Odporność danych skompresowanych na błędy — wybrane zagadnienia

Autoreferat rozprawy doktorskiej

Marek Biskup

Instytut Informatyki
Uniwersytet Warszawski

17 września 2008

Kontekst rozprawy

Błąd pojedynczego bitu w skompresowanych danych propaguje się, ponieważ zmienia on stan dekodera. W rozprawie została przedstawiona analiza odporności na błędy w skompresowanych danych. Głównym zamierzeniem jest odzyskanie największej możliwej ilości danych po wystąpieniu błędu. Rozprawa odnosi się do kodów Huffmana.

W danych zakodowanych kodem Huffmana pojedynczy błąd bitowy powoduje, że dekoder traci synchronizację z koderem [1, 2]. Błąd propaguje się, ponieważ słowa kodowe widziane przez dekoder nie są ułożone poprawnie względem tych zakodowanych. W przypadku większości kodów Huffmana dekoder w końcu zsynchronizuje się [3, 4]. Nazywane jest to *statystyczną synchronizacją*. Niemniej jednak nie istnieje a priori ograniczenie górne na liczbę niepoprawnie zdekodowanych symboli.

Statystyczna synchronizacja kodów Huffmana ma związek z istnieniem *ciągów synchronizujących* [4, 5] dla tych kodów. Ciąg synchronizujący, jeśli wystąpi w dekodowanych danych, powoduje zawsze resynchronizację dekodera. Kody posiadające ciąg synchronizujący są statystycznie synchronizowalne (przy założeniu, że źródło znaków jest ϵ-gwarantowane, patrz [4]), ponieważ taki ciąg musi w końcu wystąpić w kodowanych danych. Niektóre kody Huffmana mają *słowa synchronizujące*, czyli słowa kodowe, które są ciągami synchronizującymi [6, 7].

Optymalny binarny kod prefiksowy nie jest wyznaczony jednoznacznie przez prawdopodobieństwa liter. Algorytm Huffmana daje swobodę wyboru przyporządkowania 0 i 1 do dwóch fragmentów kodu, łączonych w każdym kroku algorytmu. Istnieją też optymalne kody, których nie można skonstruować algorytmem Huffmana. W rzeczywistości algorytm Huffmana określa optymalny rozkład długości słów kodowych. Wszystkie kody zgodne z takim rozkładem (zwane kodami Huffmana w tej rozprawie) są optymalne.

Ze względu na mnogość kodów Huffmana dla danego rozkładu długości słów kodowych, istotnym problemem jest znalezienie najlepszego, w jakimś sensie kodu. Miernikiem jakości kodu może być na przykład średni czas resynchronizacji dekodera oraz istnienie ciągu lub słowa synchronizującego. Średni czas resynchronizacji dekodera (*opóźnienie synchronizacji*) i jego wariancja mogą być liczone przy użyciu łańcuchów Markowa [8, 9, 10]. Wyniki pokazują, że kody Huffmana w średnim przypadku synchronizują się szybko, nawet po kilku znakach. Powstało kilka heurystycznych metod konstruujących kody o małym opóźnieniu synchronizacji [11, 12, 13, 14].

Jeśli największy wspólny dzielnik długości słów kodowych jest równy 1 to istnieje kod Huffmana mający ciąg synchronizujący [5]. Z drugiej strony, dla innych rozkładów prawdopodobieństwa, do takiej sytuacji można łatwo doprowadzić psując nieznacznie optymalność kodu [15]. Problem istnienia i konstrukcji kodów ze słowem synchronizującym jest znacznie trudniejszy. Istnieją częściowe wyniki dotyczące istnienia lub nieistnienia takich kodów oraz algorytmy konstrukcji takich kodów dla niektórych przypadków [2, 7, 16, 17]. Istnieją również algorytmy wprowadzające redundancję dla skonstruowania kodu ze słowem synchronizującym [6, 15].

Nawet jeśli dekoder kodu Huffmana zsynchronizuje się po błędzie bitu, liczba zdekodowanych znaków jest często inna niż rzeczywista liczba znaków zakodowanych [10, 18]. Kolejne znaki będą więc umieszczane na nieprawidłowych pozycjach. W wielu zastosowaniach takie znaki będą błędnie zinterpretowane. *Silna synchronizacja* dekodera polega dodatkowo na określeniu właściwej pozycji dekodowanych znaków. Jedną z metod zapewnienia silnej synchronizacji jest regularne wstawianie pewnych znaczników, które mogą być rozpoznane przez dekoder. Za takim znacznikiem można na przykład umieścić numer dekodowanego znaku, lub numer znacznika. Znacznikiem, w przypadku kodów Huffmana, może być *rozszerzone słowo synchronizujące* (ang. extended synchronizing codeword, ESC) [19, 16]. Konstrukcja takiego słowa polega na wydłużeniu jednego ze słów kodowych w ten sposób, że otrzymane słowo nie jest podsłowem żadnego ciągu innych słów kodowych. Oczywiście kody mające ESC nie mogą być optymalne.

