УДК 004.043

А.Я. Бомба, О.В. Шпортько

Ентропійні способи вибору предиктора для рядка пікселів у форматі *PNG*

Предложены способы выбора предиктора для каждой строки пикселов и алгоритм разбиения изображения на блоки строк с помощью энтропии перед сжатием в формате *PNG*. Приведены фрагменты программ на языке С для оценки энтропии строк пикселов непосредственно и после применения контекстно-зависимого алгоритма. Описанные способы выбора предикторов строк позволяют уменьшить коэффициенты сжатия изображений набора *ACT* в формате *PNG*.

The methods are suggested of the choice of a predictor for every line of pixels and the algorithm of laying out of the image on the blocks of lines with the help of the entropy before a compression in the PNG format. The fragments of the programs in the C language for the estimation of the entropy of lines of pixels directly before and after the application of the context-dependent algorithm are presented. The described methods of the choice of predictors of lines make it possible to decrease the coefficients of the compression of a set of the ACT ratios in the PNG format.

Запропоновано способи вибору предиктора для кожного рядка пікселів та алгоритм розбиття зображення на блоки рядків за допомогою ентропії перед стисненням у форматі *PNG*. Наведено фрагменти програм мовою С для оцінки ентропії рядків пікселів безпосередньо та після застосування контекстно-залежного алгоритму. Описані способи вибору предикторів рядків дозволяють зменшити коефіцієнти стиснення зображень набору ACT у форматі *PNG*.

Вступ. На даний час формат *PNG* [1] є одним з основних для збереження зображень без втрат. В Інтернеті, наприклад, нараховується понад 15 млн сторінок, що містять зображення у цьому форматі. Популярності формату *PNG* сприяють, насамперед, прийнятні коефіцієнти стиснення (КС – відношення розміру стиснутих до нестиснутих даних) та висока швидкість декодування. Проблема зменшення розмірів файлів зображень у цьому форматі є актуальною і залишатиметься такою в найближчому майбутньому, оскільки навіть часткове її розв'язання дає змогу прискорити завантаження *PNG*-файлів з мережі та підвищити ефективність використання дискового простору.

База дослідження. Особливості етапів обробки даних зображень у форматі *PNG*

Пікселі зображення записуються у *PNG*-файли найчастіше по рядках зверху вниз, а у кожному рядку – послідовно зліва направо (як символи у текстах), формуючи тим самим вхідний потік. Перед кодуванням ці пікселі можуть бути попередньо оброблені предикторами. У *PNG*-файлах, що використовуються сьогодні, стиснуті дані зберігаються у відокремлених блоках згідно формату словникового стиснення *DEFLATE* [2]. Відповідно до цього формату, у стиснутих блоках містяться результати застосування до вхідного потоку алгоритму *LZH* [3], згідно з яким результати контекстно-залежного словникового алгоритму *LZ*77 [4] стискаються контекстно-незалежними кодами Хафмана [5]. Для дослідження взаємовпливу етапів обробки даних зображень у форматі *PNG* нагадаємо принципи їх виконання.

*Контекстно-залежний алгоритм LZ*77 базується на заміні у вихідному потоці послідовності чергових незакодованих неподільних елементів (літералів буфера) посиланням на аналогічну послідовність літералів, що починається серед закодованих елементів (у словнику), у вигляді пари чисел <*довжина*; *зміщення від закінчення словника*> [4]. За відсутності аналогічної послідовності літералів у словнику, перший літерал буфера переноситься у вихідний потік без змін. Після цього закодовані літерали переносяться з початку буфера в кінець словника і кодування продовжується аналогічно аж до закінчення літералів вхідного потоку. Наприклад, потік кодів байтів пікселів 36 38 35 35 36 38 35 36 36 38 35 38 36 38 35 35 28 в закодованому вигляді можна записати так: 36 38 35 35 <3; 4> 36 <3; 4> 38 <4; 12> 28.

Під час декодування кодів, отриманих за алгоритмом *LZ*77, окремі літерали копіюються у

Ключові слова: безвтратне стиснення зображень, статичні предиктори, формат графічних файлів *PNG*.

вихідний потік без змін. Пари ж <*довжина*; *зміщення*> декодуються шляхом послідовного копіювання з кінця вихідного потоку за вказаним зміщенням в кінець вихідного потоку необхідної кількості літералів. Природно, що алгоритм декодування має розрізняти окремі літерали та групи <*довжина; зміщення*>. Згідно з алгоритмом *LZH*, у форматі *DEFLATE* з цією метою базові значення довжин замін та окремі літерали кодуються разом. Після базових значень довжин міститься визначена форматом кількість бітів, що разом з базовим значенням визначають довжину заміни. Зміщення зберігається після відповідної довжини аналогічно – у вигляді базового значення та додаткових бітів.

В програмній реалізації для зменшення КС алгоритму *LZ*77 [6] забезпечено можливість виходу зі словника в буфер під час кодування повторів, запроваджено модифікований алгоритм аналізу альтернативних стиснутих блоків та застосовано аналіз кількостей додаткових бітів при записі зміщень.

Ідея використання *динамічних кодів Хафмана* [5], що найчастіше застосовуються після виконання алгоритму *LZ*77 для стиснення літералів і базових значень довжин, а також для відокремленого стиснення базових значень зміщень у кожному блоці стиснутих даних, полягає у заміні чисел з більшою частотою (тут і надалі – абсолютною) кодами меншої кількості бітів, ніж для чисел з меншою частотою.

Згідно теореми Шеннона, елемент s_i з ймовірністю появи $p(s_i)$ найвигідніше кодувати – $\log p(s_i)$ бітами (тут і надалі логарифм береться за основою 2). Тоді середня довжина коду елемента після застосування контекстно-незалежного алгоритму (такого, як кодування Хафмана) має наближатися до *ентропії джерела* [3]:

$$H = -\sum_i p(s_i) \times \log\left(p(s_i)\right). \qquad (1)$$

Як відомо, ентропія джерела зменшується при збільшенні нерівномірності розподілу ймовірностей між елементами [7].

Алгоритм генерування динамічних кодів Хафмана відомий ще з середини минулого століття і зменшити його КС не видається можливим. Можна лише намагатися після мінімізації ентропії окремих рядків розбити зображення на блоки рядків пікселів з близькими значеннями ентропії. Це дозволить відокремити і локалізувати рядки з високою ентропією (межі фрагментів зображення або значні перепади яскравостей пікселів), підвищивши тим самим нерівномірності розподілу в суміжних блоках, і тому зменшити загальний КС.

Зменшити ентропію при обробці зображень у форматі *PNG намагаються також за допомогою предикторів*, що прагнуть, використовуючи значення відомих суміжних елементів, спрогнозувати значення чергового елемента [3, 7]. Якщо піксел зображення характеризується декількома компонентами (наприклад, *R*, *G*, *B*), то предиктор кожної з них прогнозує значення згідно з відповідними компонентами сусідніх пікселів. У процесі використання цієї технології обчислюють і надалі кодують відхилення чергової компоненти від прогнозованого предиктором значення [3]. Тому, у загальному випадку, процес застосування предикторів до кожної компоненти піксела у вузлі (i, j) можна записати формулою

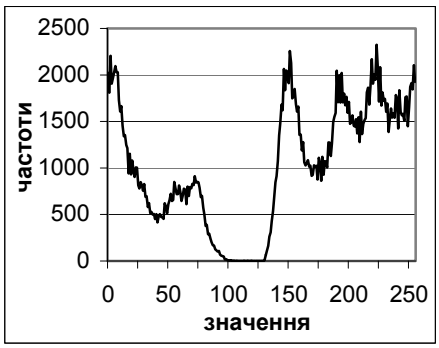
$$\Delta_{ij} = C_{ij} - predict_{ij}, \qquad (2)$$

де C_{ij} – значення компоненти до застосування предиктора, Δ_{ij} – значення компоненти після застосування предиктора, $predict_{ij}$ – значення предиктора, обчисленого для обраної компоненти.

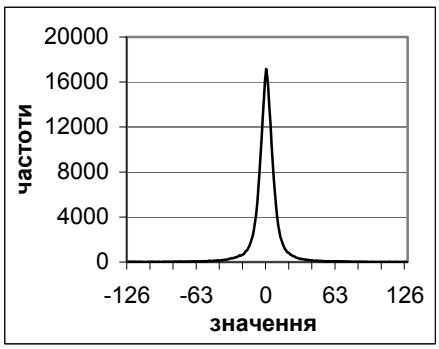
Проаналізуємо розподіл частот значень Δ_{ij}. Оскільки сусідні піксели зображення мають, як правило, близькі кольори і тому близькі значення відповідних компонент, то часто значення предиктора буде збігатися зі значенням чергового елемента, найчастіше – буде близьким до цього значення і зрідка – значно відрізнятиметься від нього. Тобто більшість значень Δ_{ij} будуть близькими до нуля. Такий перерозподіл частот значень (а отже, і ймовірностей) значно підвищує нерівномірність розподілу і тому зменшує ентропію джерела (згідно (1)), а отже, і довжину закодованої послідовності [7]. Таким чином, предиктори хоча й не виконують безпосереднього стиснення даних, але зменшують,

насамперед, КС контекстно-незалежного алгоритму. Приклад перерозподілу частот значень компонент після застосування предиктора наведено на рис. 1.

Крім цього, окремі предиктори можуть розбити існуючі або згенерувати нові повтори фрагментів зображення і цим вплинути на КС алгоритму *LZ*77, про що буде сказано далі.



а



б

Рис. 1. Розподіл частот значень зеленої компоненти даних перших 64 Кб зображення *Lena.bmp*: *а* – до застосування предиктора; *б* – після застосування *LeftPredict*

У стиснутих блоках формату *PNG* даним кожного рядка передує окремий байт, що визначає предиктор, який застосовується до компонент всіх його пікселів [8]. Сьогодні форматом передбачено п'ять можливих значень цього байта, що визначає чотири різні предиктори: 0 – дані рядка не обробляються предикторами; 1 – *LeftPredict*; 2 – *AbovePredict*; 3 – *AveragePredict*; 4 – *PaethPredict*. Для зручності опису цих предикторів введемо позначення яскравостей суміжних елементів для елемента *X* згідно схеми:

- *LeftAbove*
- *Above*
- *Left*
- *X*

Тоді предиктори формату *PNG*, згідно стандарту [1], мовою *C* запишемо так:

```
  char LeftPredict(char Left, char Above,
char LeftAbove)
   {return Left;}
  char AbovePredict(char Left, char Above,
char LeftAbove)
   {return Above;}
  char AveragePredict(char Left, char
Above, char LeftAbove)
   {return (Left+Above)/2;}
  char PaethPredict(char Left, char
Above, char LeftAbove)
   {int pp=Left+Above-LeftAbove;
    int pa,pb,pc;
    pa=abs(pp-Left);
    pb=abs(pp-Above);
    pc=abs(pp-LeftAbove);
    if (pa<=pb && pa<=pc) return Left;
    else if (pb<=pc) return Above;
         else return LeftAbove;}.
```

Відхилення між значеннями сусідніх елементів в зображеннях найчастіше зумовлені двома причинами: «сильними» коливаннями, обумовленими зображеними об'єктами – трендом, і слабкими фоновими коливаннями – шумом [3]. Перші три згаданих предиктори описують шумову модель і належать до лінійних статичних предикторів. Останній поєднує в собі трендову та шумову моделі і належить до нелінійних статичних предикторів.

