computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

АНАЛІЗ ЗОБРАЖЕНЬ ПЕРЕД СТИСНЕННЯМ У ФОРМАТІ PNG

Олександр Шпортько

Рівненський державний гуманітарний університет
вул. С. Бандери, 12, м. Рівне, 33028, Україна
e-mail: chportko@ukr.net, chportko@yandex.ru

Резюме: *У статті досліджено вплив різних варіантів предикторів та їх комбінацій на стиснення зображень у форматі PNG. Наведено фрагменти програм мовою С для оцінки ентропії рядків пікселів безпосередньо та після застосування контекстно-залежного алгоритму. Пропонується спосіб аналізу зображень перед стисненням за допомогою розбиття на мінімальні блоки рядків пікселів з близькими значеннями ентропії, вибору ефективного варіанту компресії з п'яти можливих для кожного мінімального блоку з використанням методу динамічного програмування, поєднання суміжних мінімальних блоків в однорідні блоки рядків та встановлення параметрів їх кодування. Як показують експерименти, описаний аналіз зображень дозволяє, наприклад, зменшити коефіцієнти стиснення неперервно-тонових зображень набору Kodak True Color Image у форматі PNG в середньому до 12,37 bpp.*

Ключові слова: *безвтратне стиснення зображень, стратегії розкладу LZ77, формат графічних файлів PNG.*

ВСТУП

На сьогоднішній день 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 (відношення розмірів стиснутого до нестиснутого файлів зображення, надалі – КС) та висока швидкість декодування. Проблема зменшення розмірів файлів зображень у цьому форматі є актуальною на сьогодні 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. У цій статті описується спосіб розбиття зображень на блоки однорідних рядків та визначення прогнозованих параметрів компресії кожного блоку перед стисненням, що дозволяє суттєво покращити КС у форматі PNG.

1. ЕТАПИ ОБРОБКИ ЗОБРАЖЕНЬ У ФОРМАТІ PNG. ПОСТАНОВКА ЗАДАЧІ

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 на 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 LZ77 [4] стискаються контекстно-незалежними кодами Хафмана [5]. Розглянемо етапи обробки даних зображень у форматі PNG детальніше.

Контекстно-залежний алгоритм LZ77. Описуючи словникові алгоритми, фіксовану кількість попередніх закодованих неподільних елементів (літералів) вхідного потоку називають словником, а наступних незакодованих – буфером. Алгоритм LZ77 базується на заміні у вихідному потоці послідовності чергових літералів буфера посиланням на аналогічну послідовність літералів, що починається у словнику, у вигляді пари чисел *<довжина; зміщення від закінчення словника>*. У випадку 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*. Очевидно, що для досягнення стиснення довжина співпадаючої послідовності має перевищувати 2. Оскільки більші зміщення кодуються, як правило, більшою кількістю бітів, то співпадаючі послідовності шукаються у словнику з кінця справа наліво.

Під час декодування кодів, отриманих за алгоритмом LZ77, окремі літерали копіюються у 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 довжини замін та окремі літерали кодуються разом числами в межах [0; 285]. При цьому числа з діапазону [0; 255] відповідають кодам окремих літералів, 256 позначає закінчення блоку, а числа з діапазону [257; 285] вказують на базові значення довжин. Після базових значень довжин міститься визначена форматом кількість бітів, що разом з базовим значенням однозначно визначають довжину заміни. Зміщення зберігається після відповідної довжини аналогічно – у вигляді базового значення та додаткових бітів. Базове значення зміщення знаходиться в межах [0; 29]. У форматі DEFLATE максимальне значення довжини закодованої послідовності може сягати 258, а зміщення – 32768.

На практиці для формування розкладу LZ77 найчастіше використовують такі підходи [3]:

- *жадібний розклад (greedy parsing)*, при якому на кожному кроці алгоритму для чергових літералів буфера шукають *найдовшу* співпадаючу послідовність, що починається у словнику. Цей підхід забезпечує максимальну швидкість формування розкладу, оскільки зі словника в буфер щоразу переноситься максимальна кількість літералів, але не найкращі коефіцієнти стиснення, оскільки максимальні довжини замін призводять до використання максимальних зміщень та, крім цього, в процесі розкладу не відслідковується можливість використання літерала і заміни замість послідовності двох замін. Ми використовуємо жадібний розклад лише під час попереднього аналізу зображень;
- *лінивий розклад (lazy parsing)*, під час якого максимальну довжину співпадаючої послідовності для чергової позиції *порівнюють* з аналогічною довжиною для наступної позиції. Якщо максимальна довжина співпадаючої послідовності з наступної позиції довша, то з чергової позиції кодують не заміну, а літерал. Цей розклад формується довше від жадібного, оскільки виконує майже вдвічі більше пошуків співпадаючих послідовностей, хоча й забезпечує кращі КС. Ми використовуємо модифікований лінивий розклад для аналізу та стиснення зображень у форматі PNG, порівнюючи між собою не максимальні довжини замін, а прогнозований КС чергової заміни з прогнозованим КС літерала та заміни від наступної позиції (КС розраховуються з відношення прогнозованої кількості бітів для кодування до загальної кількості закодованих бітів). Якщо другий з цих КС менший від першого, то з чергової позиції кодуємо літерал, інакше – кодуємо заміну;
- *майже оптимальний розклад (almost optimum parsing)*, при якому для кожної позиції потоку *j* обчислюють *мінімальну* прогнозовану *вартість* (кількість бітів) кодування V_j від початку потоку до цієї позиції включно за формулою:

$$V_j = \min_{l<j}\left(V_l + price_{l+1,j}\right),$$

де $price_{l+1,j}$ – прогнозована вартість зберігання пари <*довжина; зміщення*> для кодування літералів з *l+1* по *j*-й, якщо *l<j-1* (вартість нескінченна, коли пари заміни для кодування цих елементів не існує) або прогнозована вартість зберігання літерала s_j, якщо *l=j-1*. Після визначення мінімальної вартості кодування останнього літерала потоку відбирають позиції, довжини та зміщення, що дозволяють досягнути цієї мінімальної вартості, і саме з них створюється закодований потік. Цей розклад формується найдовше з розглянутих, оскільки потребує пошуків співпадаючих послідовностей з усіх позицій потоку, але й забезпечує найкращі КС. Ми використовуємо цей розклад як альтернативу лінивому для стиснення зображень у форматі PNG.

