С. 38–43. 22. Яцук В.О., Малачівський П.С. Методи підвищення точності вимірювань: підруч. – Львів : Бескид Біт, 2008. – 368 с. 23. Majewski J., Makachivwski P., Andrunyk W. Software for approximation of sensor transfer function and its sensitivity function // Pomiary, automatyka, kontrola (PAK). – 2010. – 56, N9. – S. 1107–1110. 24. Я. Чапля, П. С. Малачівський, М. І. Дзюбачик, Б. Р. Монцибович, В. А. Андруник. Пакет програм для неперервного і гладкого рівномірного сплайн-наближення високоточної низькотемпературної характеристики ("Апрокріо") // Свідоцтво про реєстрацію авт. права на твір № 20705 від 30.05.2007 / Державний департамент інтелектуальної власності МОНУ. - 80 с. 25. Корн Г., Корн Т. Справочник по математике для научных работников: [Пер. с англ]. – М.: Мир, 1997. – 831 с.

УДК 004.043

А.Я. Бомба[1], О.В. Шпортько[2], Л.В. Шпортько[3]
[1]Рівненський державний гуманітарний університет, кафедра інформатики та ПМ;
[2]Рівненський державний гуманітарний університет, кафедра економічної кібернетики;
[3]ДВНЗ "Рівненський коледж економіки та бізнесу"

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ АРИФМЕТИЧНОГО КОДУВАННЯ В ПРОЦЕСІ ПРОГРЕСУЮЧОГО ІЄРАРХІЧНОГО СТИСНЕННЯ ЗОБРАЖЕНЬ БЕЗ ВТРАТ

Запропоновано спосіб та відповідні алгоритми використання арифметичного кодування в процесі прогресуючого ієрархічного стиснення зображень без втрат, досліджено перспективи відокремленого кодування груп елементів з найбільшою ймовірністю та моделювання відносних частот елементів після застосування предикторів. Сукупна реалізація запропонованих підходів замість кодування Хафмана дає змогу, наприклад, без застосування контекстно-залежних алгоритмів зменшити коефіцієнти стиснення зображень набору ACT у середньому на 2,13 %.

Ключові слова: безвтратне прогресуюче ієрархічне стиснення зображень, арифметичне кодування.

A method and proper algorithms of the use of the arithmetic encoding in the process of progressing hierarchical compression of images without losses are offered in the article, the prospects of the separated encoding of groups of elements with most probability and design of relative frequencies of elements after application of predictors are explored. The combined realization of the suggested approaches instead of Huffman's encoding, for example, enables to decrease aspect of images of the set of ACT ratios, on average, by 2.13 % without the application of context-depended algorithms.

Key words: progressing hierarchical compression of images without losses, arithmetic coding.

Вступ

Сьогодні опрацювання яскравостей пікселів зображень у популярних графічних форматах, які виконують стиснення без втрат, найчастіше здійснюється послідовно по рядках згори донизу, а у кожному рядку – поспіль зліва направо [1]. Як наслідок, вивести стиснуте зображення у цих форматах можливо лише після декодування всіх пікселів, а декомпресія фотокарток чи малюнків з мільйонами пікселів при такому способі обходу може тривати декілька секунд незалежно від розміру області та роздільної здатності пристрою виведення.

Для прискорення виведення великих зображень у форматах компресії з втратами найчастіше застосовують прогресуюче (поступальне) ієрархічне опрацювання пікселів [2, с. 176]. В процесі застосування цього способу обходу пікселі зображення обробляють пошарово, збільшуючи щоразу роздільну здатність, причому в процесі послідовного опрацювання даних чергового шару використовують дані попередніх шарів. Зображення з пікселів чергового шару фактично є зменшеною у декілька разів (найчастіше – у чотири) копією зображення з пікселів наступного шару, а останній шар збігається з вхідним зображенням. Тому під час прогресуючого ієрархічного декодування деталі зображення проявляються поступово. Зупинити таке декодування можливо вже після декомпресії шару з кількістю пікселів, не меншою від області виведення по кожній з осей, не очікуючи відтворення всіх пікселів зображення. Отже, **розробка контекстно-незалежних способів кодування для графічного формату компресії зображень без втрат з використанням принципів прогресуючого ієрархічного опрацювання є на сьогодні актуальним завданням**.

Аналіз останніх досліджень і публікацій.
Принципи стиснення зображень без втрат з використанням предикторів

Як відомо, стиснення зображень без втрат у графічних форматах найчастіше відбувається в три етапи: на першому яскравості пікселів перетворюються за допомогою предикторів [3]; на другому контекстно-залежне кодування зменшує надлишковості між подібними фрагментами; на третьому контекстно-незалежне кодування усуває надлишковості між переважаючими значеннями яскравостей компонентів [4]. Контекстно-залежне кодування може зменшувати коефіцієнт стиснення (відношення розмірів стиснутого до нестиснутого файлів зображення, надалі – КС) в декілька разів за рахунок подібних фрагментів. Але такі фрагменти рідко трапляються у фотореалістичних зображеннях, тому єдиним універсальним етапом стиснення зображень без втрат є контекстно-незалежне кодування. Основний принцип такого кодування: **довжина коду довільного елемента з більшою ймовірністю не повинна перевищувати довжину коду будь-якого елемента з меншою ймовірністю**. Цей принцип ґрунтується на фундаментальному положенні теорії інформації, згідно з яким для мінімізації довжини коду послідовності елемент s_i (в нашому випадку під елементом розуміють яскравість окремої компоненти кожного піксела) з ймовірністю появи $p(s_i)$ доцільно кодувати $-\log_2 p(s_i)$ бітами [5, с. 17]. Тому середня довжина коду елемента блоку після застосування будь-якого контекстно-незалежного алгоритму згідно з формулою of Shannon [2, с. 611] не може бути меншою за *ентропію джерела*