Проблема вибору таких статичних предикторів для кожного рядка зображення, що дозволяють зменшити загальний КС зображення, залишалася *невирішеною* досі [8, с. 317]. Нижче подамо способи її вирішення за допомогою ентропії.

Вплив предикторів та їх комбінацій на стиснення зображень у форматі *PNG*. Оцінка ентропії рядків пікселів

Алгоритми, що використовуються у форматі *PNG* для стиснення даних, орієнтовані на зменшення різних видів надлишковості: якщо *LZ*77 усуває надлишковості між однаковими фрагментами даних, то кодування Хафмана орієнтоване на зменшення надлишковостей між переважаючими елементами розподілів. Алгоритм *LZ*77 у цьому форматі може стискувати дані максимум у 129, а кодування Хафмана – максимум у вісім разів. Саме тому алгоритм *LZ*77 і

використовується у форматі *PNG*, хоча він і зменшує нерівномірність розподілу між окремими елементами і може зменшити ефективність кодування Хафмана. З іншого боку, у природних зображеннях яскравості компонент сусідніх пікселів, як правило, близькі, але не однакові. Це робить малоефективним використання для них алгоритму *LZ*77, і тому кодування Хафмана у цьому випадку забезпечує хоча б задовільні КС. Ключову роль в процесі стиснення різних зображень і навіть окремих їх фрагментів можуть відігравати як алгоритм *LZ*77, так і кодування Хафмана. Використання різних предикторів (а також відмова від їх застосування) формату *PNG* неоднаково впливає на КС цих алгоритмів. Відповідно для кожного рядка зображення слід обирати той предиктор, який дозволить підсилити дію ключового алгоритму стиснення чергового фрагменту і допоможе тим самим забезпечити найменший загальний КС.

Дослідимо вплив різних варіантів предикторів на прикладі стиснення восьми різнотипних 24-бітних природних зображень з стандартного набору файлів *Archive Comparison Test* (*ACT*) у форматі *PNG*, характеристики яких наведено в табл. 1. Завантажити *TIFF*-версії файлів зображень цього набору можна, наприклад, з *http*: // *compression.ca/act/act-files.html*. Тестування проводилося за допомогою програми *CD* [7], у яку, крім вказаних раніше, було внесено такі модифікації: розмір блоків даних, збільшений до 64 Кб, відкинуто допоміжні текстові блоки та запроваджено поєднання рядків у блоки, що описано далі. Результати тестування наведено в табл. 2, 3. Швидкість декодування зображень майже не залежить від застосованих предикторів або їх комбінацій і складає в середньому для дискретно-тонових зображень 910 Кб/с, для штучних з перешкодами – 545 Кб/с, а для неперервно-тонових – 330 Кб/с. Тому середній час декодування всіх зображень набору складає лише 25 с.

Стиснення зображень у форматі *PNG* без застосування предикторів відбувається шляхом повторень та за наявності нерівномірності розподілу значень яскравості пікселів. Але в природних зображеннях аналогічні фрагменти та однакові значення яскравості компонент трапляються порівняно рідко, тому й таке стиснення призводить до високих КС (див. рис. 1,*а* та варіант *Без предикторів* у табл. 2, 3). Стиснення зображень без застосування предикторів ефективне, насамперед, для дискретно-тонових штучних зображень з високою роздільною здатністю. Це стиснення виконується найшвидше, оскільки не вимагає виконання додаткових обчислень значень предикторів.

Таблиця 1. Характеристика зображень набору ACT

№ файла	Назва файла	Розмір, Кб	Розміри зображення, пікселів	Унікальних кольорів серед пікселів, %	Особливості
1.	*Clegg.bmp*	2101	814x880	17,83	Неперервно-тонове, штучне, з шумами, декілька великих об'єктів
2.	*Frymire.bmp*	3622	1118x1105	0,29	Дискретно-тонове, штучне, один великий об'єкт
3.	*Lena.bmp*	769	512x512	56,56	Неперервно-тонове, природне, декілька великих об'єктів
4.	*Monarch.bmp*	1153	768x512	19,99	Неперервно-тонове, природне, один великий і багато малих об'єктів
5.	*Peppers.bmp*	769	512x512	42,47	Неперервно-тонове, природне, декілька великих об'єктів
6.	*Sail.bmp*	1153	768x512	19,26	Неперервно-тонове, природне, багато середніх об'єктів
7.	*Serrano.bmp*	1464	629x794	0,26	Дискретно-тонове, штучне, один великий фрагментований об'єкт
8.	*Tulips.bmp*	1153	768x512	30,07	Неперервно-тонове, природне, декілька великих об'єктів

Предиктор *LeftPredict* генерує горизонтальні прирости яскравості між компонентами сусідніх пікселів. Однакові прирости яскравості трапляються в природних зображеннях, як правило, частіше, ніж однакові фрагменти, що покращує КС алгоритму *LZ*77. Крім цього, застосування предиктора *LeftPredict* підвищує нерівномірність розподілу навколо нуля (рис. 1,*б*), зменшуючи тим самим КС кодування Хафмана. Саме тому середній КС зображень набору ACT із застосуванням *LeftPredict* зменшився порівняно з варіантом без застосування предикторів на 11,72%. Але

зазначимо, що при застосуванні *LeftPredict* можливе зменшення довжини повторень, що були між фрагментами оригіналу зображення, тому можливе і підвищення загального КС (дані зображень №№ 2, 7 в рядку *LeftPredict* у табл. 2). Отже, однозначно стверджувати, що стиснення з використанням *LeftPredict* ефективніше від стиснення без застосування предикторів, неможливо. Середня тривалість компресії файлів набору ACT цим способом перевищує тривалість кодування без предикторів на 14,46%, оскільки застосування предикторів потребує розрахунку їх значень і породжує більше коротких збігів послідовностей, що сповільнює формування розкладу *LZ*77.

Таблиця 2. Розміри файлів зображень набору *ACT* у форматі *PNG* після застосування різних предикторів та їх комбінацій, Кб

Назва варіанта	Номер файла								Разом	Середній КС, %
	1	2	3	4	5	6	7	8		
Без предикторів	502	**248**	702	812	650	894	**104**	958	4870	55,59
LeftPredict	447	331	502	645	441	793	135	738	4032	43,87
AbovePredict	481	349	471	645	454	808	143	710	4061	43,77
AveragePredict	1124	515	**468**	610	422	**763**	215	683	4800	47,06
PaethPredict	**440**	354	471	**609**	**415**	780	145	**667**	3881	41,77
Комбінування 1	441	347	476	705	514	838	146	889	4356	47,53
Комбінування 2	440	353	687	645	441	797	147	711	4221	46,77
Комбінування 3	440	352	463	606	414	772	147	667	3861	41,51
Комбінування 4	439	348	463	605	414	772	146	666	3853	41,46
Комбінування 5	463	311	463	607	415	763	132	669	3823	41,33
Комбінування 6	439	339	463	605	414	763	147	666	3836	41,34
Комбінування 4, 5 з аналізом блоків	439	243	463	600	414	757	104	663	3683	40,49

Аналогічно впливає на стиснення зображень і застосування предиктора *RightPredict*, який генерує вертикальні прирости яскравості компонент суміжних пікселів. Різні зображення мають різний рівень відмінностей пікселів по горизонталі та вертикалі. Саме тому КС в рядках *LeftPredict* та *RightPredict* табл. 2 різні. Саме цим пояснюється різниця в часі кодування на 34,31% зображення № 1 для даних рядків у табл. 3, оскільки це зображення містить суттєво меншу кількість однакових приростів яскравості компонент пікселів у вертикальному напрямку стосовно горизонтального, що прискорює формування розкладу *LZ*77, але й підвищує КС. Крім цього, гірший КС з використанням предиктора *RightPredict* стосовно *LeftPredict* для більшої частини зображень набору пояснюється тим, що однакові прирости по вертикалі трапляються, як правило, в сусідніх рядках. Зміщення ж між сусідніми пікселами по вертикалі при обробці даних у форматі *PNG* дорівнює довжині рядка. Тобто вони значно більші від зміщень між сусідніми пікселами по горизонталі і тому кодуються значно більшою кількістю додаткових біт.

Таблиця 3. Час кодування файлів зображень набору *ACT* у форматі *PNG* варіантами програми з застосування різних предикторів та їх комбінацій (на комп'ютері з частотою процесора 300 МГц), с

Назва варіанта	Номер файла								Разом
	1	2	3	4	5	6	7	8	
Без предикторів	9,06	13,51	4,66	6,54	4,62	6,97	5,55	6,92	57,83
LeftPredict	12,62	13,84	4,89	8,40	5,61	7,20	5,88	7,75	66,19
AbovePredict	8,29	14,77	4,95	8,41	5,22	7,03	5,99	8,02	62,68
AveragePredict	17,30	15,93	5,05	9,61	5,93	7,30	6,42	8,46	76,00
PaethPredict	11,37	15,71	5,21	9,56	5,93	7,69	6,48	8,84	70,79
Комбінування 1	14,50	20,32	6,43	10,71	6,92	9,45	8,35	9,67	86,35
Комбінування 2	15,16	22,52	6,81	10,66	7,03	9,34	9,17	10,05	90,74
Комбінування 3	14,83	21,04	6,42	11,37	7,03	9,34	8,63	10,44	89,10
Комбінування 4	14,61	21,04	6,54	11,70	7,20	9,45	8,74	10,55	89,83
Комбінування 5	16,48	24,06	7,25	12,69	7,97	10,55	9,99	11,59	100,58
Комбінування 6	17,08	24,99	7,69	13,13	8,35	11,10	10,33	12,08	104,75
Комбінування 4, 5 з аналізом блоків	49,21	67,89	20,04	42,35	25,93	30,10	28,35	37,35	301,22

Предиктор *AveragePredict* врівноважує вплив сусідніх пікселів по горизонталі та вертикалі. Він добре (відносно розглянутих варіантів) прогнозує яскравість компонент пікселів для зображень великих та розмитих природних об'єктів, хоча й дуже чутливий до різких перепадів кольорів (наприклад, ліній променів світла у зображенні № 1) та чітких меж об'єктів штучних зображень. Сповільнення кодування зображень з використанням *Average Predict* пояснюється як необхідністю виконання операції ділення для розрахунку значення предиктора кожної компоненти всіх пікселів, так і низькою ефективністю розкладу *LZ*77, що породжує багато елементів для кодування Хафмана.

