# Пристосування формату словникової компресії Deflate до ієрархічного стиснення зображень без втрат

*Олександр Шпортько*

Рівненський державний гуманітарний університет
вул. С. Бандери, 12, Рівне, Україна
chportko@ukr.net, chportko@yandex.ru

## Abstract

The expedience of progressing hierarchical lossless image compression is justified, the method of round of pixels and basic static predictors for realization of such compression are described. The algorithm of generation of the alternative compressed Deflate-blocks, choice of the shortest block from alternative and iterative diminishing of its length for the improvement of compression of information are described. The modifications of format Deflate for the increase of efficiency of hierarchical lossless image compression with the use of a few dictionaries but sliding windows are offered and grounded. As experiments show, combined application of the offered algorithms for optimization of Deflate-blocks enables to decrease aspect of images ratios approximately on 6 % but to accelerate decoding on 20 %.

## 1. Вступ

На сьогодні всі алгоритми стиснення зображень поділяються на два основні класи – з втратами та без втрат. Алгоритми компресії з втратами забезпечують значно кращі показники стиснення і тому використовуються для зберігання більшості фотореалістичних зображень, хоча й відтворюють образи з незначними відхиленнями від оригіналів. Алгоритми компресії без втрат виконують стиснення значно слабше, проте продукують образи, ідентичні оригінальним, що є визначальним фактором для збереження дискретно-тонових зображень.

Опрацювання яскравостей пікселів зображень у популярних графічних форматах, які виконують стиснення без втрат, найчастіше здійснюється послідовно по рядках зверху вниз, а у кожному рядку – поспіль зліва направо [1]. Як наслідок, вивести стиснуте зображення у цих форматах можливо лише після декодування всіх пікселів. Поряд з цим, для прискорення виводу великих зображень у форматах компресії з втратами найчастіше застосовують прогресуюче (поступальне) ієрархічне опрацювання пікселів [2, с. 176], яке дозволяє зупинити декодування громіздких зображень відразу після заповнення області виводу, не виконуючи декомпресію всього файла і тим самим пришвидшує цей процес. Тому розробка формату графічних файлів з реалізацією прогресуючого ієрархічного стиснення зображень без втрат **є на сьогодні актуальним завданням.**

## 2. Принципи стиснення зображень без втрат

Загальновідомо, що стиснення зображень без втрат відбувається **максимум в три етапи** [3]: на першому яскравості пікселів перетворюються за допомогою предикторів; на другому контекстно-залежне кодування зменшує надлишковості між окремими фрагментами; на третьому контекстно-незалежне кодування усуває надлишковості між переважаючими значеннями яскравостей компонентів пікселів.

Серед контекстно-залежних алгоритмів у форматах графічних файлів найчастіше використовується словниковий алгоритм LZ77, оскільки він забезпечує найшвидше декодування [4, с. 82]. 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 словника 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.

Основний принцип контекстно-незалежного кодування – **довжина коду довільного елемента з більшою ймовірністю не повинна перевищувати довжину коду будь-якого елемента з меншою ймовірністю.** Середня довжина коду довільного елемента блоку $s_i$ після застосування будь-якого контекстно-незалежного алгоритму, згідно з формулою of Shannon [5, с. 611], не може бути меншою *ентропії джерела*

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

Підвищити ефективність контекстно-незалежного кодування в процесі стиснення зображень без втрат найчастіше намагаються за допомогою *предикторів* [6], які під час обходу прогнозують значення яскравості кожної компоненти чергового піксела, використовуючи значення яскравостей тих самих компонентів опрацьованих раніше суміжних пікселів, оскільки дані яскравості мають між собою найбільшу степінь кореляції [7, с. 675]. В процесі використання цього підходу обчислюють і надалі кодують відхилення $\Delta_{uv}$ значення яскравості чергової компоненти піксела $F_{uv}$ від прогнозованого обраним предиктором значення $predict_{uv}$, тобто

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

($u$ та $v$ пробігають відповідно по всіх рядках та стовпцях компонентів пікселів зображення). Застосування предикторів найчастіше збільшує нерівномірність розподілу ймовірностей значень яскравостей і, як наслідок, зменшує ентропію (1).