Gwarantowana synchronizacja kodów Huffmana

W przypadku kodów Huffmana synchronizacja dekodera jest tylko statystyczna. O ile w średnim przypadku resynchronizacja następuje szybko, w pesymistycznym przypadku dekoder nie zsynchronizuje się do końca dekodowanej wiadomości. W wielu przypadkach ograniczenie na opóźnienie synchronizacji jest pożądane, na przykład jeśli potrzebne jest zdekodowanie tylko fragmentu danych. Przy takim ograniczeniu dekoder może zacząć dekodowanie odpowiednią liczbę bitów przed danym fragmentem, a czytając właściwy fragment dekoder będzie już zsynchronizowany.

Ograniczone opóźnienie synchronizacji może być łatwo zapewnione przez regularne wstawianie słowa synchronizującego. Wstawiane słowo służy w tym przypadku tylko do resynchronizacji i jest pomijane przez dekoder. Taka metoda nie jest optymalna, gdyż po usunięciu słowa synchronizującego z kodu można skrócić inne słowo kodowe. Metoda nie wykorzystuje też naturalnej tendencji kodów Huffmana do resynchronizacji. Można spodziewać się, że dla większości pozycji startowych dekodera, dekoder dostatecznie szybko zsynchronizuje się spontanicznie. Zatem tylko w niektórych miejscach słowa synchronizujace muszą być wstawione.

W rozprawie przedstawione zostały dwie metody ograniczające opóźnienie synchronizacji do pewnej liczby L bitów, która jest parametrem algorytmów. Synchronizacja jest zapewniana przez wstawianie w odpowiednie miejsca pewnych ciągów bitów, zwanych *znacznikami synchronizującymi*. Obie metody mają następujące cechy.

- Użyty kod Huffmana jest optymalny. Redundancja zależy tylko od liczby wstawionych znaczników synchronizujących. W szczególności redundancja jest zerowa, jeśli nie będzie wstawiony żaden znacznik.
- Wykorzystywana jest statystyczna synchronizacja kodów Huffmana. Jeśli dekoder, zaczynający dekodowanie od pewnego bitu, dostatecznie szybko zsynchronizuje się spontanicznie, to nie jest dla niego wstawiany żaden znacznik synchronizujący. Znaczniki wstawiane są tylko w te miejsca, gdzie w przeciwnym wypadku opóźnienie synchronizacji przekroczyłoby zadany próg.
- Redundancja może być dowolnie zmniejszona przez zwiększenie parametru L.

Pierwsza metoda [20] wymaga od dekodera znajomości pozycji aktualnie dekodowanego bitu. Metoda ta jest prosta i efektywna. Spadek wydajności kodera i dekodera używających tej metody wynosi około 20% — dodatkowy narzut jest podliniowy względem długości zakodowanego ciągu. Metoda

działa dla dowolnego kodu Huffmana, jednak dla kodów nie mających ciągu synchronizującego redundancja będzie większa (patrz drugi punkt powyżej).

Druga metoda nie wymaga od dekodera żadnej znajomości pozycji dekodowanych bitów. To kwalifikuje ją do użycia przy ograniczaniu propagacji nie tylko błędów zamiany bitu, ale również błędów wstawienia i usunięcia bitu. Metoda ta wymaga jednak aby kod miał ciąg synchronizujący.

Czas działania drugiej metody jest proporcjonalny do długości zakodowanej wiadomości, przy dodatkowym preprocessingu w czasie $O(N)$, gdzie N jest liczbą słów kodowych w kodzie. Została też przedstawiona alternatywna implementacja algorytmu, znacznie prostsza koncepcyjnie i implementacyjnie. W jej przypadku czas przetwarzania każdego słowa kodowego jest proporcjonalny do długości najdłuższego słowa kodowego. W średnim przypadku oczekuje się, że czasy działania obu implementacji będą podobne. Prostsza implementacja wymaga jednak preprocessingu w czasie $O(N^2)$, więc nadaje się tylko do małych kodów.

Obie opracowane metody są nowością. Problem ograniczenia propagacji błędów w danych zakodowanych kodem Huffmana był wcześniej rozważany [19, 21], ale opracowane dotychczas metody nie miały żadnych z trzech wymienionych wcześniej cech.