$$H = -\sum_i p(s_i) \times \log_2 p(s_i). \quad (1)$$

Ентропія джерела зменшується зі збільшенням нерівномірності розподілу ймовірностей (частот) між елементами [3]. За нашими підрахунками, застосування контекстно-незалежного алгоритму в середньому зменшує КС зображень на 33 %.

Підвищити ефективність цього кодування в процесі стиснення зображень без втрат намагаються за допомогою предикторів, які під час обходу прогнозують значення яскравості кожної компоненти чергового піксела (наприклад, для найпоширеніших 24-бітних зображень – це яскравості червоної, зеленої та синьої компоненти, записані цілими числами в окремих байтах), використовуючи значення яскравостей тих самих компонентів опрацьованих раніше суміжних пікселів, адже дані яскравості характеризуються найбільшим ступенем кореляції [6, с. 675]. У процесі використання цього підходу обчислюють і надалі кодують відхилення Δ_{uv} значення яскравості чергової компоненти піксела F_{uv} від прогнозованого обраним предиктором значення $predict_{uv}$, тобто

$$\Delta_{uv} = F_{uv} - predict_{uv} \quad (2)$$

(u та v пробігають відповідно по всіх рядках та стовпцях компонентів пікселів зображення). Суміжні пікселі зображень найчастіше мають подібні кольори, а отже, і близькі значення яскравостей відповідних компонентів, тому значення прогнозу часто збігається зі значенням яскравості чергової компоненти, найчастіше – є близьким до цього значення і рідко – значно відрізняється від нього (рис. 1). Тобто більшість значень Δ_{uv} виявляються близькими до нуля. Тим

самим застосування предикторів найчастіше збільшує нерівномірність розподілу ймовірностей значень яскравостей компонентів і, як наслідок, зменшує ентропію (1), що дозволяє зменшити середню довжину коду контекстно-незалежного алгоритму.

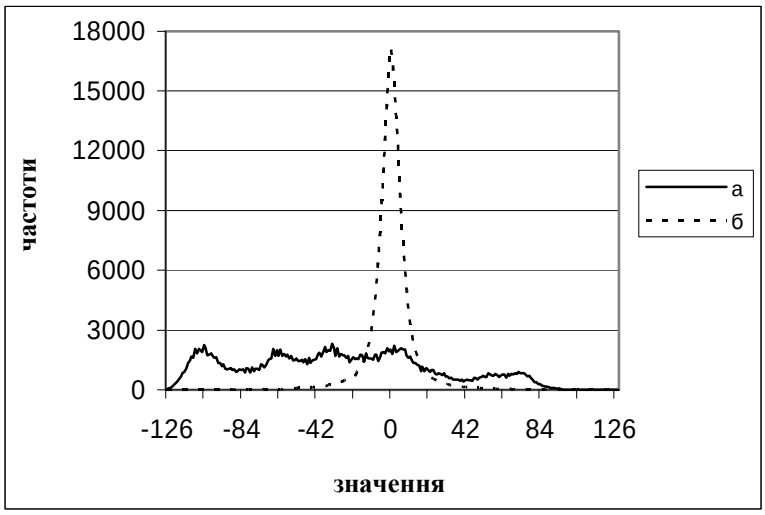

Рис. 1. Розподіл частот значень зеленої компоненти зображення Lena.bmp:
а – до застосування предиктора (H=7,59 bpb);
б – після застосування Left-предиктора (H=5,34 bpb)

На практиці сьогодні найчастіше використовують два альтернативні контекстно-незалежні методи: кодування Хафмана та арифметичне кодування.

За кодування Хафмана (HUFF) кожному елементу залежно від його ймовірності (частоти появи) відповідає фіксований префіксний код [2]. Виконують це кодування найчастіше в такій послідовності: підраховують ймовірності (частоти) окремих елементів; впорядковують елементи за спаданням ймовірностей; ітеративно поєднують до отримання одного елемента два елементи з найменшими ймовірностями (найчастіше – останні в утвореному списку), дописуючи при цьому першому з них код *0*, а другому – *1*; сумують ймовірності обраних елементів для обчислення ймовірності утвореного елемента і вставляють цей елемент у відсортований список ймовірностей; утворюють коди HUFF, записуючи сформовані коди у зворотному порядку – від вершини до кожного елемента.

Середня довжина коду HUFF збігається з ентропією джерела лише тоді, коли для всіх елементів s_i довжини їх оптимальних кодів $-\log_2 p(s_i)$ цілі [4]. Крім цього, довжина коду HUFF навіть найімовірнішого елемента не може бути меншою ніж один біт.

