Adaptation of Bit Recycling to Arithmetic Coding

WOSSPA'13

Ahmad Al-Rababa'a and Danny Dubé
Université Laval, Canada

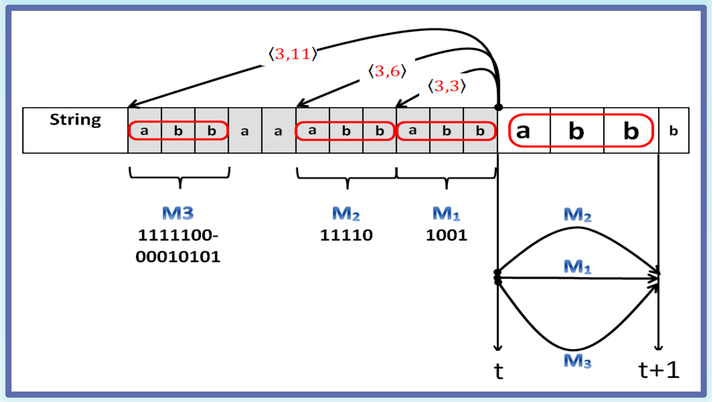
UNIVERSITÉ LAVAL

Objectives

The bit recycling compression technique has been introduced to minimize the redundancy caused by the multiplicity of encodings feature present in many compression techniques. This work aims to adapt Huffman-based Bit Recycling (HuBR) to Arithmetic Coding Bit Recycling (ACBR) in order to achieve better compression and a much wider applicability.

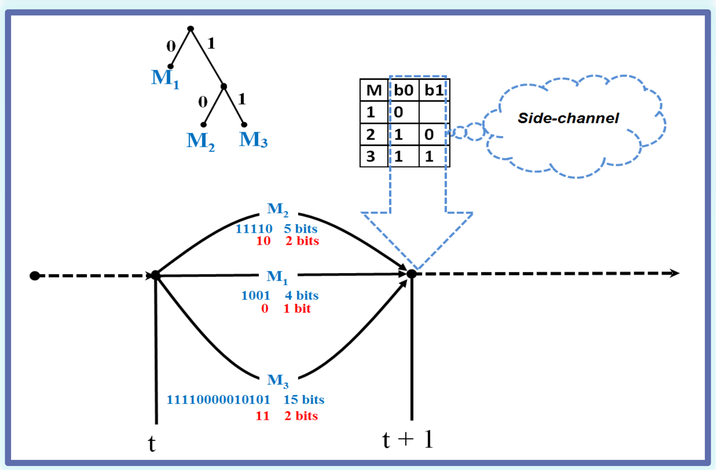
Multiplicity of Encodings

The simplest case: LZ77 Example.
Encoding the string "*abbaaabbabbabbb*".
A message is either a literal message, denoted by $[c]$, or a match message, denoted by $\langle l, d \rangle$.



Huffman-Based Bit Recycling (HuBR)

The Principle of HuBR.
How does the HuBR encoder exploit the multiplicity of encodings?



HuBR: the decompressor's bit stream

Instant I	**11110**0011011010101.....
Instant II	0011011010101.....
Instant III	**10**0011011010101....

The Weakness of HuBR

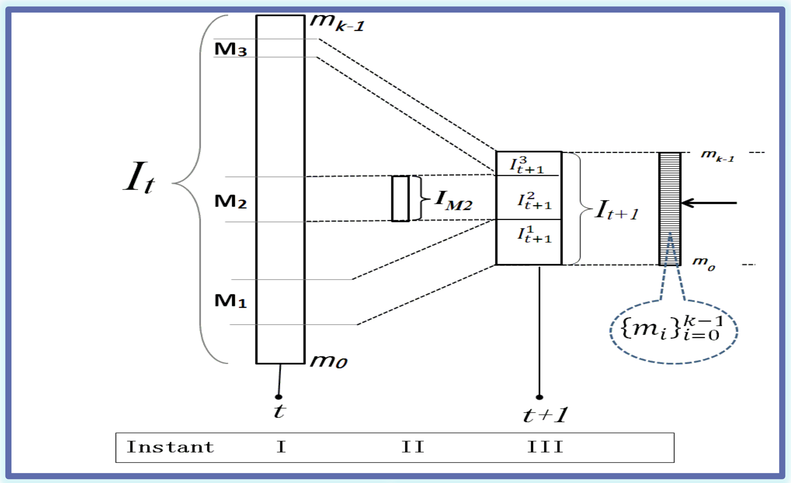
1. Huffman coding is constrained to generate codewords of integral lengths.
2. HuBR imposes an additional burden (dropping the costly messages) to avoid some situations that affect its performance negatively .

