Krzysztof RYBAK [1], Ernest JAMRO [1, 2], Maciej WIELGOSZ [1, 2], Kazimierz WIATR [1, 2]
[1] AGH, AKADEMIA GÓRNICZO-HUTNICZA, KATEDRA ELEKTRONIKI, Al. Mickiewicza 30, 30-059 Kraków
[2] ACK CYFRONET AGH, Nawojki 11, 30-950 Kraków

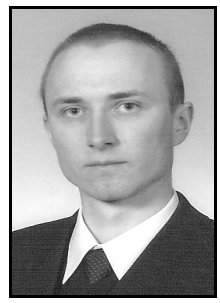
Optymalizacja kompresji Huffmana pod kątem podziału na bloki

Mgr inż. Krzysztof RYBAK

Ukończył studia pierwszego stopnia na AGH (2012), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektroni-ka i Telekomunikacja. W roku 2014 obronił pracę magisterką związaną z tematyką niniejszej publikacji na AGH Wydziale Informatyki Elektroniki i Telekomunikacji. Jego zainteresowania naukowe dotyczą kompresji danych.

e-mail: krzysztofrybak6@wp.pl

Dr inż. Maciej WIELGOSZ



Ukończył studia na AGH (2005), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obronił pracę doktorska w 2010 roku. Obecnie jest pracownikiem Katedry Elektroniki AGH i bierze czynny udział w pracach badawczych realizowanych w zespole rekonfigurowalnych systemów obliczeniowych. Jego zainteresowania naukowe dotyczą sprzętowej akceleracji obliczeń, strumieniowej analizy treści oraz architektur sprzętowych dla algorytmów sztucznej inteligencji.

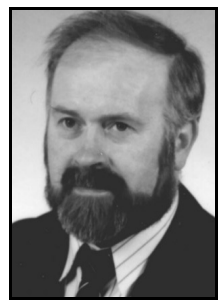
e-mail: wielgosz@agh.edu.pl

Dr inż. Ernest JAMRO

Ukończył studia na AGH na kierunku Elektronika oraz na University of Huddersfield (UK) na kierunku Elektronika i Telekomunikacja. Obronił pracę doktorską w 2001 roku na AGH na wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki. Aktualnie jest adiunktem w Katedrze Elektroniki na AGH. Jego zainteresowania naukowe to sprzętowa akceleracja obliczeń, i systemy wbudowane oparte na układach FPGA.

e-mail: jamro@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR



Studia AGH Kraków (1980), dr nauk technicznych (1987), dr habilitowany (1999) i profesor (2002). Profesor zwyczajny na Akademii Górniczo-Hutniczej oraz Dyrektor Akademickiego Centrum Komputerowego Cyfronet AGH. Prowadzone prace badawcze dotyczą komputerowego sterowania procesami, systemów wizyjnych, systemów wieloprocesorowych, układów programowalnych, rekonfigurowalnych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń.

e-mail: wiatr@agh.edu.pl

Streszczenie

Prezentowane w pracy badania dotyczą bezstratnej kompresji danych opartej o metodę Huffmana i zgodnej ze standardem deflate stosowanym w plikach .zip / .gz. Zaproponowana jest optymalizacja kodera Huffmana polegająca na podziale na bloki, w których stosuje się różne książki kodowe. Wprowadzenie dodatkowego bloku z reguły poprawia stopień kompresji kosztem narzutu spowodowanego koniecznością przesłania dodatkowej książki kodowej. Dlatego w artykule zaproponowano nowy algorytm podziału na bloki.

Słowa kluczowe: kompresja danych, kodowanie Huffmana, deflate.