Найкращі в середньому передбачення значень яскравостей компонент пікселів генерує предиктор *PaethPredict*, який прогнозує значення у напрямку найменшого приросту. Цей предиктор на дискретно-тонових зображеннях поступаєть-

ся лише предиктору *LeftPredict* та варіанту без предикторів, а на неперервно-тонових зображеннях забезпечує КС, близькі до предиктора *AveragePredict*. Але, по-перше, нелінійний предиктор *PaethPredict* обчислюється найдовше з усіх розглянутих предикторів, що негативно впливає як на час кодування, так і на час декодування, і, по-друге, цей предиктор, як і попередній, часто зменшує довжини повторень, що були між фрагментами оригіналу зображення.

Як бачимо, застосування нелінійних комбінованих предикторів в цілому ефективніше від лінійних шумових на понад 2%, хоча й потребує більшої кількості обчислень. З іншого боку, не існує універсального предиктора, застосування якого сприяло б найкращому стисненню всіх типів зображень. Навіть для сусідніх фрагментів зображень оптимальними можуть виявитися різні предиктори. Тому для досягнення кращих коефіцієнтів стиснення *необхідно використовувати комбінації предикторів*. Зрозуміло, що комбінування предикторів сповільнює кодування, але дає змогу не лише зменшити розміри файлів зображень, але й завдяки цьому прискорити декодування.

За стандартом *PNG* [1] рекомендується обирати той предиктор для кожного рядка, застосування якого зумовлює мінімальну суму значень знакових байт (з діапазону [–128; 127]). Справді, зменшення загальної суми значень рядка після використання предиктора свідчить про наближення розподілу частот до симетричного, але чи вказує цей критерій на зменшення ентропії джерела? Результати застосування такого способу вибору предиктора для кожного рядка дають негативну відповідь на це запитання (див. рядок *Комбінування* 1 в табл. 2).

Інший спосіб визначення оптимального предиктора для кожного рядка полягає у пошуку найбільшої кількості повторень однакових значень, що йдуть підряд (запропонований в [8, с. 317]). Справді, повторення значень сприяє покращенню показників стиснення алгоритмом *LZ*77, але цей спосіб не враховує можливості повторень груп різних байт, що також покращують стиснення алгоритмом *LZ*77, та змін ентропії джерела. Результати застосування такого способу вибору оптимального предиктора кожного рядка наведено у табл. 2, 3 в рядку *Комбінування* 2.

В [3, 9] запропоновано обирати той предиктор, застосування якого зумовлює мінімальну суму *модулів* значень знакових байтів (з діапазону [–128; 127]). Зменшення загальної суми модулів значень рядка після використання предиктора свідчить про збільшення нерівномірності розподілу частот навколо нуля та покращує результати застосування контекстно-незалежного алгоритму. Результати застосування такого способу вибору оптимального предиктора кожного рядка наведено у табл. 2, 3 в рядку *Комбінування* 3. Як видно з таблиць, такий спосіб вибору предиктора рядка покращує середній КС стосовно предиктора Піфа на 0,26%, причому зменшення розмірів файлів спостерігається майже для всіх зображень, хоча він і сповільнює кодування на 22%.

Для розробки ефективніших способів вибору предикторів рядків проаналізуємо вплив різних предикторів на продуктивність базових алгоритмів стиснення формату *PNG*. Алгоритм *LZ*77 досягає кращих КС зображень без застосування предикторів або після дії предикторів *LeftPredict* чи *RightPredict*. Кодування Хафмана найкраще стискає розподіли зображень після дії предикторів (особливо після *AveragePredict* або *PaethPredict*), оскільки вони підвищують нерівномірність розподілу. Обрати найкращий з трьох варіантів використання предикторів стосовно алгоритму *LZ*77 для кожного рядка пікселів неможливо без виконання трьох попередніх розкладів зображення з застосуванням кожного з цих варіантів, оскільки однакові фрагменти даних можуть траплятися в різних рядках. *Найкращий предиктор стосовно розкладу Хафмана для кожного рядка пікселів оберемо з умови мінімуму ентропії серед результатів їх дій*, до якої прямує середня довжина коду цього розкладу. Нехай кожен з елементів S_i зустрічається N_i разів в рядку довжиною $N = \sum_i N_i$. Тоді $p(s_i) = N_i / N$ і загальна довжина закодованих даних, враховуючи (1), наближається до значення

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

Мовою *C* підпрограма для наближеного обчислення ентропійного розміру блоку кодів Хафмана за частотами елементів згідно (3) з визначенням прогнозованого КС має вигляд:

```
  double sizeEntropiCode(long * freq,
short countAllFreq, double &entropiKC)
   {// freq - масив частот елементів
    // countAllFreq – загальна кількість
частот в масиві
    double size=0; long n=0;
    for (long i=0; i<countAllFreq; i++)
     {n+=freq[i]; // сумуємо частоти
      if (freq[i]>1) // для існування log
       size+=freq[i]*log(freq[i]); }
    if (n) {size=(n*log(n)-size)/log(2);
entropiKC=size/(8*n); }
    else {size=0; entropiKC=1; }
    return size; } .
```

А фрагмент програми для визначення номера предиктора, що породжує дані з найменшою ентропією елементів записується так:

```
  for (i=0; i<=4; ++i) // цикл по преди-
кторах
   {memset(freq, 0, sizeof(freq));
    for (j=0; j<row_width; ++j) // цикл
по елементах рядка
     freq[buffers[i][j]]++; // накопи-
чення частот елементів
    size=sizeEntropiCode(freq, 256,
entropiKC);
    if (size<minSize)
     {predict1=i; // запам'ятали номер
предиктора
      minSize=size;
entropiKC1=entropiKC; }}
  KC1=entropiKC1; .
```

Середній КС зображень набору *ACT* з застосуванням ентропійного способу вибору предикторів рядків зменшився в порівнянні з способом вибору предикторів на основі найменшої суми модулів значень знакових байтів на 0,05% (рядок *Комбінування* 4 з табл. 2, 3), причому розмір стиснутих файлів не збільшився для жодного зображення, а час стиснення збільшився лише на 0,82%. Використання цього способу ефективне, насамперед, для природних зображень декількох великих об'єктів. У фрагментах зображень, в яких ентропійний спосіб вибору предикторів рядків є найефективнішим, короткі заміни (3–4 літерали), як правило, не застосовуються. Оптимальними предикторами при такому способі вибору найчастіше є *AveragePredict* або *PaethPredict*.

Крім ентропії окремих елементів, для кожного рядка пікселів слід оцінити також КС після виконання коротких замін. Адже короткі заміни можуть суттєво збільшити частоти довжин замін, що їм відповідають, та зменшити частоти окремих літералів. А це, в свою чергу, призводить до збільшення нерівномірності розподілу і, відповідно, до зменшення КС. Виконувати попереднє стиснення *LZ*77 для результатів дії всіх предикторів рядків недоцільно, оскільки це спричинює різке сповільнення процесу кодування. Оцінювання КС рядка пікселів відбувалося після застосування триелементних замін за допомогою хеш-таблиці по чотирьох останніх бітах трьох суміжних елементів, в якій зберігаються абсолютні позиції останніх входжень подібних груп. З використанням значення цієї таблиці імітується розклад *LZ*77 з підрахунком частот літералів, триелементних замін та їх зміщень і додаткових бітів, після чого визначається загальний КС рядка. Мовою *C* фрагмент програми для визначення номера предиктора, що породжує дані з найменшим КС після коротких замін записується, наприклад, так:

```
  for (i=0;  ii<=4; ++i) // цикл по пре-
дикторах
   {memset(freq, 0, sizeof(freq));
     // частоти літералів/довжин
    memset(freqD, 0, sizeof(freqD));
     // частоти баз зміщень
    memset(hash, 0, sizeof(hash));
     // елементи хеш-таблиці
    plusBit=0; // додаткові біти зміщень
    j=0; // позиція обробки рядка після
дії предиктора i
    while (j<row_width-2)
     if (hash[((buffers[i][j] & 15)<<8)+
            ((buffers[i][j+1] & 15)<<4)+
            (buffers[i][j+2] & 15)])
{// подібні три байта вже були в рядку
  freq[257]++; // частота триелементної
заміни
  offset=j-hash[((buffers[i][j]&15)<<8)+
  ((buffers[i][j+1]&15)<<4)+(buffers[i][j+
                              +2]&15)]+1;
```

```
  // збільшуємо частоту бази зміщення
       freqD[codesDistance[offset]]++;
  // накопичуємо додаткові біти зміщення
  plusBit+=extrasDistance[codesDistan-
                          ce[offset]];

  // дані оброблених триелементних груп
  // записуємо в хеш-таблицю
   hash[((buffers[i][j] & 15)<<8)+
    ((buffers[i][j+1] & 15)<<4)+(buf-
              fers[i][j+2] & 15)]=j+1;
  if (j+1<row_width-2)
   hash[((buffers[i][j+1] & 15)<<8)+
  ((buffers[i][j+2] & 15)<<4)+(buf-
              fers[i][j+3] & 15)]=j+2;
  if (j+2<row_width-2)
   hash[((buffers[i][j+2] & 15)<<8)+
   ((buffers[i][j+3] & 15)<<4)+(buf-
              fers[i][j+4] & 15)]=j+3;
    j+=3; }
  else // подібна група відсутня – зано-
симо літерал
   {freq[buffers[i][j]]++;
    hash[((buffers[i][j] & 15)<<8)+
    ((buffers[i][j+1] & 15)<<4)+(buf-
               fers[i][j+2] & 15)]=j+1;
    j++; }
  for (; j<row_width ; ++j) //останні по-
зиції рядка
   freq[buffers[i][j]]++;
  size=sizeEntropiCode(freq, 256, en-
                                tropiKC)+
       sizeEntropiCode(freqD, 30, en-
                    tropiKCD)+plusBit;
  if (size<minSize)
   {predict2=i; // запам'ятали номер
предиктора
    minSize=size; entropiKC2=entropiKC; }}
  KC2=(double)minSize/(8*row_width); .
```

Середній КС зображень набору *ACT* з застосуванням ентропійного способу вибору предикторів рядків після імітації коротких замін *LZ*77 зменшився порівняно з середнім КС ентропійного поелементного предиктора на 0,13% (рядок *Комбінування* 5 з табл. 2, 3), хоча покращення спостерігається лише на дискретно-тонових зображеннях та неперервно-тонових зображеннях, для яких короткі заміни ефективні. Це й не дивно, адже розрахунок ентропійного КС після імітації коротких замін найкраще з розглянутих варіантів моделює стиснення у форматі *PNG*. Середня тривалість компресії файлів набору *ACT* внаслідок імітації коротких замін під час вибору предикторів рядків зросла на 11,96%. Оптимальними предикторами при виборі за мінімальним ентропійним КС після застосування коротких замін *LZ*77 найчастіше виявляються *LeftPredict* або *RightPredict*. Використання цього способу ефективне, насамперед, для природних зображень з багатьма об'єктами і для штучних зображень.

