Automatic Generation of Error Correcting Systems Based on Convolutional Codes

Lucas F. Muniz, Daniel G. Silva, Romis Attux, Carla N. Lintzmayer, Denis G. Fantinato

***Abstract*— Due to the wide use of digital systems, bit error control is an important task. Convolutional codes are error correcting codes widely used due to their efficiency. However, the large number of parameters involved in their generation results in a problem with high complexity. In this work, we used the metaheuristics GA and BRKGA to search for efficient coding systems, which showed relevant results in our simulations.**

***Keywords*— Convolutional codes, Error correction, Genetic Algorithms.**

I. Introduction

Recent technological advances prompted the immersion in the fast and complex digital world, which demanded reliable methods for data transmission and storage. Such task could be achieved through error-correcting codes [3], whose origins can be traced back to the pioneering work of Shannon [1], which showed that errors can be made arbitrarily low if proper encoding and decoding techniques are chosen – i.e., respecting the channel's capacity. As a result, a search for efficient encoding/decoding schemes begun. Throughout decades, several efficient methods have been proposed, achieving performance close to that defined by Shannon. Among them, convolutional codes occupy a notorious position due to their relatively simple structure to introduce redundancy through the combination of elements in a given sequence [2]. However, a reliable performance is conditioned to a suitable choice of parameters, which may be a hard task due to the considerably large search space.

Considering this complex parameter adjustment task in convolutional codes, we propose the use of two different metaheuristics for performing the search: the Genetic Algorithm (GA) and the Biased Random-Key Genetic Algorithm (BRKGA). In order to constrain the search space, we adopt decoders based on the Majority Logic (ML) [3]. The performance of the metaheuristics are compared in scenarios with Binary Symmetric Channels (BSC), with very promising results. We highlight, to the best of our knowledge, that no previous work in literature uses metaheuristics to search for convolutional encoders along with the ML decoding.

Lucas F. Muniz, Carla N. Lintzmayer, Denis G. Fantinato, Centro de Matemática, Computação e Cognição, Universidade Federal do ABC, Santo André-SP, e-mail: l.muniz@aluno.ufabc.edu.br, {carla.negri, denis.fantinato}@ufabc.edu.br. Daniel G. Silva, Dep. Engenharia Elétrica, Universidade de Brasília, e-mail: danielgs@ene.unb.br. Romis Attux, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, e-mail: attux@dca.fee.unicamp.br. This work was funded by São Paulo Research Foundation (FAPESP), grant nº 2019/16997-0, and CNPq, grant nº 433887/2018-4 and 308811/2019-4.

II. Convolutional codes

Data transmission systems consist of three parts: an encoder, which receives a *message* $\mathbf{u}$ and transforms it into a *code word* $\mathbf{v}$; a *channel*, which receives $\mathbf{v}$ for transmission, but it can change some bits due to noise, so it returns a sequence $\mathbf{r}$; and a decoder, which receives $\mathbf{r}$ and transforms it into $\hat{\mathbf{u}}$. It is desired for $\hat{\mathbf{u}}$ to be as close as possible to $\mathbf{u}$.

A *convolutional code* is an encoder that inserts redundancy bits, allowing error correction. It splits $\mathbf{u}$ into blocks of k bits, which are encoded into blocks of n bits. Each encoded block depends on k bits from the original block and on m previous message blocks, being m the encoder *memory order*. Thus, a convolutional code is defined by the parameters (n,k,m) [3].

The message encoding is defined by *generator sequences* $\mathbf{g}_\ell^{(j)} = (g_{\ell,0}^{(j)}, \dots, g_{\ell,m}^{(j)})$, where $1 \leq j \leq n$ and $1 \leq \ell \leq k$, which correspond to the memory registers responsible for generating the encoded bit. As the message block encoding depends on the m previous encoded blocks, it is characterized by a time unit t that denotes the encoding system state. Thus, each message block and code word processed by the encoding in a time t are denoted by u_t e v_t, respectively. So, the convolution operation in a discrete time $t \geq 0$ will be defined by $v_t^{(j)} = \sum_{i=0}^{m} \sum_{\ell=1}^{k} u_{t-i}^{(\ell)} g_{\ell,i}^{(j)}$, where $1 \leq j \leq n$, and the sum and multiplication operations are defined by the logic (module-2) operations exclusive-OR and AND, respectively. In this work, we generated systematic convolutional codes, which uses k original bits of the message block and encodes the $n-k$ redundancy bits.