Арифметичне стиснення (ARIC), на відміну від кодування HUFF, ставить у відповідність кожному черговому елементу інтервал з довжиною, пропорційною його ймовірності (частоті) [2]. При цьому з початкового інтервалу (найчастіше [0; 1)) обирають і надалі розглядають підінтервал, що відповідає першому елементу потоку. Цей підінтервал знову розбивають пропорційно частотам окремих елементів і з нього обирають підінтервал, що відповідає другому елементу потоку (див., наприклад, рис. 2, на якому підінтервали впорядковані за зростанням значень елементів).

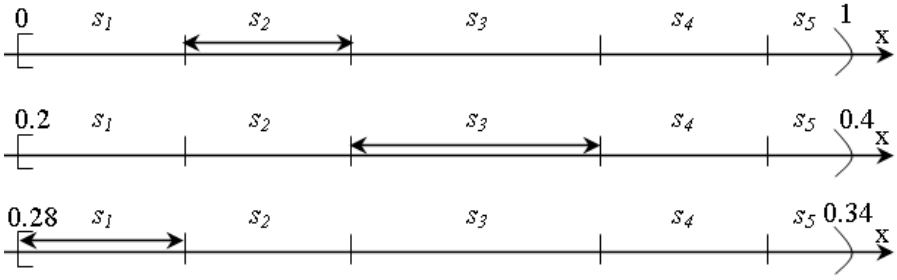

Рис. 2. Арифметичне кодування трьох перших елементів
послідовності "36 38 35 35 40 36 40 38 45 38"

Вибір підінтервалів триває аналогічно аж до закінчення елементів потоку. Тобто, кінцева довжина обраного підінтервала дорівнює добутку ймовірностей всіх елементів, а його початок залежить від порядку слідування цих елементів в потоці [4]. Результатом ARIC є будь-яке число з останнього отриманого підінтервалу. Зважаючи на скінченність розрядності чисел, для запису результуючого числа щоразу відслідковують перші значущі цифри меж інтервалів і у випадку їх співпадання записують ці цифри у вихідний потік та вилучають з подальшого розгляду. Декодування наступного елемента ARIC виконують шляхом визначення відповідного чергового підінтервала, якому належить зчитане число.

З опису принципів ARIC випливає, що для виконання такого арифметичного стиснення кодеру і, тим більше, декодеру мають бути відомі ймовірності (частоти) окремих елементів. Сьогодні широко використовуються дві стратегії формування цих частот: статична та адаптивна. У випадку статичної стратегії частоти окремих елементів передаються декодеру у стиснутих даних явно. При використанні адаптивної стратегії інтервали елементів формуються синхронно кодером та декодером в процесі опрацювання даних. Адаптивна стратегія забезпечує, як правило, кращі КС (оскільки не потребує зберігання у стиснутих даних частот окремих елементів та враховує нерівномірності розподілу поточної послідовності елементів, а не потоку загалом [4]), але суттєво сповільнює роботу декомпресора, бо вимагає перерахунку ймовірностей елементів у процесі декодування. Формати графічних файлів мають забезпечувати, насамперед, швидке декодування, тому для реалізації прогресуючого стиснення зображень без втрат ми використали статичну стратегію формування інтервалів елементів.

Враховуючи те, що термін дії основних патентів на ARIC вже минув і таке стиснення точніше кодує окремі елементи, а довжина арифметичного коду найімовірнішого елемента може бути меншою за один біт [4], дослідимо особливості застосування арифметичного кодування в процесі прогресуючого ієрархічного стиснення зображень без втрат з використанням предикторів.

Вплив ієрархічних предикторів на ентропію пікселів зображень

У роботі [7] нами запропоновано дієву схему обходу пікселів та відповідні предиктори, які реалізують прогресуюче ієрархічне стиснення зображень без втрат, коли на першому шарі пікселі зображення опрацьовуються послідовно, починаючи з першого, по рядках згори донизу, а у кожному рядку – підряд зліва направо з кроком $h_1 = 2^k$, де k визначається з умови

$$k = \left\lfloor \log_2 \left(\frac{\max\left(\min\left(height; width\right); 16\right) - 1}{15} \right) \right\rfloor,$$

де $height$ – кількість рядків, $width$ – кількість стовпців пікселів зображення. Цей крок забезпечує опрацювання на першому шарі принаймні 16 пікселів (за наявності) по кожній з осей (як у піктограмах), якщо зображення має не менші розміри. На наступних шарах ($l = \overline{2, k+1}$) проміжні пікселі зображення обробляються в два проходи: на першому послідовно опрацьовуються ті з них, які містяться на перетині діагоналей квадратів з вершинами у суміжних пікселах попереднього шару з кроком $h_l = 2^{k+2-l}$ як по рядках, так і по стовпцях, а на другому необроблені пікселі обходяться між суміжними пікселами попереднього шару і першого проходу з тим самим кроком по стовпцях і з удвічі зменшеним – по рядках (рис. 3).

