14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

A TIME IMPROVEMENT OVER THE MYCIELSKI ALGORITHM FOR PREDICTIVE SIGNAL CODING: MYCIELSKI-78

Ömer Nezih Gerek, Mehmet Fidan

Anadolu University, Dept. of Electrical Engineering,
Eskişehir, 26470 Turkey
phone: + (90) 222 3213550, fax: + (90) 222 3239501, email: ongerek@anadolu.edu.tr

ABSTRACT

The Mycielski algorithm is commonly known for applications requiring high quality predictions due to its infinite-past rule based prediction method. Since it repeatedly searches from the beginning of the data source, the complexity becomes non-polynomial, resulting in a limited practical use in multimedia applications, including coding. In this work, we present a time improvement over the Mycielski method by incorporating a codebook for the search which is dynamically constructed during the coding process. The construction method is symmetrical in the encoder and decoder parts, therefore reconstruction is assured. The idea and strategy resembles the time improvement of the celebrated LZ-78 method over the LZ-77 compression method, where the non-polynomial search is shortened by incorporating a codebook. Analogously, we call the faster method proposed here, the Mycielski-78 method.

1. GENERAL INFORMATION

Predictive signal coding is truly one of the most popular and well established techniques for signal coding [1]. The basic idea is usually to consider a finite past of the signal, define a prediction rule due to these past values, and for each sample, transmit the information related to the difference between the actual value and the predicted one. Due to practical limitations, the finite past is usually taken to be a short template. Similarly, the prediction rule is usually a linear filter or a simple and logical rule with a fast implementation. For data coding, the incorporation of information regarding the infinite past of the signal was also used. The general class of Lempel-Ziv coders can be considered to belong in the class of nonlinear rule-based predictors using infinite past information.

The Mycielski algorithm is a prediction method which utilizes the *total exact* history of the data samples. The idea is simply to look for the longest template ending at the end of the data sequence which had once appeared in the history of the sequence. Once this longest sequence is found, the prediction is made according to the predicted behaviour of the data, i.e., the value of the sample right after the repeating template. The rule estimates that if this pattern had appeared like this in the past, then it is supposed to behave the same now, too.

The Lempel-Ziv-77 (LZ-77) algorithm has a similar idea for coding a data sequence[2]. Here, the longest repeated replica of the tail template is also searched. This time, however, the next value of the sequence is not predicted, instead, a codeword describing the location and length of the past replica are encoded in the compressed bit stream. In that sense, LZ-77 does not completely fit into the class of classical predictive coders.

The LZ-78 algorithm is a time improvement over the LZ-77 [3]. It does not make a full search of patterns over the whole data sequence history. Instead, it dynamically constructs a codebook consisting of new patterns, and the final pattern is searched in this codebook. Once found, the augmented new pattern is added to the codebook, and the algorithm proceeds. Construction of such a codebook reduces the complexity of the algorithm. Indeed, although the LZ-78 has a lower compression performance than the LZ-77, LZ-78 became more popular in computers due to its speed.

This time improvement strategy inspires the fact that a similar method can be devised for speeding up the relatively slow Mycielski prediction algorithm. In this work, we have developed a method to construct a codebook while the prediction algorithm proceeds along the data sequence. The codebook constitutes a shorter search domain, and the algorithm becomes dramatically faster for long sequences. The codebook construction has a symmetrical counterpart in the decoder side, therefore the overall scheme is reversible. Due to the similarities of the improvement of LZ-78 over LZ-78, we call here the new proposed method, the Mycielski-78 method.

We have conducted several experiments using the new method and compared them with the classical Mycielski predictors. For comparisons, one dimensional data sequences with strong correlation among samples were preferred, therefore audio wave samples were used. For the complete test of the algorithm, the authors are in the stage of performing experiments over the NIST corpus which includes several data types. Especially, the "non-text" data type results are expected to be successful. Since the experiments are still carried out, only preliminary behaviour is presented here, and the tabulated results will be presented in the camera-ready version of this manuscript.

