедури модифікації.

Алгоритм V мінімізує довжину зовнішнього шляху дерева l_j і гарантує дерево мінімальної висоти $h = max\ \{l_j\}$ при умові мінімізації сумарної довжини зовнішнього шляху дерева $\sum W_j\ l_j$.

Сутність удосконалень алгоритму *V* полягає в введенні нової системи нумерації вузлів кодового дерева, яка отримала назву неявної нумерації (Implicit numbering). При неявній нумерації вузли хаффмановського дерева нумеруються в порядку збільшення за рівнями зліва на право, знизу вверх, тобто вузли більш низького рівня мають номери менші, чим

вузли наступного рівня. Важливою особливість неявної нумерації є дотримання необхідної умови побудови дерева, котра формулюється таким чином:

- для кожної ваги W всі зовнішні вузли (листя) дерева з вагою W повинні стояти попереду всіх внутрішніх вузлів ваги W.

Неважко бачити, що ця умова є одною з визначальних особливостей неявної нумерації порівняно з іншими алгоритмами.

Нумерація вузлів, що проводиться відповідно алгоритму FGK, не завжди відповідає неявній нумерації.

Однією з визначних особливостей алгоритму *V* є введення поняття блока еквівалентних вузлів. При цьому вузли x і y еквівалентні, якщо вони мають однакову вагу і обидва є внутрішніми або зовнішніми. Вузол блоку, маючий (при неявній нумерації) самий високий номер, має назву лідера блоку. Блоки упорядковуються по зростанню ваги, причому, блок листя вагою W повинен передувати блоку внутрішніх вузлів тієї ж ваги.

Особливістю алгоритму *V* також є спосіб модифікації дерева після отримання чергового символу. Головною операцією алгоритму по підтримці умови неявної нумерації є ковзання і прирощення (Slide And Inerement). Суть цієї операції полягає в тому, що вузол, об'явлений черговим обмінюється з лідером свого блоку і потім ковзає в напрямку кореню дерева по сусідньому блоку, котрий безпосередньо прилягає до блоку чергового вузла. Ковзання продовжується до тих пір, поки черговий вузол не пройде увесь блок і буде встановлено в голову цього блоку. Потім виконується прирощення ваги чергового вузла і новим черговим вузлом починається батько старого чергового вузла. Операція ковзання з прирощенням продовжується до досягнення кореня дерева. При цьому вибір батьківського вузла залежить від того, чи є черговий вузол листом, або внутрішнім вузлом. Якщо черговий вузол був листом, тоновим черговим вузлом назначається його батьківський вузол, з котрим був пов'язаний черговий вузол до початку ковзання.

Алгоритм стиснення методом LZW

Незалежно від мови і характеру текстових повідомлень у різноманітних його частинах можна виявити групи символів, що неодноразово зустрічаються тексті. Тому з метою скорочення надмірності повідомлення доцільно замість передачі груп символів, що зустрічаються повторно, робити посилання на аналогічні групи, передані в попередній частині повідомлення. Ця ідея лежить в основі методу компресії даних, запропонованого А. Лемпелем і Я. Зивом [7]. Метод Лемпеля-Зива відноситься до однопрохідних адаптивних методів стиснення послідовних даних, що не потребує апріорних знань статистичних характеристик джерела повідомлень. Алгоритм складається з правила для синтаксичного аналізу рядків повідомлення яке підлягає стисненню, що складає із символів кінцевого алфавіту, і схеми кодування. У процесі аналізу з рядків повідомлення виділяються підрядки перемінної довжини. При кодуванні цим підрядкам ставляться у відповідність кодові комбінації, що складаються з фіксованого числа елементів. Потім здійснюється пошук ідентичних підрядків у попередньому рядку. Після відшукання співпадаючих підрядків вибирається підрядок максимальної довжини і проводиться його заміна в поточному рядку повідомлення відповідною кодовою комбінацією[7].

Найбільше поширеним методом динамічного стиснення інформації є метод *LZW*. Цей метод компресії даних запропонований Т. Уелчем у 1984 р. Він одержав назву "метод *LZW*", тому що є подальшим розвитком методу *LZ88*. При кодуванні *LZW* -методом використовується таблиця рядків, що складає як з одиночних символів, так і з деяких буквосполучень. Причому кожному рядку відповідає своя двійкова комбінація (кодове слово) .

