УДК 621.391.251:681.3.053 **Жураковський Б.Ю.**

ОЦІНКА ШВИДКОДІЇ АЛГОРИТМІВ КАСКАДНОГО СТИСНЕННЯ

В роботі розглядаються запропоновані каскадні способи стиснення при обробці вимірювальної, текстової, графічної та комбінованої інформації, яка є характерною для автоматизованих систем управління. Одержані дані про швидкодію всіх запропонованих способів каскадного стиснення, що дає можливість зробити правильний вибір способу стиснення для використання його в автоматизованій системі управління.

Ключові слова: способи стиснення, алгоритм стиснення, блоки даних, компресія даних.

1. Алгоритми стиснення.

Алгоритми стиснення способом LZW і LZH. Незалежно від мови і характеру текстових повідомлень у різних 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. Зівом. Метод Лемпеля-Зіва відноситься до однопроходным адаптивних методів стиску послідовних даних, що не вимагає апріорних знань статистичних характеристик джерела повідомлень. Алгоритм перебуває з правила для синтаксичного аналізу рядків стисливого повідомлення, що перебуває із символів кінцевого алфавіту, і схеми кодування. У процесі аналізу з рядків повідомлення виділяються підстроки перемінної довжини. При кодуванні цим підстрокам ставляться у відповідність кодові комбінації, що перебувають із фіксованого числа елементів. Потім здійснюється пошук ідентичних підстрок у попередньому рядку. Після відшукання підстрок, що збігаються, вибирається підстрока максимальної довжини і робиться її заміна в поточному рядку повідомлення відповідною кодовою комбінацією.

Найбільше поширеним методом динамічного стиску 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 перебуває як з одиночних символів, так і з деяких літеросполучень. Причому кожному рядку відповідає своя двійкова комбінація (кодове слово) [1].

Метод стиску інформації LZH заснований на використанні двох методів LZ і Хаффмана. Алгоритм стиску схожий на алгоритм, що використовується в протоколі V. 42bis. У котрому також при ініціалізації формується словник котрий можна логічно представити за допомогою 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 представляє один рядок із цього набору.

Алгоритм стиснення способом Vitter. На початку 70-х років були розроблені однопрохідні методи стиску інформації, засновані на класичній процедурі кодування Хаффмена. Всі ці методи незначно відрізняються друг від друга і їхня суть полягає в тому, що передавач будує дерево Хаффмена в темпі надходження даних від джерела, тобто "на літу" [2,3]. У процесі кодування відбувається "навчання" кодера на основі статистичних характеристик джерела повідомлень, у ході якого обчислюються оцінки вихідних імовірностей повідомлення і робиться відповідна модифікація кодового дерева Хаффмена. У зв'язку з безупинною зміною кодового дерева цей процес одержав назву динамічного кодування Хаффмена. Очевидно, що для правильного відновлення стиснутих даних, декодер також повинний безупинно "учитися" поряд із кодером, здійснюючи синхронну зміну кодової таблиці на прийомній стороні. Для забезпечення синхронності процесів кодування і декодування кодер видає символ у незжатому вигляді, якщо він уперше з'явився на виході

джерела, і відзначає його на кодовому дереві. При повторній появі символу на вході кодера він передається нерівномірною кодовою комбінацією, обумовленою позицією символу на поточному кодовому дереві. Кодер коректує дерево Хаффмена збільшенням частоти передачі символів, що уже введені в дерево, або нарощує дерево, додаючи в нього нові вузли.

Вперше алгоритм синтезу динамічного коду Хаффмена був запропонований Н.Феллером у 1973 році, а потім модифікований Р.Галлагером і Д.Кнутом. У зв'язку з цим він одержав назву "алгоритм FGK".

Подальше удосконалення алгоритму динамічного кодування даних нерівномірними кодами FGK було запропоновано Д. Вітером. Цей алгоритм одержав назву алгоритму V. При пошуку шляхів оптимізації процедури кодування даних кодом Хаффмена автор виходив із того, що в новому алгоритмі число обмінів вузлів у процесі модифікації кодового дерева повинно обмежуватися деяким малим числом (у кращому випадку одиницею), а динамічне хаффменовське дерево повинно будуватися таким чином, щоб мінімізувати не тільки сумарну довжину зовнішнього шляху $\sum W_j l_j$, але і розміри $\sum l_j$, $\max\{l_j\}$. Мінімізація висоти дерева $h = \max\{l_j\}$ дозволить запобігти утворенню довгих кодових комбінацій при кодуванні чергового символу в повідомленні. Віттеру в значній мірі удалося вирішити поставлену задачу [4].