Зрозуміло, що для обчислення прогнозованих КС лінивого розкладу та вартостей кодувань майже оптимального розкладу необхідно знати приблизні кількості бітів, що будуть витрачені для кодування літералів, довжин замін та зміщень. Вгадати наперед ці значення неможливо. Більше того, як буде показано далі, прогнозовані кількості бітів

кодів можуть відрізнятися для різних блоків рядків. Тому приблизні значення кількості бітів для кодування літералів, довжин замін та зміщень розраховуються в процесі попереднього аналізу зображень.

Контекстно-незалежне кодування Хафмана. Ідея використання динамічних кодів Хафмана [3, 6], що найчастіше застосовуються після виконання алгоритму LZ77 для стиснення літералів і базових значень довжин а також для відокремленого стиснення базових значень зміщень, полягає у заміні чисел з більшою частотою (тут і надалі – абсолютною) кодами меншої кількості бітів, ніж для чисел з меншою частотою. Коди Хафмана для літералів/довжин та зміщень визначаються для кожного блоку стиснутих даних окремо, що сприяє покращенню стиснення.

Згідно з формулою 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 має наближатися до *ентропії джерела* [3]:

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

Ентропія джерела зменшується при збільшенні нерівномірності розподілу ймовірностей між елементами. Середня довжина коду Хафмана співпадає з ентропією джерела лише тоді, коли для всіх елементів s_i довжини їх оптимальних кодів $-\log p(s_i)$ цілі. Чим більше $-\log p(s_i)$ відхиляються від цілих чисел, тим більшою стає різниця між середньою довжиною коду Хафмана і ентропією джерела. Алгоритми точного обчислення розміру блоків динамічних кодів Хафмана з використанням принципу їх генерації, а також алгоритм вибору найкоротшого з альтернативних стиснутих блоків у форматі PNG детально описані в [7].

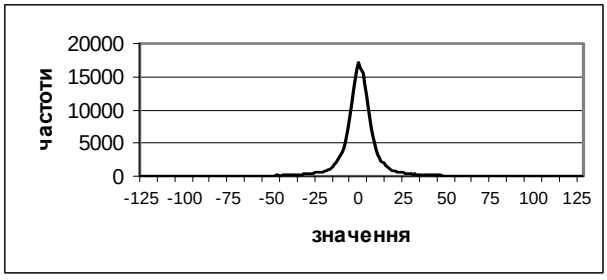
Різновиди предикторів у форматі PNG. Зменшити ентропію в процесі обробки зображень у форматі PNG намагаються за допомогою предикторів. *Предиктор* – це функція, що прогнозує (моделює) значення чергового елемента, використовуючи значення відомих суміжних елементів. Якщо піксел зображення характеризується декількома компонентами (наприклад, R, G, B), то предиктор кожної компоненти прогнозує значення згідно відповідних компонентів суміжних пікселів. У процесі використання цієї технології обчислюють і надалі кодують відхилення чергової компоненти від прогнозованого предиктором значення. 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.

Оскільки суміжні пікселі зображення мають, як правило, близькі кольори і тому близькі значення відповідних компонентів, то часто значення предиктора буде співпадати зі значенням чергового елемента, найчастіше – буде близьким до цього значення і рідко – значно відрізнятиметься від нього. Тобто більшість значень Δ_{ij} будуть близькими до 0. 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. Таким чином, предиктори хоча й не виконують безпосереднє стиснення даних, але зменшують, насамперед, КС контекстно-незалежного алгоритму [6]. Приклад перерозподілу частот значень компоненти після застосування предиктора наведено на рис. 1.

(a)



(b)

Рис. 1 – Розподіл абсолютних частот значень зеленої компоненти зображення Lena.bmp: (a) до застосування предиктора; (b) після застосування *LeftPredict*

З іншого боку, окремі предиктори можуть розбити існуючі або згенерувати нові повтори фрагментів зображення і цим вплинути на КС алгоритму LZ77.

Для однозначності декодування предиктори не можуть використовувати значення, що обходяться після чергового елемента. Оскільки елементи зображення найчастіше обходять по рядках зверху вниз, а в кожному рядку – зліва направо, то предиктори не можуть використовувати значення правіше та нижче від чергового елемента. У стиснутих блоках формату PNG даним кожного рядка передує окремий байт, що визначає предиктор, який застосовується до компонентів всіх його пікселів. На сьогодні форматом передбачено п'ять можливих значень цього байта [6], що визначає чотири різні предиктори: 0 – дані рядка не обробляються предикторами; 1 – *LeftPredict*; 2 – *AbovePredict*; 3 – *AveragePredict*; 4 – *PaethPredict*. Для опису цих предикторів позначимо значення суміжних елементів для елемента X згідно схеми:

LeftAbove	Above
Left	X

Тоді предиктори формату PNG мовою C записуються 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; }
```

На сьогоднішній день у спеціалізованому програмному забезпеченні для покращення стиснення зображень у форматі PNG найчастіше використовують метод перебору різних варіантів предикторів, розмірів блоків стиснутих даних та стратегій стиснення. **Метою** ж цієї **статті** є опис алгоритму розбиття зображень на блоки однорідних рядків та вибору для кожного блоку (а не для цілого зображення) ефективного варіанту компресії перед стисненням у форматі PNG.

2. АНАЛІЗ ВПЛИВУ РІЗНИХ ВАРІАНТІВ ПРЕДИКТОРІВ НА СТИСНЕННЯ ЗОБРАЖЕНЬ У ФОРМАТІ PNG. ОЦІНКА ЕНТРОПІЇ РЯДКІВ ПІКСЕЛІВ

Алгоритми, що використовуються у форматі PNG для стиснення даних, орієнтовані на зменшення різних видів надлишковостей: якщо LZ77 усуває надлишковості між однаковими фрагментами даних, то кодування Хафмана орієнтоване на зменшення надлишковостей між переважаючими елементами розподілів. Алгоритм LZ77 у цьому форматі може стискувати дані максимум у 129, а кодування Хафмана – максимум у 8 разів. Саме тому алгоритм LZ77 і використовується у форматі PNG, хоча він і зменшує нерівномірність розподілу між окремими елементами і може зменшити ефективність кодування Хафмана. З іншого боку, у неперервно-тонових зображеннях яскравості компонентів суміжних пікселів, як правило, близькі, але не однакові. Це робить малоефективним використання для них алгоритму LZ77, і тому кодування Хафмана у такому випадку забезпечує хоча б задовільні КС. Ключову роль в процесі стиснення різних зображень і навіть окремих їх фрагментів можуть відігравати як алгоритм LZ77, так і кодування Хафмана. Різні предиктори (чи відмова від їх використання) формату PNG орієнтуються на різні алгоритми стиснення. Відповідно, для кожного рядка зображення слід обирати той предиктор, який дозволить підсилити дію ключового алгоритму стиснення для даного фрагменту зображення і допоможе цим забезпечити найменший загальний КС.

Дослідимо вплив різних варіантів предикторів на прикладі стиснення 24 різнотипних 24-бітних неперервно-тонових зображень стандартного набору файлів Kodak True Color Images (KTCI) у форматі PNG. Завантажити ці файли можна, наприклад, з http://r0k.us/graphics/kodak/-index.html. Тестування проводилося за допомогою програми з CD до [6] з жадібним розкладом LZ77, у яку були внесені такі модифікації: забезпечена можливість виходу зі словника в буфер під час кодування повторів; запроваджений алгоритм вибору найкоротшого з альтернативних стиснутих блоків динамічних кодів Хафмана [7]; застосований аналіз кількостей додаткових бітів при записі зміщень; розмір блоків даних збільшений до 64 Кб та відкинуті допоміжні текстові блоки.

Результати тестування наведено в табл. 1, де показником компресії файлів обрано КС, виражений в bpp, тобто у кількості бітів, що в середньому витрачаються для кодування одного

піксела зображення.

Таблиця 1. Коефіцієнти стиснення (bpp) файлів зображень набору KTCI у форматі PNG після застосування різних варіантів предикторів рядків

№ файла	Без предикторів	Left Predict	Right Predict	Entropi Predict	Entropi LZ77 Predict
01	14.95	14.90	15.22	14.70	14.51
02	12.80	13.03	12.84	12.70	12.53
03	11.03	10.35	11.20	11.16	10.97
04	14.32	13.18	12.95	13.07	12.99
05	17.65	16.03	16.19	15.82	15.82
06	13.70	12.70	14.34	12.88	12.76
07	12.78	11.49	12.30	11.82	11.80
08	18.03	17.09	16.13	15.55	15.55
09	12.14	11.93	12.11	11.93	11.72
10	13.36	12.43	12.39	12.14	11.93
11	13.55	12.72	13.55	12.72	12.39
12	11.61	10.95	11.91	11.61	11.30
13	17.88	16.69	17.32	16.80	16.76
14	15.86	14.15	15.11	14.59	14.40
15	13.59	12.80	12.30	12.57	12.24
16	11.30	10.99	12.61	11.39	11.30
17	13.26	12.36	12.47	12.68	12.47
18	17.76	16.15	15.80	15.69	15.72
19	14.38	13.78	13.74	13.38	13.38
20	10.72	10.16	10.66	10.20	10.26
21	13.78	13.07	14.13	13.28	13.18
22	16.80	15.01	14.40	14.30	14.26
23	15.17	12.14	11.59	11.49	11.47
24	15.22	14.53	13.99	14.20	14.03
Average	**14.24**	**13.28**	**13.55**	**13.19**	**13.07**

Стиснення зображень у форматі PNG без застосування предикторів відбувається за рахунок повторень та наявності нерівномірності розподілу значень яскравостей пікселів. Але в неперервно-тонових зображеннях аналогічні фрагменти та однакові значення яскравостей компонентів трапляються порівняно рідко, тому й таке стиснення призводить до високих КС (рис. 1(а) та стовпець 2 в табл. 1). Стиснення зображень без застосування предикторів ефективне, насамперед, для дискретно-тонових зображень з високою роздільною здатністю. Середній час компресії файлів набору KTCI (тут і надалі – на комп'ютері з процесором AMD-K6 з частотою 300 МГц) цим способом складає 8 с.

Предиктор *LeftPredict* генерує горизонтальні прирости яскравостей між компонентами суміжних пікселів. Однакові прирости яскравостей трапляються в неперервно-тонових зображеннях, як правило, частіше, ніж однакові фрагменти, що покращує КС алгоритму LZ77. Крім цього, застосування предиктора *LeftPredict* підвищує нерівномірність розподілу навколо нуля (рис. 1 (b)), зменшуючи цим самим КС кодування Хафмана. Саме тому середній КС зображень набору KTCI з застосуванням *LeftPredict* зменшився порівняно з варіантом без застосування предикторів майже на 1 bpp (стовпець 3 з табл. 1). Використання предиктора *LeftPredict* ефективне також для штучних зображень з рівномірними приростами яскравостей компонентів пікселів по горизонталі (наприклад, у фоні). Але застосування *LeftPredict* здатне зменшувати довжини повторень, що були між фрагментами оригіналу зображення, тому може і підвищити загальний КС (дані зображення № 2 в табл. 1). Отже, однозначно стверджувати, що стиснення з використанням *LeftPredict* ефективніше від стиснення без застосування предикторів неможливо. Середня тривалість компресії файлів набору KTCI цим способом складає 9 с.

Аналогічно впливає на стиснення зображень і застосування предиктора *RightPredict*, який генерує вертикальні прирости яскравостей компонентів суміжних пікселів. Різні зображення мають різний рівень відмінностей пікселів по горизонталі та вертикалі. Саме тому КС в стовпцях 3 і 4 табл. 1 різні. Гірший середній КС предиктора *RightPredict* стосовно *LeftPredict* пояснюється тим, що однакові прирости по вертикалі зустрічаються, як правило, в суміжних рядках. Зміщення ж між суміжними пікселами по вертикалі при обробці даних у форматі PNG дорівнює довжині рядка. Тобто вони значно більші від зміщень між суміжними пікселами по горизонталі і тому кодуються значно більшою кількістю додаткових бітів.

Предиктор *AveragePredict* врівноважує вплив суміжних пікселів по горизонталі та вертикалі. Він добре прогнозує яскравості компонентів пікселів для зображень великих та розмитих об'єктів, хоча й дуже чутливий до різких перепадів кольорів (наприклад, зображень променів світла на об'єктах). Ще кращі в середньому прогнози значень пікселів генерує предиктор *PaethPredict*, який однаково добре працює як на дискретно-тонових, так і на неперервно-тонових зображеннях. Але, по-перше, він обчислюється найдовше з усіх розглянутих предикторів, що негативно впливає як на час кодування так і декодування, і, по-друге, цей предиктор, як і попередній, часто зменшує довжини повторень, що були між фрагментами оригіналу зображення.

Отже, алгоритм LZ77 досягає кращих КС зображень без дії предикторів або після застосування предикторів *LeftPredict* чи *RightPredict*. Кодування ж Хафмана найкраще стискає розподіли зображень після дії предикторів (особливо після *AveragePredict* та

PaethPredict), оскільки вони кардинально підвищують нерівномірність розподілу. Обрати найкращий з трьох варіантів використання предикторів стосовно алгоритму LZ77 для кожного рядка пікселів неможливо без виконання трьох попередніх розкладів зображення з застосуванням кожного з цих варіантів, оскільки однакові фрагменти даних можуть зустрічатися в різних рядках. А **оптимальним предиктором стосовно кодування Хафмана для кожного рядка пікселів вважатимемо той, застосування якого до даних цього рядка забезпечує мінімальну довжину ентропійного коду** (адже довжина коду Хафмана наближається до довжини цього коду). 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

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

Мовою С підпрограма для такого наближеного обчислення розміру ентропійного коду за частотами елементів має вигляд:

```
double lenEntropiCode (long *freq, short countAllFreq)
 {// 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);
  else size=0;
  return size; }
```

А фрагмент програми для визначення номера предиктора, що породжує дані з найменшою довжиною ентропійного коду записується так:

```
for (i=1; i<=4; ++i) // цикл по предикторах
 {memset(freq, 0, sizeof(freq));
  // накопичення частот елементів після дії предиктора
  for (j=0; j<row_width; ++j) // цикл по елементах рядка
   freq[buffers[i][j]]++;
  len=lenEntropiCode(freq, 256);
  if (len<minLen)
   {predict1=i; // запам'ятали номер предиктора
    minLen=len; }}
```

Середній КС зображень набору KTCI з застосуванням ентропійного способу вибору предикторів рядків (*EntropiPredict*) зменшився порівняно з предиктором *LeftPredict* на 0.09 bpp (стовпець 5 з табл. 1), хоча покращення спостерігається і не для всіх зображень (№№ 4, 15, 17, 24). Використання цього способу ефективне, насамперед, для неперервно-тонових зображень декількох великих об'єктів. Для фрагментів зображень, в яких ентропійний спосіб вибору предикторів рядків є найефективніший, короткі заміни (3-4 літерали), як правило, не використовуються. Середня тривалість компресії файлів набору KTCI з застосуванням ентропійного способу вибору предикторів складає 10 с. Оптимальними предикторами при такому способі вибору найчастіше є *AveragePredict* чи *PaethPredict*.

Крім предиктора, що забезпечує мінімальну довжину ентропійного коду окремих елементів, **для кожного рядка пікселів слід визначити також предиктор, котрий дає змогу досягнути мінімальну довжину коду після виконання коротких замін**. Адже короткі заміни можуть суттєво збільшити частоти базових значень довжин замін, що їм відповідають, та зменшити частоти окремих літералів. А це, в свою чергу, для фрагментів зображень з високою ентропією елементів може призвести до збільшення нерівномірності розподілу літералів/довжин, і відповідно, до зменшення КС. Виконувати попереднє стиснення LZ77 для результатів дії всіх предикторів над даними рядків недоцільно, оскільки це призводить до різкого сповільнення процесу кодування. Тому ми оцінювали КС результату дії кожного предиктора над даними чергового рядка пікселів після виконання триелементних замін за допомогою хеш-таблиці по чотирьох останніх бітах трьох суміжних елементів, в якій зберігаються абсолютні позиції останніх входжень подібних триплетів. Використовуючи значення цієї таблиці, імітується розклад LZ77 з підрахунком частот літералів, триелементних замін та їх зміщень і додаткових бітів, після чого визначається загальна прогнозована довжина коду. Мовою С фрагмент програми для визначення номера предиктора, що породжує дані з найменшою прогнозованою довжиною коду після виконання коротких замін записується, наприклад, так:

```
for (i=1; i<=4; ++i) // цикл по предикторах
 {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]++; // збільшуємо частоту триелем. 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[codesDistance[offset]];
  // дані оброблених триелементних груп
  // записуємо в хеш-таблицю
  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 // подібна група відсутня – заносимо літерал
  {freq[buffers[i][j]]++;// збільшуємо частоту елемента
   hash[((buffers[i][j] & 15)<<8)+((buffers[i][j+1] & 15)
        <<4)+(buffers[i][j+2] & 15)]=j+1;  j++; }
 for (; j<row_width ; ++j) // останні позиції рядка
  freq[buffers[i][j]]++;
 len=lenEntropiCode(freq, 258)+
      lenEntropiCode(freqD, 30)+plusBit;
 if (len<minLen)
  {predict2=i; // запам'ятали номер предиктора
   minLen=len; }}
```

Середній КС зображень набору KTCI з застосуванням ентропійного способу вибору предикторів рядків після виконання коротких замін LZ77 (*EntropiLZ77Predict*) зменшився порівняно з середнім КС після використання ентропійного поелементного способу вибору предикторів рядків на 0.12 bpp (стовпець 6 з табл. 1), хоча покращення спостерігається і не на всіх зображеннях (№№ 18, 20). Це й не дивно, адже розрахунок ентропійного КС після дії коротких замін найкраще з розглянутих варіантів моделює стиснення у форматі PNG. Середня тривалість компресії файлів набору KTCI з застосуванням таких предикторів складає 11 с. Оптимальними предикторами при виборі за мінімальним ентропійним КС після застосування коротких замін LZ77 найчастіше виявляються *LeftPredict* чи *RightPredict*. Використання цього способу ефективне, насамперед, для неперервнотонових зображень з багатьма об'єктами.

Отже, **для кожного рядка зображення найефективнішим може виявитися один з 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 виконання коротких замін LZ77.

3. АНАЛІЗ ЗОБРАЖЕНЬ З РОЗБИТТЯМ НА МІНІМАЛЬНІ ТА ОДНОРІДНІ БЛОКИ РЯДКІВ

Звичайно, кожен рядок зображення можна опрацювати найефективнішим для нього предиктором і стиснути в окремий блок. Але загалом таке стиснення не буде найефективнішим, оскільки, по-перше, алгоритм LZ77 в процесі стиснення чергового рядка використовує словник з даних попередніх рядків, тобто вибір предиктора рядка впливає на стиснення не лише цього рядка, а й на компресію найближчих наступних рядків, і, по-друге, в заголовку кожного стиснутого блоку динамічних кодів Хафмана міститься опис довжин кодів його розподілів літералів/довжин та зміщень, який може займати до 1324 бітів (в середньому – 1000 бітів), а такі витрати неприпустимі для кожного рядка.

Як показують експерименти, для суміжних рядків зображення з близькими нерівномірностями розподілів кожен з варіантів компресії забезпечує, яку правило, близькі КС. Тому найефективнішими для таких рядків найчастіше виявляються однакові варіанти компресії. Отже, для підвищення КС алгоритму LZ77, **поєднаємо перед кодуванням суміжні рядки з близькими нерівномірностями розподілів в мінімальні блоки** (до 32 Кб) і визначимо для кожного з них ефективний варіант компресії. Вказаний максимальний розмір мінімального блоку рядків, що рівний розміру словника LZ77, дає змогу оперативно змінювати ефективний варіант компресії між суміжними фрагментами зображення та забезпечує залежність КС чергового мінімального блоку в основному лише від попереднього мінімального блоку. Обирати варіант компресії кожного мінімального блоку рядків слід так, щоб забезпечити загальний мінімальний КС зображення. Під час вибору варіантів компресії для мінімальних блоків рядків слід враховувати також, що кожна зміна цих варіантів між суміжними блоками зменшує ймовірність повторень фрагментів даних для алгоритму LZ77 і, внаслідок зміни характеру розподілу літералів/довжин, призводить до виникнення нового стиснутого блоку (а, отже, і до необхідності зберігання його заголовка).

Для підвищення ефективності кодування Хафмана забезпечимо компресію блоків рядків з різними нерівномірностями розподілу в різних стиснутих блоках. З цією метою **поєднаємо суміжні мінімальні блоки рядків з однаковими варіантами компресії та близькими нерівномірностями розподілів в**

однорідні блоки рядків. Кожен однорідний блок рядків в процесі подальшого стиснення може належати одному, а може бути й розбитим на декілька стиснутих блоків, адже розмір стиснутого блоку, як правило, не перевищує 64 Кб. Але різні однорідні блоки рядків мають обов'язково належати різним стиснутим блокам, адже вони мають різні характери нерівномірностей розподілів.

Отже, аналіз зображень перед стисненням у форматі PNG необхідно виконувати поетапно:

1. Поєднати суміжні рядки з близькими нерівномірностями розподілів у мінімальні блоки;
2. Визначити для кожного мінімального блоку найефективніший варіант компресії (з п'яти) та відповідні предиктори рядків так, щоб забезпечити загальний мінімальний прогнозований КС зображення;
3. Поєднати суміжні мінімальні блоки рядків з однаковими варіантами компресії та близькими нерівномірностями розподілів в однорідні блоки рядків;
4. Розрахувати для кожного однорідного блоку рядків прогнозовані довжини кодів розподілів літералів/довжин та зміщень для подальшого ефективного кодування алгоритмом LZ77 з лінивим чи майже оптимальним розкладом.

Розглянемо питання доцільності поєднання суміжних мінімальних чи однорідних блоків рядків. З одного боку, кожне поєднання зменшує прогнозовану кількість заголовків стиснутих блоків на одиницю. Розмір заголовка стиснутого блоку можна розраховувати точно, орієнтуючись на його формат [6], а можна покласти рівним середньому розміру таких заголовків, тобто 1000 бітів. З іншого боку, поєднання суміжних блоків найчастіше зменшує нерівномірність розподілу і тому збільшує розмір динамічних кодів Хафмана. Алгоритм та підпрограма для розрахунку розмірів кодів окремих та сумісного блоків за частотами їх елементів наведено в [7]. Тому виконувати поєднання мінімальних та однорідних блоків слід ітеративно, обираючи щоразу пару суміжних блоків, що забезпечує максимальне зменшення прогнозованого розміру закодованих даних з врахуванням розмірів заголовків блоків (тобто пару з найближчими нерівномірностями розподілів), доки такі зменшення наявні або перевищують визначене порогове значення. Перераховувати прогнозовані зменшення розмірів між всіма суміжними блоками після кожної ітерації не потрібно. Достатньо виконати такі перерахунки лише між створеним та суміжними з ним блоками.

Покроково алгоритм попереднього аналізу зображень перед стисненням у форматі PNG, що реалізує зазначені етапи, записується так:

1. Визначити для кожного рядка пікселів номер предиктора, що забезпечує мінімальний ентропійний КС окремих елементів та номер предиктора, який дає змогу досягнути мінімального КС після виконання коротких замін, як описано вище. Запам'ятати для кожного рядка частоти окремих елементів після застосування першого з цих предикторів;
2. Віднести кожен рядок до окремого мінімального блоку рядків;
