УДК 338.48 (477.8)
С.В. Гринюк, Т.В. Максимчук, Н.А. Христинець
Луцький національний технічний університет

АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ СТИСНЕННЯ ІНФОРМАЦІЇ

В статті розглянуто деякі алгоритми та методи стиснення інформації, характерною особливістю якої є надлишковість, залежить від їх типу та прийнятої системи кодування. Встановлено, що стиснення інформації широко використовується як при їх зберіганні на електронних носіях, так і при передачі каналами зв'язку.
Ключові слова: *стиснення інформації, теорія інформації, кодування інформації, алгоритми і методи стиснення інформації, моделі вхідного потоку інформації.*

Постановка проблеми. Стиснення інформації широко використовується як при їх зберіганні на електронних носіях, так і при передачі каналами зв'язку. Оскільки пропускна здатність сучасних каналів зв'язку є ще не достатньо великою, а процедура передачі значного обсягу даних є ще достат ньо дорогою, тому стиснення даних – надзвичайно актуальна проблема. При цьому на передавальній станції процедура стиснення даних зводиться до кодування вхідної інформації за спеціальним алгоритмом, а приймальна станція виконує їх декодування за тим самим або й зовсім іншим алгоритмом [2].

Аналіз останніх досліджень і публікацій. Детальніші відомості з вирішення цієї проблеми можна знайти, наприклад, у багатьох початкових роботах Р.Е. Кричевського [1989], Б.Я. Рябка [1980], I.H. Witten [1987], J. Rissanen [1981], D.A. Huffman [1952], R.G. Gallager [1978], D.E. Knuth [1985], J.S. Vitter [1986] і ін.

Невирішені частини проблеми. Існує декілька різних теорій вирішення проблеми стиснення інформації. Одні мають надто складну математичну основу, інші базуються на властивостях інформаційного потоку, внаслідок чого алгоритмічна їх реалізація є достатньо простою. Загалом будь-який метод і алгоритм, що реалізують стиснення інформації, призначені для зниження обсягу вихідного потоку інформації в бітах.

Метою дослідження є огляд основних особливостей стиснення інформації, проаналізувати основні алгоритми і методи його реалізації (див. рис. 1).

Основні результати дослідження. Зазначимо, що надалі будемо розглядати компресор (compressor) як програму, що перетворює масив символів деякого вхідного алфавіту в інший, дещо менший за розміром [6]. Часто як масив використовується безструктурний двійковий файл (подібний до файла MS-DOS або UNIX), а масивом символів вхідного алфавіту є 256 можливих значень байта (але не завжди). Відповідно, декомпресор (decompressor) – програма, що виконує однозначно зворотне перетворення. Отож з подальшого аналізу вилучимо методи і алгоритми стиснення інформації, що втрачають вхідну інформацію. Наприклад, метод стиснення зображень JPEG базується на перетворенні кольорів, які практично неможливо розрізнити людським оком.

Кодування інформації. Існує декілька різних теорій вирішення проблеми стиснення інформації. Одні мають надто складну математичну основу, інші базуються на властивостях інформаційного потоку, внаслідок чого алгоритмічна їх реалізація є достатньо простою. Загалом будь-який метод і алгоритм, що реалізують стиснення інформації, призначені для зниження обсягу вихідного потоку інформації в бітах. Основними їх технічними характеристиками є:

- міра стиснення (compress rating) або відношення (ratio) обсягів вхідного і вихідного потоків;
- швидкість стиснення – час, що витрачається на кодування вхідного потоку інформації до отримання з нього перетвореного вихідного потоку;
- якість стиснення – показник, який характеризує ефективність ущільнення вихідного потоку після повторного застосування до нього цього самого або іншого алгоритму.

Найбільш відомим підходом до вирішення проблеми стиснення інформації є алгоритми кодування довжин серій (run length encoding – RLE). Його зміст полягає у заміні послідовності символів, що повторюються, на один цей самий символ та лічильник повторюваності.

Проблема тут полягає в розробленні такого декодера, який давав би змогу відрізнити у вихідному потоці закодовану серію символів від інших символів. Її вирішення полягає у додаванні до такої послідовності символів заголовків, які використовують перший біт як ознаку закодованої серії.

Розроблений за наведеним алгоритмом метод є досить ефективним для кодування графічних зображень у форматі "байт на піксель" (наприклад, формат PCX використовує метод кодування

RLE). Його недоліками є низька адаптація до багатьох типів файлів, наприклад, текстових, а також реально можна стискати тільки послідовності проміжків на початку абзаців. Саме тому цей метод отримав практичне застосування тільки у комбінації з вторинним кодуванням. Зокрема, його реалізовано в алгоритмі кодування факсів: спочатку за допомогою методу RLE зображення ділиться на чорні та білі крапки, а потім вони кодуються методом Хаффмена за спеціально підібраним бінарним деревом.