Для різних зображень і навіть для різних фрагментів одного зображення короткі заміни *LZ*77 можуть виявитися як ефективними (тобто кількість біт для зберігання заміни є не більшою від кількості біт для зберігання окремих елементів), так і неефективними, отже передбачити необхідність імітації коротких замін неможливо. Тому для кожного рядка пікселів оберемо предиктор, що забезпечує прогнозований мінімальний КС. Отримаємо цей КС за результатами порівняння мінімального поелементного КС (в попередніх фрагментах програм – *KC*1) та мінімального КС (*KC*2) після застосування коротких замін *LZ*77 стосовно результатів дії всіх предикторів, оскільки ці варіанти вибору предикторів забезпечують в середньому найкращі результати. Враховуючи те, що довгі ефективні заміни частково враховані в *KC*2 і не враховані в *KC*1, обирати предиктор, що забезпечує *KC2*, будемо лише тоді, коли *KC*2 менший за *KC*1 більше, ніж на 4%. Інакше обиратимемо предиктор, що забезпечує *KC*1:

```
  If (KC2<KC1-0.04) currentPredict=
=predict2;
  else currentPredict=predict1;  .
```

Такий комбінований спосіб вибору предикторів рядків хоча й поступається за середнім КС на 0,01% ентропійному способу після коротких замін *LZ*77 і формується на 4,15% повільніше від нього (рядок *Комбінування* 6 у табл. 2, 3), але забезпечує не гірші КС за всіма зображеннями набору стосовно способу вибору предикторів *Комбінування* 3 на основі найменшої суми модулів значень знакових байтів.

Розбиття зображень на блоки однорідних рядків за допомогою ентропії

Звичайно, кожен рядок зображення можна розмістити в окремому стиснутому блоці фор-

мату *PNG*. Але загалом таке стиснення не буде найефективнішим, оскільки в заголовку кожного стиснутого блоку динамічних кодів Хафмана міститься опис довжин кодів його розподілів літералів/довжин та зміщень, який може займати понад 1500 бітів, а такі витрати неприпустимі для кожного рядка. З іншого боку, поєднання у стиснутих блоках даних довільних суміжних рядків в цілому зменшує нерівномірність розподілу і може негативно вплинути на загальний КС. Тому пропонується в процесі попереднього аналізу зображень перед стисненням у форматі *PNG поєднати суміжні рядки з близькими прогнозованими ентропійними КС розподілів у блоки*. Кожен однорідний блок рядків в процесі подальшого стиснення може належати одному, а може бути й розбитим на кілька стиснутих блоків, адже розмір стиснутого блоку, як правило, не перевищує 64 Кб. Але різні блоки рядків мають обов'язково належати різним стиснутим блокам, оскільки вони мають різні ентропійні КС (в попередніх фрагментах програм – *entropiKC1*), а отже, і різні нерівномірності розподілів. Зрозуміло, що поєднувати між собою суміжні блоки рядків доцільно лише тоді, коли прогнозоване збільшення довжини кодів від їх поєднання не перевищує розміру заголовка нового стиснутого блоку (тобто 1500).

Розрахуємо прогнозоване збільшення довжини кодів від поєднання двох суміжних блоків. Нехай перший блок рядків містить l_1 елементів з прогнозованою ентропією (середньою довжиною коду) e_1, а другий – l_2 елементів з ентропією e_2. Тоді прогнозована довжина кодів першого блоку становить $l_1 \times e_1$, а другого – $l_2 \times e_2$. Внаслідок поєднання блоків рядків з подібними нерівномірностями розподілу (див. рис. 1,*б*) ентропія поєднання не може перевищити максимуму ентропій поєднаних частин, отже прогнозована довжина коду поєднання не перевищує $(l_1 + l_2) \times \max(e_1; e_2)$. Тому прогнозоване збільшення довжини коду внаслідок поєднання двох суміжних блоків не перевищує

$$\Delta l_k^{'} = (l_1 + l_2) \times \max(e_1; e_2) - l_1 \times e_1 - l_2 \times e_2 = \begin{cases} (e_1 - e_2) \times l_2, & e_1 \geq e_2 \\ (e_2 - e_1) \times l_1, & e_1 < e_2 \end{cases}. \quad (4)$$

На практиці прогнозоване збільшення довжини коду від поєднання залежить не лише від кількості елементів вхідного блоку з меншою ентропією, а й від кількості елементів іншого вхідного блоку (наприклад, короткий перший вхідний блок з високою ентропією не може максимально збільшити ентропію другого довгого вхідного блоку), тому розраховувати його доцільно за формулою, що враховує відношення розмірів вхідних блоків:

$$\Delta l_k^{''} = \begin{cases} |e_1 - e_2| \times l_2 \times \log(l_1/l_2 + 1), & l_1 \geq l_2 \\ |e_1 - e_2| \times l_1 \times \log(l_2/l_1 + 1), & l_1 < l_2 \end{cases}. \quad (5)$$

Ентропійне кодування елементів розподілів виконується у форматі *PNG* після формування розкладу *LZ*77. Визначити наперед кількість елементів такого розкладу неможливо, але ця кількість елементів перебуває у монотонно не спадній залежності від ентропії: чим менша ентропія, тим більша ймовірність існування повторень значень яскравості компонентів і, отже, тим менше елементів буде містити розклад *LZ*77. Прогнозована кількість елементів рядків обчислювалася після розкладу *LZ*77 як добуток початкової кількості елементів рядків та мінімуму з одиниці і ентропії, збільшеної у 4/3 рази (вважали, що рядки з ентропією понад 0,75 не піддаються стисненню *LZ*77). Алгоритм формування блоків однорідних рядків представимо *покроково*:

1. Визначити для кожного рядка пікселів зображення предиктор, що породжує коди з мінімальною ентропією згідно (3) та прогнозовану кількість елементів після розкладу *LZ*77, як описано раніше. Віднести кожен рядок до окремого блоку рядків.

2. Розрахувати для всіх пар суміжних блоків прогнозоване збільшення довжини коду внаслідок їх можливого поєднання за допомогою (5), враховуючи максимум 64 тис. елементів з кожного блоку. Визначити пару суміжних блоків, що породжує найменше таке збільшення довжини коду.

3. Якщо залишився лише один блок рядків або найменше збільшення довжини коду перевищує 1500 бітів, то завершити виконання ал-

горитму. Інакше – поєднати пару суміжних блоків, що породжує це збільшення, і повернутися до кроку 2.

Саме цей алгоритм формування блоків однорідних рядків за допомогою ентропії реалізовано в модифікаціях програми, використаної для тестування. Він дозволяє зменшити розміри більшості стиснутих зображень у форматі *PNG* мінімум на 1 Кб.

Досягнути менших КС зображень можливо, якщо: розбити їх на менші блоки однорідних рядків згідно (4); для кожного блоку однорідних рядків обрати предиктори з умови мінімуму розрахованих за допомогою (3) КС серед п'яти варіантів компресії (без застосування предикторів, з використанням *LeftPredict*, з застосуванням *RightPredict*, з використанням ентропійного поелементного способу вибору предикторів та з застосуванням ентропійного способу вибору предикторів після здійснення коротких замін *LZ*77); укрупнити блоки однорідних рядків, використовуючи (5). При цьому слід враховувати також відхилення від варіантів компресії, що забезпечують мінімальний КС для попереднього і наступного однорідних блоків рядків, адже зміна варіантів компресії між сусідніми блоками призводить до виникнення нового стиснутого блоку (а, отже, і його заголовка) і зменшує ймовірність повторень фрагментів даних для алгоритму *LZ*77. Тобто **автори пропонують в процесі аналізу зображення обирати предиктори, які мінімізують коефіцієнт стиснення кожного блоку однорідних рядків**, а не лише ентропію компонент пікселів окремих рядків. Результати застосування такого способу стиснення наведено у рядках *Комбінування* 4, 5 *з аналізом блоків* табл. 2, 3. Описаний спосіб стиснення зображень, що обирає мінімальний КС для кожного блоку однорідних рядків, хоча й виконується в середньому майже втричі повільніше від ентропійного варіанту і майже у шість разів повільніше від стиснення без предикторів, (оскільки формує п'ять додаткових розкладів *LZ*77), зате забезпечує найменші КС для всіх зображень набору АСТ.

Висновки. Для підвищення ефективності стиснення даних у форматах, що послідовно використовують алгоритми декількох методів, доцільно враховувати взаємний вплив цих алгоритмів. Під час попередньої обробки даних перед стисненням у таких форматах для розрахунку параметрів компресії бажано опрацьовувати всі види надлишковостей, що обробляються їх окремими алгоритмами, а також розбивати дані на однорідні блоки.

Для кожного рядка пікселів зображення предиктори слід обирати залежно від обмежень на час компресії: у випадку найшвидшого стиснення доцільно використати нелінійний предиктор *PaethPredict* для всіх рядків; під час стандартного стиснення варто скористатися ентропійним способом вибору предикторів (*Комбінування* 4); для повільного максимального стиснення слід застосувати вибір предикторів на основі аналізу КС п'яти варіантів компресії кожного блоку однорідних рядків.

Запропонований спосіб попередньої обробки з аналізом даних дозволяє отримати найменші КС зображень у форматі *PNG*, насамперед, шляхом поділу зображень на блоки рядків пікселів з близькими значеннями ентропії та вибору для кожного блоку рядків способу стиснення, що забезпечує для нього найменший КС.