3. Ітеративно поєднати між собою суміжні мінімальні блоки рядків, що забезпечують максимальне зменшення прогнозованого розміру закодованих даних та сумарно не перевищують 32 Кб. Оскільки визначити наперед частоти елементів розкладу LZ77 для кожного мінімального блоку рядків неможливо, то для визначення прогнозованих розмірів цих блоків чи їх поєднань використати частоти елементів після застосування предикторів, що забезпечують мінімальний ентропійний КС окремих елементів рядків. Ці частоти для початкових мінімальних блоків (окремих рядків) розраховуються на першому кроці алгоритму і мають сумуватися для поєднуваних блоків. Зауважимо, що розбиття кожного зображення на мінімальні блоки рядків неоднозначне, оскільки воно залежить від оцінки розміру заголовка стиснутого блоку та порогового значення для завершення ітерацій, але вибір цих параметрів несуттєво впливає на кінцевий КС зображення. Цим кроком завершується перший етап аналізу зображень перед стисненням. Приклад меж мінімальних блоків рядків для тестового зображення (стиснуте в 4 рази по горизонталі та вертикалі зображення № 21 з набору KTCI) наведено на рис. 2 зліва. Восьмий мінімальний блок тестового зображення складається лише з останнього рядка, що містить однакові пікселі. Як видно з цього прикладу, мінімальні блоки складаються з рядків, що мають близькі нерівномірності розподілів елементів.

Рис. 2 – Межі мінімальних та однорідних блоків тестового зображення

4. Визначити для кожного мінімального блоку рядків *i* після стиснення кожним з п'яти варіантів компресії *j* (0 – без застосування предикторів, 1 –

з використанням *LeftPredict*, 2 – з застосуванням *RightPredict*, 3 – з використанням ентропійного поелементного способу вибору предикторів, 4 – з застосуванням ентропійного способу вибору предикторів після виконання коротких замін LZ77) прогнозовані розміри $size_{i,j}$ в бітах ($i = \overline{0, m-1}$, де m – кількість мінімальних блоків рядків після поєднання) за допомогою алгоритму мінімізації розмірів стиснутих блоків [7]. Запам'ятати також для кожного мінімального блоку рядків і кожного варіанту компресії частоти елементів розподілів літералів/довжин та зміщень (а не лише окремих елементів, як це було для рядків). Для реалізації цього кроку до пікселів зображення необхідно почергово застосувати кожен з п'яти варіантів предикторів, виконати жадібний розклад LZ77 отриманих результатів та визначити мінімальні розміри стиснутих даних в межах кожного з мінімальних блоків рядків. Виконання цього кроку займає 50-71 % від загального часу стиснення, причому більша частина цього часу витрачається на формування п'яти додаткових розкладів LZ77. Реалізація цього кроку добре піддається розпаралеленню з використанням паралельних процесів для кожного з п'яти варіантів компресії, адже визначення прогнозованого розміру чергового мінімального блоку рядків після застосування кожного варіанту компресії не залежить від інших варіантів компресії. Прогнозовані розміри мінімальних блоків зображення рис. 2 після стиснення різними варіантами компресії наведено в табл. 2.

Таблиця 2. Прогнозовані розміри мінімальних блоків рядків тестового зображення після стиснення різними варіантами компресії, бітів

№ мінімального блоку рядків	№ варіанта компресії				
	0	1	2	3	4
0	36103	39575	43398	37950	42293
1	62868	61936	63845	60749	61479
2	70769	66953	72486	66509	66558
3	17815	16997	18249	17232	16985
4	31019	29139	31359	29309	29201
5	114948	115870	121578	117652	117417
6	58604	55905	62519	56454	56451
7	10	15	3694	16	16

5. Визначити методом динамічного програмування для кожного мінімального блоку рядків i оптимальний варіант компресії $compres_i$ та обрати відповідні частоти елементів розподілів літералів/довжин та зміщень цього варіанту для подальшого аналізу (ці частоти були розраховані для всіх варіантів компресії на попередньому кроці). Застосувати цей метод дає змогу максимальний розмір мінімальних стиснутих блоків (32 Кб), який забезпечує залежність прогнозованого розміру стиснутого чергового блоку в основному лише від попереднього блоку.

З цією метою встановимо штрафи $fine_{k,j}$ за відхилення варіанта компресії чергового мінімального блоку рядків j від варіанта компресії попереднього блоку k (див. табл. 3).

Таблиця 3. Штрафи за відхилення між варіантами компресії суміжних блоків, бітів

№ варіанта компресії попереднього блоку	№ варіанта компресії чергового блоку				
	0	1	2	3	4
0	0	2000	2000	1800	1800
1	7000	0	1500	1300	1300
2	7000	1500	0	1300	1300
3	7000	1500	1500	0	1300
4	7000	1500	1500	1300	0

Ці штрафи обумовлені як необхідністю зберігання заголовка нового стиснутого блоку, що неодмінно виникає у випадку зміни варіанту компресії (біля 1000 бітів), так і змінами словників LZ77, що погіршують КС. Штраф у 7000 бітів за перехід від варіанта компресії з предикторами до варіанта без предикторів обумовлений відсутністю потрібного словника LZ77 для стиснення початку блоку рядків без предикторів, за рахунок якого, в основному, і відбувається його компресія. Штраф за зворотний перехід складає лише 1800 – 2000 бітів, оскільки стиснення початку блоку рядків з предикторами відбувається, в основному, за рахунок використання кодів Хафмана, хоча такий перехід і змінює кардинально елементи словника LZ77. Штраф за перехід між варіантами стиснення з предикторами складає всього 1300 – 1500 бітів, оскільки такі переходи хоча й призводять до створення нових блоків, але не змінюють кардинально елементи словника LZ77.

Під час прямого ходу методу динамічного програмування визначимо мінімальний накопичений прогнозований розмір $size'_{i,j}$ від початку зображення до кожного мінімального блоку рядків i включно за умови застосування до нього варіанта компресії $j = \overline{0,4}$:

$$size'_{0,j} = size_{0,j} ;$$

$$size'_{i,j} = \min_{k=0,4}\left(size'_{i-1,k} + fine_{k,j}\right) + \\ + size_{i,j},\ i = \overline{1,\ m-1}.$$

Тобто мінімальний накопичений прогнозований розмір зображення від початку до чергового блоку для кожного варіанту компресії визначається з умови мінімуму накопичених розмірів від початку до попереднього блоку для всіх варіантів компресії з врахуванням штрафів за відхилення між варіантами компресії цих блоків.

Мінімальний прогнозований розмір всього зображення *minSize* обчислимо з умови мінімуму накопичених розмірів до останнього блоку включно по всіх варіантах компресії, а оптимальний варіант компресії кожного блоку $compres_i$ визначимо під час зворотного ходу методу динамічного програмування так, щоб забезпечити цей мінімальний розмір:

$$minSize = \min_{j=0,4} size'_{m-1,j},$$

$$compres_{m-1} = k \mid size'_{m-1,k} = minSize,$$

$$compres_i = k \mid size'_{i+1,compres_{i+1}} = size'_{i,k} + \\ + fine_{k,compres_{i+1}},\ i = \overline{m-2,0}.$$

На практиці під час прямого ходу методу динамічного програмування для кожного блоку і кожного варіанту компресії запам'ятовують номер варіанту компресії попереднього блоку, що дозволяє мінімізувати його накопичений розмір, а в процесі зворотного ходу лише використовують ці номери. Прогнозовані накопичені розміри тестового зображення від початку до кожного мінімального блоку за умови застосування до нього різних варіантів компресії наведено в табл. 4. У цій же таблиці мінімальний прогнозований розмір всього зображення виведено на темно-сірому фоні, накопичені прогнозовані розміри блоків з оптимальними варіантами компресії, що забезпечують цей мінімум – на сірому фоні, а послідовність визначення оптимальних варіантів компресії для мінімальних блоків відображено стрілками.