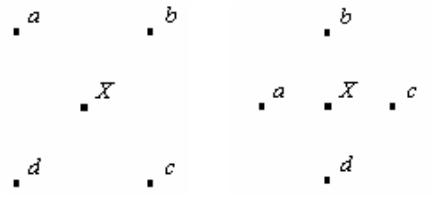
Для реалізації алгоритму LZ77 та контекстно-незалежного кодування в процесі прогресуючого ієрархічного стиснення зображень без втрат ми модифікували і використали формат Deflate [8], оскільки він успішно застосовується, наприклад, в популярному архіваторі PKZIP та графічному форматі PNG [1, с. 249-318] і не потребує придбання ліцензій для використання в прикладному ПЗ [1, с. 284]. Для реалізації контекстно-незалежного кодування ми модифікували формат Deflate, застосувавши замість кодування Хафмана арифметичний range-кодер Е. Шелвіна [4, с. 361-363]. Цей кодер виконує зчитування/запис байтів, а не бітів даних і за рахунок цього значно прискорює виконання кодування/декодування.

## 3. Схема обходу пікселів та ієрархічні предиктори для реалізації прогресуючого стиснення зображень без втрат

Для реалізації прогресуючого ієрархічного стиснення зображень без втрат ми розробили схему обходу, за якою на першому шарі піксели зображення опрацьовуються послідовно, починаючи з першого, по рядках зверху вниз, а у кожному рядку – підряд зліва направо з кроком $h_1 = 2^k$, де $k$ визначається з умови $k = \left\lfloor \log_2 \left( \frac{\max(\min(height; width); 16) - 1}{15} \right) \right\rfloor$, $height$ – кількість рядків, $width$ – кількість стовпців пікселів зображення. Цей крок забезпечує опрацювання на першому шарі принаймні 16 пікселів (при наявності) по кожній з осей (як у піктограмах). На наступних шарах ($l = \overline{2, k+1}$) проміжні піксели зображення обробляються в три проходи з кроком $h_l = 2^{k+2-l}$ як по рядках, так і по стовпцях: на першому послідовно опрацьовуються ті з них, які містяться на перетині діагоналей квадратів з вершинами у суміжних пікселах попередніх шарів, на другому обходяться рівновіддалені піксели між пікселами першого проходу по горизонталі, а на третьому – по вертикалі (рис. 1).

| | | | | | |
|---|---|---|---|---|---|
| П | 3 | П | 3 | П | ... |
| 2 | 1 | 2 | 1 | 2 | ... |
| П | 3 | П | 3 | П | ... |
| 2 | 1 | 2 | 1 | 2 | ... |
| П | 3 | П | 3 | П | ... |
| ... | ... | ... | ... | ... | ... |

*Рисунок 1*: Схема прогресуючого ієрархічного опрацювання пікселів зображення на шарах, починаючи з другого: П – піксели попереднього шару; 1 – піксели першого проходу; 2 – піксели другого проходу; 3 – піксели третього проходу




(а) (б)

*Рисунок 2*: Схеми розміщення суміжних опрацьованих елементів для елемента $X$ на шарах, починаючи з другого: а) для першого проходу; б) для другого та третього проходів

Для опису базових ієрархічних предикторів позначимо значення яскравостей аналогічних компонентів **найближчих** (суміжних) опрацьованих раніше пікселів з попереднього і чергового шару чи проходу через *a, b, c, d* (рис. 2). Фактично, на третьому проході можна було б використати всі вісім суміжних опрацьованих елементів навколо чергового елемента (див. рис. 1), але, як показали результати експериментів, це суттєво не впливає на ентропію (1) після застосування предикторів. Використовуючи ці позначення, розглянемо принципи прогнозування двох основних симетричних ієрархічних предикторів [3], орієнтованих на базове середнє арифметичне тих двох протилежних елементів з найближчих чотирьох, які найменше між собою відрізняються. Перший з цих предикторів (*ProgresPredict1*) повертає базове середнє арифметичне, а другий (*ProgresPredict2*) – визначає і повертає найближче значення серед *a, b, c, d* до базового середнього арифметичного, коли таке значення єдине. Якщо ж серед суміжних опрацьованих елементів є два найближчих рівновіддалених значення до базового середнього арифметичного, то серед них повертається те, яке частіше повторюється, а коли й кількість їх повторень однакова – то менше з цих двох значень. Загальна орієнтація на менші значення дає змогу змістити ненульові відхилення, обчислені згідно (2), в бік додатних значень і цим самим підвищити нерівномірність їх розподілу.