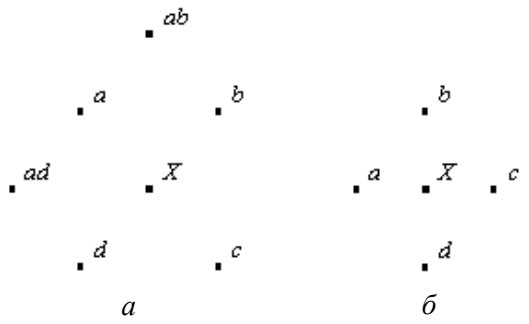

Рис. 3. Схеми розміщення суміжних опрацьованих елементів для елемента X на шарах, починаючи з другого: а – для першого проходу; б – для другого проходу

Порівняємо ентропію (1) пікселів 24-бітних зображень стандартного тестового набору Archive Comparison Test (ACT, табл. 1) після застосування комбінацій запропонованих у [7] ієрархічних предикторів на різних шарах і проходах цього обходу (табл. 2). Цей набір містить як синтезовані (№№ 1 (з шумами), 2, 7), так і фотореалістичні (всі інші) зображення. Завантажити їх TIFF-версії можна, наприклад, з http://www.compression.ca/act/act-files.html чи з http://www.compression.ru/arctest/act/act-tif.htm. Ентропія яскравостей компонентів пікселів у цих таблицях і надалі подається у bpb, тобто у кількості бітів, які витрачаються на кодування одного байта зображення.

Таблиця 1

Характеристика зображень набору ACT

№ файла	Назва файла	Розмір, Кб	Розміри, пікселів	Ентропія, bpb
1	Clegg.bmp	2101	814 x 880	7.54
2	Frymire.bmp	3622	1118 x 1105	4.65
3	Lena.bmp	769	512 x 512	7.75
4	Monarch.bmp	1153	768 x 512	7.50
5	Peppers.bmp	769	512 x 512	7.66
6	Sail.bmp	1153	768 x 512	7.32
7	Serrano.bmp	1464	629 x 794	5.99
8	Tulips.bmp	1153	768 x 512	7.66

Таблиця 2

Ентропія яскравостей компонентів пікселів зображень набору ACT після застосування комбінацій ієрархічних предикторів, bpb

Шар	Про-хід	№ файла								Середня ентропія
		1	2	3	4	5	6	7	8	
2	1	7.35	4.61	6.90	6.70	7.03	7.19	5.99	7.28	6.63
	2	7.01	4.60	6.14	6.42	6.04	6.96	5.22	7.04	6.18
3	1	7.01	4.48	6.35	6.43	6.48	7.15	5.23	6.86	6.25
	2	6.66	4.57	5.80	6.01	5.69	6.84	4.27	6.50	5.79
k+1	1	2.51	1.57	4.80	4.11	4.38	5.60	1.42	4.53	3.62
	2	0.23	1.19	4.46	3.61	3.79	4.91	0.96	3.88	2.88
Разом		2.26	1.69	4.73	4.04	4.19	5.41	1.53	4.42	3.53

Бачимо, що застосування комбінацій прогресуючих предикторів на останньому шарі забезпечує істотно меншу ентропію від використання цих самих комбінацій на початкових шарах, оскільки рівень кореляції яскравостей компонентів суміжних опрацьованих пікселів зі збільшенням номера шару чи проходу зростає через зменшення відстані між ними. На перших шарах для громіздких зображень іноді навіть доводиться відмовлятися від використання предикторів, оскільки їх застосування не зменшує ентропію, адже піксели, які при цьому використовуються для прогнозування, суттєво віддалені один від одного і, як наслідок, мають низький рівень кореляції яскравостей відповідних компонентів.

Реалізація арифметичного кодування в процесі прогресуючого стиснення зображень без втрат

Для реалізації ARIC в процесі прогресуючого ієрархічного стиснення зображень без втрат ми модифікували програми з CD до [1], які виконують кодування/декодування зображень для формату PNG, забезпечивши зберігання спочатку номерів застосованих предикторів, а після них – яскравостей пікселів згідно із ієрархічним обходом та застосували в цих програмах арифметичний range-кодер Е. Шелвіна [4] замість кодування HUFF, оскільки він виконує зчитування/запис байтів, а не бітів даних і внаслідок цього значно прискорює виконання кодування/декодування. Цей кодер

для кожного елемента використовує інтервал з цілочисловою довжиною, пропорційною його ймовірності. З метою зберігання довжини інтервалу для кожного елемента у заголовку блоку стиснутих даних замість довжини коду HUFF нами записується **кількість бітів для зберігання відповідної довжини інтервалу** в загальному інтервалі [0; 32768), а після заголовка – **двійковий запис цієї довжини без першого біта** (який завжди дорівнює одиниці). З метою прискорення декодування, для уникнення циклу пошуку початку інтервала чергового елемента, після зчитування заголовка блоку стиснутих даних доцільно створити байтовий масив, у якому для кожного значення загального інтервалу зберегти номер елемента, що йому відповідає. Застосування такого 32 Кб масиву дозволяє прискорити декодування в середньому на 46 %.

Проаналізуємо результати застосування описаного способу арифметичного кодування порівняно з кодуванням HUFF для зображень набору ACT (1–2 рядки після заголовка табл. 3–5). Тестування проводили за допомогою згаданих програм та їх модифікацій, у яких додатково розмір блоків даних був збільшений до 64 Кб та було відкинуто допоміжні текстові блоки. У табл. 3 показником компресії файлів обрано КС, виражений, як і ентропія у табл. 2, в bpb.

