УДК 621.391+512.64

Р.В. Скуратовський

МЕТОД ШВИДКОГО ТАЙМЕРНОГО КОДУВАННЯ

A statistical-oriented method of data compression using non-traditional timer encryption. Made the necessary estimates of compression. Proposed method of multiple system of labels for data compression and decompression method of collection of texts in one area of knowledge. He also uses a system of multiple labels for a set of texts, which are combined in solid archive. As shown, the joint unzipping is more effective on the number of operations as a result – on time. Allowed a virtual software implementation. Basis for a fundamentally new method of the timer marks on archiving is first proposed the principle of using a set of timer labels for prefixes and suffixes of words this labels a Ьфву are belo Ьфву ngs to another alphabet, but in a more general case for arbitrary subgroups of words that have a relatively large size. First time here implemented the principle of self-learning system, which is used in the next step after finding the first prefix label and receipt of accurate data. The probabilistic-statistical analysis are maden and evaluation of the effectiveness of the method. This method is used in conjunction with fast timers counters, that have patents on inventions in Ukraine and Russia.

Вступ

Задача швидкого стиснення текстових даних з нижньою межею стиснення, меншою, ніж ентропія даної мови чи даного тексту, є досі не розв'язаною. Можливим підходом до розв'язання цієї задачі є застосування не символьного, а блочного кодування.

Можливостями таймерного кодування займалася плеяда науковців на чолі з В.Ф. Бардаченком [1]. Вони виявили можливість таймерного маскування інформації [1], побудови корпоративної системи захисту інформації [2], шифрування [3–5]. Головною ж перевагою таймерного кодування є можливість кодування зі стисненням інформації. Цим питанням займалися В.Ф. Бардаченко і Є.О. Осадчий.

Всім добре відомо, що інколи при архівуванні даних на результат необхідно чекати годину чи дві. Якщо обсяг інформації перевищує 1 Гб, то методи стиснення (LamplZiv та ін.), які застосовує RAR, будуть працювати не менше однієї години при роботі на ноутбуку. Складнісна оцінка останнього методу – $O(n\log n)$. Важливою характеристикою також є об'єм стиснутої інформації. Тому цільовою функцією може бути така:

$$F = k_1 x + k_2 y \to \min \ ,$$

де y – час, x – об'єм (адитивна форма), $x \le Q_1$, $y \le t_1$.

Постановка задачі

Мета роботи – досягти найбільшого стиснення інформації з найменшими витратами часу за допомогою блочного кодування з використанням таймерних міток. Головні цілі – довести, що таке стиснення тексту можна зробити за поліноміальний час, а також знайти цей час. Також необхідно показати, що запропоноване впорядкування символів для роботи текстового генератора є оптимальним у класі послідовних генераторів тексту.

Вихідні положення

Граничною величиною стиснення довільного текстового файлу будемо вважати 8 біт – це пам'ять, достатня для однієї мітки [5]. Практично при таймерному кодуванні маємо лінійний ріст об'єму виділеної пам'яті залежно від кількості міток. Загалом викладення матеріалу ведеться у термінах попередніх робіт [6–8]. Наш метод стиску можна застосувати після стиснення тексту вже відомим архіватором, який використовує методи символьного кодування. Ми доведемо, що складнісна оцінка процесу стиснення тексту з n символів – це величина $3(\text{length}\,|A|)\cdot\left(\left[\frac{n}{2}\right]+1\right)n$, що, на відміну від попередніх методів таймерного кодування, набагато менше.

Обмеження. Об'єднання текстів, що відповідають розкодованим міткам, однозначно визначає весь текст, швидкість розархівування більша, ніж s_0. Для порівняння межі стискання нагадаємо, що при символьному кодуванні з однозначним декодуванням, тобто в префіксному кодуванні, існує код із найменшою довжиною коду символу

$$L(c,\xi):\ H(\xi) \le L(c,\xi) \le H(\xi)+1,$$

де $H(\xi)$ – ентропія джерела, тобто його нижня межа стиснення; $H(\xi) = \sum_{i=1}^{n} p_i \log_2\left(\frac{1}{p_i}\right)$, або середньо очікувана кількість інформації, яку несе один символ. Зрозуміло, що зі зростанням довжини тексту T довжина коду, наприклад *арифметичного*, буде лінійно зростати. Запропонований метод, у випадку використання однієї мітки, застосовує пам'ять, необхідну для мітки і для зберігання алфавітного впорядкування символів відкритого тексту. При цьому довжина коду зростає не більше, ніж лінійно, зі збільшенням довжини тексту, а коли використовується система міток з періодом 3 байти, то навіть після досягнення нижньої межі стиснення для символьного кодування за Шенноном можна досягти стиснення щонайбільше у 24 рази після застосування довільного з архіваторів. Нагадаємо, що нижня межа стиснення для 32 символьного алфавіту становить 5 біт на символ (а з урахуванням нерівномірності розподілу літер і взаємозалежностей у послідовності літер це 2 біти). Саме тому в статті вперше запропоновано нові методи таймерного кодування, доведено їх ефективність, розроблено відповідне математичне забезпечення для його оптимізації.