## 4. Пристосування алгоритму LZPR до формату Deflate

Для модифікації формату Deflate ми застосували спрощений варіант алгоритму LZPR [10], врахувавши, що найбільше однакових послідовностей виявляється, як правило, між даними пікселів вхідного зображення без застосування предикторів. Тому пошук однакових послідовностей ми виконували серед пікселів вхідного зображення, зберігаючи зміщення між пікселами, а не між байтами, а окремі елементи переносили з потоку результатів застосування предиктора з найменшою ентропією для кожного рядка чергового проходу. Це дозволило, по-перше, не змінювати внутрішню структуру зберігання результатів алгоритму LZ77 у форматі Deflate, по-друге, зменшити середні КС зображень за рахунок використання фактично двох словників і, по-третє, прискорити декодування, оскільки результати застосування замін LZ77 не потрібно в подальшому розкодовувати.

## 5. Організація додаткового пошуку однакових послідовностей алгоритму LZ77 з найближчих опрацьованих пікселів

Як слідує з реалізації спрощеного варіанту алгоритму LZPR [10], під час пошуку однакових послідовностей найменші зміщення у словнику будуть відповідати опрацьованим пікселам поточного рядка чергового проходу, більші – пікселам попереднього рядка цього ж проходу, ще більші – пікселам попередніх проходів.

З іншого боку очевидно, що однаковий колір з кольором чергового піксела найімовірніше може зустрітися серед кольорів суміжних опрацьованих раніше пікселів (рис. 1). Такі піксели можуть опрацьовуватися на різних шарах чи проходах, а, отже, можуть мати різні зміщення у словнику чи навіть знаходитися вже поза ним. Відповідно, послідовності, що починаються з суміжних елементів, теж можуть виявитися рознесеними по словнику чи навіть в нього не входити. Тому пошук однакових послідовностей доцільно виконували не лише у словнику, а й починаючи з суміжних опрацьованих раніше елементів пікселів (рис. 2) з кроком чергового проходу. Якщо, розпочинаючи з цих елементів, для чергових елементів буфера вдавалося віднайти довші однакові послідовності, ніж у словнику, то ми кодували їх найменшими можливими зміщеннями (від 1 до 4). Тому, для забезпечення однозначності декодування, зміщення словника довелося кодувати числами, збільшеними на 4, що внесло додаткові модифікації у формат Deflate.

*Таблиця 1*: Ентропія яскравостей компонентів пікселів зображень набору ACT після застосування різних варіантів програм, bpb

| Варіант програми | № файла | | | | | | | | Середня ентропія |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| З застосуванням всіх замін LZ77 | 2.41 | 1.25 | 5.32 | 4.33 | 4.67 | 5.39 | 1.26 | 4.76 | 3.67 |
| З аналізом альтернативних Deflate-блоків | 2.17 | 1.25 | 4.75 | 4.05 | 4.22 | 5.39 | 1.26 | 4.45 | 3.44 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та застосуванням всіх замін LZ77 | 2.52 | 1.04 | 5.09 | 4.54 | 4.73 | 5.47 | 1.17 | 4.94 | 3.69 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та аналізом альтернативних Deflate-блоків | 2.01 | 1.02 | 4.74 | 3.99 | 4.21 | 5.27 | 1.01 | 4.43 | 3.34 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та з найближчих опрацьованих пікселів, з аналізом альтернативних Deflate-блоків | 1.47 | 0.71 | 4.73 | 3.97 | 4.21 | 5.26 | 0.72 | 4.43 | 3.19 |