Процес кодування (encoding) має справу з потоком символів у деякому алфавіті, частоти повторюваності яких різні. Його завдання полягає у перетворенні вхідного потоку даних на потік бітів мінімальної довжини. Це досягається завдяки зменшенню надлишковості вихідного потоку даних шляхом врахування частоти появи символів на вході. При цьому довжина отриманого коду здебільшого є пропорційною обсягові інформації, що міститься у вхідному потоці. Якщо розподіл частот повторюваності символів наперед відомий, то можна будувати оптимальний алгоритм кодування. Якщо ж розподіл частот заздалегідь невідомий, то існують два різних підходи до вирішення цієї проблеми.

Перший підхід базується на тому, що після перегляду вхідного потоку використовується алгоритмом кодування на основі зібраної статистики появи символів. Загалом це потребує двох проходів файлом, що звужує сферу застосування таких алгоритмів. Окрім цього, у вихідний потік записується кодова таблиця, за допомогою якої потім декодер здійснює відтворення вхідного потоку.

Прикладом такого підходу є статистичне кодування методом Хаффмена. Цей алгоритм співставляє вхідним символам, що надходять послідовностями бітів однакової довжини (наприклад, 8-бітовими байтами), послідовності бітів змінної довжини. Довжина коду пропорційна (з округленням до цілого) двійковому логарифму від його частоти, взятому з оберненим знаком. Оскільки таке кодування є префіксним, то воно дає змогу його легко декодувати однопрохідним алгоритмом. У префіксному кодуванні код будь-якого символа не є префіксом коду жодного іншого символа. Зазвичай він представляється у вигляді двійкового дерева, в якому символи знаходяться на листках, а ребра позначені 0 або 1. Тоді код символа можна задати як шлях від кореня дерева до листка, що містить цей символ.

Другий підхід використовує так званий адаптивний кодер (adaptive coder), який змінює алгоритм кодування залежно від структури вхідного потоку даних. Такий алгоритм є однопрохідним і не потребує передачі інформації про використану кодову таблицю в явному вигляді. Замість цього декодер, зчитуючи закодований потік, синхронно з кодером змінює алгоритм кодування, починаючи з деякого початкового символу. Адаптивний алгоритм кодування забезпечує більший ступінь стиснення даних, оскільки враховуються локальні зміни
частот появи символів. Прикладом такого підходу є динамічне кодування методом Хаффмена. Цей алгоритм здійснює постійне корегування двійкового дерева відповідно до статистичних показників вхідного потоку, які постійно змінюються. Реалізація цього алгоритму, як правило, потребує значних витрат часу на балансування двійкового дерева відповідно до нових частот появи символів на кожному кроці.

Перевагами методу Хаффмена є його достатньо висока швидкодія та якість стиснення інформації. Цей метод порівняно з іншими давно відомий і, як на сьогодні, широко розповсюджений, наприклад, програма compact ОС UNIX (програмна реалізація), а також стандарт кодування факсів (апаратна реалізація). Кодування даних методом Хаффмена має мінімальну надлишковість за умови, що кожний символ кодується окремою послідовністю у алфавіті {0, 1}.

Недоліком кодування методом Хаффмена є залежність ефективності стиснення від близькості ймовірностей символів до від'ємних степенів Проблема в тому, що кожен символ кодується цілою кількістю бітів. Найбільш помітно це відображається при кодуванні двосимвольного алфавіту, при якому стиснення взагалі неможливе, незважаючи на відмінності ймовірностей появи символів; алгоритм фактично "округляє" їх до 1/2!

Проте, ця проблема частково вирішується блокуванням вхідного потоку (тобто внесення до розгляду нових символів такого вигляду: "ab", "abc",…, де a, b, c – символи початкового алфавіту). Однак це не дає змоги повністю позбутися втрат (вони тільки зменшуються пропорційно розмірові блока) і призводить до стрімкого зростання розміру двійкового дерева. Якщо, наприклад, символами вхідного алфавіту є 8-бітові байти зі значеннями 0…255, то при блокуванні по два символи отримуємо алфавіт (і бінарне дерево) із 65536 символів, а при блокуванні по три – 16777216. За таких обставин значно зростають вимоги до обсягу використовуваної пам'яті та тривалості побудови двійкового дерева. У випадку використання адаптивного алгоритму

кодування зростає і тривалість його поновлення, а значить і загальна тривалість процесу стиснення інформації. Втрати ж складають у середньому 1/2 біт на символ при відсутності блокування, а за його наявності – 1/4 і 1/6 біт для блоків довжини 2 та 3 біти відповідно.