Основні математичні поняття і властивості досліджуваного методу

Символи тексту мають відносні частоти

$$f_s,\ s \in A,$$

де A – алфавіт символів з T, за якими ми їх впорядковуємо за спаданням частоти. Нагадаємо, що генератор генерує символи в заданому спадному порядку послідовно по розрядах. Час і результат роботи однозначно визначаються таймерною міткою.

***Означення* 1**. Генератор текстів, статистично орієнтований (Gs), – це генератор, який реалізує необхідний ряд розподілу частот ρ символів $n_i < n_j$.

***Означення* 2**. Генератор текстів з алфавітним впорядкуваням (Gp) – це генератор, який послідовно реалізує тексти, генеруючи символи в алфавітному порядку, тобто не враховуючи частоти появи цих символів у певному виді текстів.

Множина міток є 1-зсувом B [10], а на множині блоків визначимо n-зсув A. Відображення $\varphi: A \to B$ є блочним кодуванням. Текстовий генератор генерує послідовно всі тексти довжиною k, розглядаючи кожен із них в *окремому вікні* (*блоці довжиною k*). Якщо після генерування одного з текстів на керуючий вхід не прийшов сигнал призупинки, то він починає генерувати всі тексти довжиною k+1 і т.д. Коли цей процес закінчиться, що може відбутися тільки з надходженням сигналу на призупинку від таймерного генератора, що діє згідно з відповідною цьому блоку таймерною міткою, ми отримуємо однозначно визначений символом з Y блок тексту довжиною m. Таким чином, маємо *бієктивне* відображення $\varphi^{-1}: B \to A$, де Y – алфавіт простору зсуву B. Тому для скорочення перебору текстів при розархівації даних у код стиснутого тексту C доцільно включити одразу і довжину початкового тексту. Це кодування не є символьним кодуванням.

***Означення* 3**. Таймерна мітка відповідає необхідному стану текстового генератора. В загальному випадку таймерний генератор – це пристрій, що реалізує відлік (інкрементацію) у вибраній системі числення (СЧ) з постійними частотою і дискретизацією. Як СЧ для таймерного генератора може бути вибрана і часова система відліку, таймерні мітки набувають значення з дискретної множини з однаковими відстанями $a_1 a_N a_N \ldots a_N$ між точками. Ці відстані обернено пропорційно залежать від швидкості роботи Gs.

***Означення* 4.** Номером символу $s \in A$ при заданому впорядкуванні ρ є його порядковий номер $n_s \in \mathbb{N}$ у цьому впорядкуванні, яке здійснюється для статистичного генератора Gs згідно з імовірностями, характерними для символів вибраного тексту. Зауважимо, що вагою символу буде саме його номер n_i у частотному впорядкуванні, такому, що якщо $f_i > f_j$, то $n_j > n_i$.

***Означення* 5**. Вагою слова W чи тексту $T = a_1 a_2 a_3 \ldots a_n$ назвемо величину

$$Wh(T) = \sum_{i=1}^{n} n_i |A|^{n-i}, \qquad (1)$$

де n_i – номер літери, яка стоїть на i-му місці тексту в заданому для генератора впорядкуванні, n – довжина тексту T, $|A| = N - a_1 a_N a_N \ldots a_N$ – потужність множини символів.