Pierwsza metoda została zastosowana do podziału danych skompresowanych kodem Huffmana na bloki, w celu późniejszej niezależnej dekompresji bloków. Testy pokazały, że redundancja wprowadzona przez tę metodę jest od jednego do kilku rzędów wielkości niższa niż w przypadku innych sposobów podziału na bloki. Pomyślne testy równoległej dekompresji danych zakodowanych przy użyciu tej metody zostały przeprowadzone na kompresji Jpeg [27].

Znacznik dla silnej synchronizacji

Silna synchronizacja kodów Huffmana może być zapewniona przez regularne wstawianie ESC [19, 16]. Ta metoda jednak używa nieoptymalnego kodu Huffmana. Redundancja zależy więc nie tylko od liczby wstawionych ESC, ale nawet przy braku wstawień rośnie liniowo z długością zakodowanego ciągu.

W rozprawie przedstawiona została nowa metoda silnej synchronizacji [22]. Kod używany w tym przypadku jest optymalny. Metoda polega na wstawianiu pewnego znacznika synchronizującego z dołączoną informacją pozycyjną. Dekoder, podobnie jak w przypadku ESC, może zawsze rozpoznać wstawiony znacznik i odczytać dodatkową informację. Znacznik synchronizujący jest konstruowany z ciągu bitów, który nie występuje w zakodowanej normalnie wiadomości.

Metoda wymaga aby znacznik synchronizujący był przesłany do dekodera. Ta stała redundancja jest proporcjonalna do logarytmu z długości zakodowanej wiadomości. Jest to mniej niż liniowa redundancja metody ESC.

Metoda jest ogólna i działa dla dowolnych kodów Huffmana, a nawet może być zastosowana do innych danych binarnych.

Długość ciągu synchronizującego

W rozprawie zostało pokazane, że ciąg synchronizujący kodu Huffmana jest ciągiem synchronizującym pewnego automatu skończonego.

Dla automatu skończonego, ciąg synchronizujący to taki, który przeprowadza każdy stan automatu do jednego, wspólnego stanu. Dla automatów skończonych istnieje wciąż nie udowodniona hipoteza Černý'ego, mówiąca że długość najkrótszego ciągu synchronizującego dla automatu o N stanach nie przekracza $(N-1)^2$ [23]. To ograniczenie zostało udowodnione tylko dla pewnych klas automatów skończonych (w [24] przedstawiony jest obszerny przegląd tematyki). Dla ogólnego automatu najlepsze znane ograniczenie to $O(N^3)$. Klasa automatów odpowiadających kodom Huffmana nie była dotychczas rozważana.

Rozprawa zawiera konstruktywny dowód ograniczenia górnego na długość najkrótszego ciągu synchronizującego dla kodów Huffmana. W najprostszej (aczkolwiek nie najmocniejszej) formie ograniczenie to wynosi

$$O\left(\sum_i |w_i| \log N\right), \tag{1}$$

gdzie $w_1 \dots w_N$ są słowami kodowymi. Dla większości kodów, choć nie dla wszystkich, jest ono lepsze niż ograniczenie z hipotezy Černý'ego. Jest to również najlepsze znane ograniczenie dla tej klasy automatów. Algorytm konstrukcji ciągu synchronizującego spełniającego to ograniczenie ma złożoność (z grubsza)

$$O\left(\sum_i |w_i| \log^2 N\right). \tag{2}$$

Jest to lepsza złożoność niż złożoność $O(N^3)$ algorytmu Eppsteina [25], najlepszego znanego algorytmu konstrukcji ciągu synchronizującego dowolnego automatu.

Zaproponowany został też nowy algorytm sprawdzający, czy dany kod ma ciąg synchronizujący. Ma on złożoność jedynie $O(\sum |w_i|)$. Jest to znów lepszy czas działania niż $O(N^2)$ algorytmu Eppsteina [25].

Rozprawa przedstawia również wyniki automatycznego wyszukiwania kodów o pesymistycznej długości najkrótszego ciągu synchronizującego. Zostały przeanalizowane wszystkie kody o liczbie słów kodowych N nie przekraczającej 20 i wszystkie kody o długości najdłuższego słowa kodowego h, nie większej niż 5. Znaleziono dwie pesymistyczne klasy kodów, jedną dla ustalonego N, drugą dla ustalonego h. Dla tych klas kodów została policzona dokładna długość najkrótszego ciągu synchronizującego.