Співставляючи дані табл. 2 і табл. 4, приходимо до висновку, що оптимальний варіант компресії блоку не завжди співпадає з варіантом, що забезпечує мінімальний розмір цього блоку (мінімальні блоки №№ 3, 5, 7), адже оптимальна стратегія покликана забезпечити мінімальний розмір стиснутого зображення і вона враховує зміни між варіантами компресії суміжних блоків.

Таблиця 4. Прогнозовані накопичені розміри тестового зображення для кожного мінімального блоку рядків після їх стиснення різними варіантами компресії, бітів

№ мінімального блоку рядків	№ варіанта компресії				
	0	1	2	3	4
0	36103	39575	43398	37950	42293
1	98971	100039	101948	98952	99392
2	169740	166992	172638	165161	165940
3	187555	183658	184910	182393	182925
4	218574	212797	215252	211702	212126
5	333522	328667	334780	329354	329543
6	392126	384572	392696	385808	385994
7	391582	384587	389766	385824	385888

Цей крок завершує другий етап аналізу зображень перед стисненням. Наступні два кроки реалізують третій, а останній – четвертий етап аналізу;

6. Віднести кожен мінімальний блок рядків до окремого однорідного блоку;
7. Ітеративно поєднати суміжні однорідні блоки рядків з однаковими варіантами компресії, що забезпечують максимальне зменшення прогнозованого розміру закодованих даних, в однорідні блоки рядків. Для розрахунку прогнозованих розмірів блоків та їх поєднань використати частоти розподілів літералів/довжин та зміщень, обраних на п'ятому кроці. Під час поєднань однорідних блоків сумувати частоти відповідних розподілів. Межі однорідних блоків рядків для тестового зображення наведено на рис. 2 справа. Бачимо, що однорідні блоки рядків дають змогу поєднати мінімальні блоки рядків з близькими нерівномірностями розподілів та ліквідовують найменші такі блоки для зменшення КС;
8. Розрахувати для кожного однорідного блоку рядків прогнозовані довжини кодів розподілів літералів/довжин та зміщень з використанням прогнозованих частот цих розподілів за алгоритмом генерації кодів Хафмана [3, 5, 6] для подальшого ефективного кодування алгоритмом LZ77 з лінивим чи майже оптимальним розкладом у різні стиснуті блоки формату PNG.

Уточнити розраховані для кожного однорідного блоку рядків прогнозовані довжини кодів Хафмана розподілів літералів/довжин замін та зміщень (у випадку відсутності обмежень по часу) можна з розподілів, що утворюються за результатами додаткового проходу по даних зображення з застосуванням обраних предикторів, лінивого розкладу LZ77, кодування Хафмана та алгоритму мінімізації розміру стиснутих блоків [7].

4. РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

На завершення розглянемо результати застосування описаного способу попереднього аналізу зображень для стиснення у форматі PNG файлів вже згаданого набору KTCI програмою з CD до [6] з описаними вище модифікаціями та, додатково, з розрахунками можливостей поєднань блоків лише за частотами розподілів літералів/довжин замін (для економії оперативної пам'яті). Відповідні коефіцієнти стиснення наведено в табл. 5, а час компресії – в табл. 6. Результати аналізу застосовувалися для лінивого та майже оптимального розкладу LZ77 безпосередньо (відповідно, четвертий та шостий стовпці таблиць) та після додаткового проходу для уточнення прогнозованих довжин кодів (відповідно, п'ятий та сьомий стовпці).

Таблиця 5. Коефіцієнти стиснення файлів зображень набору KTCI у форматі PNG після застосування різних варіантів програм, bpp

№ файла	OptiPng стандарт	OptiPng макс.	lazy parsing	lazy parsing з проходом	Almost optimum parsing	Almost optimum parsing з проходом
01	14.70	14.65	14.03	14.03	13.59	13.59
02	12.57	12.55	12.22	12.20	11.95	11.91
03	11.01	10.22	10.22	10.20	9.89	9.87
04	12.97	12.84	12.61	12.59	12.36	12.32
05	15.76	15.74	15.61	15.59	15.34	15.34
06	12.93	12.59	12.55	12.53	12.16	12.16
07	11.51	11.34	11.30	11.26	11.01	10.97
08	15.44	15.42	15.38	15.34	15.13	15.11
09	11.86	11.80	11.18	11.16	10.74	10.72
10	12.07	12.05	11.72	11.68	11.41	11.34
11	12.68	12.61	12.16	12.14	11.84	11.80
12	11.57	10.80	10.68	10.66	10.26	10.24
13	16.86	16.71	16.51	16.49	16.13	16.13
14	14.55	14.07	13.99	13.97	13.63	13.61
15	12.47	12.20	11.99	11.97	11.72	11.68
16	11.30	10.87	10.80	10.78	10.39	10.39
17	12.59	12.24	12.07	12.05	11.74	11.70
18	15.61	15.59	15.49	15.44	15.24	15.24
19	13.43	13.41	13.03	12.99	12.68	12.63
20	10.18	10.01	9.93	9.91	9.72	9.66
21	13.45	12.95	12.68	12.66	12.22	12.22
22	14.28	14.26	14.05	14.01	13.82	13.78
23	11.34	11.32	11.20	11.18	11.01	10.95
24	14.11	13.93	13.80	13.78	13.53	13.51
Average	**13.14**	**12.92**	**12.72**	**12.69**	**12.40**	**12.37**

Крім цього, для порівняння ефективності стиснення у другому та третьому стовпцях таблиць наведено результати популярної серед Web-дизайнерів програми OptiPng (http://www.optipng.sourceforge.net), яка генерує короткі PNG-файли за результатами, відповідно, стандартного та максимального перебору предикторів, розмірів стиснутих блоків та стратегій стиснення.

Таблиця 6. Час стиснення файлів зображень набору KTCI у форматі PNG різними варіантами програм на комп'ютері з процесором AMD-K6 з частотою 300 МГц, с

№ файла	OptiPng стандарт	OptiPng макс.	lazy parsing	lazy parsing з проходом	Almost optimum parsing	Almost optimum parsing з проходом
01	23	810	31	37	37	42
02	33	1090	36	44	45	52
03	45	1281	39	48	57	65
04	28	1030	36	44	45	52
05	21	791	32	39	37	43
06	33	1018	33	40	42	48
07	41	1487	38	47	53	61
08	24	482	32	38	36	42
09	36	706	37	43	44	50
10	36	679	36	45	47	54
11	36	596	33	40	44	51
12	38	686	36	42	46	53
13	20	433	30	36	34	38
14	24	484	31	37	36	42
15	35	618	38	45	49	56
16	31	636	35	41	45	51
17	31	650	37	44	46	53
18	21	484	32	38	36	42
19	28	526	37	44	44	51
20	57	813	42	50	59	68
21	29	830	35	41	42	47
22	24	940	34	41	39	46
23	37	822	42	52	55	65
24	29	626	34	41	43	49
Average	**32**	**772**	**35**	**42**	**44**	**51**

Визначення предикторів рядків та кожен жадібний розклад LZ77 зображень набору KTCI в середньому триває 5 с, лінивий розклад LZ77 та додатковий прохід для уточнення прогнозованих довжин кодів – 7 с, майже оптимальний розклад LZ77 – 9 с. Аналізуючи дані табл. 1, 5 та 6, приходимо до висновків, що в середньому по набору KTCI:

- ентропійний вибір предикторів рядків після застосування коротких замін LZ77 (стовпець 6 з табл. 1) дозволяє досягнути кращих КС, ніж програма OptiPNG з параметрами стандартного перебору (стовпець 2 з табл. 5), витрачаючи при цьому в 2.91 рази менше часу;
- використання описаного способу

попереднього аналізу зображень перед лінивим розкладом LZ77 дає змогу отримати кращі КС від програми OptiPNG з параметрами максимального перебору на 0.2 bpp, а перед майже оптимальним розкладом LZ77 – на 0.52 bpp, витрачаючи для цього відповідно у 22 та 17 разів менше часу, причому зменшення КС спостерігається для всіх зображень набору;

- застосування додаткового проходу для уточнення прогнозованих довжин кодів розподілів однорідних блоків дає змогу зменшити КС на 0.03 bpp, хоча й збільшує час кодування у середньому на 7 с;
- попередній аналіз зображень дозволив, ефективно використовуючи розбиття зображень на мінімальні та однорідні блоки рядків, прогнозування довжин кодів розподілів, лінивий та майже оптимальний розклад LZ77, зменшити КС файлів набору до 12.37 bpp (на 783 Кб стосовно загального розміру оригінальних файлів набору у форматі PNG).