2. MYCIELSKI ALGORITHM

The Mycielski algorithm predicts a current data sample using all the past values of the data sequence. The general predictor can be expressed as:

$$\hat{x}[n+1] = f_{n+1}\left(x[1],\cdots,x[n]\right)$$

The function $f(\cdot)$ performs an iterative algorithm that starts from the shortest data segment at the end (i.e., $x[n]$), then one by one increases the data segment length as $\left(x[n-1],x[n]\right)$, $\left(x[n-2],x[n-1],x[n]\right)$, etc. Meanwhile, the segments are searched throughout the past data by sliding them over the samples. At one point, a probably long segment will not be encountered anywhere in the past sequence. At that point, the prediction is made as the previously encountered (1 shorter) data segment's next sample value. As a consequence, the algorithm searches through the whole data sequence repeatedly for each prediction step. The overall scheme can be analytically expressed as follows:

$$m = \arg\max_{k} \left\{ \begin{matrix} x[k]=x[n], x[k-1]=x[n-1] \\ ,\ldots, x[k-L+1]=x[n-L+1] \end{matrix} \right\}, \qquad (1)$$
$$f_{n+1} = \hat{x}[n+1] = x[m]$$

The above predictor can be re-described in words as an attempt to estimate the next sample in the currently ongoing random process as the most probable sample which had occurred in the history of the data sequence. The most probable is taken as the longest repeating chain of data samples.

Clearly, the estimate is a strong one, but its determination complexity is beyond the limits for any practical consideration, even using fast computers.

A similar complexity problem exists in the celebrated LZ-77 algorithm. The time improvement over the LZ-77 algorithm was achieved by the LZ-78 algorithm. Due to the structural resemblances, the LZ-78 improvement style over the LZ-77 has been investigated for Mycielski prediction algorithm.

3. LZ-77 AND LZ-78 ALGORITHMS

LZ-77 and LZ-78 algorithms are commonly used for lossless compression purposes. Unlike Shannon-type methods, they do not consider encoding each symbol or fixed groups separately. Instead, they consider the combinations of samples in a sequence.

The LZ-77 algorithm strongly resembles the Mycielski algorithm in its search for the longest repeating sub-sequence in the past data samples. Mycielski uses this sequence for making the prediction. LZ-77, on the other hand, spots the repeating segment and encodes its occurring location and length. In that way, a great deal of redundancy is exploited if the sequence has repeating patterns.

The original LZ-77 algorithm is slow due to its exhaustive search strategy. Although there are some finite-past and finite code-length modifications in the literature, the real time improvement was achieved by developing the LZ-78 method. Instead of exhaustively searching the whole data samples, LZ-78 generates a codebook consisting of newly encountered patterns as codewords. The search is, then, performed only along the codewords, without performing a sliding operation along the data sequence. Although the codebook size increases as samples proceed, the search strategy is significantly faster. Since the codebook generation is symmetrical in the decoder, data recovery is assured.

A LZ-78 code generation process is illustrated as follows:

Sequence: aaabbabaabbabaaaaababababab

Codebook		**Code**
1.	a	(0,a)
2.	aa	(1,a)
3.	b	(0,b)
4.	ba	(3,a)
5.	baa	(4,a)
6.	bb	(3,b)
7.	ab	(1,b)
8.	aaa	(2,a)
9.	aab	(2,b)
10.	aba	(7,a)
11.	bab	(4,b)

Output: 0a1a0b3a4a3b1b2a2b7a4b

Since the search is only performed over the dynamically generated codebook, the algorithm speed becomes polynomial time at the expense of the lower prediction performance as compared to that of LZ-77, which utilizes all possible past sequences.

4. THE MYCIELSKI-78 ALGORITHM

The infinite past prediction step of the Mycielski algorithm makes it almost impossible to use in practical compression schemes. The time improved version of the Mycielski algorithm, inspired by the improvement of LZ-78 over LZ-77, produces a polynomial-time method. Similar to the analogy of Mycielski and LZ-77 algorithms, the newly developed algorithm (which will be called the Mycielski-78 algorithm) has a codebook construction and search method similar to that of LZ-78. In other words, the prediction domain search is conducted over a dynamically growing codebook instead of the whole past data sequence. Since

- the codebook search is not an overlapping search, and
- the search can be carried out only for codewords with equal length,

the time required for the execution of Mycielski-78 is dramatically shorter than that for the original Mycielski algorithm.