Описані способи вибору предикторів рядків та розбиття зображення на блоки однорідних рядків за допомогою ентропії не потребують модифікації декодера чи програм перегляду зображень і за умови зменшення розмірів файлів лише прискорюють їх роботу. Саме тому вони можуть бути ефективно застосовані у процесі компактного збереження даних у форматах, що використовують різні предиктори для окремих рядків пікселів, таких, як формат *PNG*.

1. *Boutell T.* PNG Specification, Vers. 1.0, PNG Development Group, October, 1996.
2. *L. Peter Deutsch* DEFLATE Compressed Data Format Specification, Vers. 1.3, RFC 1951, 1996.
3. *Методы* сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов и др. – М.: ДИАЛОГ-МИФИ, 2003. – 384 с.
4. *Ziv J.*, *Lempel A*. A universal algorithm for sequential data compression // IEEE Transactions on Inform. Theory. – May 1977. – **23**, N 3. – P. 337–343.

5. *Huffman D*. A Method for the Construction of Minimum Redundancy Codes // Proc. of the IRE. – **40**, N 9. – P. 1098–1101.
6. *Бомба А.Я*., *Шпортько О.В*. Алгоритм оптимізації вибору фільтра для попереднього опрацювання зображень перед стисненням на основі методу «предиктор – коректор» // Вісн. Нац. ун-ту «Львівська політехніка». (Серія: Інформаційні системи та мережі). – 2008. – № 621. – С. 46–54.
7. *Бредихин Д.Ю*. Сжатие графики без потерь качества. – http://www.compression.ru/download/articles/i_lless/bredikhin_2004_lossless_image_compression_doc.rar
8. *Миано Дж*. Форматы и алгоритмы сжатия изображений в действии: Учеб. пособие. – М.: Триумф, 2003. – 336 с.
9. *Шпортько О.В*. Використання альтернативних блоків стиснутих даних у форматі PNG // Комп'ютерні науки та інформаційні технології: Матеріали третьої Міжнар. конф. *CSIT*'2008. – Львів: Вежа і Ко, 2008. – С. 149–153.

Поступила 10.09.2009
Тел. для справок: (0362) 26-04-44, 64-72-87, 22-60-51 (Ровно)
E-mail: *abomba@ukr.net*, *chportko@yandex.ru*, *chportko@ukr.net*

●

А.Я. Бомба, О. В. Шпортько

Энтропийные способы выбора предиктора для строки пикселей в формате *PNG*

Введение. Сегодня формат *PNG* [1] – один из основных для сохранения изображений без потерь. В Интернете, например, насчитывается свыше 15 млн страниц, содержащих изображения в этом формате. Популярности формата *PNG* способствуют, в первую очередь, приемлемые коэффициенты сжатия (КС) – отношение размера сжатых к несжатым данным и высокая скорость декодирования. Проблема уменьшения размеров файлов изображений в этом формате актуальна и будет оставаться такой в ближайшем будущем, поскольку даже частичное ее решение дает возможность ускорить загрузку *PNG*-файлов из сети и повысить эффективность использования дискового пространства.

База исследования. Особенности этапов обработки данных изображений в формате *PNG*

Пикселы изображения записываются в *PNG*-файлы чаще всего по строкам сверху вниз, а в каждой строке – последовательно слева направо (как символы в текстах), формируя тем самым входной поток. Перед кодированием эти пикселы могут быть предварительно обработаны предикторами. В *PNG*-файлах, используемых сегодня, сжатые данные сохраняются в раздельных блоках согласно формату словарного сжатия *DEFLATE* [2]. В соответствии с этим форматом, в сжатых блоках содержатся результаты применения к входному потоку алгоритма *LZH* [3], согласно с которым результаты контекстно-зависимого словарного алгоритма *LZ*77 [4] сжимаются контекстно-независимыми кодами Хаффмана [5]. Для исследования взаимовлияния этапов обработки данных изображений в формате *PNG* напомним принципы их выполнения.

*Контекстно-зависимый алгоритм LZ*77 базируется на замене в выходном потоке последовательности очередных незакодированных неделимых элементов (литералов буфера) ссылкой на аналогичную последовательность литералов, начинающихся среди закодированных элементов (в словаре), в виде пары чисел <*длина*; *смещение от окончания словаря*> [4]. В случае отсутствия аналогичной последовательности литералов в словаре, первый литерал буфера переносится в выходной поток без изменений. После этого закодированные литералы переносятся с начала буфера в конец словаря и кодирование продолжается аналогично вплоть до окончания литералов входного потока. Например, поток кодов байтов пикселов 36 38 35 35 36 38 35 36 36 38 35 38 36 38 35 35 28 в закодированном виде можно записать так: 36 38 35 35 <3; 4> 36 <3; 4> 38 <4; 12> 28.

В процессе декодирования кодов, полученных по алгоритму *LZ*77, отдельные литералы копируются в выходной поток без изменений. Пары же <*длина*; *смещения*> декодируются путем последовательного копирования с конца выходного потока по указанному смещению в конец выходного потока необходимого количества литералов. Естественно, что алгоритм декодирования должен различать отдельные литералы и группы <*длина; смещение*>. Согласно с алгоритмом *LZH*, в формате *DEFLATE* с этой целью базовые значения длин замен и отдельные литералы кодируются. После базовых значений длин содержится определенное форматом количество битов, которые вместе с базовым значением однозначно определяют длину замены. Смещение сохраняется после соответствующей длины аналогично – в виде базового значения и дополнительных битов.

В программной реализации для уменьшения КС алгоритма *LZ*77 [6] обеспечена возможность выхода из словаря в буфер во время кодирования повторов, использован модифицированный алгоритм анализа альтернативных сжатых блоков и проведен анализ количеств дополнительных битов для записи смещений.

Ключевые слова: сжатие изображений без потерь, статические предикторы, формат графических файлов *PNG*.

Идея использования *динамических кодов Хаффмана* [5], чаще всего применяемых после выполнения алгоритма *LZ*77 для сжатия литералов и базовых значений длин, а также для отдельного сжатия базовых значений смещений в каждом блоке сжатых данных, заключается в замене чисел с большей частотой (здесь и в дальнейшем – абсолютной) кодами меньшего количества битов, чем для чисел с меньшей частотой.

В соответствии с теоремой Шеннона, элемент s_i с вероятностью появления $p(s_i)$ выгоднее всего кодировать – $\log p(s_i)$ битами (здесь и в дальнейшем логарифм берется по основанию 2). Тогда средняя длина кода элемента после применения контекстно-независимого алгоритма (такого, как кодирование Хаффмана) должна приближаться к *энтропии источника* [3]:

$$H = -\sum_i p(s_i) \times \log(p(s_i)). \qquad (1)$$

Как известно, энтропия источника уменьшается при увеличении неравномерности распределения вероятностей между элементами [7].

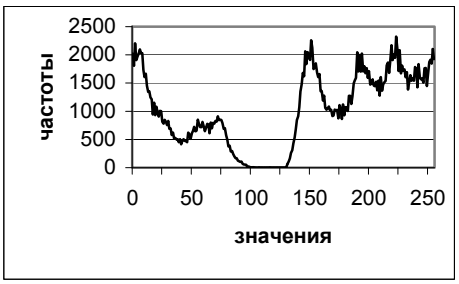
Алгоритм генерирования динамических кодов Хаффмана известен еще со средины прошлого века и уменьшить его КС не представляется возможным. Можно только пытаться после минимизации энтропии отдельных строк разбить изображение на блоки строк пикселов с близкими значениями энтропии. Это позволит отделить и локализовать строки с высокой энтропией (границы фрагментов изображения или значительные перепады яркости пикселов), повысив тем самым неравномерности распределения в смежных блоках, и поэтому уменьшить общий КС.

Уменьшить энтропию при обработке изображений в формате *PNG пытаются также с помощью предикторов*, стремящихся, используя значение известных смежных элементов, спрогнозировать значение очередного элемента [3, 7]. Если пиксел изображения характеризуется несколькими компонентами (например, *R*, *G*, *B*), то предиктор каждой из них прогнозирует значение согласно соответствующим компонентам соседних пикселов. В процессе использования этой технологии вычисляют и в дальнейшем кодируют отклонение очередной компоненты от прогнозируемого предиктором значения [3]. Поэтому, в общем случае, процесс применения предикторов к каждой компоненте пиксела в узле (i, j) можно записать формулой

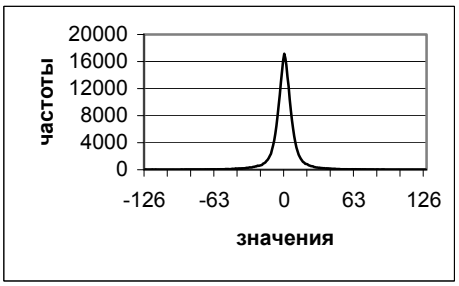
$$\Delta_{ij} = C_{ij} - predict_{ij}, \qquad (2)$$

где C_{ij} – значение компоненты до применения предиктора, Δ_{ij} – значение компоненты после применения предиктора, $pregict_{ij}$ – значение предиктора, вычисленного для выбранной компоненты.

Проанализируем распределение частот значений Δ_{ij}. Поскольку соседние пикселы изображения имеют, как правило, близкие цвета и поэтому близкие значения соответствующих компонент, то часто значение предиктора будет совпадать со значением очередного элемента, чаще всего – будет близким к этому значению и изредка – значительно отличаться от него. То есть большинство значений Δ_{ij} будут близкими к нулю. Такое перераспределение частот значений (а следовательно, и вероятностей) значительно повышает неравномерность распределения и потому уменьшает энтропию источника (согласно (1)), а значит, и длину закодированной последовательности [7]. Таким образом, предикторы хотя и не выполняют непосредственное сжатие данных, но уменьшают, в первую очередь, КС контекстно-независимого алгоритма. Перераспределение частот значений компонент после применения предиктора показано на рис. 1.



а



б

Рис. 1 Распределение частот значений зеленой компоненты данных первых 64 Кб изображения *Lena.bmp*: *а* – до применения предиктора; *б* – после применения *LeftPredict*

Кроме того, отдельные предикторы могут разбить существующие или сгенерировать новые повторы фрагментов изображения и этим повлиять на КС алгоритма *LZ*77, о чем будет сказано далее.

В сжатых блоках формата *PNG* данным каждой строки предшествует отдельный байт, определяющий предиктор, применяемый к компонентам всех его пикселов [8]. Сегодня форматом предусмотрено пять возможных значений этого байта, который определяет четыре различных предиктора: 0 – данные строки не обрабатываются предикторами; 1 – *LeftPredict*; 2 – *AbovePredict*; 3 – *AveragePredict*; 4 – *PaethPredict*. Для удобства описания этих предикторов введем обозначение яркости смежных элементов для элемента *X* по схеме:

- *LeftAbove*
- *Above*
- *Left*
- *X* .