Interesujące jest, że długość najkrótszego słowa kodowego, odpowiednio $O(N)$ i $O(h^2)$, jest znacznie mniejsza od udowodnionej wcześniej granicy, czyli około $O(Nh \log N)$. Sformułowana została hipoteza, że te kody mają najgorszą długość ciągu synchronizującego wśród wszystkich kodów o ustalonym, odpowiednio, N i h. Pozostaje to otwartym problemem.

Algorytmy

W rozprawie zostały przedstawione dwa algorytmy znajdujące wszystkie słowa synchronizujące danego kodu Huffmana. Pierwszy algorytm jest bardzo prosty i działa w czasie $O(\sum |w_i|)$ (to jest jednak szybciej od naiwnego algorytmu $O(\sum |w_i|^2)$ lub $O(N \sum |w_i|)$). Drugi algorytm jest bardziej złożony. Wymaga on preprocessingu $O(N)$, a potem może sprawdzić w czasie $O(|w|)$ czy pojedyncze słowo kodowe w jest synchronizujące. W szczególności sprawdzenie, czy pojedyncze słowo kodowe jest synchronizujące można zrobić w czasie $O(N)$. Problem znajdowania wszystkich słów synchronizujących nie był dotychczas rozważany.

Rozprawa zawiera też algorytm, za pomocą którego dekoder może stwierdzić, że już jest zsynchronizowany. Algorytm nie określa dokładnego momentu synchronizacji, ale jego ograniczenie górne. Algorytm może być użyty na przykład do odzyskiwania danych, których początkowy fragment jest stracony. Przy jego pomocy można stwierdzić od kiedy dekodowane dane są z pewnością poprawne.

Uwagi końcowe

Nowe wyniki naukowe, przedstawione w rozprawie, zostały również zawarte w następujących artykułach:

- M. T. Biskup, "Guaranteed synchronization of Huffman codes," in *Proc. 18th IEEE Data Compression Conference (DCC'08)*, pp. 462–471, IEEE Computer Society, (Los Alamos, CA, USA), 2008, [20].

- M. T. Biskup, "A word that does not appear in the encoded message as a resynchronization marker," in *Proceedings of the IEEE Information Theory Workshop*, Porto, Portugal, 2008, [22].

- M. T. Biskup, "Shortest Synchronizing Strings for Huffman Codes" in *Mathematical Foundations of Computer Science 2008,* E. Ochmański and J. Tyszkiewicz, eds., *Lecture Notes in Computer Science* **5162**, pp. 120–131, Springer, 2008, [26].

- M. T. Biskup, "Synchronization of Huffman Codes," 2008, nieopublikowane, [27].

Poniżej zebrane są główne wyniki rozprawy wraz z odnośnikami do prac, w których wyniki zostały przedstawione:

- metoda gwarantowanej synchronizacji kodów Huffmana dla dekodera, który zna swoją pozycję w dekodowanym ciągu [20],

- metoda gwarantowanej synchronizacji kodów Huffmana dla dekodera, który nie zna swojej pozycji w dekodowanym ciągu [27],

- metoda wyboru znacznika silnej synchronizacji, który może być używany wraz z optymalnym kodem Huffmana [22],

- ograniczenie górne na długość najkrótszego ciągu synchronizującego kodu Huffmana [26],

- algorytm konstrukcji ciągu synchronizującego dla kodu Huffmana [26],

- algorytm sprawdzający, czy kod Huffmana ma ciąg synchronizujący [26],

- dwa algorytmy znajdowania wszystkich słów synchronizujących kodu Huffmana [27],

- metoda określania swojego stanu synchronizacji przez dekoder [27],

- dwie klasy kodów Huffmana z długimi najkrótszymi ciągami synchronizującymi oraz dokładna analiza długości ich najkrótszych ciągów synchronizujących [26],

- podział danych skompresowanych kodem Huffmana na bloki w celu późniejszej niezależnej dekompresji bloków oraz testy metody na kompresji Jpeg [27].

Literatura

[1] B. Rudner, "Construction of minimum-redundancy codes with an optimum synchronizing property," *IEEE Trans. Inform. Theory* **17**, pp. 478–487, July 1971.

[2] T. J. Ferguson and J. H. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Trans. Inform. Theory* **30**, pp. 687–693, July 1984.

[3] C. F. Freiling, D. S. Jungreis, F. Théberge, and K. Zeger, "Almost all complete binary prefix codes have a self-synchronizing string," *IEEE Trans. Inform. Theory* **49**, pp. 2219–2225, September 2003.

[4] R. M. Capocelli, L. Gargano, and U. Vaccaro, "On the characterization of statistically synchronizable variable-length codes," *IEEE Trans. Inform. Theory* **34**, pp. 817–825, July 1988.