Оскільки Gs генерує тексти послідовно навіть коли знає відразу, яка довжина шуканого тексту, то процес генерації можна описати так: поставивши неправильно 1-шу літеру, генератор вимушений перебирати на наступних позиціях після неї всі літери з A – множини символів, доки він не дійде до величини часу більшої, ніж таймерна мітка для T. Наприклад, якщо закодовано число $T = 1567$, а генератор поставив 1-ою літеру 2 (що відповідає впорядкуванню 2, 3, 4, 5, 6, 7, 8, 9, 0, 1), то він буде перебирати усі цифри, спочатку збільшивши символ в 2-му розряді, потім усі комбінації в наступних розрядах. Так він згенерує $|A|^{n-1} = 10^3$ комбінацій, дійшовши до тексту $a_1 a_N a_N a_N$, де a_N – останній символ з алфавіту A в заданому впорядкуванні. Потім повернеться до 1-ої позиції, де замінить символ 2 на наступний за порядком символ 3 (який теж хибний щодо закодованого числа) і знову почне генерувати символи на наступних позиціях. Так буде, доки він не дійде в першому розряді до 1. Таким чином, у кожному розряді від 2 до 4-го було використано від 9 до 10 цифр, а у 1-му перебрано всі 10 цифр. Тому верхня границя перебраних комбінацій при неправильно вибраному a_1 буде $n_1 \cdot 10^3$ для даного T. Якщо неправильно вибрано a_2, то це $n_2 \cdot 10^2$ комбінацій і т.д. Тоді вага становить

$$Wh(1567) = n_1 \cdot 10^3 + n_2 \cdot 10^2 + n_3 \cdot 10 + n_4,$$

де $n_1 = 10,\ n_2 = 4,\ n_3 = 5,\ n_4 = 6$, а при оптимальному впорядкуванні $n_1 = 1, n_2 = 2,\ n_3 = 3,\ n_4 = 4$.

Оціночна величина перебору у випадку неправильно поставленого a_1 становить $n_1 \cdot |A|^{n-1}$. В цьому випадку величина $|A|^{n-1}$ є вагомістю позиції першого символу чи першого розряду. В загальному випадку величина $|A|^{n-i}$ рівна вазі i-го символу.

Таким чином, $Wh(T)$ – це кількість перебраних гіперслів при вибраному впорядкуванні літер, після якої слідує шукане T.

Основні результати

***Властивість* 1**. Знаючи ваги символів, можна швидко обчислити вагу всього тексту.

Дійсно, оскільки текст відомо, то маємо символьне впорядкуванням ρ з вагами n_i символів з T. Тому можна скористатись формулою $Wh(T) = \sum_{i=1}^{n} n_i |A|^{n-i}$.

Дослідимо деякі властивості таймерних міток тексту. Нехай з однаковою швидкістю і фіксованим впорядкуванням генерується текст. Під Gp розуміємо генератор, який діє не залежно від розподілу ймовірностей даного тексту, а генерує символи в детермінованому алфавітному порядку.

***Властивість* 2**. За двійковим записом числа однозначно встановлюється таймерна мітка цього числа, причому для цього не потрібне послідовне генерування всіх попередників даного тексту при вибраному порядку.

Дійсно таймерну мітку даного двійкового числа a, отже, і відповідного йому тексту, можна отримати з формули

$$t = \frac{N_2(a)}{V}, \qquad (2)$$

де $N_2(a)$ – двійкове число, що відповідає коду (вазі тексту при даному порядку генерування символів) даного тексту, V – швидкість (частота) роботи генератора. $|N_2(a)| = Wh_p(T)$, де $Wh_p(T)$ – складність породження (вага) за (1) тексту T при певному лінійному впорядкуванні. Для статистично орієнтованого генератора з символьним впорядкуванням ρ це

$$t(\mu_0) = \frac{Wh_s(T)}{V}, \qquad (3)$$

де $Wh_s(T)$ – складність породження тексту T при певному лінійному статистичному впорядкуванні. Знаючи вагу тексту, можна розділити текст на k частин, для кожної з них обчислити свою вагу і порахувати $t(\mu_i)$ – величину таймерної мітки $1 \le i \le k$. Таким чином, отримано систему таймерних міток $y = t(\mu_\Sigma)$ за допомогою обчислення суми ряду, а не генераторного перебору множини гіперслів згідно з отриманим лексикографічним впорядкуванням за ρ.

Послідовність величин $|A|^n$ обчислюється рекурсивно і має часову складність $\text{length}(|A|^2) \times \times \left(\left[\frac{n}{2}\right] + 1\right) = 2(\text{length}\,|A|) \cdot \left(\left[\frac{n}{2}\right] + 1\right) = 2(\log_2 A) \times$

$\times\left(\left[\frac{n}{2}\right]+1\right)$ [9], а довжини символів з A обмежені одним байтом. Тому точний загальний час отримуємо, врахувавши $n=|T|$ множень на ваги n_i. В оцінці часової складності це відобразиться так:

$$3(\text{length}\,|A|)\cdot\left(\left[\frac{n}{2}\right]+1\right)n.$$

Таким чином, отримаємо час роботи – таймерну мітку генератора без процесу послідовного перебору текстів (який вимагає експоненційного часу від n) для отримання мітки.