Таблиця 3

Коефіцієнти стиснення файлів зображень набору ACT після застосування різних варіантів програм ієрархічної компресії, bpb

Варіант програми	№ файла								Середній КС
	1	2	3	4	5	6	7	8	
З кодуванням HUFF	2.68	2.02	4.78	4.06	4.24	5.47	1.92	4.47	3.70
З послідовним ARIC	2.21	1.61	4.75	4.04	4.22	5.45	1.48	4.45	3.53
З ARIC по проходах	2.20	1.61	4.76	4.04	4.22	5.45	1.48	4.45	3.53
З ARIC по проходах з виділенням груп трійок нулів	2.24	1.58	4.76	4.03	4.22	5.44	1.50	4.44	3.53

Таблиця 4

Час кодування файлів зображень набору ACT різними варіантами програм ієрархічної компресії на комп'ютері з частотою 300 МГц, с

Варіант програми	№ файла								Середній час
	1	2	3	4	5	6	7	8	
З кодуванням HUFF	14.66	25.82	5.71	8.24	5.55	8.73	10.06	8.30	10.88
З послідовним ARIC	13.68	23.63	5.27	7.80	5.22	8.12	9.56	7.86	10.14
З ARIC по проходах	13.46	22.79	5.27	7.69	5.22	7.91	9.28	7.69	9.91
З ARIC по проходах з виділенням груп трійок нулів	13.07	21.81	5.22	7.64	5.16	7.97	8.85	7.69	9.68

Таблиця 5

Час декодування файлів зображень набору ACT різними варіантами програм ієрархічної компресії на комп'ютері з частотою 300 МГц, с

Варіант програми	№ файла								Середній час
	1	2	3	4	5	6	7	8	
З кодуванням HUFF	7.36	11.52	3.02	4.23	2.91	4.83	4.78	4.34	5.37
З послідовним ARIC	6.21	10.32	2.41	3.41	2.31	3.68	4.28	3.52	4.52
З ARIC по проходах	6.26	10.22	2.48	3.40	2.36	3.68	4.23	3.52	4.52
З ARIC по проходах з виділенням груп трійок нулів	6.05	9.88	2.41	3.46	2.36	3.74	4.17	3.52	4.45

Як видно з цих таблиць, використання ARIC замість кодування HUFF в програмах ієрархічного стиснення зображень без втрат без контекстно-залежних алгоритмів призводить до покращення КС зображень набору ACT в середньому на 2.13 % переважно за рахунок синтезованих зображень, для яких ARIC, на відміну від кодування HUFF, здатне забезпечити на останніх шарах довжину коду найімовірнішого елемента менше одного біта. При цьому скорочення часу кодування

спостерігається в середньому на 6.8 %, а декодування – на 15.83 %. Менші значення КС окремих синтезованих зображень (табл. 3) від ентропії після застосування предикторів (табл. 2) пояснюються можливістю арифметичного кодування ефективно стискувати найімовірніші елементи (нулі), які йдуть в кінці блоку даних (зокрема, в кінці останнього проходу) [4].

Розбиття даних зображень на однорідні блоки перед застосуванням арифметичного кодування

Як вже зазначалося, для однозначного декодування блоку ARIC на його початку має міститися заголовок, який дає змогу кожному елементу поставити в однозначну відповідність інтервал з довжиною, пропорційною його ймовірності, оскільки ймовірності елементів у різних блоках можуть відрізнятися. На перший погляд здається, що стиснення даних зображення в єдиний блок покращує КС, адже тоді буде використано лише один заголовок блоку замість багатьох. З іншого боку, як буде показано далі, розбиття даних на блоки, як правило, не збільшує (найчастіше – зменшує) загальну довжину арифметичного коду (без врахування довжин їх заголовків). Доведемо це математично.

Оскільки середня довжина ARIC близька до ентропії [2], то загальна довжина блоку таких кодів для послідовності елементів наближено дорівнює сумі довжин їх ентропійних кодів, тобто *довжині ентропійного коду послідовності*. Виведемо формулу для обчислення цієї довжини. Нехай кожен з елементів s_i зустрічається N_i разів у послідовності довжини $N = \sum_i N_i$. Згідно із статистичним означенням ймовірності, $p(s_i) = N_i / N$, тому загальна довжина ентропійного коду послідовності, враховуючи (1), наближається до значення

$$L_H = N \times H = N \log(N) - \sum_i N_i \log(N_i). \quad (3)$$

Твердження. Довжина ентропійного коду (до якої наближається довжина коду ARIC) поєднання двох послідовностей елементів є не меншою від суми довжин ентропійних кодів цих послідовностей.

Доведення. Нехай у першій послідовності довжини $N^{'}$ елемент s_i зустрічається $N_i^{'}$ разів, а в другій послідовності довжини $N^{''}$ цей самий елемент зустрічається $N_i^{''}$ разів. Тоді довжина ентропійного коду таких елементів у коді першої послідовності становитиме $L_i^{'} = N_i^{'} \log\left(\frac{N^{'}}{N_i^{'}}\right)$, у другій становитиме $L_i^{''} = N_i^{''} \log\left(\frac{N^{''}}{N_i^{''}}\right)$, а у послідовності, створеній в результаті поєднання цих двох послідовностей, довжина ентропійного коду елементів s_i дорівнюватиме $L_i = \left(N_i^{'} + N_i^{''}\right) \log\left(\frac{N^{'} + N^{''}}{N_i^{'} + N_i^{''}}\right)$.