Хорошу якість стиснення інформації, яке не залежить від близькості значень ймовірностей появи символів до степенів 1/2, дає алгоритм арифметичного кодування інформації. Відповідний метод кодування дає змогу стискати вхідне повідомлення без втрат за умови, що відомий розподіл частот появи його символів. Концепцію цього методу було наведено у роботах Еліаса ще в 60-х роках XX ст. Пізніше метод було удосконалено та значно модифіковано. Вважається, що метод арифметичного кодування є найбільш оптимальним, позаяк досягається теоретична межа стиснення інформації – ентропії вхідного потоку. Текст, який стиснено арифметичним кодером, розглядається як деякий двійковий дріб з інтервалу [0, 1). Результат стиснення можна подати у вигляді послідовності двійкових цифр із цього дробу.

Ідея методу полягає в тому, що початковий текст розглядається як запис цього дробу, де кожен вхідний символ є "цифрою" з вагою, що пропорційна ймовірності його появи. При реалізації цього методу виникають такі проблеми: необхідно використовувати дійсну арифметику необмеженої точності; результат кодування стає відомим тільки після припинення надходження символів вхідного потоку. Однак, проведені дослідження [2-4] показали, що можна практично без втрат обмежитися цілочисельною арифметикою невеликої точності (16-32 розряди), а також домогтися покрокової (інкрементальної) роботи алгоритму: цифри коду можуть видаватися послідовно при зчитуванні вхідного потоку.

Моделі вхідного потоку інформації [3, 4]. Кодування даних є тільки частиною процесу їх стиснення. Не менш важливим є так зване моделювання (modelling) самого процесу. Як уже зазначалося вище, арифметичне кодування інформації має мінімальну надлишковість за певної частоти розподілу символів вхідного алфавіту. Але звідки береться цей розподіл і що собою представляє вхідний алфавіт? Відповіді на ці запитання дає модель вхідного потоку інформації, яка представляє деякий спосіб передбачення розподілу ймовірностей при надходженні кожного наступного си-мвола. Саме кожного, оскільки статичні моделі, у яких розподіл не змінюється в процесі ко-дування, здебільшого не забезпечують максимальної якості стиснення інформації.

Значно цікавіші так звані адаптивні моделі, які враховують поточний контекст вхідного потоку інформації. Такі моделі дають змогу будувати швидкі однопрохідні алгоритми стиснення інформації, які не вимагають апріорних знань про вхідний потік інформації і будують розподіл ймовірностей появи символів під час роботи. Також виділяють клас "локально адаптивних" алгоритмів, які при побудові розподілу появи символів алфавіту відзначають більшою вагою частоти тих символів, які надійшли пізніше. Відомі різні підходи до вирішення цієї проблеми. Найпростіший з них – збирання статистики появи кожного символа незалежно від інших, наприклад, при моделюванні потоку лічильником Бернуллі ймовірність появи поточного символа не залежить від того, які символи траплялися перед ним. Інший варіант – використання марківської моделі, коли розподіл ймовірностей появи символів будується з урахуванням деякої кількості попередніх символів.

Аналізуючи моделі потоку інформації та адаптивні алгоритми стиснення інформації, заслуговує уваги простий та достатньо ефективний метод кодування джерела з невідомим розподілом частот появи символів. Цей метод відомий як стиснення інформації за допомогою "купки книжок" [5], який було вперше відкрито та досліджено Б.Я. Рябком ще у 1980 р., а потім повторно відтворено у 1986 р. Бентлі, Слейтером, Тар'яном і Вей.

Ідея роботи методу полягає у створенні алфавіту джерела інформації $A = \{a_i, i = \overline{1,m}\}$. Кодер зберігає список символів $B = \{b \in A, j = \overline{1,n}\}$, який представляє деяку перестановку символів алфавіту. При надходженні на вхід символа c, що має в цьому списку номер i, кодер присвоює йому номер i. Потім кодер переміщує символ c на початок списку, збільшуючи на одиницю номери всіх символів, що знаходилися перед c. Внаслідок виконання таких дій найбільш "популярні" символи сконцентруються на початку списку і матимуть коротші коди, а усі інші – у кінці списку зі значно довшими кодами.