Об'єм стиснутої інформації обчислюється з урахуванням суми міток, які застосовані до кожного з блоків тексту. Кожному блоку тексту, який представлений двійковим числом $N_2(a)$, відповідає таймерна мітка t, об'єм якої $Q(t)$, де $Q(t)$ – кількість біт у двійковому числі t. Тоді об'єм після стиснення всього тексту становить $Q(T)=\sum_{i=1}^{k}Q(t_i)$, де a_i – це i-й символ алфавіту з порядком ρ. Отже, при довжині мітки i-го блоку $\text{length}\,(t_i)=l_i$ маємо об'єм після стиснення $Q(t)=\sum_{i=1}^{k}l_i$ біт.

Нагадаємо, що вірогідність появи слова T – це кумулятивна ймовірність

$$P(T)=p_1p_2\ldots p_n,$$

де p_i – ймовірність появи i-ої літери в тексті з даної галузі знань. Самі тексти можна впорядкувати лексикографічно.

Тоді середньо статистична вага тексту рівна математичному сподіванню від усіх ваг слів:

$$M(Wh(W))=\sum_{i=1}^{k}P(W_i)W_i, \qquad (4)$$

де k – кількість слів у тексті.

Нехай $T=(a_1\ldots a_n)$ – випадковий текст із заданим розподілом $p(a_i=s_{i_j})=f_j,\ j\in(1,\ldots,N)$, де f_j – частота появи j-го символу, $\sum_{j=1}^{N}f_j=1$, $f_j\ge 0$.

Нехай $N=|A|,\ n=|T|$, тоді текст $T=(a_1\ldots a_n)$ – випадкова величина. $T=s_{i_1}\ldots s_{i_n}$, $i_j\in\{1,\ldots,N\},1\le j\le n$, – це конкретне значення, яке спостерігаємо.

Теорема. Математичне сподівання величини складності відновлення тексту генератором Gs менше, ніж ця складність для Gp.

Обчислимо математичне сподівання відновлення тексту генератором Gs:

$$E_S(\rho(\sigma))=\sum_{i=1}^{N}P(s_i=a_1)\cdot(Wh(s_i)),$$

де $Wh(s_i)$ – кількість операцій, які необхідно зробити до появи на першому місці символу $s_i\in A$, тобто необхідно виконати умову $a_1=s_1$. Тут символ на першому місці в тексті позначимо a_1. Тоді $E_{\#}(\rho(A),Wh(T))=\sum_{i=1}^{N}P(s_i=a_1)\times$ $\times(Wh(s_i))=\sum_{i=1}^{N}f_i(i)\,i$ – математичне сподівання кількості операцій при довільному детермінованому лінійному порядку (наприклад, алфавітному, тобто для Gp)

$$\sum_{i=1}^{N}f_{\sigma(i)}\cdot\sigma(i)\cdot((N-1)^n+\ldots)\le\sum_{i=1}^{N}f_i\cdot i\cdot((N-1)^n+\ldots),$$

$f_i\ge f_{\sigma(i)}$ при $i<\sigma(i)$.

$$E_S(\rho(\sigma))=\sum_{i=1}^{N}P(s_i=a_1)\cdot(Wh(s_i))=\sum_{i=1}^{N}f_{\sigma(i)}\cdot\sigma(i),$$

де $\rho(\sigma)$ – перестановка символів при порядку ρ. Оцінимо $\sum_{i=1}^{N}f_{\sigma(i)}\cdot\sigma(i)\cdot((N-1)^n+\ldots)$ зверху як

$$\sum_{i=1}^{N}f_{\sigma(i)}\cdot\sigma(i)\cdot((N-1)^n+\ldots)\le\sum_{i=1}^{N}f_i\cdot(i-1)M+\tilde{M}=$$

$$=\sum_{i=1}^{N}f_i\cdot(i-1)M+\tilde{M}\sum_{1}^{n}f_i=\ldots,$$ де M – оцінка для доданків меншого степеня, і порівняємо $E_S(\rho(\sigma))$ з $E_{\#}(\rho(A),Wh(T))$. Маємо

$$E_{\#}(\rho(A),Wh(T))=$$

$$=\sum_{i=1}^{N}P(s_1=a_1)\cdot(Wh(s_i))=\sum_{i=1}^{N}f_i\cdot(i)\,i,$$

нехай $E_{\#}(Wh(s_{i_2}|s_{i_1}))$ – вага T за умови, що на другій позиції стоїть символ s_{i_2} за (4).