Обчислимо приріст довжини ентропійного коду цих елементів внаслідок поєднання двох послідовностей:

$$dL_i = L_i - L_i^{'} - L_i^{''} = \left(N_i^{'} + N_i^{''}\right) \times \log\left(\frac{N^{'} + N^{''}}{N_i^{'} + N_i^{''}}\right) - N_i^{'} \log\left(\frac{N^{'}}{N_i^{'}}\right) - N_i^{''} \log\left(\frac{N^{''}}{N_i^{''}}\right) \quad (4)$$

Дослідивши функцію (4) на екстремум за змінними $N_i^{'}$ та $N_i^{''}$, доходимо висновку, що вона досягає мінімального значення, яке дорівнює нулю, при $\frac{N_i^{'}}{N^{'}} = \frac{N_i^{''}}{N^{''}}$. У всіх інших випадках приріст довжини ентропійного коду елементів s_i додатний. Оскільки приріст довжини ентропійного коду кожного з елементів внаслідок поєднання двох послідовностей невід'ємний, то й загальна довжина ентропійного коду внаслідок такого поєднання не зменшується.

Розглянемо, наприклад, залежність приросту довжини ентропійного коду, який утворюється внаслідок поєднання двох послідовностей по 100 елементів, від частот елемента s_i у цих вхідних послідовностях (рис. 4).

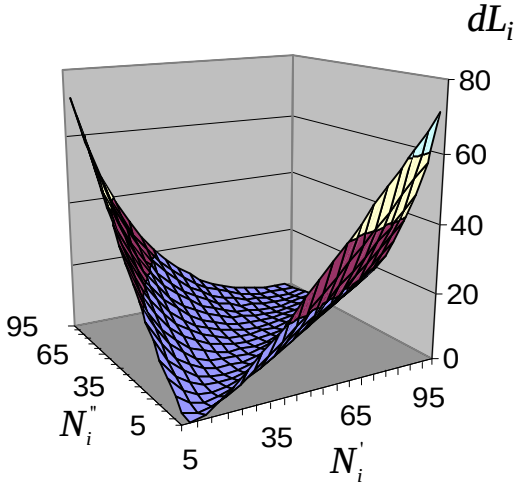

Рис. 4. Залежність приросту довжини ентропійного коду поєднання двох вхідних послідовностей від частот елемента s_i у цих вхідних послідовностях

Як видно з цього рисунка, приріст довжини ентропійного коду дорівнює нулю лише тоді, коли відносні частоти (ймовірності) елемента у двох вхідних послідовностях однакові. У всіх інших випадках цей приріст додатний, причому він зростає зі збільшенням різниці між відносними частотами елемента у вхідних послідовностях. Отже, **довжина ентропійного коду незначно зростає лише внаслідок поєднання блоків з близькими відносними частотами (ймовірностями) окремих елементів**.

Розглянемо тепер послідовність елементів “1, 2, 1, 1, 2, 1, 2, 2, 2, 1, 3, 4, 3, 4, 4, 3, 3, 3, 4, 4”, яка містить чотири різні елементи “1”, “2”, “3”, “4” з однаковою частотою 5 та ймовірністю 0.25. Якщо згенерувати арифметичні коди для всієї послідовності як єдиного блоку, то будь-якому елементу відповідатиме код довжиною 2 біти і вся послідовність буде закодована в 40 бітів. Якщо ж розбити цю послідовність на два однакові блоки, то в кожному з них зустрінуться лише по два різні елементи з ймовірностями 0.5, і будь-якому з яких відповідатиме код довжиною 1 біт, а дані всієї послідовності закодуються в 20 бітів.

Отже, створення малих блоків ARIC призводить до зайвих витрат пам'яті для зберігання заголовків, а великих – до неврахування змін характеристик даних. **Тому стиснуті дані перед ARIC доцільно розбивати на блоки, які мають різні відносні частоти (ймовірності) окремих елементів.** Як відомо, після застосування предикторів розподіли частот значень компонентів окремих блоків зображення близькі до розподілу Лапласа [2] і тому найбільший вплив на довжину ентропійного коду стиснутих даних (3) мають елементи, близькі до 0 (див. рис. 1б), а **мірою відмінності частот окремих елементів блоків може бути ентропія** (1). Ентропія даних зображень в процесі ієрархічного обходу пікселів значно змінюється (зменшується) при переході між шарами та проходами (табл. 2). Тому **дані різних проходів мають стискуватися у різні блоки ARIC**.

Розбиття даних зображень на блоки в наших програмних реалізаціях ієрархічного стиснення без втрат виконується автоматично, оскільки максимальний розмір блоку даних обмежується 64 Кб, що вже частково виконує цю вимогу. Тому проаналізуємо зміни показників стиснення внаслідок додаткового розбиття блоків по проходах (2, 3 рядки табл. 3–5). Як і слід було чекати, створення цих додаткових блоків зменшує розмір арифметичних кодів (зображення № 1), хоча й створює їх додаткові заголовки (зображення № 3), тому й значно не впливає на середній КС набору, хоча й прискорює кодування на 2.27 %, що й свідчить про доцільність такого розбиття.

Про доцільність виділення послідовностей однакових значень перед застосуванням арифметичного кодування

Проаналізуємо доцільність виділення груп однакових елементів перед застосуванням ARIC у процесі стиснення зображень без втрат, адже після застосування предикторів виділення, наприклад, пар чи трійок нулів суттєво зменшує довжину послідовності, але чи зменшує воно довжину відповідного ентропійного коду? Дослідимо це питання детальніше.