Тогда предикторы формата *PNG*, согласно стандарту [1], на языке *C* запишем так:

```
   Char LeftPredict(char Left, char Above,
char LeftAbove)
    {return Left;}
   char AbovePredict(char Left, char Above,
char LeftAbove)
    {return Above;}
   char AveragePredict(char Left, char Above,
char LeftAbove)
    {return (Left+Above)/2;}
   char PaethPredict(char Left, char Above,
char LeftAbove)
    {int pp=Left+Above-LeftAbove;
     int pa,pb,pc;
     pa=abs(pp-Left);
     pb=abs(pp-Above);
     pc=abs(pp-LeftAbove);
     if (pa<=pb && pa<=pc) return Left;
     else if (pb<=pc) return Above;
        else return LeftAbove;}.
```

Как отмечено в [3], отклонения между значениями соседних элементов в изображениях чаще всего предопределены двумя причинами: «сильными» колебаниями, обусловленными изображенными объектами – трендом, и слабыми фоновыми колебаниями – шумом. Первые три из упомянутых предикторов описывают шумовую модель и принадлежат к линейным статическим предикторам. Последний объединяет в себе трендовую и шумовую модели и принадлежит к нелинейным статическим предикторам.

Проблема выбора таких статических предикторов для каждой строки изображения, которые позволяют уменьшить общий КС изображения, оставалась *нерешенной* до сих пор [8, с. 317]. Далее рассмотрим способы ее решения с помощью энтропии.

Влияние предикторов и их комбинаций на сжатие изображений в формате *PNG*. Оценка энтропии строк пикселов

Алгоритмы, используемые в формате *PNG* для сжатия данных, ориентированы на уменьшение разных видов избыточности: если *LZ*77 устраняет избыточности между одинаковыми фрагментами данных, то кодирование Хаффмана ориентировано на уменьшение избыточности между преобладающими элементами распределений. Алгоритм *LZ*77 в этом формате может сжимать данные максимум в 129, а кодирование Хаффмана – максимум в восемь раз. Именно поэтому алгоритм *LZ*77 и используется в формате *PNG*, хотя он и уменьшает неравномерность распределения между отдельными элементами и может уменьшить эффективность кодирования Хаффмана. С другой стороны, в естественных изображениях яркости компонент соседних пикселов, как правило, близки, но не одинаковы. Это делает малоэффективным использование для них алгоритма *LZ*77, и поэтому кодирование Хаффмана в этом случае обеспечивает хотя бы удовлетворительные КС. Ключевая роль в процессе сжатия различных изображений и даже отдельных их фрагментов могут выполнять как алгоритм *LZ*77, так и кодирование Хаффмана. Использование разных предикторов (в том числе и отказ от их применения) формата *PNG* неодинаково влияет на КС этих алгоритмов. Поэтому для каждой строки изображения следует выбирать тот предиктор, который позволит усилить действие ключевого алгоритма сжатия соответствующего фрагмента, чем и поможет обеспечить наименьший общий КС.

Таблица 1. Характеристики изображений набора *ACT*

№ файла	Название файла	Размер, Кб	Размеры изображения, пикселов	Уникальных цветов среди пикселов, %	Особенности
9.	*Clegg.bmp*	2101	814x880	17,83	Непрерывно-тоновое, искусственное, с помехами, несколько больших объектов
10.	*Frymire.bmp*	3622	1118x1105	0,29	Дискретно-тоновое, искусственное, один большой объект
11.	*Lena.bmp*	769	512x512	56,56	Непрерывно-тоновое, естественное, несколько больших объектов
12.	*Monarch.bmp*	1153	768x512	19,99	Непрерывно-тоновое, естественное, один большой и много малых объектов
13.	*Peppers.bmp*	769	512x512	42,47	Непрерывно-тоновое, естественное, несколько больших объектов
14.	*Sail.bmp*	1153	768x512	19,26	Непрерывно-тоновое, естественное, много средних объектов
15.	*Serrano.bmp*	1464	629x794	0,26	Дискретно-тоновое, искусственное, один большой фрагментированный объект
16.	*Tulips.bmp*	1153	768x512	30,07	Непрерывно-тоновое, естественное, несколько больших объектов

Исследуем влияние различных вариантов предикторов на примере сжатия восьми разнотипных 24-битных естественных изображений из стандартного набора файлов *Archive Comparison Test* (*ACT*) в формате *PNG*, характеристики которых приведены в табл. 1. Загрузить *TIFF*-версии файлов изображений этого набора можно, например, из *http*: //*compression.ca*/*act*/*act-files.html*. Тестирование проводилось с помощью программы *CD* [7], в которую, кроме указанных, были внесены такие модификации: размер блоков данных увеличен до 64 Кб, отброшены вспомогательные текстовые блоки и внедрено объединение строк в блоки, как описано далее. Результаты тестирования приведены в табл. 2, 3. Скорость декодирования изо-

бражений почти не зависит от примененных предикторов или их комбинаций и составляет в среднем для дискретно-тоновых изображений 910 Кб/с, для искусственных с помехами – 545 Кб/с, а для непрерывно-тоновых – 330 Кб/с. Поэтому среднее время декодирования всех изображений набора составляет только 25 с.

Таблица 2. Размеры файлов изображений набора *ACT* в формате *PNG* после применения различных предикторов и их комбинаций, Кб

Название варианта	Номер файла								Вместе	Средний КС, %
	1	2	3	4	5	6	7	8		
Без предикторов	502	**248**	702	812	650	894	**104**	958	4870	55,59
LeftPredict	447	331	502	645	441	793	135	738	4032	43,87
AbovePredict	481	349	471	645	454	808	143	710	4061	43,77
AveragePredict	1124	515	**468**	610	422	**763**	215	683	4800	47,06
PaethPredict	**440**	354	471	**609**	**415**	780	145	**667**	3881	41,77
Комбинирование 1	441	347	476	705	514	838	146	889	4356	47,53
Комбинирование 2	440	353	687	645	441	797	147	711	4221	46,77
Комбинирование 3	440	352	463	606	414	772	147	667	3861	41,51
Комбинирование 4	439	348	463	605	414	772	146	666	3853	41,46
Комбинирование 5	463	311	463	607	415	763	132	669	3823	41,33
Комбинирование 6	439	339	463	605	414	763	147	666	3836	41,34
Комбинирования 4, 5 с анализом блоков	439	243	463	600	414	757	104	663	3683	40,49

Таблица 3. Время кодирования файлов изображений набора *ACT* в формате *PNG* вариантами программ с применением различных предикторов и их комбинаций (на компьютере с частотой процессора 300 МГц), с

Название варианта	Номер файла								Вместе
	1	2	3	4	5	6	7	8	
Без предикторов	9,06	13,51	4,66	6,54	4,62	6,97	5,55	6,92	57,83
LeftPredict	12,62	13,84	4,89	8,40	5,61	7,20	5,88	7,75	66,19
AbovePredict	8,29	14,77	4,95	8,41	5,22	7,03	5,99	8,02	62,68
AveragePredict	17,30	15,93	5,05	9,61	5,93	7,30	6,42	8,46	76,00
PaethPredict	11,37	15,71	5,21	9,56	5,93	7,69	6,48	8,84	70,79
Комбинирование 1	14,50	20,32	6,43	10,71	6,92	9,45	8,35	9,67	86,35
Комбинирование 2	15,16	22,52	6,81	10,66	7,03	9,34	9,17	10,05	90,74
Комбинирование 3	14,83	21,04	6,42	11,37	7,03	9,34	8,63	10,44	89,10
Комбинирование 4	14,61	21,04	6,54	11,70	7,20	9,45	8,74	10,55	89,83
Комбинирование 5	16,48	24,06	7,25	12,69	7,97	10,55	9,99	11,59	100,58
Комбинирование 6	17,08	24,99	7,69	13,13	8,35	11,10	10,33	12,08	104,75
Комбинирования 4, 5 с анализом блоков	49,21	67,89	20,04	42,35	25,93	30,10	28,35	37,35	301,22

Сжатие изображений в формате *PNG* без применения предикторов происходит путем повторений и при наличии неравномерности распределения значений яркости пикселов. Но в естественных изображениях аналогичные фрагменты и одинаковые значения яркости компонентов случаются сравнительно редко, поэтому и такое сжатие приводит к высоким КС (см. рис. 1,*а* и вариант *Без предикторов* в табл. 2, 3). Сжатие изображений без применения предикторов эффективно, в первую очередь, для дискретно-тоновых искусственных изображений с высокой разрешающей способностью. Это сжатие выполняется быстрее всего, поскольку не требует выполнения дополнительных вычислений значений предикторов.

Предиктор *LeftPredict* генерирует горизонтальные приросты значений яркости между компонентами соседних пикселов. Одинаковые приросты значений яркости случаются в естественных изображениях, как правило, чаще, чем одинаковые фрагменты, что улучшает КС алгоритма *LZ*77. Кроме того, применение предиктора *LeftPredict* повышает неравномерность распределения вокруг нуля (рис. 1,*б*), уменьшая тем самым КС кодирования Хаффмана. Именно поэтому средний КС изображений набора *ACT* с применением *LeftPredict* уменьшился в сравнении с вариантом без применения предикторов на 11,72%. Но отметим, что при применении *LeftPredict* возможно уменьшение длины повторений, которые были между фрагментами оригинала изображения, поэтому возможно и повышение общего КС (данные изображений № 2, 7 в строке *LeftPredict* табл. 2). Следовательно, однозначно утверждать, что сжатие с использованием *LeftPredict* эффективнее сжатия без применения предикторов невозможно. Средняя продолжительность компрессии файлов набора *ACT* этим способом превышает продолжительность кодирования без предикторов на 14,46%, поскольку применение предикторов требует расчета их значений и порождает больше коротких совпадающих последовательностей, что замедляет формирование разложения *LZ*77.

Аналогично влияет на сжатие изображений и применение предиктора *RightPredict*, генерирующего вертикальные приросты значений яркости компонент смежных пикселов. Различные изображения имеют разный уровень отличий пикселов по горизонтали и вертикали. Именно поэтому КС в строках *LeftPredict и RightPredict* табл. 2 различны. Этим объясняется разница во времени кодирования на 34,31% изображение № 1 для данных строк в табл. 3, поскольку это изображение содержит существенно меньшее количество одинаковых приростов значений яркости компонент пикселов в вертикальном направлении относительно горизонтального, что ускоряет формирование разложения *LZ*77 и повышает КС. Кроме этого, худшие КС с использованием предиктора *RightPredict* относительно *LeftPredict* для большей части изображений набора объясняются тем, что одинаковые приросты по вертикали встречаются, как правило, в соседних строках. Смещение же между соседними пикселами по вертикали при обработке данных в формате *PNG* равно длине строки, т.е. они значительно больше смещений между соседними пикселами по горизонтали и поэтому кодируются значительно большим количеством дополнительных бит.