Алгоритм дворівневого кодування Лемпеля-Зіва [10]. Усі методи та моделі кодування інформації, проаналізовані вище, розглядали як вхідні дані послідовності символів у деякому нескінченому алфавіті. При цьому залишалося відкритим питання про зв'язок цього вхідного алфавіту кодера з даними, що підлягають стисненню, і представлені у вигляді послідовності у іншому алфавіті, який зазвичай складається із 256 символів-байтів.

Значно якісніше стиснення інформації можна отримати шляхом виділення із вхідного потоку послідовностей, що повторюються, після чого здійснити кодування посилань на ці послідовності. Цей метод належить Лемпелю та Зіву і називається LZ77-compression (за роком публікації). Його робота полягає у виконанні таких дій: компресор постійно зберігає певну кількість останніх оброблених символів у деякому буфері (який називається ковзаючим словником – sliding dictionary). Назва "ковзаючий" пов'язана з тим, що за постійної довжини буфера кожного разу, коли компресор кодує наступну послідовність символів, він дописує їх у кінець словника та відкидає таку саму кількість символів на його початку. При обробленні вхідного потоку символи, що надійшли пізніше, потрапляють у кінець буфера, зсуваючи попередні символи та витісняючи найперші.

Розміри цього буфера є різними в різних реалізаціях алгоритму LZ77. Зазвичай використання буфера більших розмірів дає змогу отримати дещо якісніший результат стиснення інформації. Алгоритм виділяє (шляхом пошуку в словнику) найдовший початковий підрядок вхідного потоку, що збігається з одним із підрядків у словнику, і подає на вихід пару (length, distance), де length – довжина знайденого у словнику підрядка, а distance – відстань від нього до вхідного підрядка (тобто фактично індекс підрядка в буфері, віднятий від його розміру). У випадку, коли такого підрядка не знайдено, до вихідного потоку просто додається черговий символ вхідного потоку.

У найпершій версії алгоритму Лемпеля-Зіва виконувався найпростіший лінійний пошук усім словником. Тривалість стиснення інформації за такої реалізації була пропорційною добуткові довжини вхідного потоку на розмір буфера, що значно сповільнювало роботу алгоритму. Але потім було використано збалансоване бінарне дерево для пошуку в словнику відповідного підрядка символів, що дало змогу на порядок збільшити швидкість роботи алгоритму.

Таким чином, алгоритм Лемпеля-Зіва [7] перетворює один потік вхідних символів на два паралельних потоки length та distance. Очевидно, що ці потоки є потоками символів у алфавітах L та D, і до них можна застосувати один із наведених вище методів (RLE, кодування методом Хаффмена, арифметичне кодування). Алгоритм роботи цього методу називається дворівневим кодуванням і вважається найефективнішим серед усіх, що використовуються сьогодні на практиці. При реалізації цього методу є проблема узгодженості виведення обох потоків даних у один файл, яка зазвичай розв'язується шляхом почергового запису кодів символів із обох потоків.

Сімейство алгоритмів LZ78 (LZW, MW, AP, Y). Серед цього сімейства насамперед заслуговує уваги алгоритм Лемпеля-Зіва-Велча (Lempel-Ziv-Welch compression – LZW, LZ78), який вирізняється від інших високою швидкодією роботи при кодуванні та декодуванні будь-якої інформації, невибагливістю до пам'яті та простою апаратною реалізацією. Недолік – менша ефективність стиснення інформації порівняно з алгоритмом дворівневого кодування [1, 2].

Принцип роботи цього алгоритму полягає у виконанні таких дій. Спочатку створюється словник, що зберігає рядки тексту і містить близько 2-3-8 тисяч пронумерованих комірок. Потім записуються до перших 256 комірок рядки, що складаються з одного символа, номер якого збігається з номером комірки. Алгоритм проглядає вхідний потік, розбиваючи його на підрядки, і додає нові комірки в кінець словника.

Після зчитування із вхідного потоку по одному символу, кожного разу відбувається перевірка, чи є вже зчитана послідовність символів у словнику. Внаслідок виконання таких дій отримуємо рядок S – найдовший префікс вхідного тексту (якщо його розглядати як рядок In), який вже є у словнику. Нехай його знайдено в комірці з номером n. Виведемо число n у вихідний потік, змістимо вказівник вхідного потоку на length(S) символів наперед та додамо до словника нову комірку, що містить рядок Sc, де c – черговий символ на вході (відразу після S). За побудованою послідовністю символів S послідовності Sc у словнику немає. Алгоритм перетворює потік символів на вході на потік індексів комірок словника на виході. При розмірі словника в 4096 комірок можна передавати 12 біт на індекс кожної послідовності, що знайдено.