A sequence $\mathbf{r}$ received from the channel can be modeled as $\mathbf{r} = \mathbf{v} + \mathbf{e}$, where $\mathbf{e}$ is the *error sequence*. From $\mathbf{r}$, we define a *syndrome sequence* which is given by the operation $\mathbf{s} = \mathbf{r}\mathbf{H}^T$, where $\mathbf{H}$ is the *parity-check matrix*. The matrix $\mathbf{H}$ is based on the generator sequences and is constructed so that if a sequence $\mathbf{v}$ belongs to the encoding, then the property $\mathbf{v}\mathbf{H}^T = 0$ is valid. Thus, $\mathbf{s} = (\mathbf{v}+\mathbf{e})\mathbf{H}^T = \mathbf{v}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$, and $\mathbf{s}$ only depends on the channel error.

The ML method uses the orthogonal check sum concept. Each syndrome bit or sum of syndrome bits represents a sum of the channel errors, called *parity-check sums*. If there are errors in $\mathbf{r}$, some syndrome bits will value 1. From a set with J orthogonal sums over bit $e_i^{(\ell)}$ in position ℓ of block i, for $1 \leq \ell \leq k$, we define $t_{\mathbf{ML}} = \lfloor J/2 \rfloor$ as the *majority logic decoding rule*. If more than $t_{\mathbf{ML}}$ sums have value 1, then the estimated error bit value will be $\hat{e}_i^{(\ell)} = 1$. The bit recovery is performed using the module-2 operation $\hat{u}_i^{(\ell)} = r_i^{(\ell)} + \hat{e}_i^{(\ell)}$.

III. Results

Given that the convolutional code structure is based on a binary domain, the population based metaheuristics GA and BRKGA were chosen. All algorithms were implemented using *Julia*[1] language. The GA and BRKGA implementations followed the classic approaches [4], [5].

For simulation, we implemented a channel that changes a bit with probability p, hence the probability for a bit in $\mathbf{e}$ to be 1 is p, and otherwise it is $1-p$. The encoder uses a code given by the metaheuristics and the decoder uses the ML method. To evaluate error correcting property of a code C over a sequence $\mathbf{u}$, we used the accumulation error measure $E(C,\mathbf{u}) = \int_0^1 t_e(C,\mathbf{u},p)dp$, where p is the probability of error, and $t_e(C,\mathbf{u},p) = N_e(C,\mathbf{u},p)/N$ is the measured error rate, being $N_e(C,\mathbf{u},p)$ the number of measured errors in $\mathbf{v}$ (the encoding of $\mathbf{u}$ using C) after transmission through the channel, and N the total number of transmitted bits. This measure indicates the error correcting capacity of C, since it represents the total variation of errors introduced by the channel and not recovered by the code – bit inversion for error rate larger than 0.5 was not considered. Thus, low values of $E(C,\mathbf{u})$ indicate that C has high error correcting capacity.

For the search, random bit sequences $\mathbf{u}$ with 1500 bits each were generated, and $p \in [0,1]$ was varied in steps of 0.05 over the interval. Parameters (n,k,m) generated by the metaheuristics were defined in the intervals $2 \leq n \leq 20$ and $1 \leq k,m \leq 20$, to guarantee higher simulation speed and to limit the search space. Using $E(C,\mathbf{u})$, five objective functions were created, in order to guide the search for good encoder/decoder with low structural complexity (the encoder structure has spacial complexity of $O(nmk)$) and low accumulation error $E(C,\mathbf{u})$. The objective function that obtained the best results was $f(C) = R/(E(C,\mathbf{u}) + nkm)$, where $R = k/n$ is the code rate.

The GA implementation has parameters (n_G, t_p, p_m, p_p), where $n_G \in [50,200]$ is the number of generations, $t_p \in [50,150]$ is the population size, $p_m \in [0.2,0.4]$ is the percentage of mutated individuals in the population, and $p_p \in [0.1,0.3]$ is the percentage of individuals selected by the crossover operation. The parameters' values with best results during the search were (50, 150, 0.2, 0.1). The chromosomes are formed by $(n-k)k$ sequences of $m+1$ bits (systematic convolutional code) that directly represent the generator functions. We also keep parameters n, k, and m for each chromosome. The best convolutional code C_1 obtained by GA has structure (2,1,1) and is defined by the generator functions $\mathbf{g}_1^{(1)} = (1,0)$, $\mathbf{g}_1^{(2)} = (1,1)$. Its error capacity showed average $E(C_1,\mathbf{u}) \approx 0.4$.

