All-Match LZ77 Bit Recycling*

Danny Dubé† Vincent Beaudoin‡
Université Laval, Canada

Recently, a technique called *bit recycling* (BR) was introduced to help reduce the redundancy caused by the multiplicity of encodings. It has been used to improve LZ77 compression, which is especially prone to allow for the existence of numerous different compressed files for some given original file F. The multiplicity of encodings causes redundancy. Instead of trying to eliminate or reduce the multiplicity itself, BR exploits it and extracts a *compensation* from it. It uses *implicit communication* that happens when, at some point in the compression process, we have that: 1. the compressor $\mathcal{C}$ has more than one option; and 2. the decompressor $\mathcal{D}$ is able to recognize that situation. The mere fact that $\mathcal{C}$ has the liberty to select one among many options allows it to implicitly send bits to $\mathcal{D}$. The particularity of BR is that it avoids storing as many bits as possible in the compressed file by implicitly sending them instead.

Previous work presented a technique that recycles bits based on the existence of multiple longest matches, called *longest-match BR* (LMBR) [1]. This work presents a more general, and more powerful, technique, called *all-match BR* (AMBR) that exploits shorter matches also. LMBR is simple because two longest matches $\langle l, d_1\rangle$ and $\langle l, d_2\rangle$ (same length but different distances) both describe the same l characters. No matter which match $\mathcal{C}$ selects, $\mathcal{D}$ decodes the same number of characters. However, AMBR is more complex because matches $\langle l_1, d_1\rangle$ and $\langle l_2, d_2\rangle$ of different lengths (say, $l_1 < l_2$) do not describe the same characters. By selecting the first match, $\mathcal{C}$ describes strictly fewer characters. Moreover, when $\mathcal{D}$ receives the first match, it does not have enough information to realize that $\mathcal{C}$ had another option. On the other hand, if $\mathcal{D}$ were to receive the second match, it would realize immediately that $\mathcal{C}$ had both options. Since BR can only be performed by $\mathcal{D}$ based on the information that it possesses, $\mathcal{D}$ only considers the options that $\mathcal{C}$ had in order to describe the prefix of F that $\mathcal{D}$ has seen until now. Consequently, AMBR has to deal with the possible parses for the prefixes of F. The latter are much too numerous and have to be manipulated in groups that are characterized by the length of the prefix that is described and the last match that is used. It was shown in previous work that the most efficient encoding is obtained using optimal recycling. The latter requires options to be given costs. Here, the costs of the options are the expected costs of the encodings of the parses for each prefix of F. To summarize, bits get recycled from the selection, by $\mathcal{C}$, of a particular parse of F.

Our experiments demonstrated that the use of AMBR considerably improves compression efficiency. On average, the files compressed using AMBR are 9.2% smaller than those compressed without using any BR while those compressed using LMBR are only 2.9% smaller.

[1] D. Dubé and V. Beaudoin. Bit recycling with prefix codes. In *Proc. of DCC*, page 379, Snowbird, Utah, USA, mar 2007.

*Funded by NSERC. †Danny.Dube@ift.ulaval.ca ‡Vincent.Beaudoin.1@ulaval.ca