Optimization of Huffman compression employing different block sizes

Abstract

According to deflate [2] standard (used e.g. in .zip / .gz files), an input file can be divided into different blocks, which are compressed employing different Huffman [1] codewords. Usually the smaller the block size, the better the compression ratio. Nevertheless each block requires additional header (codewords) overhead. Consequently, introduction of a new block is a compromise between pure data compression ratio and headers size. This paper introduces a novel algorithm for block Huffman compression, which compares sub-block data statistics (histograms) based on current sub-block entropy $E(x)$ (1) and entropy-based estimated average word bitlength $E_{mod}(x)$ for which codewords are obtained for the previous sub-block (2). When $E_{\mathrm{mod}}(x) - E(x) \geq T$ (T – a threshold), then a new block is inserted. Otherwise, the current sub-block is merged into the previous block. The typical header size is 50 B, therefore theoretical threshold T for different sub-block sizes S is as in (3) and is given in Tab. 2. Nevertheless, the results presented in Tab. 1 indicate that optimal T should be slightly different – smaller for small sub-block size S and larger for big S. The deflate standard was selected due to its optimal compression size to compression speed ratio [3]. This standard was selected for hardware implementation in FPGA [4, 5, 6, 7].

Keywords: data compression, Huffman coding, deflate.

1. Wstęp

Kompresja danych metodą Huffmana [1] polega na zastosowaniu zmiennej długości kodów w zależności od częstości występowania symboli, dzięki czemu sumaryczna długość danych ulega skróceniu. Metoda Huffmana stanowi podstawę do wielu standardów kompresji, np. deflate [2], który jest stosowany w plikach .zip, .gz. Istnieje wiele lepszych standardów kompresji np. bzip2, niemniej standard deflate wydaje się najlepszym standardem w momencie kiedy ważny jest stosunek stopnia kompresji do jej czasu [3]. Dlatego też standard deflate został wybrany do sprzętowej kompresji danych realizowanej w układach FPGA [4, 5]. Zaproponowano również moduł sprzętowy służący do kompresji Huffmana [6] oraz moduł programowy służący do adaptacji kodowania Huffmana pod implementację sprzętową [7].

W standardzie deflate istnieje możliwość podziału danych wejściowych na bloki, które są kodowane różnymi książkami kodowymi. Czym mniejsza wielkość bloku tym kodowanie Huffmana z reguły daje lepsze wyniki ponieważ wykorzystuje lokalność statystyki znaków (histogramu danych wejściowych). Niemniej dla każdego bloku konieczne jest przesłanie książki kodowej co z kolei powoduje dodatkowy narzut kompresji. Standardową metodą jest stosowanie stałego rozmiaru bloku, np. 5–12 kB [8]. W niniejszym artykule zaproponowano nową metodę podziału na bloki, dzięki czemu podział następuje tylko wtedy, kiedy wpłynie on wydatnie na stopień kompresji.

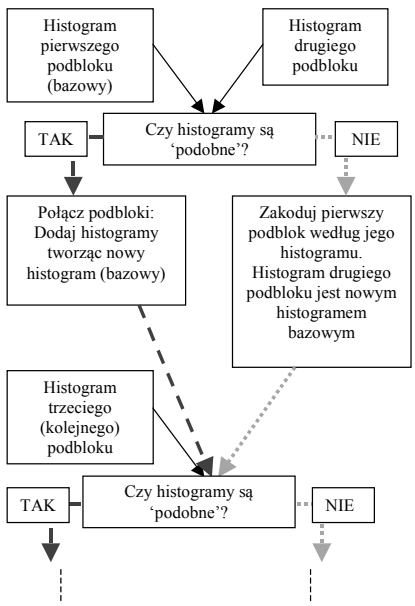
W niniejszym opracowaniu skupiono się na optymalizacji stopnia kompresji poprzez podział danych wejściowych na bloki o różnej długości z określonym kwantem, czyli wybór czy zmienić książkę kodową (stworzyć nowy blok) następuje nie co jeden znak wejściowy ale co określoną długość podbloku, np. 1kB. Histogram kolejnego podbloku jest porównywany z bieżącym histogramem i na tej podstawie podejmowana jest decyzja czy dla kolejnego podbloku ma być tworzona nowa książka kodowa, czy też histogramy są wystarczająco podobne aby zakodować je wspólną tablicą kodową. Myślą wiodącą jest osiągnięcie kompromisu pomiędzy degradacją stopnia kompresji związaną z kodowaniem kilku podbloków wspólną tablicą kodową a narzutem kompresji związanym z przesłaniem nowej książki kodowej. Przedstawiona została zarówno metoda porównania histogramów jak i ustalenia progu podobieństwa między nimi.