$E(T) = \sum_{i=1}^{n} Wh(s_1)P(a_1 = s_1)$ – очікувана вага тексту з даної галузі знань.

$$Wh(s_{i_1}) = (n_{i_1} - 1)(N)^{n-1} +$$

$$+ \sum_{i_2=1}^{N} P(a_2 = s_{i_2}) Wh(s_{i_2} | a_1 = s_{i_1}) = ... = , \quad (5)$$

де

$$Wh(s_{i_2} | a_1 = s_{i_1}) = (n_{i_2} - 1)(N)^{n-2} +$$

$$+ \sum_{i_3=1}^{N} P(a_3 = s_{i_3}) Wh(s_{i_3} | a_1 = s_{i_1}, a_2 = s_{i_2}) \quad (6)$$

є математичним сподіванням ваги для варіанта, коли вже перший символ поставлено правильно і генератор шукає правильний варіант для 2-го символу.

Зауважимо, що в правій частині маємо умовну вагу (умовне математичне сподівання) i_k-го символу, яка залежить від попередніх i символів. Тут використовуються рекурсивно задані функції.

$$Wh(s_{i_3} | a_1 = s_{i_1}, a_2 = s_{i_2})) = (n_{i_3} - 1)N^{n-3} +$$

$$+ \sum_{i_4=1}^{N} P(a_4 = s_{i_4}) Wh(s_{i_4} | a_1 = s_{i_1},$$

$$a_2 = s_{i_2}, a_3 = s_{i_3})| \quad (7)$$

і так далі. Останнім виразом є $Wh(s_{i_n} | a_1 = s_{i_1}, ..., a_{n-1} = s_{i_{n-1}})$ – умовне математичне сподівання від ваги залишку тексту.

Зрозуміло, що $Wh(s_{i_n} | a_1 = s_{i_1}, ..., a_{n-1} = s_{i_{n-1}}) = n_{i_n}$.

Тому

$$Wh(s_{i_{n-1}} | a_1 = s_{i_1}, ..., a_{n-2} = s_{i_{n-2}}) =$$

$$= (n_{i_{n-1}} - 1)N + \sum_{i_n=1}^{N} P(a_n = s_{i_n}) n_{i_n} =$$

$$= (n_{i_{n-1}} - 1)N + \sum_{i_n=1}^{N} f_{i_n} n_{i_n} = (n_{i_{n-1}} - 1)N + \sum_{i=1}^{n} f_i n_i .$$

Рухаючись далі у зворотному порядку, отримаємо

$$Wh(s_{i_{n-2}} | a_1 = s_{i_1}, ..., a_{n-3} = s_{i_{n-3}}) = (n_{i_{n-2}} - 1)N^2 +$$

$$+ \sum_{i_{n-1}=1}^{N} P(a_{n-1} = s_{i_{n-1}})((n_{i_{n-1}} - 1)N + \sum_{i=1}^{N} f_i n_i) =$$

$$= (n_{i_{n-2}} - 1)N^2 + \sum_{i_{n-1}=1}^{N} f_{i_{n-1}}((n_{i_{n-1}} - 1)N +$$

$$+ \sum_{i=1}^{N} f_i n_i) = (n_{i_{n-2}} - 1)N^2 + N\sum_{i_{n-1}=1}^{N} f_{i_{n-1}} n_{i_{n-1}} -$$

$$- N\sum_{i_{n-1}=1}^{N} f_{i_{n-1}} + \sum_{i_{n-1}=1}^{N} f_{i_{n-1}}(\sum_{i=1}^{N} f_i n_i) =$$

$$= (n_{i_{n-2}} - 1)N^2 + N\sum_{i=1}^{N} f_i n_i - N + \sum_{i=1}^{N} f_i n_i =$$

$$= (n_{i_{n-2}} - 1)N^2 + (N+1)\sum_{i=1}^{N} f_i n_i - N.$$

Послідовно згорнувши, матимемо

$$Wh(s_{i_{n-k}} | a_1 = s_{i_1}, ..., a_{n-k-1} = s_{i_{n-k-1}}) = (n_{i_{n-k}} - 1)N^k +$$

$$+ (N^{k-1} + N^{k-2} + ... + 1)\sum_{i=1}^{N} f_i n_i - N - N^2 - ... N^{k-1}.$$