Процедура динамічного кодування *LZW* - методом полягає в наступному. Відбувається початкове заповнення *LZW* - таблиці в неї вносяться рядки, що відображають статистику

повідомлень на визначеної мові. При відсутності такої статистики таблиця містить тільки рядки, що складаються з одиночних символів. Потім, у міру надходження даних від джерела інформації, формуються рядки, що складаються з декількох символів. Тому що таблиця має обмежений розмір, то рядки, що зустрічаються в тексті рідко, виключаються, а на їхнє місце вносяться рядки, що мають велику частоту появи. Таким чином, у процесі накопичення статистики про повідомлення що стискається відбувається динамічна перебудова таблиці кодування й адаптація її до характеру переданих даних. Компресія даних починається з ініціалізації таблиці, при якій в неї вносяться рядки, що складаються з одиночних символів. Потім надходить перший вхідної символ, аналізований як префікс деякого рядка PREFIX. Після цього вводиться такий символ CHARACTER і утвориться розширений рядок шляхом об'єднання префікса й одиночного символу PREFIX + CHARACTER. Далі здійснюється зіставлення знову утвореного рядка з рядками, що існують у таблиці кодування. Якщо рядок PREFIX + CHARACTER є в таблиці, то вона стає новим префіксом, тобто PREFIX = PREFIX + CHARACTER і вводиться такий символ CHARACTER і процедура зіставлення рядка в таблиці повторюється знову. У противному випадку, якщо послідовності PREFIX + CHARACTER у таблиці рядків немає, то на вихід кодера виводиться кодова комбінація, що відповідає рядку PREFIX, у таблицю вноситься додатковий рядок PREFIX + CHARACTER, а символ CHARACTER стає новим префіксом. Якщо вхідна послідовність не вичерпана, те процедура утворення рядків і їх співставлення повторюється[8].

Алгоритм стиснення методом LZH

Метод стиснення *LZH* заснований на використанні двох методів стиснення даних: словниковий (*LZ*) і статистичний (*Haffman*) метод.

На початку виконання алгоритму стиснення відбувається ініціалізація словника. Словник логічно являє собою сукупність абстрактних структур даних, що містять набір деревоподібних структур, у котрих кожний корінь відповідає визначеному знаку алфавіту. При 8-розрядному форматі символу кількість таких дерев дорівнює 256. Деревоподібні структури являють собою набір відомих рядків, що починаються одним визначеним символом, а кожний вузол дерева надає один рядок із цього набору.

При виконанні компресії циклічно формується новий рядок шляхом додавання чергового одиночного символу до існуючого рядка, що знаходиться в таблиці статей словника, що виражається в додаванні нового вузла до кодового дерева [9].

Далі відбувається співставлення рядка, при якому послідовність символів із потоку даних співставлення зі статтею словника. Якщо рядок відповідає статті словника і не є створеним під час останнього виклику процедури зіставлення рядка, то вводиться наступний символ і добавляється до рядка. Потім цей крок повторюється. Якщо рядок не відповідає статті словника, або відповідає статті, створеної під час останнього виклику процедури зіставлення рядків, то останній символ убирається, а кодове слово відповідній статті словника видається на вихід кодера. Отриманий укорочений рядок представляє, таким чином, самий довгий зіставлений рядок, а останній (відкинутий) символ є не зіставленим символом, із якого починається таке зіставлення рядків що надходять від джерела даних [9].

Матричний метод стиснення

З матричних методів стиснення найбільш придатним для використання в системах управління є адаптивно-матричний [6].

Для роботи алгоритму необхідно задати розмір блока даних. Для зручності і наочності вхідний потік розбивається на рядки символами “переведення рядка - повернення каретки”. У реальних умовах довжина рядка матриці визначається довжиною запису даних, що передаються.

Інформація зчитується блоками з вхідного файлу, одночасно робиться її аналіз і пошук елементів, що повторюються, у сусідніх рядках. Якщо в двох сусідніх рядках знайдені однакові ланцюжки символів, алгоритм здійснює перегляд наступних рядків, визначаючи кількість рядків матриці. Розміри матриці і номери рядків для кожного рядка матриці записуються в допоміжний масив. Якщо розміри матриці задовольняють визначеній умові, то вона позначається як та, що підлягає стисненню.

При записі інформації у вихідний потік для кожної виділеної матриці передається її перший рядок, перед яким ідуть ознака стиснення і розміри матриці. Наступні рядки матриці опускаються.

Даний метод не забезпечує найбільш оптимального виділення матриць у вхідному потоці (наприклад, якщо дві матриці перекриваються, то виділена буде перша що зустрілася, навіть якщо вона менша за розміром). Однак цей метод має високу швидкість стиснення, оскільки виділення матриць відбувається одночасно з прийомом даних.

