УДК 004.657

О.В. Тарасов, Є.В. Онопко

Харківський національний економічний університет, Харків

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ БЛОЧНО-СТАТИСТИЧНОГО МЕТОДУ СТИСНЕННЯ ІНФОРМАЦІЇ

У статті розглянутий блочно-статистичний алгоритм стиснення інформації, котрий є модифікацією класичного алгоритму Хаффмана, який до цього часу є затребуваним у сучасних алгоритмах стиснення та обробки інформації. Було проведено дослідження ефективності розробленого алгоритму для різних типів вхідних даних у порівнянні з оригінальним. Зроблені висновки щодо доцільності використання алгоритму для стиснення у реальних базах даних, та необхідності його подальшого дослідження.

Ключові слова: *метод Хаффмана, стиснення даних, архів, коефіцієнт стиснення.*

Вступ

Стиснення інформації є однією з тих проблем, яка нерозривно пов'язана з обробкою даних з використанням засобів обчислювальної техніки. Текст і звук, графіка і відео – для кожного з цих видів інформації існують свої найбільш відповідні методи стиснення. Метою процесу стиснення, як правило, є отримання компактнішого способу представлення початкових даних, який мінімізує об'єм займаної пам'яті за допомогою деякого їх перетворення.

Існує декілька різних підходів до проблеми стиснення інформації, які базуються або на складних математичних алгоритмах або засновані на властивостях інформаційного потоку і алгоритмічно досить прості [1]. Проте усі способи стиснення можна розділити на дві категорії: оборотне і безповоротне стиснення. У першому випадку можливо повне і безпомилкове відновлення початкових даних, які були піддані стисненню. У другому – це неможливо. Проте з точки зору практичного застосування результат, отриманий в процесі стиснення, може бути цілком задовільний, наприклад, для графіки або звукових повідомлень.

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 [2]. 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 відносяться 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 [3].

Якщо при стисненні даних відбувається тільки зміна структури даних, то метод стиснення є зворотнім. У цьому випадкові 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, тощо. Існує багато різних практичних методів стиснення без втрати інформації, які, як правило, мають різну ефективність для різних типів даних та різних обсягів.

Особливий інтерес викликає застосування методів стиснення у базах даних для зменшення об'єму текстових полів. По-перше, для таких даних неприпустимо використання алгоритмів безповоротного стиснення, а по-друге, застосування алгоритмів, закладених в широко відомих архіваторах, наприклад, RAR або ZIP, є неефективним з причини того, що стисненню піддаються текстові поля невеликого об'єму і в цьому випадку отриманий при стисненні результат, з урахуванням даних для декодування, може займати об'єм пам'яті більший, ніж початкові дані [4]. Найбільш часто, в таких випадках, застосовується метод Хаффмана[8].

Основна частина

Код Хаффмана (Huffman code) це мінімально-надлишковий префіксний код (minimum-redundancy prefix code). Розглянемо основні ідеї коду Хаффмана та зробимо дослідження ряду важливих властивостей алгоритму.

Основною ідеєю алгоритму Хаффмана є те, що кодування символів вхідного алфавіту здійснюється різним числом біт. Символи, які зустрічаються частіше, будуть закодовані меншим числом біт, ніж ті, які зустрічаються рідше. Отриманий код буде оптимальний або, іншими словами, мінімально-надлишковий.

Ідея алгоритму була опублікована Девідом Хаффманом в 1952 році. Алгоритм Хаффмана двопрохідний. На першому проході будується частотний словник і генеруються коди. На другому проході відбувається безпосередньо кодування.

За 50 років з дня опублікування, код Хаффмана нітрохи не втратив своєї актуальності і значущості. Так з упевненістю можна сказати, що ми стикаємося з ним, в тій чи іншій формі (справа в тому, що код Хаффмана рідко використовується окремо, частіше працюючи у зв'язці з іншими алгоритмами), практично кожен раз, коли архівуємо файли, дивимося фотографії, фільми, посилаємо факс або слухаємо музику [5].

Стискаючи файл за алгоритмом Хаффмана перше, що ми повинні зробити, – це прочитати файл повністю і підрахувати скільки разів зустрічається кожен символ з розширеного набору ASCII, тобто визначити їх відносну частоту у файлі. Якщо ми будемо враховувати усі 256 символів, то не буде різниці в стисненні текстового і EXE файлу.

Після підрахунку частоти входження кожного символу, необхідно переглянути отримані результати, впорядкувати символи за убуванням частоти їх входження у файл. Фактично не змінюючи місцезнаходження кожного символу в таблиці – відсортувати таблицю посилань на них за спаданням частоти. Кожне посилання з останньої таблиці назвемо "вузлом".