Перенумеруємо значення (елементи s_i) байтів відхилень (2) яскравостей компонентів пікселів від значень, прогнозованих предикторами в межах [-127 .. 128] (див. рис. 1, *б*), тобто покладемо $s_i = i$ та $P(s_i) = P_i$. В процесі ARIC можливе кодування пари однакових елементів s_i, якщо ці елементи йдуть підряд (ймовірність такої події становить P_i^2) і перед цією парою міститься інший елемент (з ймовірністю $1 - P_i$), або перед цією послідовністю міститься така сама пара елементів, перед якою – інший елемент і т.д. Застосовуючи формулу суми членів нескінченно спадної геометричної прогресії, отримаємо

$$P(s_i s_i) = P_i^2 (1 - P_i) + P_i^4 (1 - P_i) + \ldots = P_i^2 \times (1 - P_i)\left(1 + P_i^2 + P_i^4 + \ldots\right) = \frac{P_i^2 (1 - P_i)}{1 - P_i^2}. \quad (5)$$

Аналогічно, можна показати, що

$$P(s_i s_i s_i) = \frac{P_i^3 (1 - P_i)}{1 - P_i^3}, \quad (6)$$

а ймовірність появи послідовності *n* однакових елементів s_i становить $\frac{P_i^n (1 - P_i)}{1 - P_i^n}$. Очевидно, що найбільшу ймовірність після застосування предикторів у загальному випадку матиме елемент s_0 (див. рис. 1, *б*), і тому максимальне зменшення довжини послідовності відбувається від виділення груп саме таких елементів. Якщо, наприклад $P_0 = \frac{1}{2}$, то, використовуючи (5), $P(s_0 s_0) = \frac{1}{6}$, тобто дві третини таких елементів увійдуть в пари, а одна третина залишиться у вигляді непоєднаних елементів. Зрозуміло, що чим більша ймовірність елемента s_0, тим більше пар зустрічатиметься, але значна частина елементів все одно залишиться відокремленими, і ARIC доведеться кодувати пари $s_0 s_0$ як новий елемент. Виділення груп однакових елементів зменшує ймовірність непоєднаних таких елементів і тому збільшує довжину їх коду та зменшує довжину коду інших елементів внаслідок скорочення послідовності і додає довжину коду утворених груп, тобто нерівномірність розподілу загалом зменшується.

На практиці, після застосування предикторів до трикомпонентних зображень, трійки s_0 можуть зустрічатися з більшою ймовірністю, ніж згідно із (6), оскільки, якщо предиктор точно спрогнозував яскравості двох компонентів піксела, то, найімовірніше, він так само точно спрогнозує значення і третьої компоненти. Проаналізуємо, наприклад, результати ARIC зображень набору ACT після застосування предикторів і виділення трійок нульових елементів (останній рядок табл. 3–5). Бачимо, що хоча й для чотирьох зображень набору KC незначно зменшився, але для двох синтезованих зображень він збільшився і в середньому по набору залишився незмінним. Тобто загальної тенденції до зменшення KC від виділення трійок послідовних нулів не спостерігається. Це саме стосується і груп однакових елементів інших довжин. Отже, виділяти послідовності однакових значень елементів перед ARIC загалом недоцільно.

Моделювання ймовірностей елементів у блоках арифметичних кодів

На завершення опишемо варіанти моделювання значень ймовірностей елементів блоків ARIC для зменшення розмірів їх заголовків. Як зазначалося вище, розподіли ймовірностей елементів

після застосування предикторів близькі до симетричних (див. рис. 1, *б*) і найбільшу ймовірність мають елементи навколо нуля. Але ці розподіли, як правило, не симетричні, оскільки предиктори можуть мати систематичні відхилення [2] і не враховують повною мірою всі особливості перепадів яскравостей пікселів зображень. Тому ймовірності елементів “0”, “-1” та “1”, які найбільше впливають на нерівномірність розподілу, доцільно передавати у заголовках стиснутих блоків ARIC явно:

$$\hat{P}_0 = P_0, \hat{P}_{-1} = P_{-1}, \hat{P}_1 = P_1, \tag{7}$$

де $\hat{}$ вказує на прогнозоване значення ймовірності. Ймовірності решти елементів змоделюємо дискретними значеннями показникових функцій

$$\hat{P}_i = P_1 t^{i-1} \; (i = \overline{2,128}), \quad \hat{P}_i = P_{-1} t^{-i-1} \; (i = \overline{-127,-2}). \tag{8}$$

Параметр t з (8) визначимо, враховуючи, що елементи $s_i (i = \overline{-127,128})$ повністю вичерпують можливі значення байтів відхилень (2), тобто

$$\sum_{i=-127}^{128} \hat{P}_i = 1. \tag{9}$$

Підставивши (7) та (8) в (9) отримаємо, що

$$t = \frac{1 - \hat{P}_0 - \hat{P}_{-1} - \hat{P}_1}{1 - \hat{P}_0}. \tag{10}$$

Отже, використовуючи описаний спосіб моделювання значень ймовірностей у заголовку блоку ARIC можна зберігати лише три ймовірності (7), а решту значень розраховувати за (10) та (8). У цьому випадку для забезпечення однозначності декодування не тільки декодер, а й кодер мають використовувати змодельовані значення ймовірностей елементів. Зрозуміло, що моделювання значень ймовірностей доцільно застосовувати тоді, коли зменшення розміру заголовка блоку ARIC перевищує збільшення довжини ентропійного коду цього блоку

$$\hat{L}_H - L_H = -\sum_{i=-127}^{128} N_i \log_2 \left(\hat{P}_i \right) + \sum_{i=-127}^{128} N_i \log_2 \left(P_i \right) = \sum_{i=-127}^{128} N_i \log_2 \left(\frac{P_i}{\hat{P}_i} \right). \tag{11}$$

З (11) випливає, що моделювання значень ймовірностей елементів доцільно використовувати насамперед для невеликих блоків ARIC. Але, як показали експерименти, застосування такого способу моделювання зменшує розміри стиснутих зображень лише на десятки байтів, оскільки заголовки цих блоків невеликі. Наприклад, розмір зображення № 1 з набору ACT внаслідок такого моделювання значень ймовірностей зменшився на 49 байтів.