З останньої рекурентної формули та (5)–(7) отримаємо, що

$$Wh(s_{i_2} | a_1 = s_{i_1}) = (n_{i_{n-2}} - 1)N^{n-2} + (N^{n-3} +$$

$$+ N^{n-2} + ... + 1)\sum_{i=1}^{N} f_i n_i - N - N^2 - ... - N^{n-3}$$

та, остаточно,

$$Wh(s_{i_1}) = (n_{i_1} - 1)N^{n-1} +$$

$$+ \sum_{i_2=1}^{N} P(a_2 = s_{i_2}) Wh(s_{i_2} | a_1 = s_{i_1}) = ... =$$

$$= (n_{i_1} - 1)N^{n-1} + \sum_{i_2=1}^{N} P(a_2 = s_{i_2})((n_{i_2} - 1)N^{n-2} +$$

$$+ \sum_{k=3}^{N} P(a_k = s_{i_k}) Wh(s_{i_k} | a_1 = s_{i_1}, a_2 = s_{i_2})) = ... =$$

$$= Wh(s_{i_1}) = (n_{i_1} - 1)N^{n-1} + (N^{n-2} +$$

$$+ N^{n-3} + ... + 1)\sum_{i=1}^{N} f_i n_i - N - N^2 - ... N^{n-2} .$$

Отже, середня вага всього тексту дорівнюватиме

$$E(Wh(T)) = \sum_{i_1=1}^{N} Wh(s_{i_1}) P(a_1 = s_{i_1}) =$$

$$= \sum_{i_1=1}^{N} ((n_{i_1} - 1)N^{n-1} + (N^{n-2} + N^{n-3} + ... + 1) \times$$

$$\times \sum_{i=1}^{N} f_i n_i - N - N^2 - ... N^{n-2}) f_{i_1} =$$

$$= (N^{n-1} + N^{n-2} + ... + 1)\sum_{i=1}^{N} f_i n_i - N -$$

$$- N^2 - ... - N^{n-1}.$$

Оскільки у виразі для середньої ваги одного символу $\sum_{i=1}^{N} f_i n_i$ у статистично-орієнтова-

ному порядку впорядкованим за зростанням частотам $f_1 \ge f_2 \ge ... \ge f_N$ відповідають впорядковані за спаданням ваги $n_1 \le n_2 \le ... \le n_N$, то згідно з нерівністю Чебишова, ця вага є найменшою серед усіх можливих впорядкувань.

Теорему доведено.

Отже, в процесі формування тексту генератор отримує його таймерну мітку з алфавіту Y (або одиницю у СЧ таймерного генератора і тому має мінімальний обсяг пам'яті [7]) для кожного варіанта тексту для порівняння згенерованого числа, яке, зрозуміло, відповідає певному тексту з міткою оригінального тексту (позначимо її μ_0). Мітка, зроблена для статистичного генератора μ_0 (3), відповідає меншому двійковому числу, ніж те, що відповідає мітці μ_1 для Gp (2), який діє у нестатистичному порядку. Це спричиняється тим, що літери, які мають більшу ймовірність появи, кодуються меншими двійковими числами, тобто мають менше значення n_i у величині $Wh(T) = \sum_{i=1}^{n} n_i |A|^{n-i}$, ніж при нумерації цих же літер під час роботи перебірного генератора. А тому символи текстів, що є більш притаманними даній галузі знань, теж отримають менші номери у зробленому статистичному впорядкуванні, і в одиничній системі числення [8] їм відповідають менші таймерні мітки:

$$t(\mu_0) = \frac{Wh(T)}{V}. \qquad (5)$$

А для перебірного генератора вага збігається з $N_2(a)$, тому $t(\mu_1) = \frac{N_2(a)}{V}$. Але $N_2(a) > Wh(T)$, бо для більшості значень i буде виконуватися $m_i > n_i$, тому маємо $t(\mu_0) < t(\mu_1)$.