*Таблиця 2*: Час кодування зображень набору ACT різними варіантами програм, с

| Варіант програми | № файла | | | | | | | | Середній час |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| З застосуванням всіх замін LZ77 | 15.44 | 24.77 | 6.15 | 11.97 | 6.98 | 9.45 | 10.10 | 10.71 | 11.95 |
| З аналізом альтернативних Deflate-блоків | 18.24 | 25.71 | 6.87 | 13.07 | 7.74 | 10.33 | 10.44 | 11.81 | 13.03 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та застосуванням всіх замін LZ77 | 16.81 | 25.59 | 6.15 | 9.01 | 6.10 | 9.28 | 10.44 | 9.18 | 11.57 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та аналізом альтернативних Deflate-блоків | 17.68 | 26.48 | 6.37 | 9.83 | 6.48 | 10.05 | 11.20 | 9.78 | 12.23 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та з найближчих опрацьованих пікселів, з аналізом альтернативних Deflate-блоків | 15.38 | 28.17 | 6.38 | 10.27 | 6.59 | 10.55 | 11.15 | 10.49 | 12.37 |

*Таблиця 3*: Час декодування зображень набору ACT після застосування різних варіантів програм, с

| Варіант програми | № файла | | | | | | | | Середній час |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| З застосуванням всіх замін LZ77 | 6.10 | 9.45 | 2.69 | 3.62 | 2.52 | 3.90 | 3.96 | 3.79 | 4.50 |
| З аналізом альтернативних Deflate-блоків | 6.48 | 9.61 | 2.41 | 3.46 | 2.36 | 3.95 | 4.01 | 3.52 | 4.48 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та застосуванням всіх замін LZ77 | 5.33 | 4.78 | 2.30 | 2.97 | 2.14 | 3.25 | 1.98 | 3.19 | 3.24 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та аналізом альтернативних Deflate-блоків | 5.50 | 4.72 | 2.31 | 3.08 | 2.14 | 3.24 | 2.36 | 3.24 | 3.32 |
| З пошуком однакових послідовностей пікселів у словнику даних зображення та з найближчих опрацьованих пікселів, з аналізом альтернативних Deflate-блоків | 4.78 | 6.05 | 2.41 | 3.24 | 2.30 | 3.36 | 2.53 | 3.41 | 3.51 |

## 6. Результати експериментів

На завершення розглянемо результати застосування описаних та згаданих алгоритмів (табл. 1-3) для компресії восьми різнотипних 24-бітних зображень стандартного набору файлів ACT (завантажити їх TIFF-версії можна, наприклад, з http://compression.ru/arctest/act/act-files.html). Цей набір містить як синтезовані (№№ 1 (з шумами), 2, 7), так і фотореалістичні (всі інші) зображення. Тестування виконували на комп'ютері з процесором AMD-K6, 300МГц, 128 Mb RAM модифікаціями програми з CD до [1].

Бачимо, що аналіз альтернативних Deflate-блоків [10] без використання спрощеного алгоритму LZPR дає змогу зменшити КС зображень в середньому по набору ACT на 0.23 bpb, а у випадку його застосування – на 0.35 bpb, хоча й збільшує час кодування на 7-9 %. Цей алгоритм ефективний насамперед для фотореалістичних зображень та дискретно-тонових зображень з шумами.

Використання спрощеного алгоритму LZPR без аналізу альтернативних Deflate-блоків (ефективний, в першу чергу, для дискретно-тонових зображень без шумів) в середньому навіть збільшує КС, але сукупне їх використання в середньому зменшує цей показник на 0.33 bpb, сповільнює кодування на 2.34 % та прискорює декодування на 26 %.

Додаткове використання алгоритму пошуку однакових послідовностей, починаючи з найближчих опрацьованих пікселів, в середньому зменшує КС на 0.15 bpb, хоча й сповільнює кодування на 1 % та декодування – на 6 %. Сумісне застосування розглянутих модифікацій у середньому зменшує КС на 0.48 bpb, сповільнює кодування на 3.5 %, але прискорює декодування на 20 %.