The compromise of the newly obtained algorithm also resembles the compromise between LZ-77 and LZ-78. The later one is much faster, but since the prediction domain is sub-optimal, the overall performance is slightly worse.

Codebook generation in LZ-78 was illustrated with an example in the previous section. We also present the Mycielski-78 method on a basic sequence example. It can be noted

that the generated code is totally sufficient to reconstruct the original data sequence. Due to this symmetry, exact recovery is possible.

Mycielski-78 example:

Sequence: 10110110101101101010110 1

Seq:	Repeat	Prediction	Diff.	new codebook entry
1.	1 none	0	1-0=1	-
2.	0 none	0	0-0=0	-
3.	1 none	0	1-0=1	-
4.	1 1.(1)	2.[1](0)	1-0=1	(1)1
5.	0 1.(1)	2.[1](0)	0-0=0	-
6.	1 2.(0)	3.[1](1)	1-1=0	(2)0
7.	1 1.(1)	2.[1](0)	1-0=1	-
8.	0 3.(11)	4.[1](0)	0-0=0	(3)11
9.	1 2.(0)	3.[1](1)	1-1=0	-
10.	0 4.(01)	5.[1](1)	0-1=-1	(4)01
11.	1 5.(10)	6.[1](1)	1-1=0	(5)10
12.	1 4.(01)	5.[1](1)	1-1=0	-
13.	0 3.(11)	4.[1](0)	0-0=0	-
14.	1 5.(10)	6.[1](1)	1-1=0	-
15.	1 6.(101)	7.[1](1)	1-1=0	(6)101
16.	0 3.(11)	4.[1](0)	0-0=0	-
17.	1 5.(10)	6.[1](1)	1-1=0	-
18.	0 6.(101)	7.[1](1)	0-1=-1	-
19.	1 5.(10)	6.[1](1)	1-1=0	-
20.	0 6.(101)	7.[1](1)	0-1=-1	-
21.	1 5.(10)	6.[1](1)	1-1=0	-
22.	1 6.(101)	7.[1](1)	1-1=0	-
23.	0 7.(1011)	8.[1](0)	0-0=0	(7)1011
24.	1 5.(10)	6.[1](1)	1-1=0	-

Sequence: 101101101 01101101 01 01101
Prediction: 000001001 11101101 11 11101
Pred. error: 101100100-10000000-10-10000

In the above example, the initial predictions are taken to be zero. As a consequence, the prediction error sequence has the first two symbols to be equal to the original first two symbols of the data sequence. In the forthcoming prediction steps, the key is searched up to the one before last element in the codebook. If a match occurs, the prediction corresponds to the first bit of the symbol right after the matching word in the codebook. It must be noted that exactly the same codebook can be dynamically constructed by observing the prediction error sequence. The codebook size increases together with the length of the sequence. However, the search is much simpler: one can search through codebook symbols which have exactly the same length as the search sub-string. Furthermore, there is no overlapping in the search due to sliding. Finally, the codebook is never as long as the original data sequence, itself.

A technical detail must be clarified in order to utilize this method for practical compression purposes. Although every data sequence can be converted to a binary bit sequence as given in the above examples, the sample-wise correlations are not fully exploited in that way. One expects to be able to use the data symbols in its original domain. In other words, use of integer values with higher dynamic range is a desired property. This approach was adopted in this work, as well. However, the step of finding a "match" in the codebook or history becomes inferior if the exact match is sought after. In other words, the Hamming distance is not a good metric to use in the search algorithm. In normal multimedia signals such as samples of an audio file, a pattern hardly occurs again exactly within the data sequence. On the other hand, structurally quite similar patterns may repeat themselves. In such cases, a more flexible similarity measure better fits to the application. The following metric was adopted in our experiments where audio samples were tested:

$$d(x_1,x_2)\begin{cases}0, & |x_1-x_2|<\dfrac{\max\left(|x_1|,|x_2|\right)}{10}\\ 1, & |x_1-x_2|\geq\dfrac{\max\left(|x_1|,|x_2|\right)}{10}\end{cases}$$

In order to make a fair comparison, this new metric was used in both the original Mycielski, and the Mycielski-78 algorithms.