Тепер для заархівування тексту, що рівносильно створенню його мітки чи системи міток, достатньо обчислити $t(\mu_0) = \frac{Wh(T)}{V}$ або для генератора Gp це $t(\mu_1) = \frac{N_2(a)}{V}$. Для цього необхідно зробити зчитування усіх символів з T, де $|T| = N$, і виконати $N-1$ множень на вагу розряду, тобто на $|A|^k, 0 \le k \le N-1$. Час на обчислення добутку чисел рівний добутку довжин цих чисел. Послідовність величин $|A|^n$ обчислюється рекурсивно: спочатку обчислюється A^2, а потім його кратні добутки і добуток з A, і має часову складність $\text{length}(|A|^2) \cdot \left(\left[\frac{n}{2}\right]+1\right) = (\text{length}|A|)^2 \cdot \left(\left[\frac{n}{2}\right]+1\right)$, а довжини символів з A обмежені одним байтом. Тому сучасний комп'ютер таке обчислення – визначення $t(\mu_0)$ – зробить за частки секунди, бо його складність – $O(n^2)$, на відміну від експоненційної залежності при старому методі архівування, що використовував посимвольне генерування тексту Gp, яке передбачало перебір великої кількості варіантів і залежало від частот символьного і таймерного генераторів, а порівняно з витратами часу класичного символьного кодування (наприклад, за методом LamplZiv (його складнісна оцінка $n \log n$) і допоміжним методом Burrowse-Wheller transformation (де використовують циклічні зсуви усіх підстрічок і їх зрізи), який використовує архіватор RAR) ці оцінки належать до одного класу.

Як відомо, для тексту з рівномірним розподілом ймовірностей символів (нехай їх 32, тоді $H(\zeta) = \sum_{i=1}^{32} p_i \log_2 p_i^{-1} = \sum_{i=1}^{32} 32^{-1} \log_2 2^5 = 5$) на один символ тексту потрібно як мінімум 5 біт. З урахуванням взаємозалежностей між символами, тобто умовних ймовірностей і того, що ймовірності символів розподілені не рівномірно, ентропія джерела (нижня межа стискання) значно зменшується (2 біти на символ з 32-літерного алфавіту за дослідженнями Шеннона). Якщо застосувати спочатку класичне символьне кодування, а потім таймерне кодування з відстанню 3 байти між мітками, то матимемо повторне стиснення в 24 рази. Справді, 3 байтам відповідає блок довжиною 24 біти, і замість них ми зберігаємо всього $Q(a)$ біт на мітку, що має менший об'єм [8].

Твердження. Якщо згідно з лексикографічним впорядкуванням слів виконується $w_1 < w_2$, то $Wh(w_1) < Wh(w_2)$ і $t(w_1) < t(w_2)$.

Доведення. Це випливає з того, що існує $\exists i : n_i < m_i,\ n_j = m_j,\ \forall j < i$, тому $Wh(w_1) < Wh(w_2)$. Оскільки символи з більшою вагою генеруються після символів з меншою вагою згідно зі статичним порядком символів, то $t(w_1) < t(w_2)$.

***Означення* 1.** Довжина вихідного потокового коду L_{out} – це сума довжин кодів слів (або

блоків слів) у новому алфавіті *Y*. Середня довжина обчислюється так:

$$L_{\text{out}} = \sum_{i=1}^{N} l(C(w_i))p(w_i),$$

де N – кількість блоків, на які розбито текст, кожному з них відповідає мітка – символ з *Y*.

***Означення* 2.** Довжина вхідного потокового коду – це сума довжин $L_{\text{in}} = \sum_{i=1}^{N} l(w_i)$, середня довжина – це $L_{\text{in}} = \sum_{i=1}^{N} l(w_i)p(w_i)$, де $l(w_i)$ – довжина *i*-го блоку тексту, $p(w_i)$ – ймовірність появи цього блоку тексту.

Формула для коефіцієнта стиснення дає результат $k = \dfrac{\sum_{i=1}^{N} l(C(w_i))p_i}{\sum_{i=1}^{N} l(w_i)p_i} = \dfrac{N \cdot p_i}{N \cdot m \cdot p_i} = \dfrac{1}{m}$, оскільки довжина мітки рівна 1 байту (при найкращому впорядкуванні символів). (Детальніше про довжину мітки, поданої в іншій кодифікації, – в [1, 6, 7].) Звідси випливає, що коефіцієнт стиснення при такому коді рівний довжині блоку w_i, який відображається в символ з нового алфавіту *Y*. Отже, при *n*-зсуві, тобто кодуванні блоками довжиною *m*, коефіцієнт стиснення рівний $\dfrac{1}{m}$, що порівняно з кодуванням алгоритмом LZW, в якому коефіцієнт стиснення в межах від $O\left(\dfrac{\ln n}{n}\right)$ до $O(1)$, де n – довжина тексту, значно менше, бо m можна збільшити аж до n. Величина n істотно залежить від швидкодії текстового генератора.