Koder Huffmana dokonujący kompresji z użyciem jednej tablicy oraz wielu tablic kodowych został napisany w języku C w środowisku Eclipse.

2. Koder Huffmana

2.1. Algorytm działania

Jak już wspomniano, myślą przewodnią optymalizacji kodera Huffmana jest podział całego bufora danych na podbloki. Rozmiar każdego podbloku wynosi S (ostatni podblok ma najczęściej mniejszą długość). Zapamiętywany jest histogram bazowego (pierwszego) podbloku. Następnie odbywa się porównanie histogramu drugiego podbloku z bazowym. Jeżeli histogramy są wystarczająco 'podobne', to zostają do siebie dodane tworząc nowy histogram bazowy: dane obu podbloków będą kodowane tą samą książką kodową. Jeżeli jednak histogramy znacząco się różnią, to wstawiany jest nowy blok: dotychczasowy bufor jest kodowany według jego histogramu, a nowy histogram bazowy jest tworzony z nowego podbloku: z tym histogramem będzie porównywany kolejny podblok. Opisany proces został przedstawiony na rys. 1.



Rys. 1. Algorytm podziału bufora na bloki
Fig. 1. Block division algorithm

2.2. Metoda porównania histogramów

Do tej pory posługiwano się stwierdzeniem, że dwa histogramy są 'podobne' lub nie. Celem porównania histogramów jest podjęcie decyzji: czy dane w dwóch podblokach są na tyle podobne, że można je połączyć w jeden blok, czy też na tyle różne, że należy stworzyć osobne bloki z różnymi książkami kodowymi. Należy teraz przedstawić bliżej metodę 'stwierdzenia podobieństwa'. Sprawdzono kilka metod porównania histogramów, np. metodę korelacji rang Spearmana [10], jednak najlepsza wydaje się metoda oparta na entropii.

Dla histogramu podbloku zostaje obliczona entropia danych według wzoru:

$$E(x) = \sum_{i=0}^{255} p(i) \log_2 \left(\frac{1}{p(i)}\right) \tag{1}$$

gdzie: $E(x)$ – entropia źródła x, $p(i)$ – prawdopodobieństwo wystąpienia i-tego symbolu.

Obliczona entropia reprezentuje średnią długość słów zakodowanych według książki kodowej, które jest utworzone z histogramu podbloku. W rzeczywistości średnia długość słowa jest większa, gdyż entropia jest dolną granicą stopnia kompresji.

Zostaje również oszacowana średnia długość słowa kodowego podbloku według książki kodowej utworzonej z histogramu bazowego (czyli poprzedniego podbloku), według wzoru:

$$E_{\mathrm{mod}}(x) = \sum_{i=0}^{255} p(i) \log_2 \left(\frac{1}{p_{base}(i)}\right) \tag{2}$$

gdzie: $E_{\mathrm{mod}}(x)$ – oszacowana średnia długość słowa, $p(i)$ – prawdopodobieństwo wystąpienia i-tego symbolu w histogramie podbloku, $p_{\mathrm{base}}(i)$ – prawdopodobieństwo wystąpienia i-tego symbolu w histogramie bazowym.