Предиктор *AveragePredict* уравновешивает влияние соседних пикселов по горизонтали и вертикали. Он хорошо (относительно рассмотренных выше вариантов) прогнозирует значения яркости компонент пикселов для

изображений больших и размытых естественных объектов, хотя и очень чувствителен к резким перепадам цветов (например, линий лучей света в изображении № 1) и четким границам объектов искусственных изображений. Замедление кодирования изображений с использованием *AveragePredict* объясняется как необходимостью выполнения операции деления для расчета значения предиктора каждой компоненты всех пикселов, так и низкой эффективностью разложения *LZ*77, порождающего много элементов для кодирования Хаффмана.

Самые лучшие в среднем ожидания значений яркости компонент пикселов генерирует предиктор *PaethPredict*, прогнозирующий значение в направлении наименьшего прироста. Этот предиктор на дискретно-тоновых изображениях уступает лишь предиктору *LeftPredict* и варианту без предикторов, а на непрерывно-тоновых изображениях обеспечивает КС, близкие к предиктору *AveragePredict*. Но, во-первых, нелинейный предиктор *PaethPredict* вычисляется дольше всех рассмотренных предикторов, что негативно влияет как на время кодирования, так и на время декодирования, и, во-вторых, этот предиктор, как и предыдущий, часто уменьшает длины повторений, которые были между фрагментами оригинала изображения.

Как видим, применение нелинейных комбинированных предикторов в целом эффективнее линейных шумовых на более чем 2%, хотя и требует выполнения большего количества вычислений. С другой стороны, не существует универсального предиктора, применение которого способствовало бы наилучшему сжатию всех типов изображений. Даже для соседних фрагментов изображений оптимальными могут оказаться разные предикторы. Поэтому для достижения лучших коэффициентов сжатия *необходимо использовать комбинации предикторов*. Понятно, что комбинирование предикторов замедляет кодирование, однако оно дает возможность не только уменьшить размеры файлов изображений, но и вследствие этого ускорить декодирование.

По стандарту *PNG* [1] рекомендуется выбирать тот предиктор для каждой строки, применение которого предопределяет минимальную сумму значений знаковых байт (из диапазона [–128; 127]). Действительно, уменьшение общей суммы значений строки после использования предиктора свидетельствует о приближении распределения частот к симметричному, но указывает ли этот критерий на уменьшение энтропии источника? Результаты применения такого способа выбора предиктора для каждой строки дают негативный результат (см. строку *Комбинирования* 1 в табл. 2).

Другой способ определения оптимального предиктора для каждой строки заключается в поиске наибольшего количества повторений одинаковых последовательных значений [8, с. 317]. Действительно, повторение значений способствует улучшению показателей сжатия алгоритмом *LZ*77, но этот способ не учитывает возможности повторений групп различных байт, что также улучшают сжатие алгоритмом *LZ*77, и изменений энтропии источника. Результаты применения такого способа выбора оптимального предиктора каждой строки приведены в табл. 2, 3 в строке *Комбинирования* 2.

В [3 и 9] предложено выбирать тот предиктор, применение которого предопределяет минимальную сумму *модулей* значений знаковых байтов (из диапазона [–128; 127]). Уменьшение общей суммы модулей значений строки после использования предиктора свидетельствует об увеличении неравномерности распределения частот вокруг нуля и улучшает результаты применения контекстно-независимого алгоритма. Результаты применения такого способа выбора оптимального предиктора каждой строки приведены в табл. 2, 3 в строке *Комбинирования* 3. Как следует из данных таблиц, такой способ выбора предиктора строки улучшает средний КС относительно предиктора Пифа на 0,26%, причем уменьшение размеров файлов наблюдается почти для всех изображений, хотя он и замедляет кодирование на 22%.

Для разработки более эффективных способов выбора предикторов строк проанализируем влияние разных предикторов на производительность базовых алгоритмов сжатия формата *PNG*. Алгоритм *LZ*77 достигает лучших КС изображений без применения предикторов после воздействия предикторов *LeftPredict* или *RightPredict*. Кодирование Хаффмана лучше всего сжимает распределения изображений после воздействия предикторов (особенно после *AveragePredict* или *PaethPredict*), поскольку они повышают неравномерность распределения. Выбрать лучший из трех вариантов использования предикторов относительно алгоритма *LZ*77 для каждой строки пикселов невозможно без выполнения трех предварительных разложений изображения с применением каждого из этих вариантов, поскольку одинаковые фрагменты данных могут встречаться в разных строках. *Лучший предиктор относительно разложения Хаффмана для каждой строки пикселов выберем из условия минимума энтропии среди результатов их действий*, к которой стремится средняя длина кода этого разложения. Пусть каждый из элементов S_i встречается N_i раз в строке длиной $N = \sum_i N_i$. Тогда $p(s_i) = N_i / N$ и общая длина закодированных данных, учитывая (1), приближается к значению

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

На языке *C* подпрограмма для приближенного вычисления энтропийного размера блока кодов Хаффмана по частотам элементов согласно (3) с определением прогнозируемого КС имеет вид:

```
  double sizeEntropiCode(long * freq, short
countAllFreq, double &entropiKC)
   {// freq – массив частот элементов
    // countAllFreq – общее количество час-
       тот в массиве
    double size=0; long n=0;
    for (long i=0; i<countAllFreq; i++)
```

```
 {n+=freq[i]; // сумируем частоты
  if (freq[i]>1) // для существования log
    size+=freq[i]*log(freq[i]); }
 if (n) {size=(n*log(n)-size) /log(2);
     entropiKC=size/(8*n); }
 else {size=0; entropiKC=1; }
 return size; }   .
```

Фрагмент программы для определения номера предиктора, порождающего данные с наименьшей энтропией элементов записывается так:

```
for (i=0; i<=4; ++i) // цикл по предикторам
 {memset(freq, 0, sizeof(freq));
  for (j=0; j<row_width; ++j) // цикл по
                              элементам строки
   freq[buffers[i][j]]++; // накопление
                              частот элементов
  size=sizeEntropiCode(freq, 256, entropiKC);
  if (size<minSize)
   {predict1=i; // запомнили номер предик
                                       тора
    minSize=size; entropiKC1=entropiKC; }
KC1=entropiKC1;    .
```

Средний КС изображений набора *ACT* с применением энтропийного способа выбора предикторов строк уменьшился в сравнении со способом выбора предикторов на основе наименьшей суммы модулей значений знаковых байтов на 0,05% (строка *Комбинирования 4* из табл. 2, 3), причем размер сжатых файлов не увеличился ни для одного изображения, а время сжатия увеличилось только на 0,82%. Использование этого способа эффективно, в первую очередь, для естественных изображений нескольких больших объектов. Для фрагментов изображений, в которых энтропийный способ выбора предикторов строк является самым эффективным, короткие замены (три–четыре литерала), как правило, не применяются. Оптимальными предикторами при таком способе выбора чаще всего есть *AveragePredict* или *PaethPredict*.

Кроме энтропии отдельных элементов, для каждой строки пикселов следует оценить также КС после выполнения коротких замен, так как они могут существенно увеличить частоты соответствующих длин замен и уменьшить частоты отдельных литералов. А это, в свою очередь, приводит к увеличению неравномерности распределения и, соответственно, к уменьшению КС. Выполнять предварительное сжатие *LZ*77 для результатов действия всех предикторов строк нецелесообразно, поскольку это приводит к резкому замедлению процесса кодирования. Оценивались КС строки пикселов после применения триэлементных замен с помощью хеш-таблицы по четырем последним битам трех смежных элементов, в которой сохраняются абсолютные позиции последних вхождений подобных групп. Используя значение этой таблицы, можно имитировать разложение *LZ*77 с подсчетом частот литералов, триэлементных замен и их смещений, а также дополнительных битов, после чего определяется общий КС строки. На языке *C* фрагмент программы для определения номера предиктора, порождающего данные с наименьшим КС, после коротких замен записывается, например, так:

```
for (i=0; ii<=4; ++i) // цикл по предикторам
 {memset(freq, 0, sizeof(freq)); // частоты
                                 литералов/длин
  memset(freqD, 0, sizeof(freqD)); // час
                                 тоты баз смещений
  memset(hash, 0, sizeof(hash)); // эле
                                 менты хеш-таблицы
  plusBit=0; // дополнительные биты смещений
  j=0; // позиция обработки строки после
                                 действия предиктора i
  while (j<row_width-2)
   if (hash[((buffers[i][j]& 15)<<8)+
              ((buffers[i][j+1]& 15)<<4)+
                (buffers[i][j+2]& 15)])
    {// подобные три байта уже были в строке
     freq[257]++; //  частота триэлементной
                                       замены
     offset=j-hash[((buffers[i][j]&15)<<8)+
    ((buffers[i][j+1]&15)<<4)+(buffers[i][j+
     +2]&15)]+1;
      // увеличиваем частоту базы смещения
      freqD[codesDistance[offset]]++;
      // накапливаем дополнительные биты
                                       смещения
      plusBit+=extrasDistance[codesDistan-
       ce[offset]];
      // данные обработанных триэлементных
                                       групп
      // записываем в хеш-таблицу
      hash[((buffers[i][j]&15)<<8)+
      ((buffers[i][j+1]&15)<<4)+
       +(buffers[i][j+2]&15)]=j+1;
      if (j+1<row_width-2)
       hash[((buffers[i][j+1]&15)<<8)+
       ((buffers[i][j+2]&15)<<4)+
         (buffers[i][j+3]&15)]=j+2;
      if (j+2<row_width-2)
       hash[((buffers[i][j+2]&15)<<8)+
       ((buffers[i][j+3]&15)<<4)+
         (buffers[i][j+4]&15)]=j+3;
      j+=3; }
    else // подобная группа отсутствует –
             заносим литерал
     {freq[buffers[i][j]]++;
       hash[((buffers[i][j]&15)<<8)+
        ((buffers[i][j+1]&15)<<4)+(buf-
           fers[i][j+2]&15)]=j+1;
       j++; }
    for (; j<row_width ; ++j) //последние
                                 позиции строки
     freq[buffers[i][j]]++;
    size=sizeEntropiCode(freq,256, entropiKC)+
         sizeEntropiCode(freqD, 30, entro-
                                 piKCD)+plusBit;
  if (size<minSize)
   {predict2=i; // запомнили номер предиктора
    minSize=size; entropiKC2=entropiKC; }
KC2=(double) minSize/(8*row_width); .
```

Средний КС изображений набора *ACT* с применением энтропийного способа выбора предикторов строк после

имитации коротких замен *LZ*77 уменьшился в сравнении со средним КС энтропийного поэлементного предиктора на 0,13% (строка *Комбинирования* 5 из табл. 2, 3), хотя улучшение наблюдается только на дискретно-тоновых и непрерывно-тоновых изображениях, для которых короткие замены эффективны. Это и не удивительно, так как расчет энтропийного КС после имитации коротких замен лучше всего из рассмотренных вариантов моделирует сжатие в формате *PNG*. Средняя длительность компрессии файлов набора *ACT* вследствие имитации коротких замен во время выбора предикторов строк возросла на 11,96%. Оптимальными предикторами в случае выбора по минимальному энтропийному КС после применения коротких замен *LZ*77 чаще всего оказываются *Left-Predict* или *RightPredict*. Использование этого способа эффективно в первую очередь для естественных изображений с многими объектами и для искусственных изображений.