Алгоритм отримання класичного коду Хаффмана зображений на рис. 1.

Рис. 1. Блок схема
класичного алгоритму Хаффмана

Виникає питання. А чи не можна зменшити середню довжину коду, розбивши увесь вхідний алфавіт на дві або більше груп і провівши кодування по Хаффману для кожної з них окремо? Правда в цьому випадку у вхідному алфавіті з'являються нові символи, що характеризують перехід до тієї або іншої групи, але довжина коду у кожній окремій групі може бути зменшена. Був проведений експеримент, який для вхідного алфавіту з 80 символів визначив середню довжину коду символу для алгоритму Хаффмана, при розподілі частот символів, що підкоряються закону Ципфа [6].

Далі початковий алфавіт був розбитий на дві рівні по кількості символів групи. Сумарна імовірність символів в кожній групі, при цьому, склала: $P_1 = 0{,}862$ і $P_2 = 0{,}138$. Вірогідність переходів з групи в групу при цьому дорівнює $P_{12}=P_{21}=P_1 x P_2=0{,}119$. Наступним кроком, символи в кожній групі були окремо піддані стисненню по методу Хаффмана [7]. Блок-схема модифікованого алгоритму зображена на рис. 2. Результати експерименту наведені в табл. 1.

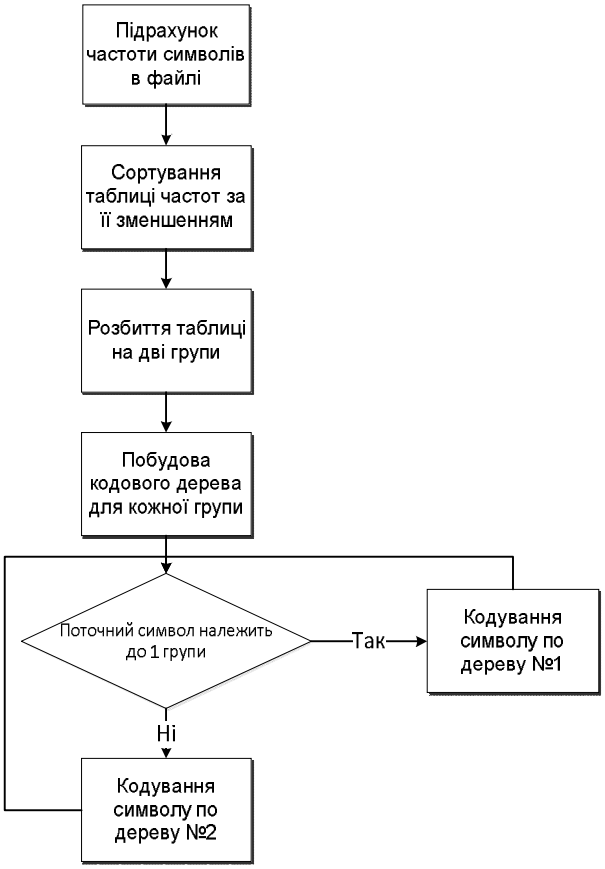

Рис. 2. Блок-схема модифікованого алгоритму

Таблиця 1

Результати розрахунку середньої довжини коду

Для усього тексту	Для блоку №1 (P_1=0,862)	Для блоку №2 (P_2=0,138)
4,91	4,36	3,49
	Середня довжина=4,24	

Для отримання оптимального результату були проведені експерименти з розбиттям таблиці частот символів на групи різні за розмірами. На першому

кроці до першої групи були віднесені 5% символів, котрі містяться в вхідному тексті, до другої – 95%. На кожному наступному кроці кількість символів для першої групи збільшувалась на 5%, а для другої – зменшувалась на 5%.

Моделювання було проведено за допомогою розробленої програми, яка отримує на вхід текстовий документ, проводить кодування по класичному алгоритму Хаффмана (будує кодове дерево та кодує кожен символ) та по його модифікації (розбиває таблицю символів на дві групи згідно їх частот, будує кодове дерево для кожної з них та кодує символи за допомогою них), підраховує розмір стиснених даних за оригінальним алгоритмом та модифікацією.

Для проведення експериментів використовувалися дані трьох типів.

На першому етапі були використані сгенеровані дані, в яких символи розташовані по убуванню їх частоти, а самі частоти підкоряються закону Ципфа.

На другому етапі були використані сгенеровані дані, в яких розподіл символів також підкоряється закону Ципфа, але розташовані вони були у довільному випадковому порядку.

Третій етап експерименту включав дослідження ефективності модифікації алгоритму в порівнянні з класичним на даних, котрі були отримані з реальних баз даних.