Różnica wartości $E_{\mathrm{mod}}(x) - E(x)$ estymuje ile bitów na słowo więcej będzie użytych przy zakodowaniu nowego podbloku według książki kodowej utworzonej z histogramu bazowego. Otrzymana wartość jest porównywany z progiem T i na podstawie wyniku porównania podejmowana jest decyzja czy histogramy są podobne czy też nie. Narzut związany z przesłaniem nowej książki kodowej w testowanych przypadkach jest rzędu 400 bitów. Teoretyczny próg wynosi zatem:

$$\mathrm{T} \approx \frac{400}{\mathrm{S}} \tag{3}$$

Czyli, np. dla $S = 1024$ B, $T \approx 0.4$ [bit/słowo].

3. Wyniki

Plikiem testowym jest bufor składający się z plików *bib, paper1, paper2, paper5, paper6, progc, progl* oraz *trans,* wszystkie z pakietu Calgary Corpus [9] używanego w standardowych testach kompresji. Pliki różnią się charakterem danych. Dzięki temu różnica stopnia kompresji między zaproponowaną a standardową metodą będzie wyższa. Wielkość takiego pliku wynosi 501 632 bajtów, entropia pliku wynosi 5,36, entropia pomnożona przez długość bufora wejściowego to 336 147 – jest dolnym ograniczeniem kodera Huffmana z użyciem jednej tablicy. Standardowa metoda Huffmana dokonuje kompresji ze stopniem 67,43%, rozmiar bufora wyjściowego wynosi 338 259 bajtów. Algorytm wykorzystujący wiele tablic kodowych ma kilka opcji kompresji i według nich zostaną przedstawione wyniki. Tab. 1 zawiera zestawienie wyników kompresji w funkcji rozmiaru podbloków.

Tab. 1. Wyniki kompresji w funkcji rozmiaru podbloku S, przedstawiono próg porównania T dla którego osiągnięto najlepszy stopień kompresji
Tab. 1. Results of compression ratio versus sub-blocks size S, for each block size the comparison threshold T is included for which the best compression was obtained

S	T	liczba bloków	narzut kompresji [B]	zakodowane dane [B]	suma [B]
256B	1,30	108	4 741	311 548	316 290
512B	0,79	98	4 600	311 387	315 988
1kB	0,41	94	4 573	310 911	315 485
2kB	0,29	63	3 218	312 501	315 720
4kB	0,10	62	3 256	313 296	316 533
8kB	0,07	36	1 989	315 270	317 260
16kB	0,07	20	1 140	317 178	318 319

Tab. 2 zawiera podobne zestawienie, ale z użyciem progu porównania T wyznaczonego według (3). Tab. 3 zawiera zestawienie wyników kompresji w funkcji progu porównania T dla przykładowego rozmiaru podbloków. Podane wyniki odnoszące się do rozmiaru (np. narzutu) zostają podane w bajtach.

Analizując wyniki w tab. 1–3, granica, jaką jest entropia całego pliku, została przekroczona. Dla każdego rozmiaru bloku uzyskano stopień kompresji lepszy niż w przypadku metody z jedną tablicą. Najlepszy stopień kompresji uzyskano dla rozmiaru podbloku o wartości 1024 bajtów i wyniósł 62,89%. Oszacowanie progu porównania histogramów T przedstawione w (3) okazało się bardzo trafne, szczególnie dla rozmiaru podbloku, dla którego uzyskano najlepszy stopień kompresji. Najlepszy stopień kompresji zostaje uzyskany gdy osiągnięty jest kompromis pomiędzy narzutem kompresji związanym z przesłaniem kolejnych tablic kodowych a degradacją kompresji związaną z kodowaniem wielu tablic wspólną tablicą Huffmana. Dla podbloków o rozmiarze

1024 bajtów taki kompromis został osiągnięty dla progu porównania 0,4. Poniżej tej wartości próg jest zbyt rygorystyczny i tworzonych jest zbyt wiele tablic, co związane jest ze zwiększeniem narzutu kompresji. Powyżej optymalnego progu tworzonych jest zbyt mało tablic, gdyż próg jest bardziej liberalny i dopasowywane są histogramy, którym w rzeczywistości nie odpowiadają podobne drzewa Huffmana.