[5] M. P. Schützenberger, "On synchronizing prefix codes," *Information and Control* **11**(4), pp. 396–401, 1967.

[6] B. L. Montgomery and J. Abrahams, "Synchronization of binary source codes," *IEEE Trans. Inform. Theory* **32**, pp. 849–854, November 1986.

[7] A. Escott and S. Perkins, "Binary Huffman equivalent codes with a short synchronizing codeword," *IEEE Trans. Inform. Theory* **44**, pp. 346–351, January 1998.

[8] J. C. Maxted and J. P. Robinson, "Error recovery for variable length codes," *IEEE Trans. Inform. Theory* **31**, pp. 794–801, November 1985.

[9] M. E. Monaco and J. M. Lawler, "Corrections and additions to 'error recovery for variable length codes' by J.C. Maxted and J.P. Robinson," *IEEE Trans. Inform. Theory* **33**, pp. 454–456, May 1987.

[10] P. F. Swaszek and P. DiCicco, "More on the error recovery for variable-length codes," *IEEE Trans. Inform. Theory* **41**, pp. 2064–2071, November 1995.

[11] G. Zhou and Z. Zhang, "Synchronization recovery of variable-length codes," *IEEE Trans. Inform. Theory* **48**, pp. 219–227, January 2002.

[12] M. R. Titchener, "The synchronization of variable-length codes," *IEEE Trans. Inform. Theory* **43**(2), pp. 683–691, 1997.

[13] T.-C. Yang and S. Kumar, "A low complexity error recovery technique for wavelet image codecs with inter-subband dependency," *IEEE Transactions on Consumer Electronics* **48**, pp. 973 – 981, November 2002.

[14] Y. Takishima, M. Wada, and H. Murakami, "Error states and synchronization recovery for variable length codes," *IEEE Trans. Communications* **42**, pp. 783–792, Feb/Mar/Apr 1995.

[15] R. M. Capocelli, A. D. Santis, L. Gargano, and U. Vaccaro, "On the construction of statistically synchronizable codes," *IEEE Trans. Inform. Theory* **38**, pp. 407–414, March 1992.

[16] S. Perkins and A. E. Escott, "Synchronizing codewords of q-ary Huffman codes," *Discrete Math.* **197-198**, pp. 637–655, 1999.

[17] Y.-M. Huang and S.-C. Wu, "Shortest synchronizing codewords of a binary Huffman equivalent code," in *ITCC '03: Proceedings of the International Conference on Information Technology: Computers and Communications*, p. 226, IEEE Computer Society, (Washington, DC, USA), 2003.

[18] S. Malinowski, H. Jegou, and C. Guillemot, "Synchronization recovery and state model reduction for soft decoding of variable length codes," *IEEE Trans. Inform. Theory* **53**, pp. 368–377, Jan. 2007.

[19] W.-M. Lam and A. Reibman, "Self-synchronizing variable-length codes for image transmission," *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-92* **3**, pp. 477–480, 1992.

[20] M. T. Biskup, "Guaranteed synchronization of Huffman codes," in *Proc. 18th IEEE Data Compression Conference (DCC'08)*, pp. 462–471, IEEE Computer Society, (Los Alamitos, CA, USA), 2008.

[21] S. Hemami, "Robust image transmission using resynchronizing variable-length codes and error concealment," *IEEE Journal of Selected Areas in Communications* , June 2000.

[22] M. T. Biskup, "A word that does not appear in encoded message as a resynchronization marker," in *Proceedings of the IEEE Information Theory Workshop*, Porto, Portugal, 2008.

[23] J. Černý, "Poznámka k. homogénnym experimentom s konecnými automatmi," *Mat. fyz. čas SAV* **14**, pp. 208–215, 1964.

[24] A. Roman, *Problemy synchronizacji automatów skończonych*. PhD dissertation, Jagiellonian University, 2006. In Polish.

[25] D. Eppstein, "Reset sequences for monotonic automata," *SIAM J. Comput.* **19**(3), pp. 500–510, 1990.

[26] M. T. Biskup, "Shortest synchronizing strings for huffman codes," in *MFCS*, E. Ochmanski and J. Tyszkiewicz, eds., *Lecture Notes in Computer Science* **5162**, pp. 120–131, Springer, 2008.

[27] M. T. Biskup, "Synchronization of Huffman codes." unpublished, April 2008.

[28] M. T. Biskup, "Synchronization of Huffman codes." Submitted to European Symposium on Algorithms, April 2008.