Покажемо на прикладах текстової та вимірювальної інформації, що оптимальна кількість методів в каскаді – два.

Розглянемо випадок, коли в каскаді застосовуються три методи, причому на перших двох етапах застосовуються методи, що не є найефективнішими для цього типу інформації, а на третьому етапі – один з найефективніших методів (рис. 1).

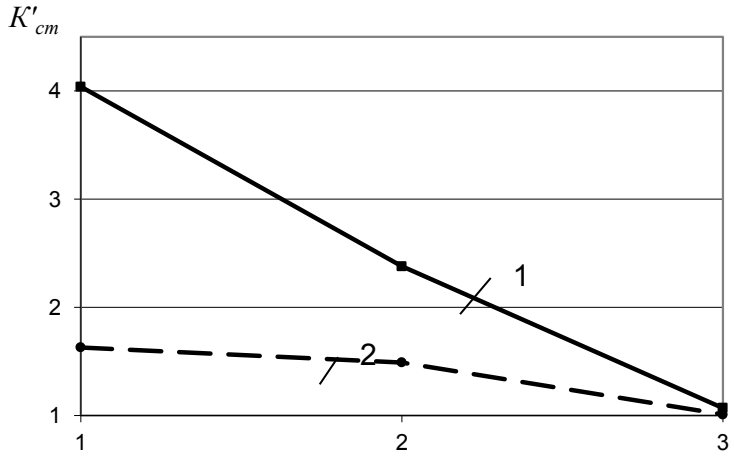

Рис. 1 Залежність ефективності стиснення інформації від кількості методів у каскаді:1 – каскад Віттер-LZH-LZW для текстової інформації,2 – каскад Віттер-матричний-LZW для вимірювальної інформації

На рис.1$K'_{ст}$ – це коефіцієнт стиснення на етапі каскаду:

$$K'_{ст} = V'_{п} / V'_{ст},$$

де $V'_{п}$ – об'єм даних на вході методу стиснення поточного етапу, $V'_{ст}$ – об'єм даних на виході методу стиснення поточного етапу. Як бачимо, метод стиснення, що застосовується на третьому етапі, практично не зменшує обсягу даних (на третьому етапі $K'_{ст} \approx 1$), тобто він практично не впливає на коефіцієнт стиснення каскадного методу в цілому. Отже, необхідно використовувати каскади, які вміщують два методи стиснення.

Висновок

Таким чином, проведений аналіз дає можливість зробити висновок, що при збільшенні

кількості методів стиснення в каскаді, коефіцієнт стиснення не збільшується. Причому це не залежить від того, які методи стиснення використовуються, орієнтовані на даний тип інформації, чи ні. Оптимальною кількістю методів в каскаді є два методи.

Література

1. Цымбал В.П.Теория информации и кодирование / В.П.Цымбал. - К.: Вища шк., 1992. - 261 с.

2.Кузьмин И.В., Кедрус В.А. Основы теории информации и кодирования / Кузьмин И.В., Кедрус В.А. - Киев: Вища школа, 1977. - 280 с.

3.Хэмминг Р.В. Теория кодирования и теория информации / Р.В. Хэмминг - М.: Радио и связь, 1983. - 176 с.

4.Жураковский Ю. П., Назаров В. Д. Каналы связи. / Жураковский Ю. П., Назаров В. Д — К.: Вища шк. Головное изд-во, 1985. — 236 с.

5.Жураковський Ю. П. Полторак В. П. Теорія інформації та кодування./ Жураковський Ю. П. Полторак В. П – К.: «Вища школа», 2001. – 256 с.

6. Жураковський Б.Ю. Моделювання процесів стиснення повідомлень /Жураковський Б.Ю. // Радіотехніка 2009, вип.159, с.279-282.

7. Розоринов Г.Н. Уменьшение избыточности цифровых изображений / Розоринов Г.Н.// Вісник ДУІКТ № 4, 2013 р. – стр.5 -12.

8. Кричевский Р. Е. Сжатие и поиск информации./ Кричевский Р. Е. — М.: Радио и связь, 1989. —168с.

9. Кохманюк Д. Сжатие данных: как это делается. / Кохманюк Д – IndexPro, 1992, №1, с.18-29; 1993, №2, с.30-49.

Надійшла 22.01.2015 р. Рецензент: д.т.н., проф. Розорінов Г.М.