5. ВИСНОВКИ

1. У форматі PNG 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 швидкого стиснення варто скористатися ентропійним поелементним способом вибору предикторів; для звичайного стиснення найкраще застосувати ентропійний спосіб вибору предикторів після виконання коротких замін LZ77; у випадку повільного максимального стиснення доцільно використати предиктори, визначені за допомогою попереднього аналізу згідно запропонованого у цій статті алгоритму.
2. Розглянутий спосіб попереднього аналізу зображень перед стисненням дає змогу отримати найменші КС у форматі PNG у порівнянні з результатами інших програм, відомих авторам, за рахунок: поділу зображень на мінімальні блоки рядків пікселів з близькими значеннями ентропії; вибору за допомогою методу динамічного програмування для кожного мінімального блоку рядків варіанта компресії, що забезпечує мінімальний прогнозований КС всього зображення; поєднання мінімальних блоків рядків в однорідні блоки та визначення прогнозованих довжин кодів розподілів отриманих блоків для наступного LZ-стиснення.
3. Розбиття зображень на блоки рядків пікселів з близькими значеннями ентропії з наступним поблочним стисненням дає змогу підвищити нерівномірність розподілу окремих фрагментів і тому покращує КС контекстно-незалежного алгоритму, а отже може з успіхом використовуватись у всіх форматах, де такі алгоритми застосовуються.
4. Реалізація описаного способу попереднього аналізу зображень не вимагає модифікації декодерів чи програм перегляду зображень і за рахунок зменшення розмірів файлів лише прискорює їх роботу. Саме тому розглянутий алгоритм доцільно використовувати для забезпечення мінімальних КС зображень у форматі PNG та інших подібних форматах.

6. СПИСОК ЛІТЕРАТУРИ

[1] T. Boutell et all. *PNG Specification*. Version 1.0, PNG Development Group, October 1996.

[2] L. Peter Deutsch. *DEFLATE Compressed Data Format Specification*. Version 1.3, RFC 1951, 1996.

[3] *Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео.* Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. М.: ДИАЛОГ-МИФИ, 2003, с. 17–106.

[4] J. Ziv., A. Lempel. A universal algorithm for sequential data compression, *IEEE Transactions on Information Theory*, May 1977, vol. 23, no. 3, pp. 337–343.

[5] D. Huffman. A method for the construction of minimum-redundancy codes, *Proceedings of the I.R.E.*, September 1952, vol. 40, no. 9, pp. 1098–1101.

[6] Миано Дж. *Форматы и алгоритмы сжатия изображений в действии: учеб. пособ.* М.: Триумф, 2003, с. 249–318. – (Практика программирования).

[7] Шпортько О. В. Вибір найкоротшого з альтернативних стиснутих блоків динамічних кодів Хафмана у форматі PNG, *Комп'ютинг,* 2009, Том. 8, вип. 2, с. 58-67.

***Олександр Шпортько**, аспірант кафедри інформатики та прикладної математики Рівненського державного гуманітарного університету. Закінчив Рівненський державний педагогічний інститут у 1997 році за спеціальністю "Прикладна математика". Наукові інтереси: стиснення даних без втрат, теорія інформації, теоретичні основи програмування, проектування та розробка баз даних.*

computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

IMAGES ANALYSIS BEFORE THE COMPRESSION IN PNG FORMAT

Alexander Shport'ko

Rivne State Humanitarian University,
12, S. Bandery street, Rivne 33028 Ukraine
e-mail: chportko@ukr.net, chportko@yandex.ru

Abstract: *An influence of predictor's different variants and their combination on PNG image compression is examined in this article. Fragments in C programming language for estimation of pixel lines entropy before and after application of context-dependent algorithm have been presented. A method for image analysis before the compression through breakdown of pixel lines with close entropy values into minimum blocks is offered. The method is based on the choice of an effective compression variant from five possible ones for every minimum block using dynamic programming and on the combination of contiguous minimum blocks in the homogeneous blocks of lines and on adjustment of their code parameters. It is experimentally showed that the images analysis, for example, allows to decrease the continuous tone images compression aspect of Kodak True Color PNG format image set on the average up to 12,37 bpp.*

Keywords*: lossless images compression, LZ77 diving strategy, PNG files graphic format.*

At present time PNG is one of the widespread lossless formats for images storing. The Internet offers over 15 million pages containing images in this format, so reducing image file's size is an actual problem and its solution will help to accelerate downloading PNG-files from the network and to improve the disk space usage.

This article analyzes known algorithms for PNG format image processing, in particular: (i) Context-dependent algorithm LZ77, (ii) Context-independent Hafman encoding, (iii) Types of predictors in PNG format. On the basis of provided analysis was suggested method of image partitioning into blocks of uniform lines and determination of the predicted compression settings before each block compression to significantly improve the compression coefficient for PNG format.

Results of experimental studies are following:

1. For each image pixels line in PNG format the predictors should be chosen according to compression time limitations: in case of fastest compression it is rational to use nonlinear predictor *PaethPredict* for all lines; in case of rapid compression the entropic method of piecewise predictor selection is the best; for the ordinary compression the entropic method of predictors choice after implementation of short replacements LZ77 is worth trying. In case of slow maximal compression it is rational to use predictors determined by previous analysis according to the offered algorithm.
2. The presented method of previous analysis of images before the compression enables to receive the least AR in the format of PNG in comparison to those, obtained using other programs of famous authors for diving of images into minimum blocks of pixel lines with the close entropy values, the choice for every minimum block of lines in the variant of compression which provides minimum forecasted AR of all the image, combination of minimum blocks and determination of forecasted code lengths of division for the received blocks in the next LZ-compression.
3. Dividing of the images into blocks of pixel lines with close entropy values with the future block-structured compression enables to raise the unevenness of division for separate fragments, that is why promote the unevenness in division of separate fragments and that is why it improves AR of context-independent algorithm and consequently can successfully be used in all formats where such algorithms are used.
4. The realization of the developed method for a previous analysis of images does no require modifications in decoders or image-viewing programs and due to the decreased sizes of files only speedups their work. For this reason the algorithm is rational to be used for providing of minimum AR for PNG images.