Алгоритм матричного способу стиснення. З матричних способів стиснення найбільш придатний для використання в системах управління адаптивно-матричний.

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 файла, 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 спосіб 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 спосіб має високу швидкість стиснення, оскільки виділення матриць робиться одночасно з прийомом даних.

Ціллю даного дослідження є оцінка швидкості роботи кожного з алгоритмів стиску для різних видів інформації і різних параметрів стиски, на підставі отриманих результатів виробити рекомендації по застосуванню методів стиску в системах, критичних до швидкодії алгоритму.

Тестування проводилося на процесорах AMD-2400, *Intel Pentium* -2000. Для винятку впливу розміру і швидкості кэша другого рівня на швидкість роботи алгоритму кеш-пам'ять системи відключалася. Для тестування використовувалися фрагменти даних довжиною 4096 байт, при цьому вимір робився в циклі 1000 разів для підвищення точності результатів.

2. Швидкодія і час стиснення різних методів стиснення.

Швидкодія і час стиснення методом LZH. На рисунку 1 показана залежність часу стиснення від розміру, стисливого блока. Як видно з малюнка, чим більше стисливий блок, тим більше часу затрачається на його стиск.

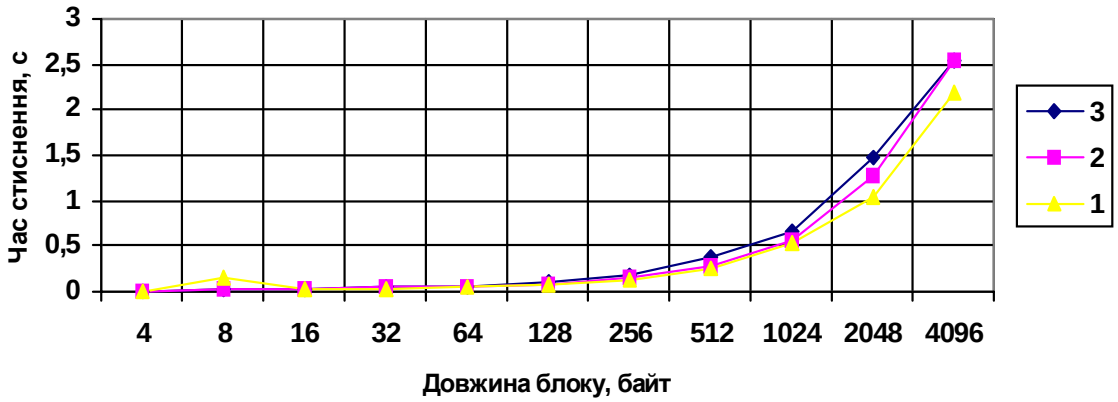

Рис. 1. Залежність часу стиснення від розміру, стисливого блока

Графік 1 - відповідає вимірювальної інформації. Тут час саме мінімальне, це зв'язаний з однотипною інформацією, для котрої кодова таблиця утворюється не дуже великою, що скорочує час пошуку відповідного коду. Для інших даних, графічної інформація графік 2 , текстовий і комбінованої графік 3, час збільшується зі збільшенням стисливого блока. Оцінюючи максимальну пропускну спроможність алгоритму для кожного з процесорів. Для AMD-2400 масив із 4096 символів кодується 1000 разів за $T_{\max} = 285,72$ із, отже, час опрацювання одного символу $t_c = 285,72/(4096 \times 1000) = 0,000697$ с. Звідси пропускна спроможність алгоритму $R_{\max} = 1/t_c = 7303$. 75 символів/с.

Для процесора *Pentium*-2000 $T_{\max} = 27,45$ с, $t_c = 27,45/(4096 \times 1000) = 6,70$ мкс, $R_{\max} = 147656,73$ символів/с.

Швидкодія і час стиснення методом LZW. На рисунку 2 подана залежність часу стиснення для LZW методу від розміру блока.