Tab. 2. Wyniki kompresji w funkcji rozmiaru podbloku
Tab. 2. Results of compression ratio versus sub-block size

S	*T*	liczba bloków	narzut nagłówków	zakodowane dane	suma
256B	1,6	46	2109	316 071	318 181
512B	0,8	97	4564	311 459	316 024
1kB	0,4	97	4698	310 851	315 550
2kB	0,2	84	4246	311 561	315 808
4kB	0,1	62	3256	313 296	316 533
8kB	0,05	38	2092	315 206	317 298
16kB	0,025	24	1363	317 007	318 371

Tab. 3. Wyniki kompresji w funkcji progu porównania histogramów dla rozmiaru podbloku równego 1kB
Tab. 3. Results of compression ratio versus histogram comparison threshold for sub-block size equal to 1kB

T	liczba bloków	narzut nagłówków	zakodowane dane	suma
0	490	21 731	305 475	327 207
0,1	340	15 306	306 622	321 929
0,2	210	9 764	308 115	317 881
0,3	145	6 824	309 350	316 176
0,4	97	4 698	310 851	315 550
0,5	53	2 704	312 875	315 580
0,6	40	2 070	314 143	316 215
0,7	27	1 713	315 534	316 948
0,8	18	947	316 656	317 605

Wprowadzenie nowego bloku wymaga większej mocy obliczeniowej zarówno w koderze jak i w dekoderze. Wiąże się to z budową książki kodowej zgodnie z algorytmem zaproponowanym przez Huffmana [1]. Dodatkowo w standardzie deflate aby zmniejszyć wielkość nagłówka sam nagłówek jest kompresowany algorytmem RLE (Run Length Encoding) a następnie kodowaniem Huffmana [2]. Dodatkowo podczas kodowania i dekodowania z reguły używa się tablicy LUT (Look Up Table) [4, 6], dotyczy to zarówno metody sprzętowej jak i programowej. Przy małym rozmiarze bloku – porównywalnym z wielkością tablicy LUT, czas samego programowania tablicy LUT jest porównywalny z czasem zapisu tej tablicy. Podsumowując, aby zwiększyć szybkość kodowania i dekodowania lepiej jest używać większych rozmiarów bloków.

W równaniach (1) i (2) użyto skomplikowanej funkcji logarytmu, co wydaje się nieakceptowalnie komplikować zaproponowany algorytm. Niemniej dana wejściowa (wartość histogramu) jest z reguły 4–8 bitowa. Na przykład, dla wielkości bloku 4kB maksymalna wartość histogramu może być zapisana na 12 bitach (2^{12} = 4k). Wartość średnia histogramu wynosi 4096 / 256 = 16, czyli może być zapisana na 4–5 bitach. Dlatego do obliczenia wartości funkcji $\log_2()$ można użyć relatywnie prostej pamięci LUT. Co więcej nawet mnożenie w (1) razem z obliczaniem funkcji $\log_2()$ może być zaimplementowane w jednej pamięci LUT.

Tab. 4 zawiera analogiczne zestawienie jak w tab. 1. Różnica polega na tym, że bieżący histogram zostaje porównany z wszystkimi histogramami utworzonymi do tej pory. W przypadku kiedy bieżący histogram jest podobny do któregoś z poprzednich, nie jest przesyłana po raz kolejny wybrana książka kodowa, ale tylko jej numer. Takie kodowanie jest niezgodne ze standardem deflate, niemniej wydaje się być prostą metodą jego ulepszenia, np. w momencie kiedy wprowadzane są wstawki w obcym języku, równania, itp. Niestety wyniki pokazują, że stopień kompresji w metodzie zgodnej ze standardem deflate jest porównywalny do rezultatów uzyskanych w metodzie niezgodnej ze specyfikacją. Przy czym moc obliczeniowa wymagana do porównania histogramów wszystkich poprzednich bloków jest znacząca. Dlatego niniejsza modyfikacja nie powinna być wprowadzana.