Для невеликих блоків ARIC ефективним може виявитися також моделювання значень ймовірностей рівномірним розподілом (замість (8))

$$\hat{P}_i = \frac{1 - \hat{P}_0 - \hat{P}_{-1} - \hat{P}_1}{253} \; (i = \overline{-127,-2}, \overline{2,128}), \tag{12}$$

яке, наприклад, зменшує розмір цього самого зображення на 129 байтів.

Висновки і перспективи подальших досліджень

1. У форматах графічних файлів байт-орієнтоване арифметичне кодування зі статичною стратегією формування інтервалів елементів є дієвою альтернативою кодуванню Хафмана. Для здійснення такого кодування, враховуючи структуру ARIC, у стиснутих блоках необхідно забезпечити відокремлене зберігання арифметичних кодів кожного розподілу.

2. Для реалізації статичної стратегії формування інтервалів елементів ARIC у форматах графічних файлів у заголовку кожного блоку стиснутих даних доцільно зберігати кількості бітів для запису довжин інтервалів, а після заголовка – двійкові коди цих довжин без першого біта.

3. Дані різних шарів та проходів ієрархічного обходу пікселів після застосування предикторів доцільно стискувати у різні блоки ARIC, оскільки вони мають різну ентропію. ARIC слід застосовувати до елементів безпосередньо, без виділення послідовностей однакових значень. Розміри невеликих блоків ARIC можна незначно зменшити, якщо зберігати в їхніх заголовках дані

лише найімовірніших елементів, а ймовірності решти елементів моделювати за допомогою показникової чи рівномірної функції.

4. Прискорити декодування арифметичних кодів зі статичним формуванням інтервалів більш ніж на 40 % дає змогу використання допоміжного масиву, в якому для кожного значення загального інтервалу зберігається номер елемента, що йому відповідає.

Надалі, з метою подальшого зменшення КС в процесі прогресуючого ієрархічного стиснення зображень без втрат, нами планується розробити способи попереднього зменшення ентропії та пристосувати контекстно-залежні методи компресії для цього способу обходу пікселів.

1. Миано Дж. Форматы и алгоритмы сжатия изображений в действии: учеб. пособ. / Дж. Миано. – М. : Триумф, 2003. – С. 249-318. – (Практика программирования). 2. Сэломон Д. Сжатие данных, изображений и звука / Д. Сэломон. – М.: Техносфера, 2006. – 368 с. – (Мир программирования: цифровая обработка сигналов). 3. Бредихин Д. Ю. Сжатие графики без потерь качества [Електронний ресурс] / Д. Ю. Бредихин. – 2004. – http: //www.compression.ru/ download/ articles/ i_lless/ bredikhin_2004_lossless_image_compression_ doc.rar. 4. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : ДИАЛОГ-МИФИ, 2003. – 384 с. 5. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1072 с. 6. Прэтт Э. Цифровая обработка изображений: Пер. с англ. / Э. Прэтт. – М.: Мир, 1982. – Кн. 2, 480 с., ил. 7. Шпортько О. В. Використання предикторів в процесі прогресуючого ієрархічного контекстно-незалежного стиснення зображень без втрат / О. В. Шпортько // Вісник Національного університету "Львівська політехніка". – 2013. – № 771. – С. 354–364. – (Серія: Комп'ютерні науки та інформаційні технології).

УДК 004.652

А.С. Василюк, Т.М. Басюк
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

ПІДСИСТЕМА ЗНИЩЕННЯ ФОРМУЛ АЛГОРИТМІВ

Описано означення процесів знищення формул алгоритмів. Наведено алґоритм комп'ютерного знищення формул алгоритмів. Синтезовано, мінімізовано, побудовано математичну модель і досліджено алґоритм знищення формул абстрактних алгоритмів.

Ключові слова: знищення, алгоритм, математична модель.

This article is about the determination of the process of deleting formulas of algorithms. The algorithm of computer deleting of formulas of algorithms was given. Synthesized, minimizing the mathematical model was synthesized, minimized and the algorithm of deleting of formulas of abstract algorithms was studied.

Key words: deleting, algorithms, mathematical model.

Вступ. Постановка проблеми

Відома [1,2] алгебра алгоритмів, яка має оригінальні операції, наприклад, секвентування, елімінування, паралелення та циклічні операції, котрі позначаються спеціальними знаками, яких немає серед відомих математичних знаків. Для набору та редагування формул абстрактних алгоритмів розроблено спеціалізовану комп'ютерну підсистему МОДАЛ [3], але нею не знищуються формули абстрактних алгоритмів.