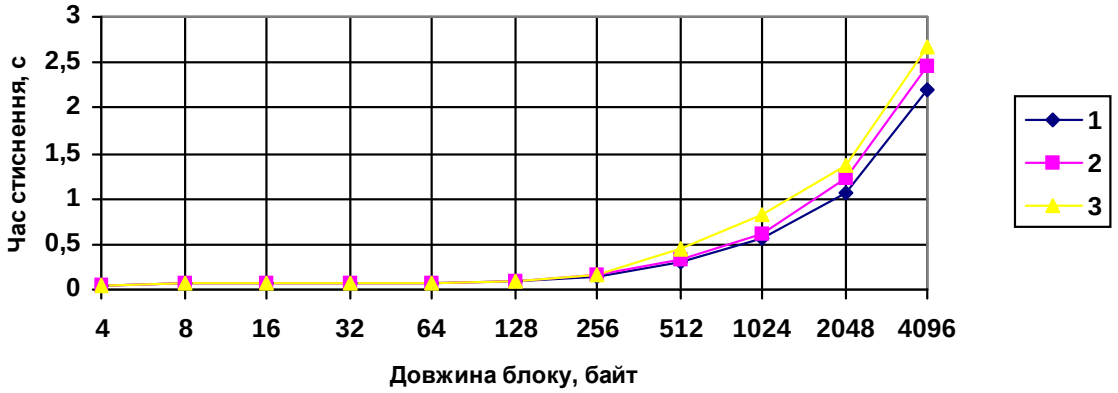

Рис.2. Залежність часу стиснення для LZW методу від розміру блока

Як видно з рисунка найменший час стиску має вимірювальна інформація (графік 1). Графік 2 відповідає текстової інформації, 3- графічна і комбінована інформація. Стиск блока розміром 1024 байт ще дає припустимий час необхідне для стиску в каналах зв'язку. Понад 1024 байти час стисливої інформації збільшується до секунд, що приводить до затримки і що неприпустимо в системах реального часу

Оцінюючи максимальну пропускну спроможність алгоритму для кожного з процесорів. Для AMD-2400 масив із 4096 символів кодується 1000 разів за $T_{\max} = 271,52$ із, отже, час опрацювання одного символу $t_c = 271,25/(4096 \times 1000) = 0,000697$ с. Звідси пропускна спроможність алгоритму $R_{\max} = 1/t_c = 14635,75$ символів/с.

Для процесора *Pentium*-2000 $T_{\max} = 23,57$ с, $t_c = 27,45/(4096\times1000) = 6,72$ мкс, $R_{\max} = 147656,73$ символів/с.

Швидкодія і час стиску методом Vitter. На рисунку 3 подана залежність часу стиску для методу Vitter від розміру блока. Даний метод стиску дає найменший час стиску з розглянутих алгоритмів, як видно з малюнка. На графіку 1 показана залежність для вимірювальної інформації. Мінімальний час стиску пояснюється тим, що процедура, що робить побудову дерева, виконується за постійний час не залежно від розміру стисливого блока. Графік 2 відповідає графічної інформації. Час стиску для текстової і комбінованої інформації показано на графіку 3. З графіка видно, що стиск відбувається за однаковий час, а деякі стрибки пояснюються впливом зовнішніх впливів, хоч і робилася оцінка часу в циклі 1000 разів.

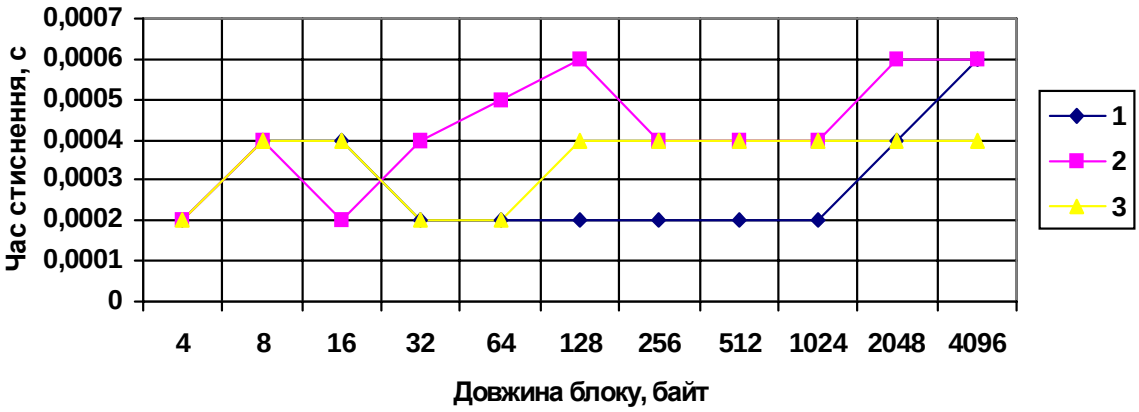

Рис.3. Залежність часу стиску для методу Vitter від розміру блока

Оцінюючи максимальну пропускну спроможність алгоритму для кожного з процесорів. Для AMD-2400 масив із 4096 символів кодується 1000 разів за $T_{\max} = 0,44$ із, отже, час опрацювання одного символу $t_c = 0,44/(4096\times1000) = 1,075$ мкс. Звідси пропускна спроможність алгоритму $R_{\max} = 1/t_c = 9309091,3$ 3 символів/с.