При реалізації цього алгоритму треба враховувати й те, що будь-яка комірка словника, за винятком найперших, які містять односимвольні послідовності, зберігає копію деякої іншої комірки, до кінця якої приписано один символ: використовується літерно-інкрементний словник.

Таким чином, алгоритм LZW має важливу властивість префіксності: якщо деяка послідовність символів S міститься в словнику, то всі його префікси – також. Далі, кодер використовує дві операції зі словником: здійснює пошук рядка символів Sc за умови, що рядок S є в словнику (та відома його позиція в ньому), а c – наступний зчитаний символ; виконує додавання

нової комірки, яка містить цей рядок. Внаслідок цього можна обійтися структурою даних, що називається trie – деревом послідовного політерного пошуку.

Тривалість роботи цього алгоритму є лінійною за довжиною вхідного потоку та не залежить від розміру словника. Для кожного символа, що зчитується, відбувається або один пошук у словнику, або додавання однієї нової комірки. Саме тому він є досить популярним у апаратних реалізаціях, наприклад – протокол v42bis.

Оскільки кодер алгоритму LZW додає до словника одну комірку для кожної розпізнаної послідовності символів, то його можна назвати методом із фразо розширюваним словником. Коли словник переповнюється, кодер може або припинити його заповнення ("заморозити" словник, як це використано у класичному алгоритмі LZW), або очистити (повністю – compress, частково – алгоритм PKZIP shrinking).

Алгоритм MW (Miller and Wegman), на відміну від алгоритму LZW, утворює нові комірки словника шляхом склеювання двох рядків, тобто використовує фразо-інкрементний словник. Розпізнавши за аналогією з алгоритмом LZW послідовність символів S на вході, він додає до словника нову комірку PS – конкатенацію послідовності символів P (попередньо розпізнаної послідовності) та символів S (нової). Завдяки цьому він, як правило, швидше адаптується до тексту, але може пропустити деякі послідовності. Тому він стискає вхідний потік інформації не набагато краще за алгоритм LZW, а деколи навіть і гірше.

Висновки. Більшість типів даних характеризується надлишковістю, яка залежить від типу даних і прийнятої системи кодування. Стиснення інформації широко використовується як при їх зберіганні на електронних носіях, так і при передачі каналами зв'язку. Існує декілька різних теорій вирішення проблеми стиснення інформації. Одні мають надто складну математичну основу, інші базуються на властивостях інформаційного потоку, внаслідок чого алгоритмічна їх реалізація є достатньо простою. Загалом будь-який алгоритм чи метод, що реалізують стиснення інформації, призначені для зниження обсягу вихідного потоку інформації в бітах.

1. Баранов Г. Обзор методов сжатия данных / Геннадий Баранов. [Електронний ресурс]. – Доступний з http://www.compression.ru/arctest/descript/methods.htm
2. Берлекамп Є. Алгебраическая теорія кодирування [Текст]: пер. с англ./ Є. Берлекамп; под ред. С.Д. Бирмана – М.: Мир, 1971. – 477 с.
3. Ватолин Д. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : Изд-во "Диалог-МИФИ", 2002. С. 384 с.
4. Кассами Т. Теория кодирования [Текст]: пер. с яп. / Т. Кассами, Н. Токура, Е. Ивадари, Я. Инагави; под ред. Б.С. Цыбакова, С.И. Гельфанда. - М.: Мир, 1987. - 392 с.
5. Кохманюк Д. Сжатие информации: как это делается / Д. Кохманюк // Index PRO, 1993. – К., № 1-2. – С. 23-28.
6. Кричевский Р. Е. Сжатие и поиск информации / Р. Е. Кричевский. – М. : Изд-во "Радио и связь", 1989. – 364 с.
7. Рябко Б. Я. Сжатие информации с помощью стопки книг / Б.Я. Рябко // Проблемы передачи информации. – 1980. – Т. 16, № 4. – С. 16-21.
8. Сэломон Д. Сжатие данных, изображения и звука / Д. Сэломон. – М. : Изд-во "Техносфера", 2004. – 368 с.
9. Шеннон К. Математическая теория связи [Текст]. Работы по теории информации и кибернетике : пер. с англ. / К. Шеннон; под ред. Р.Л. Добрушина и О.В. Лупанова - М.: ИЛ, 1963. - 830 с.
10. Ziv J. Compression of individual sequences via variable-rate coding / J. Ziv, A. Lempel // IEEE Transactions, IT-24. – No. 5 (Sept. 1978). – PP. 530-536.