The BRKGA implementation has parameters (n_G, t_p, ρ_e, p_m, p_e), where n_G and t_p are defined as in GA, $\rho_e \in [0.8,0.9]$ is the probability of a generated individual to inherit a key from an elite parent, $p_m \in [0.2,0.4]$ is the percentage of mutants added to the next generation, and $p_p \in [0.1,0.3]$ is the percentage of individuals that will be selected for the elite population. The parameters' values with best results during the search were (50, 50, 0.85, 0.2, 0.2). The BRKGA chromosome is formed by $(n-k)k$ real numbers, resulting in a set of real values $\{r_1,\ldots,r_{(n-k)n}\}$, each of which represent bit sequences of $m+1$ bits (generator functions). Each number r_i belongs to an interval $I_a = [ca, c(a+1))$, where $a \in \mathbb{N}$ and $c = 0.1$ is a scale factor that defines the intervals' length. Each interval I_a encodes a binary sequence S_a of length $m+1$, being $(S_a)_2 = (a)_{10}$ and the least significant bit the left-most position. The decoded generator sequence is the set of binary sequences $(S_a)_2$ that match the interval of each number r_i, for $i = 1,\ldots,(n-k)k$, from the chromosome's generator sequences set. We also keep parameters n, k, and m for each chromosome.

[1]Official website of the language: `https://julialang.org`.

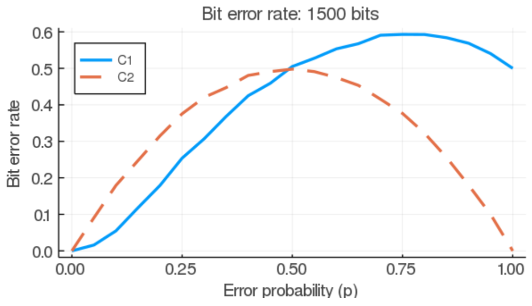

Fig. 1

Error rate of C_1 (solid) and C_2 (dashed) as a function of p.

The best convolutional code C_2 obtained by BRKGA has structure (3,2,1) and is defined by the generator functions $\mathbf{g}_1^{(1)} = (1,0)$, $\mathbf{g}_1^{(2)} = (0,0)$, $\mathbf{g}_1^{(3)} = (0,1)$, $\mathbf{g}_2^{(1)} = (0,0)$, $\mathbf{g}_2^{(2)} = (1,0)$, $\mathbf{g}_2^{(3)} = (0,1)$. The error capacity showed average $E(C_2,\mathbf{u}) \approx 0.33$.

Figure 1 shows the average error rate of C_1 and C_2, using sequences of 1500 bits when p was varied in steps of 0.05 in the interval $[0,1]$. Note that for p in the interval $[0,0.5)$, C_1 showed smaller error rate than C_2 (greater correction capacity), while the contrary occurred in the interval $(0.5,1]$.

IV. Conclusion

We proposed the use of metaheuristics GA and BRKGA to generate efficient convolutional codes, and both have found solutions with low structural complexity and low error rate. The solutions found depended directly on the objective function. Thus, the search for other objective functions could be important to find even more efficient convolutional codes.

References

[1] C. E. Shannon, "A mathematical theory of communication", *Bell system technical journal*, vol. 27, no. 3, pp. 379 - 423, 1948.

[2] W. C. Huffman and V. Pless, *Fundamentals of error-correcting codes*. Cambridge university press, 2010.

[3] S. Lin and D. J Costello, *Error control coding*. Pearson Education India, 2004.

[4] R. Martí, M. Pardalos and M. G. C. Resende, Eds., *Handbook of Heuristics*, Springer International Publishing, 2018.

[5] I. Boussaïd, J. Lepagnot and P. Siarry, "A survey on optimization metaheuristics", *Information Sciences*, vol. 237, pp. 82 - 117, 2013