Висновки

Для досягнення найбільшого зменшення об'єму після стиснення всього тексту можна використовувати лише одну мітку. Її об'єм можна визначити, скориставшись відповідно СЧ, де це число є одиницею [7]. Серію нулів *s* можна умовно розбити на блоки *b* одного порядку з самою серією і запам'ятати їх кількість і довжину такого блоку, крім того, ще буде залишок від ділення *s* на $\text{length}(b)$. Довжину ж блоку $\text{length}(b)$ доцільно вибрати рівну числу 2 в деякому степені. Крім того, якщо у користувача, який буде розархівовувати, є впорядкування символів для даного типу текстів, то об'єм архівованих даних буде рівний об'єму запису однієї мітки. При цьому час на розгортання тексту буде дещо більшим, ніж при множинній системі міток, але його можна зменшити нарощенням частоти таймерного лічильника.

Головним результатом є те, що час стиску тексту тепер не залежить від частоти символьного генератора і таймерного лічильника та не визначається в результаті процесу послідовного статистично-орієнтованого перебору чи простого перебору. Він визначається як сума степеневого ряду з ваговими коефіцієнтами відповідного символьного впорядкування ρ. Це дає лише квадратичну залежність складності від довжини тексту *n*, а саме $3(\text{length}|A|) \cdot \left(\left[\dfrac{n}{2}\right] + 1\right) n$, що порівняно з експоненційною залежністю від *n* при методі послідовного статистичного перебору є принциповим проривом у величині складності за архівування цим методом. При цьому практично для сучасного комп'ютера з ресурсами ноутбука програмно вдалося реалізувати таймерні лічильники з частотою приблизно рівною половині тактової частоти процесора. Цей швидкісний метод може бути успішно використано для стиску SMS-повідомлень. При цьому не потрібно вбудовувати таймерний лічильник у мобільний телефон, бо можна скористатись системою супутникового зв'язку JPRS для з'єднання з еталонним атомним годинником і синхронізатором, який можна реалізувати на базі телефонної станції. А оскільки атомний годинник має більшу величину дискретизації часу, ніж таймерний лічильник, що можна реалізувати на базі ноутбука, то можна використати генератор текстів з більшою частотою, що також підвищує швидкість роботи.

Перспективою подальших досліджень є розвиток і впровадження нового принципу стиснення даних.

1. *Бардаченко В.Ф., Джунин Е.Н.* Анализ характеристик таймерных маскраторов информации // УСИМ. – 1994. – № 3. – С. 16.

2. *Бардаченко В.Ф., Кишинский С.И.* Построение корпоративной системы фондовой биржи с использовани-

ем таймерной технологий обработки и защиты информации // Там же. – 2000. – № 5. – С. 66.

3. *Бардаченко В.Ф., Колесицкий О.К.* Анализ современных средств аутентификации для систем защиты информации // Там же. – 2004. – № 3. – С. 81–92.
4. *Осадчий Є.О.* Підхід до покращення таймерних методів захисту інформації // Вісник ТАНГ. – 1999. – № 1. – С. 30–35.
5. *Осадчий Є., Осадчий О.* Таймерні методи та засоби захисту інформації // Тези доп. Міжнар. наук.-практ. конф. "Проблеми впровадження інформаційних технологій в економіці та бізнесі". – Ірпінь, 2000. – С. 354–355.
6. *Методи* та засоби захисту інформації в часі / Є. Осадчий, В. Курченко, Д. Грищенко, О. Осадчий // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – 2000. – Вип. 1. – С. 73–76.
7. *Осадчий Є.О.* Інформаційні трансформерні технології побудови машин та механізмів: Монографія. – К.: Наук. світ. – 2004. – 168 с.
8. *Таймерне* кодування в прогресивних інформаційних технологіях. Т. 3 / *Є.О. Осадчий, О.Є. Осадчий, В.В. Ткаченко, В.Д. Курченко* // Тез. допов. 12 Міжнар. конф. з автоматизації керування "Автоматика 2005". – К., 2005. –С. 55.
9. *N. Koblitz*, "Algebraic aspects of cryptography", in Algorithms and Computation in Mathematics, vol. 3. Berlin, Springer-Verlag, 2004, 207 pp.
10. *D. Lind and B. Marcus*, An introduction to symbolic dynamics and coding. Cambridge, UK: Cambridge University Press, 1995, 490 pp.

Рекомендована Радою
фізико-математичного факультету
НТУУ "КПІ"

Надійшла до редакції
12 травня 2011 року