Базуючись на результатах проведених досліджень першого експерименту можна зробити висновок, що розроблена модифікація алгоритму Хаффмана є більш ефективною в порівнянні з класичним алгоритмом. Для даних, довжина котрих приблизно дорівнює 50 символам, ефективність досягає 30% у порівнянні з класичним алгоритмом. В середньому ефективність досягає 23%. Для цього типу даних доцільно розділяти символи тексту на дві групи, котрі також, приблизно рівні за обсягом. На рис. 3 зображений графік, котрий демонструє перевагу модифікації при стисненні різного за довжиною тексту.

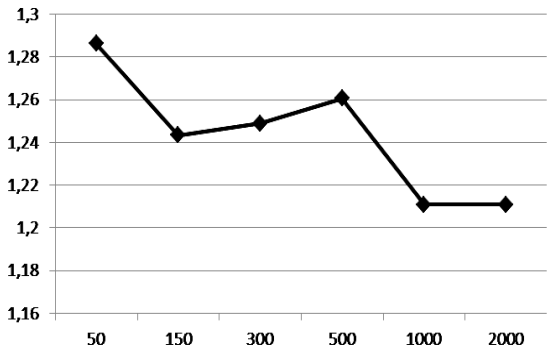

Рис. 3. Ефективність стиснення для даних, у яких символи розподілені за законом Ципфа та йдуть по порядку

Для даних, у яких символи розподілені за законом Ципфа та розташовані у довільному порядку, модифікований алгоритм найбільш ефективний для даних, довжина котрих приблизно дорівнює 50. На цих даних ефективність алгоритму на 13% вище у порівнянні з кодом Хаффмана. Також для цього типу даних доцільно розділяти символи тексту на дві групи, приблизно рівні за обсягом. На рис. 4 зображений графік, котрий демонструє перевагу модифікації при стисненні тексту, що відрізняється довжиною.

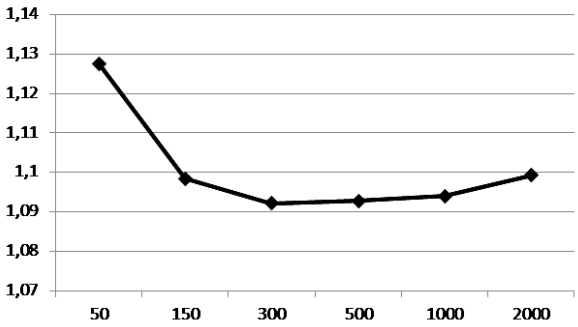

Рис.4. Ефективність стиснення для даних, у яких символи розподілені за законом Ципфа та розташовані у довільному порядку.

Тестування для реальних даних проводилося для двох наборів даних.

Перший набір містив рядки тексту з символів кирилиці довжиною від 10 до 80. Ефективність розробленої модифікації досягає приблизно 13%. Цей показник вірний для даних, котрі суттєво відрізняються довжиною. Найбільший коефіцієнт стиснення отриманий при розділенні символів тексту на дві рівні групи. На рис. 5 зображені узагальнені дані, що характеризують залежність розміру стиснутих даних від розміру двох груп символів (за частотою). За віссю ординат відображене середнє значення довжини рядка після стиснення, за віссю абсцис – розмір першої групи символів.

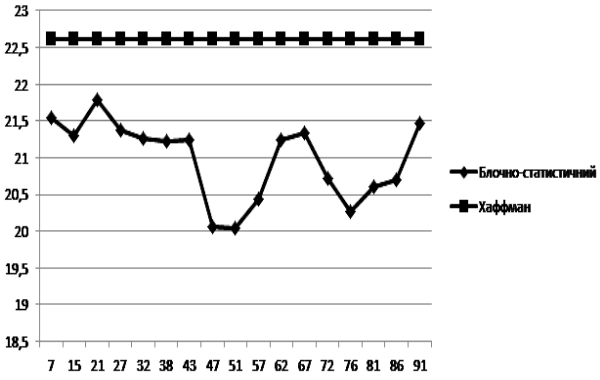

Рис. 5. Залежність об'єму стиснутих даних від кількості символів у групах у порівнянні з алгоритмом Хаффмана.

На рис. 6 зображений графік, котрий демонструє перевагу модифікації при стисненні реальних даних.