Для различных изображений и даже для разных фрагментов одного изображения короткие замены *LZ*77 могут оказаться как эффективными (т.е. количество бит для сохранения замены является не большим количества бит для сохранения отдельных элементов), так и неэффективными, следовательно, предусмотреть необходимость имитации коротких замен невозможно. Поэтому для каждой строки пикселов выберем предиктор, обеспечивающий прогнозируемый минимальный КС. Получать этот КС будем по результатам сравнения минимального поэлементного КС (в предыдущих фрагментах программ – *KC*1) и минимального КС (*KC*2) после применения коротких замен *LZ*77 относительно результатов действия всех предикторов, поскольку эти варианты выбора предикторов обеспечивают в среднем наилучшие результаты. Учитывая, что длинные эффективные замены частично учтены в *KC2* и не учтены в *KC*1, выбирать предиктор, обеспечивающий *KC2*, будем только тогда, когда *KC*2 меньше *KC*1 более, чем на 4%. Иначе будем выбирать предиктор, обеспечивающий *KC*1:

```
If (KC2<KC1-0,04) currentPredict=predict2;
Else currentPredict=predict1; .
```

Такой комбинированный способ выбора предикторов строк хотя и уступает по среднему КС на 0,01% энтропийному способу после коротких замен *LZ*77 и формируется на 4,15% медленнее его (см. строку *Комбинирования* 6 в табл. 2, 3), но обеспечивает КС по всем изображениям набора не хуже способа выбора предикторов *Комбинирования* 3 на основе наименьшей суммы модулей значений знаковых байтов.

Разбиение изображений на блоки однородных строк с помощью энтропии

Каждую строку изображения можно, конечно, разместить в отдельном сжатом блоке формата *PNG*. Но в целом такое сжатие не будет самым эффективным, поскольку в заглавии каждого сжатого блока динамических кодов Хаффмана содержится описание длин кодов его распределений литералов/длин и смещений, которое может занимать свыше 1500 битов, а такие расходы недопустимы для каждой строки. С другой стороны, объединение в сжатых блоках данных произвольных смежных строк в целом уменьшает неравномерность распределения и может негативно повлиять на общий КС. Поэтому предлагается в процессе предварительного анализа изображений перед сжатием в формате *PNG объединить смежные строки с близкими прогнозируемыми энтропийными КС распределений в блоки*. Каждый однородный блок строк в процессе последующего сжатия может принадлежать одному, а может и быть разбит на несколько сжатых блоков, поскольку размер сжатого блока, как правило, не превышает 64 Кб. Но разные блоки строк должны обязательно принадлежать различным сжатым блокам, так как имеют разные энтропийные КС (в предыдущих фрагментах программ – *entropiKC*1), а следовательно, и разные неравномерности распределений. Понятно, что объединять смежные блоки строк целесообразно только тогда, когда прогнозируемое увеличение длины кодов от их объединения не превышает размера заглавия нового сжатого блока (т.е. 1500).

Рассчитаем прогнозируемое увеличение длины кодов от объединения двух смежных блоков. Пусть первый блок строк содержит l_1 элементов с прогнозируемой энтропией (средней длиной кода) e_1, а второй – l_2 элементов с энтропией e_2. Тогда прогнозируемая длина кодов первого блока составит $l_1 \times e_1$, а второго – $l_2 \times e_2$. Вследствие объединения блоков строк с подобными неравномерностями распределения (см. рис. 1,*б*) энтропия объединения не может превысить максимума энтропий объединенных частей, следовательно прогнозируемая длина кода объединения не превышает $(l_1 + l_2) \times \max(e_1; e_2)$. Поэтому прогнозируемое увеличение длины кода в результате объединения двух смежных блоков не превышает

$$\Delta l_k^{'} = (l_1 + l_2) \times \max(e_1; e_2) - l_1 \times e_1 - l_2 \times e_2 = \begin{cases} (e_1 - e_2) \times l_2, & e_1 \geq e_2 \\ (e_2 - e_1) \times l_1, & e_1 < e_2 \end{cases}. \quad (4)$$

На практике прогнозируемое увеличение длины кода от объединения зависит не только от количества элементов входного блока с меньшей энтропией, но и от количества элементов другого входного блока (например, короткий первый входной блок с высокой энтропией не может максимально увеличить энтропию второго длинного входного блока), поэтому рассчитывать его целесообразно по формуле, которая учитывает отношение размеров входных блоков:

$$\Delta l_k^{''} = \begin{cases} |e_1 - e_2| \times l_2 \times \log(l_1 / l_2 + 1), & l_1 \geq l_2 \\ |e_1 - e_2| \times l_1 \times \log(l_2 / l_1 + 1), & l_1 < l_2 \end{cases}. \quad (5)$$

Энтропийное кодирования элементов распределений выполняется в формате *PNG* после формирования разложения *LZ*77. Предопределить количество элементов такого разложения невозможно, но это количество элементов находится в монотонно неубывающей зависимости от энтропии: чем меньше энтропия, тем больше вероят-

ность существования повторений значений яркости компонентов и, следовательно, тем меньше элементов будут содержать разложение *LZ*77. Вычисляли прогнозируемое количество элементов строк после разложения *LZ*77 как произведение начального количества элементов строк и минимума из единицы и энтропии, увеличенной в 4/3 раза (считали, что строки с энтропией свыше 0,75 не поддаются сжатию *LZ*77). Алгоритм формирования блоков однородных строк представим пошагово:

1. Определить для каждой строки пикселов изображения предиктор, порождающий коды с минимальной энтропией согласно (3) и прогнозируемое количество элементов после разложения *LZ*77, как описано ранее. Отнести каждую строку к отдельному блоку строк.

2. Рассчитать для всех пар смежных блоков прогнозируемое увеличение длины кода вследствие их возможного объединения с помощью (5), учитывая максимум 64 тыс. элементов из каждого блока. Определить пару смежных блоков, порождающих наименьшее такое увеличение длины кода.

3. Если остался только один блок строк или наименьшее увеличение длины кода превышает 1500 битов, то завершить выполнение алгоритма. Иначе объединить пару смежных блоков, порождающих это увеличение и вернуться к шагу 2.

Именно этот алгоритм формирования блоков однородных строк с помощью энтропии реализован в модификациях программы, использованной для тестирования. Он позволяет уменьшить размеры большинства сжатых изображений в формате *PNG* минимум на 1 Кб.

Достичь меньших КС изображений возможно, если: разбить их на меньшие блоки однородных строк согласно (4); для каждого блока однородных строк выбрать предикторы из условия минимума, рассчитанных с помощью (3) КС среди пяти вариантов компрессии (без применения предикторов, с использованием *LeftPredict*, с применением *RightPredict*, с использованием энтропийного поэлементного способа выбора предикторов и с применением энтропийного способа выбора предикторов после осуществления коротких замен *LZ*77); укрупнить блоки однородных строк, используя (5). При этом следует учитывать также отклонение от вариантов компрессии, которые обеспечивают минимальный КС для предыдущего и следующего однородных блоков строк, поскольку изменение вариантов компрессии между соседними блоками приводит к возникновению нового сжатого блока (а следовательно, и его заглавия) и уменьшает вероятность повторений фрагментов данных для алгоритма *LZ*77. То есть *предлагается в процессе анализа изображения выбирать те предикторы, которые минимизируют коэффициент сжатия каждого блока однородных строк*, а не только энтропию компонент пикселов отдельных строк. Результаты применения такого способа сжатия приведены в строках *Комбинирования* 4, 5 *с анализом блоков* табл. 2, 3. Описанный способ сжатия изображений, выбирающий минимальный КС для каждого блока однородных строк, хотя и выполняется в среднем почти втрое медленнее энтропийного варианта и почти в шесть раз медленнее сжатия без предикторов (поскольку формирует пять дополнительных разложений *LZ*77), но обеспечивает наименьшие КС для всех изображений набора *ACT*.

Заключение. Для повышения эффективности сжатия данных в форматах, последовательно использующих алгоритмы нескольких методов, целесообразно учитывать взаимное влияние этих алгоритмов. Во время предварительной обработки данных перед сжатием в таких форматах для расчета параметров компрессии желательно учитывать все виды избыточностей, обрабатываемых их отдельными алгоритмами, а также выполнять разбиение данных на однородные блоки.

Для каждой строки пикселов изображения предикторы следует выбирать в зависимости от ограничений на время компрессии: в случае самого быстрого сжатия целесообразно использовать нелинейный предиктор *PaethPredict* для всех строк; во время стандартного сжатия стоит воспользоваться энтропийным способом выбора предикторов (*Комбинирование4*); для медленного максимального сжатия следует применить выбор предикторов на основе анализа КС пяти вариантов компрессии каждого блока однородных строк.

Предложенный способ предварительной обработки с анализом данных позволяет получить наименьшие КС изображений в формате *PNG* путем, в первую очередь, разбиения изображений на блоки строк пикселов с близкими значениями энтропии и выбора для каждого блока строк способа сжатия, обеспечивающего для него наименьший КС.

Описанные способы выбора предикторов строк и разбиения изображения на блоки однородных строк с помощью энтропии не требуют модификации декодера или программ просмотра изображений и при уменьшении размеров файлов лишь ускоряют их работу. Именно поэтому они могут быть эффективно применены в процессе компактного сохранения данных в форматах, использующих разные предикторы для отдельных строк пикселов, таких, как формат *PNG*.