5. RESULTS

The relation between the run-time speeds and compression performances of LZ-77 and LZ-78 was also observed between the Mycielski and Mycielski-78 algorithms that we have applied over sampled audio waveforms. The sampling was done at 22 kHz with 8 bits/sample quantization. The results are presented over the number of samples axes. In this way, the increase rate of the energy, entropy, and execution time is visible. In order to give a fair comparison, the LZ-77 and LZ-78 algorithms were implemented using the same compiler and software style as used in Mycielski and Mycielski-78 algorithms. In Figures 1 and 2, we present the performances of LZ-77 and LZ-78 methods. It can be observed that LZ-78 has a worse asymptotical compression performance in terms of energy and entropy. In Figure 3, however, it can be seen that the execution time of LZ-78 is significantly less than that of LZ-77.

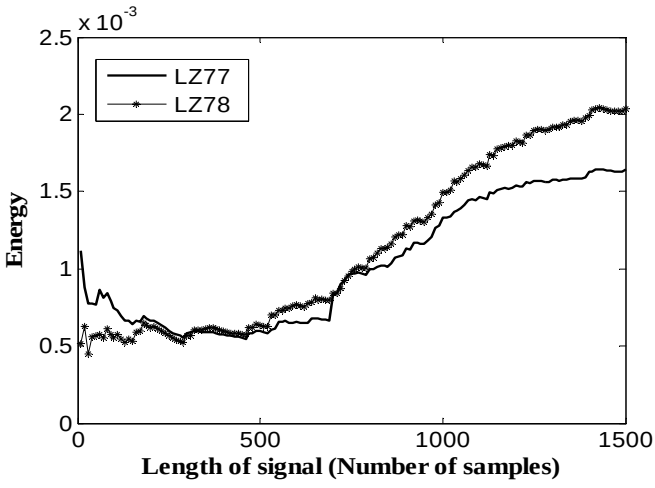

Figure 1: Energies of encoded LZ-77 and LZ-78 streams as number of processed samples increases.

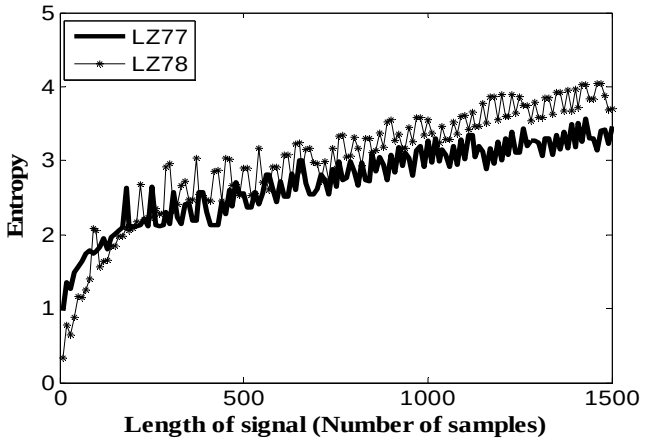

Figure 2: Sample entropies of encoded LZ-77 and LZ-78 streams as number of processed samples increases.

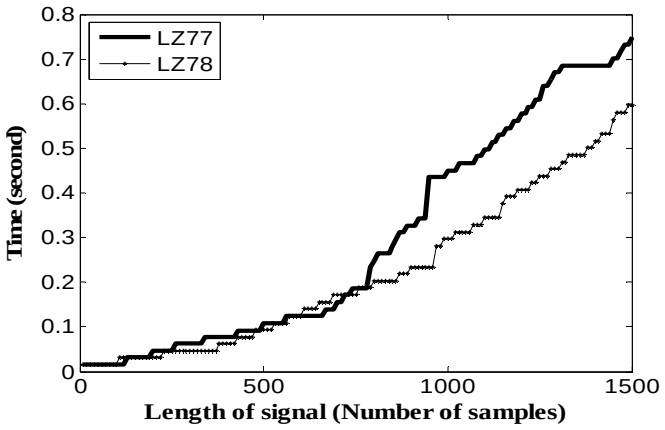

Figure 3: Execution times of LZ-77 and LZ-78.