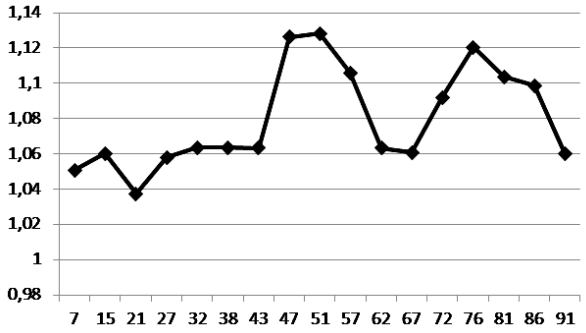

Рис.6. Ефективність стиснення реальних даних

Другий набір реальних даних містив текст змішаного характеру, тобто включав символи як кирилиці, так і латиниці. Довжина рядка коливалась від 10 до 170 символів. Отримані результати для другого набору, теж свідчать про більшу ефективність блочно-статистичного методу. В середньому коефіцієнт стиснення дорівнює 11%. Графік, котрий демонструє перевагу модифікації при стисненні другого набору реальних даних зображений на рис. 7.

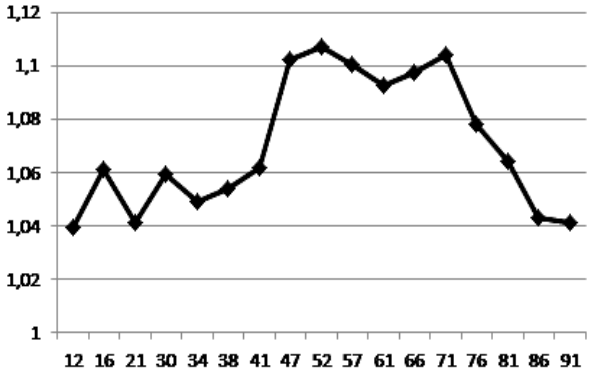

Рис.7. Ефективність стиснення реальних даних.

Висновки

В результаті проведених досліджень був розроблений блочно-статистичний метод стиснення даних, що базується на класичному алгоритмі Хаффмана. Сутність методу полягає в розбитті вхідного алфавіту на дві групи та кодуванні кожної групи символів окремо, включаючи й додаткові символи переключення з першої групи на друга й навпаки. Проведені дослідження показали, що блочно-статистичний метод дає більш високий ступінь стиснення різних типів текстових даних. Найбільша ефективність при цьому досягається у тому разі, коли сумарна частота символів у обох групах приблизно однакова.

Розроблена модифікація підтвердила свою ефективність для різних типів текстових даних, та заслуговує на подальше вивчення.

Список літератури

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

2. Селомон Д. Сжатие данных,изображений и звука. // Д. Селомон – М.: Техносфера, 2004. – 368 с.

3. Артюшенко В. М., Шелухин О. И. Цифровое сжатие информации // Артюшенко В. М., Шелухин О.И. – М.: Дашков и Ко, 2004. – 426 с.

4. Седжвик Р. Фундаментальные алгоритмы на С++. Части 1-4. Анализ. Структуры данных. Сорттровка. Поиск. // Р. Седжвик Р. – 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 // Кричевский Р.Е. – М.: Радио и связь, 198 9. – 424 с.

7. Тарасов О.В. Блочно-статистичний метод стиснення інформації // Тарасов О.В., Онопко Є.В. – Х.: ХУПС, 2011. – 172 с.

8. Левитин А.В. Алгоритмы: введение в разработку и анализ // А.В. Левитин. – М.: Вильямс, 2006. – С. 392-398.

Надійшла до редколегії 23.03.2012

Рецензент: д-р техн. наук, проф. В.П. Авраменко, Харківський національний університет радіоелектроніки, Харків.

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ БЛОЧНО-СТАТИСТИЧЕСКОГО МЕТОДА СЖАТИЯ ИНФОРМАЦИИ

А.В. Тарасов, Е.В. Онопко

В статье рассмотрен блочно-статистический алгоритм сжатия, который является модификацией классического алгоритма Хаффмана, который и до сего времени является востребованным в современных алгоритмах сжатия и обработки информации. Было проведено исследование эффективности разработанного алгоритма для различных типов входных данных по сравнению с оригинальным. Сделаны выводы о целесообразности использования алгоритма для сжатия в реальных базах данных и необходимости его дальнейшего исследования.

Ключевые слова: *метод Хаффмана, сжатие данных, архив, коэффициент сжатия.*

RESEARCHING THE EFFICIENCY OF BLOCK STATISTICAL DATA COMPRESSION METHOD

A.V. Tarasov, E.V. Onopko

This article discusses statistical block compression algorithm, which is a modification of the classical Huffman algorithm, and it's still being used in modern compression algorithms and information processing. Research was conducted to find out the efficiency of the modified algorithm for different types of input data. Conclusions were made about using algorithm in real databases for data compression and the need for further study.

Keywords*: Huffman method, data compression, archive, compression rate.*