## 7. Висновки і перспективи подальших досліджень

1. Зі збільшенням номера шару і номера проходу (якщо другий і третій проходи сприймати разом) ієрархічного обходу нерівномірність розподілу ймовірностей елементів після використання предикторів зростає і, як наслідок, ентропія (1) зменшується, тому для зменшення КС зображень дані різних проходів мають стискуватися у різні блоки модифікованого формату Deflate.

2. Для зменшення розміру чергового Deflate-блоку з усіма можливими замінами доцільно: розрахувати параметри альтернативних стиснутих блоків [10] з різними максимальними довжинами неврахованих замін; обрати серед них найкоротший блок; ітеративно зменшити його довжину та використати для зберігання даних. Цей алгоритм дозволяє відкидати неефективні заміни LZ77 Deflate-блоків скрізь, де такі блоки використовуються.

3. Застосовуючи формат словникового стиснення Deflate для компресії зображень без втрат, у нього доцільно внести такі модифікації: використати арифметичне стиснення замість кодування Хафмана; кодувати зміщення не між окремими компонентами, а між цілими пікселами; віднаходити однакові послідовності між пікселами без застосування предикторів, а у випадку їх відсутності – кодувати яскравості компонентів чергового піксела після їх використання, тобто застосувати спрощений алгоритм LZPR.

4. Сукупне використання алгоритму аналізу альтернативних Deflate-блоків, спрощеного алгоритму LZPR та додаткового пошуку однакових послідовностей, починаючи з найближчих опрацьованих пікселі у середньому зменшує КС на 6 % та значно прискорює декодування. Саме тому ці алгоритми доцільно комплексно застосовувати в процесі стиснення зображень без втрат.

Надалі, з метою додаткового зменшення розмірів файлів стиснутих зображень без втрат і прискорення декодування, нами планується підвищити ефективність застосування симетричних і асиметричних предикторів за допомогою різницевих колірних моделей [11], які планується використати як для цілих сцен, так і для окремих їх фрагментів.

## 8. Список використаних джерел


[1] *Миано Дж.* Форматы и алгоритмы сжатия изображений в действии: Учеб. пособ. – М.: Триумф, 2003. – 336 с., ил. – (Серія: Практика программирования).

[2] *Сэломон Д.* Сжатие данных, изображений и звука. – М.: Техносфера, 2006. – 368 с. – (Серія: Мир программирования: цифровая обработка сигналов).

[3] *Шпортько О. В.* Використання предикторів в процесі прогресуючого ієрархічного контекстно-незалежного стиснення зображень без втрат // Вісник Національного університету "Львівська політехніка". – 2013. – № 771. – С. 354-364. – (Серія: Комп'ютерні науки та інформаційні технології).

[4] *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М. : ДИАЛОГ-МИФИ, 2003. – 384 с.

[5] *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. – М.: Техносфера, 2005. – 1072 с.

[6] *Бредихин Д. Ю.* Сжатие графики без потерь качества [Електронний ресурс]. – 2004. – Режим доступу: http://www.compression.ru/download/articles/i_lless/bredikhin_2004_lossless_image_compression_doc.rar.

[7] *Прэтт Э.* Цифровая обработка изображений: Пер. с англ. – М.: Мир, 1982. – Кн. 2, 480 с., ил.

[8] *Deutsch P.* DEFLATE Compressed Data Format Specification version 1.3 // RFC 1951, 1996, Alladin enterprises. – May 1996. – 15 p.

[9] *Шпортько О. В.* Підвищення ефективності стиснення кольорових зображень у форматі PNG. – Дис. … канд. техн. наук. – Рівненський державний гуманітарний університет. – Рівне, 2010. – 195 с.

[10] *Шпортько О. В.* Оптимізація застосування модифікованого формату словникової компресії Deflate у процесі прогресуючого ієрархічного стиснення зображень без втрат // Науковий вісник Чернівецького національного університету імені Юрія Федьковича. Серія: Комп'ютерні системи та компоненти. – 2013. – Т. 4. – Вип. 4. – С. 40-52.

[11] *Шпортько О. В.* Використання різницевих кольорових моделей для стиснення RGB-зображень без втрат // Відбір і обробка інформації. – 2009. – № 31 (107). – С. 90-97.