The next set of figures (Figures 4, 5, and 6) represent the respective results for Mycielski based prediction algorithms. Three variants of the Mycielski algorithm were presented in these figures. Mycielski corresponds to the original algorithm. Mycielski78-1 corresponds to the time improved version using the standard Hamming distance for matching. Mycielski78-2, uses the time improved version with the relaxed distance metric described in Section 4.

The observations made here can be listed as follows: As compared to the energies and entropies of the original sequence, the Mycielski prediction is capable of exploiting significant amount of redundancy. As a within-comparison, the original Mycielski algorithm performs best in terms of entropy and energy reduction. In the long run, however, the performances of all the variants converge to a similar result. The Mycielski-78-2 algorithm which uses the time improvement with the relaxed Hamming metric performs slightly better than the Mycielski-78-1 algorithm, itself. On the other hand, considering Figure 6, the situation for the execution times is quite different. The execution times of Mycielski-78-1 and Mycielski-78-2 algorithms are quite similar; however both are orders of magnitudes faster than the original Mycielski algorithm. As a result, the time improvement of the algorithm is well justified as compared to the slight deterioration in the compression performance.

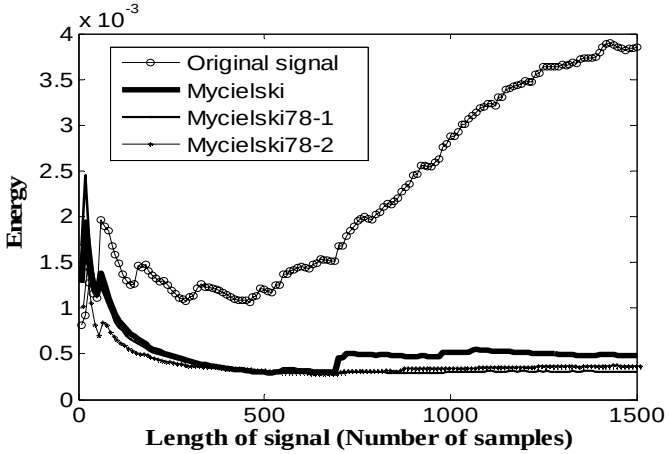

Figure 4: Energies of encoded Mycielski and Mycielski-78 streams as number of processed samples increases.

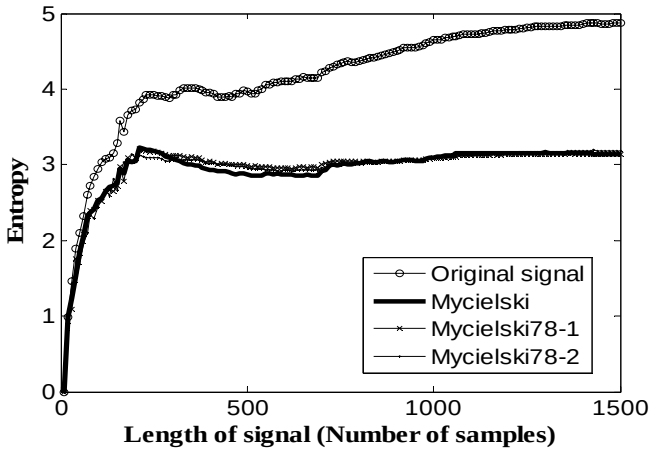

Figure 5: Sample entropies of encoded Mycielski and Mycielski-78 streams as number of processed samples increases.

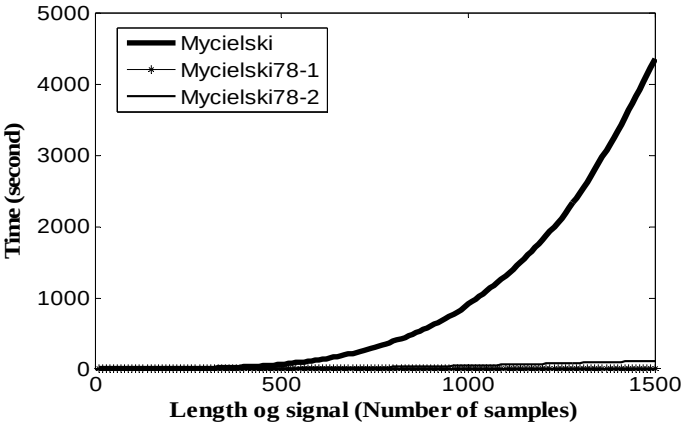

Figure 6: Execution times of Mycielski and Mycielski-78 methods.