Для процесора *Pentium*-2000 $T_{\max} = 0,036$ с, $t_c = 0,036/(4096\times1000) = 0,000117$ мкс, $R_{\max} = 8,5449\times10^9$ символів/с.

Швидкодія і час стиску матричним методом. На рисунку 4 подана залежність часу стиску для матричного методу від розміру блока.

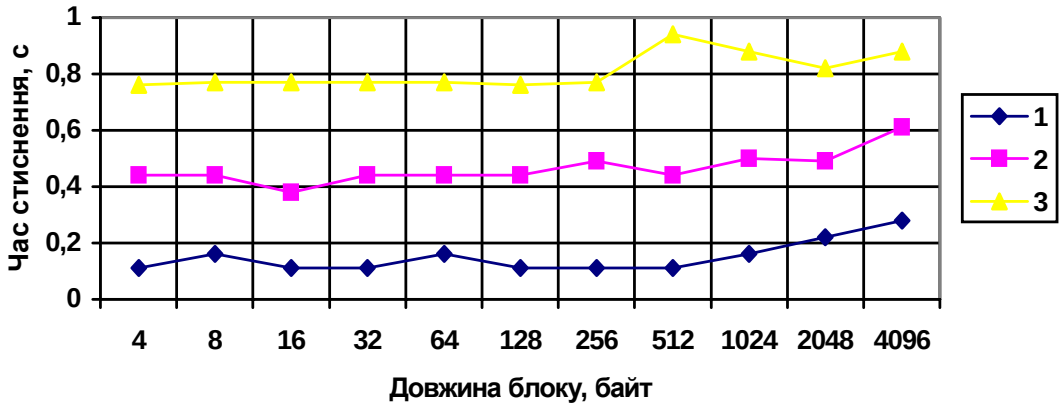

Рис. 4. Залежність часу стиску для матричного методу від розміру блока

Матричний метод, як видно з рисунку, виконує стиск практично однаковий час, але для різних потоків по різному. Для вимірювальної інформації (графік 3) час найбільше, це пояснюється тим, що пошук і адаптація до даного виду інформації відбувається найбільший час. На графіку 2 показана залежність для текстової і графічної інформації. Час тут менше ніж у попередньому випадку, зв'язане це з тим, що стиск не відбувається і незжата інформація видається на вихід. Також ця залежність спостерігається і для комбінованої інформації (графік 1).

Оцінюючи максимальну пропускну спроможність алгоритму для кожного з процесорів. Для AMD-2400 масив із 4096 символів кодується 1000 разів за $T_{\max} = 46{,}8$ с, отже, час опрацювання одного символу $t_c = 46{,}8/\left(4096 \times 1000\right) = 0{,}00001147$ с. Звідси пропускна спроможність алгоритму $R_{\max} = 1/t_c = 87521{,}36$ символів/с.

Для процесора *Pentium*-2000 $T_{\max} = 4{,}2$ с, $t_c = 4{,}2/\left(4096 \times 1000\right) = 0{,}000001024$ із, $R_{\max}$ 975238,74 символів/с.

Висновок. Таким чином в роботі розглянуті запропоновані каскадні способи стиснення при обробці вимірювальної, текстової, графічної та комбінованої інформації. Одержані дані про швидкодію всіх запропонованих способів каскадного стиснення дають можливість зробити правильний вибір способу стиснення для використання його в автоматизованій системі управління.

ЛІТЕРАТУРА

1. Кохманюк Д. Сжатие данных: как это делается / Кохманюк Д. // Index Pro. - 1992. - №1. - С. 18-29; 1993. - №2. - С.30-49.

2. Кричевский Р.Е. Сжатие и поиск информации / Р.Е. Кричевський. - М.: Радио и связь, 1989. – 168 с.

3. Чернега В.С. Сжатие информации в компьютерных сетях / В.С. Чернеча. – Севастополь: СевГТУ, 1997. - 214 с.

4. Ziv J., Lempel A. Compression of individual sequences via variable-rate coding. // IEEE Trans. On Inform. Theory. - 1978. - Vol.24, №5. - P. 530-536.

Надійшла: 04.01.2013

Рецензент: д.т.н., проф. Лєнков С.В.