Tab. 4. Wyniki kompresji w funkcji rozmiaru podbloku *S*, przedstawiono próg porównania *T*, dla którego osiągnięto najlepszy stopień kompresji. Bieżący histogram jest porównywany z wszystkimi utworzonymi na danym etapie. Nagłówek danych nie jest zgodny ze standardem deflate
Tab. 4. Results of compression ratio versus size of sub-blocks *S*, for each block size the comparison threshold *T* is included, for which the best compression was obtained. The current histogram is compared with every created to this point histogram. The header of the data block is not compatible with the deflate specification

S	*T*	liczba bloków	narzut nagłówków	zakodowane dane	suma
256B	0,85	75	5087	310 709	315 796
512B	0,46	98	5409	309 604	315 013
1kB	0,34	60	3412	311 671	315 083
2kB	0,22	57	3218	312 342	315 560
4kB	0,11	47	2568	313 937	316 505
8kB	0,05	38	2123	315 206	317 329
16kB	0,04	20	1160	317 178	318 338

4. Wnioski

Zastosowana metoda podziału bufora wejściowego na bloki zwiększa stopień kompresji. Dla każdego rozmiaru bloków otrzymano wynik lepszy niż w przypadku standardowej metody z jedną tablicą kodową: w najlepszym scenariuszu stopień kompresji wyniósł 62.89% a więc o 4.54% mniej niż w metodzie standardowej. Wadą zaproponowanego algorytmu jest potrzeba utworzenia wielu tablic kodowych w koderze zaproponowanego algorytmu, co jest procesem czasochłonnym. Zmniejszenie liczby tablic kodowych można uzyskać na kilka sposobów: np. zwiększając próg porównania histogramów lub rozmiar podbloków.

Praca finansowana z Działalności Statutowej nr 11.11.230.017 oraz z NCBiR PLGrid Plus POIG.02.03.00-00-096/10.

5. Literatura

[1] Huffman D.A.: A method for the construction of minimum-redundancy codes, Proc. Inst. Radio Eng, Vol.40, No.9, pp.1098-1101, Sep. 1952.
[2] P. Deutsch DEFLATE Compressed Data Format Specification version 1.3: RFC1951, 05/1996.
[3] Collin L.: A Quick Benchmark: Gzip vs. Bzip2 vs. LZMA, http://tukaani.org/lzma/benchmarks.html, 2005-05-31.
[4] Jamro E., Wiatr K.: Implementacja w układach FPGA dekompresji danych zgodnie ze standardem Deflate, Pomiary Automatyka Kontrola, 2013 vol. 59 nr 8, s. 739–741.
[5] Gwiazdoń M., Jamro E., Wiatr K.: Optymalizacja sprzętowej architektury kompresji danych metodą słownikową, Pomiary Automatyka Kontrola, 2013 vol. 59 nr 8, s. 827–829.
[6] Jamro E., Wielgosz M., Wiatr K.: FPGA Implementation of the Dynamic Huffman Encoder, Proc. IFAC Workshop on Programmable Devices and Embedded Systems, Brno, Feb. 14-16, 2006, pp.60-65.
[7] Rybak K., Jamro E., Wiatr K.: Realizacja kompresji danych metodą Huffmana z ograniczeniem długości słów kodowych, Pomiary, Automatyka, Kontrola, 2012 vol. 58 nr 7 s. 662–664.
[8] Abdul Mannan M., Kaykobad M.: Block Huffman coding, Computers & Mathematics with Applications, Volume 46, Issues 10–11, November-December 2003, Pages 1581–1587.
[9] ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/, styczeń 2014.
[10] http://www.statystyka.org/wspolczynnik_korelacji_rang_spearmana.php, styczeń 2014.

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*