$$NC = \sum_{i=1}^{n} (c_i - |r_i|) \cdot \frac{1}{2^{|r_i|}}$$

Let p_1, p_2, and p_3 be 0.0625, 0.03125, and 0.0000305 respectively.
- *NC* for M_1,M_2, and M_3 is 5.5 bits.
- *NC* for M_1 and M_2 is 3.5 bits.
- *NC* for M_1 is 4 bits.

But the minimum (bit-fractional) $NC = -\log_2 \sum_{i=1}^{n} p_i =$ 3.414 bits.

Solution: Arithmetic Coding Bit Recycling



Arithmetic Encoding:

$$C_{n+1} = C_n + (A_n \times Q_n)$$

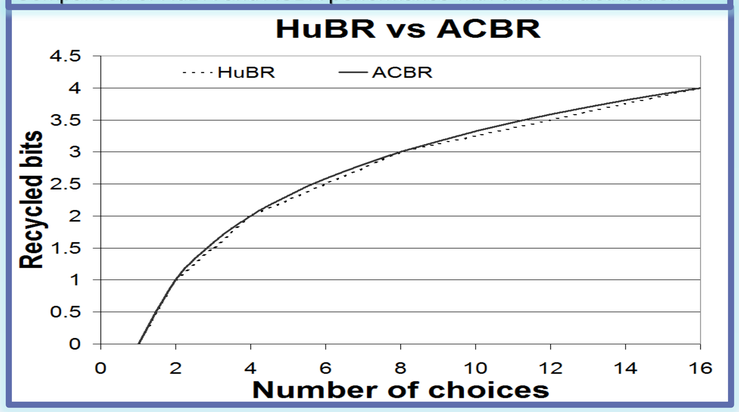
$$A_{n+1} = A_n \times \boldsymbol{p_n} \Longrightarrow P(M_1) + \boldsymbol{P(M_2)} + P(M_3)$$

Arithmetic Decoding:

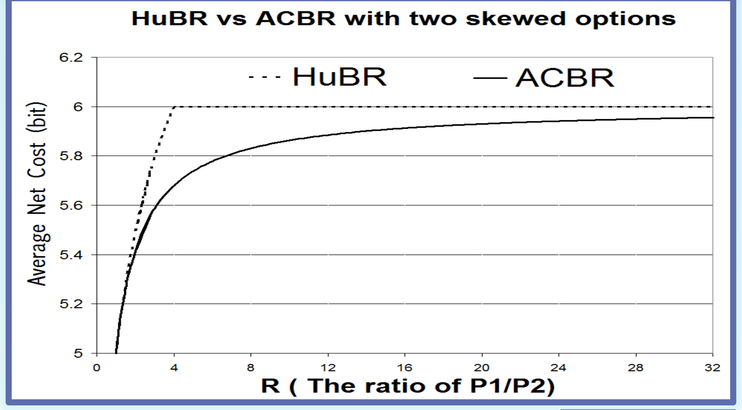
$$C_{n+1} = C_n - (A_n \times Q_n)$$

Theoretical Analysis

Comparison of HuBR and ACBR performance with uniform distribution.



Comparison of HuBR and ACBR performance with two skewed choices.



$$NC\_H = \min\left\{\left(-\frac{1}{2}\log_2(p_1 \cdot p_2)\right) - 1, -\log_2 p_1, -\log_2 p_2\right\}$$

$$NC\_A = -\log_2(p_1 + p_2)$$

Assume:
- $p_1 \geq p_2$
- $p_1 = (1/64)$

Conclusion and Future Work

The theoretical analysis showed that ACBR achieves perfect recycling in all cases whereas HuBR achieves perfect recycling only in particular cases. Consequently, a significantly better compression can be achieved by ACBR. We intend to adjust the ACBR scheme so that it can be implemented using fixed-length registers, since it currently uses arbitrary-precision calculations. Afterwards, ACBR can be implemented and applied on many applications include the multiplicity of encoding property, such as LZ77 and its variants, some variants of Prediction by Partial Matching (PPM) technique, Volf and Willems switching-compression technique, and Knuth's algorithm for the generation of balanced codes.