According to the conducted experiments, the Mycielski-78 algorithm is found to be a plausible method applicable for data compression. Its prediction performance is found to be very similar to the strong Mycielski predictor. On the other hand, the time improvement by the incorporation of the dynamically built codebook is significant, and this improvement makes the algorithm to be of practical concern. The overall energy and entropy results for 8 audio sequences are given in Table I. The test sequences are selected as audio waves of various classical music compositions because such files contain strong correlative behavior enabling good predictive coding performance. It can be seen that variants of Mycielski are performing better than the LZ variants for this particular data type.

A more comprehensive test is performed over five file types obtained from the "Large Conterbury Corpus" and "Maptask 2001 Corpus". The file types are ".exe", ".wav", ".bmp", and ".txt". The average compression results are presented in Table II. The results for lossless Mycielski-78 are given in terms of how many times the entropy has changed, but the results for LZ-77 and LZ-78 are given in terms of reciprocal of the actual compression ratio. According to the results, it can be seen that Mycielski-78 performs well for files that exhibit strong correlation properties among samples such as the audio wave files and XML files that contain repeated keywords and phrases throughout the document. As a result of these results and due to the proposed implementation time improvement, the new Mycielski prediction (Mycielski-78) may be considered as an alternative in lossless compression for specific types of sources.

REFERENCES

[1] P. Jacquet, W. Szpankowski, and I. Apostol, "A Universal Predictor Based on Pattern Matching" , *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp.1462-1472, june 2002.

[2] David Salomon, "LZ77 (Sliding Window)", "Data Compression-The Complete Reference" , Section 3-2, p. 104-106, Springer-Verlag Newyork, 1998.

[3] David Salomon, "LZ78", "Data Compression-The Complete Reference" , Section 3-5, p. 112-116, Springer-Verlag Newyork, 1998.

Table I: Energy (σ^2) and sample entropy ($\mathcal{H}$) comparisons of audio test sequences.

		Signals with 1500 samples							
		wave1	**wave2**	**wave3**	**wave4**	**wave5**	**wave6**	**wave7**	**wave8**
Original Signal	σ^2	0,0039	0,0209	0,0015	0,0013	0,00065	0,000017	0,01757	0,00319
	$\mathcal{H}$	4,8717	6,1761	4,3026	4,2278	3,5952	1,0639	6,0670	4,8443
LZ77	σ^2	0,0016	0,0084	0,00065	0.00066	0,00035	0,000033	0,00675	0,00128
	$\mathcal{H}$	3,4637	3,7039	3,2335	3,1371	2,9859	2,4372	3,5667	3,3476
LZ78	σ^2	0,0020	0,0084	0,00078	0.00073	0,00040	0,000022	0,0074	0,00162
	$\mathcal{H}$	3,6955	4,1691	3,7719	3,5271	3,6345	1,9820	3,9254	3,9037
Mycielski	σ^2	0,00047	0,0068	0,00029	0,00016	0,00019	0,000019	0,00457	0,00049
	$\mathcal{H}$	3,1393	5,0978	2,9562	2,1027	2,7045	1,2247	4,7173	3,3130
Mycielski78	σ^2	0,00030	0,0089	0,00022	0,00008	0,00017	0,000005	0,00672	0,00041
	$\mathcal{H}$	3,1311	5,3170	2,9658	1,9315	2,7708	0,4554	5,1371	3,3556

Table II: Compression results for Mycielski-78, LZ-77, and LZ-78 (in terms of 1/CR).

	.exe	.wav	.bmp	.txt	.xml	Average
LZ-77	0.83	0.79	0.52	0.52	0.24	0.58
LZ-78	0.71	0.64	0.52	0.50	0.44	0.56
Mycielski-78	1.07	0.69	0.77	1.02	0.42	0.79