Applications of Neural Networks to Telecommunications
pp. 119–124 (1993). ISBN 0-8058-1560-0

Using Genetic Algorithms to Design Constant Weight Codes

Francesc Comellas and Ramon Roca

Departament de Matemàtica Aplicada i Telemàtica
Universitat Politècnica de Catalunya
ETSE Telecomunicació, Campus Nord C-3, Gran Capitán s/n
E-08071 Barcelona, Catalonia, Spain
comellas@mat.upc.es

Abstract

Genetic algorithms have been used successfully for solving different combinatorial optimization problems. We give here a new efficient application for the search of constant weight codes. The algorithm presented is intended to generate codes with the maximum number of codewords for a given length, constant weight and minimum Hamming distance. A comparison with the simulated annealing technique is also discussed. On the other hand, the algorithm introduces a new scaling for the cost function that proved to be more effective than other scalings used in the literature.

1 Introduction

The design of good codes is of fundamental importance in a communication system. Information to be transmitted is represented via a *source code* as the ASCII code, Huffman code, etc. This source-encoded information is then sent over a channel, such as a telephone line, optical fiber, microwave link, etc. To have a reliable transmission the data are encoded again by using an *error-correcting code* that enables the detection and the correction of possible errors introduced during the transmission of the message (see Figure 1). Several modes of efficient codification are known: Hamming codes, Reed-Solomon codes, convolution codes, etc. The error-correcting code that

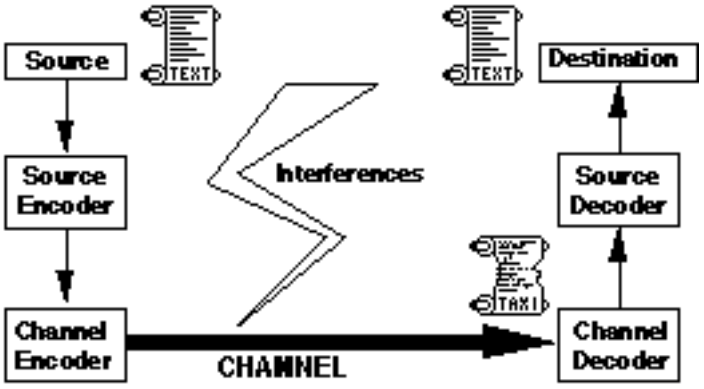

Figure 1: The coding process.

we consider in this paper is a set of binary n-tuples $C = \{x_1, x_2, \ldots, x_n\}$. Each member $w \in C$ is called a codeword. The Hamming distance $d_H(v, w)$ between two codewords is the number of bit positions where they differ. The minimum distance of C is $d_{\min} = \min_{\substack{v,\, w \in C \\ v \neq w}} d_H(v, w)$. When the error-correcting code is used in a communication system, up to $d_{\min}/2$ transmission errors can be detected and corrected. Therefore a large minimum distance $d_{\min}$ is a desirable property of an error-correcting code.

The Hamming weight of a codeword $w \in C$ is the number of ones in w. We consider only *constant-weight codes* where all codewords have the same Hamming weight.

The optimization problem studied consists of finding an error-correcting code of size as near as possible to the theoretical maximum size $A(n, d, w)$ for a fixed length n, minimum distance d (or more) and constant weight w.

Codes with size $A(n, d, w)$ are extremely difficult to find, see [1, 4]. So it is interesting to give simple methods that explicitly construct codes with size as high as possible and in this way to obtain a bound for $A(n, d, w)$. One of the methods that has proved useful is simulated annealing. El Gamal *et al.* in [2] describe the application of simulated annealing to several codification problems and in particular to the design of constant weight codes. In this paper we present a new method based on a genetic algorithm. The work of El Gamal *et al.* is used as a reference to compare both techniques. In Section 2 we describe the algorithm and also a new scaling procedure for the cost function that was developed for it. The performance of this scaling is compared with other known scalings. In Section 3 we give the results obtained and a comparison is made with the results coming from simulated annealing. Finally, in Section 4 the conclusions are presented.

2 The Genetic Algorithm

The genetic algorithm considered is based on the Simple Genetic Algorithm from the book of Goldberg [3]. Each generation has the same constant number of individuals and three genetic operators are used: reproduction, crossover and mutation.

The main aspects to decide are the representation of the solutions, the cost function and the crossover and mutation operators.

We are interested in the design of constant weight binary codes of size S, length L, weight w and minimum distance $d_{\min}$ or more. We code each possible solution by a list of $M = SL$ elements. Each position in the list has values 0 or 1. The list may be seen like the concatenation of all codewords from the corresponding code.

Cost function

The cost function used in the algorithm is:

$$f = K - \sum_{\substack{x,\, y \in C \\ x \neq y}} \frac{1}{g^2 + 1} \quad g = \begin{cases} d_H(x,y) & \text{if } d_H(x,y) < d_{\min} \\ d_{\min} & \text{if } d_H(x,y) \geq d_{\min} \end{cases}$$

where K must prevent the cost function from taking negative values.

This cost function has a clear advantage over the cost function $f = \sum_{\substack{x,\, y \in C \\ x \neq y}} [d_H(x,y)]^{-2}$ used by El Gamal *et al.* [2]. Namely, in the El Gamal's function a large distance within two given codewords may compensate a short distance between two others and the total cost function be the same than in a code where all words are at equal distance. As a consequence, this function can not discriminate correctly between codes.

Exponential scaling

The study of standard scaling mechanisms used in the selection process lead us to the introduction of a new scaling, the *exponential scaling*:

$$s = e^{\frac{C(f - avg)}{max - avg}}$$

where f is the cost function and C a constant that enables the control of the probability of selection of an individual with maximum fitness in relation to an individual with average fitness. Usual values for C are between 2 and 3.

We have compared this new scaling mechanism with the linear, and rank scalings [3].

Figure 1 shows the maximum fitness at each generation (average of 50 runs) for the three scalings. The problem was the search of a code of size 26, length 23, minimum distance 10 and constant weight 9. A population of 300 individuals was considered. Other parameters were set as follows: $p_{cross} = 0.9$, $p_{mut} = 0.6$, $K = 20$ and $C = 2$. Similar performances were found for related problems.

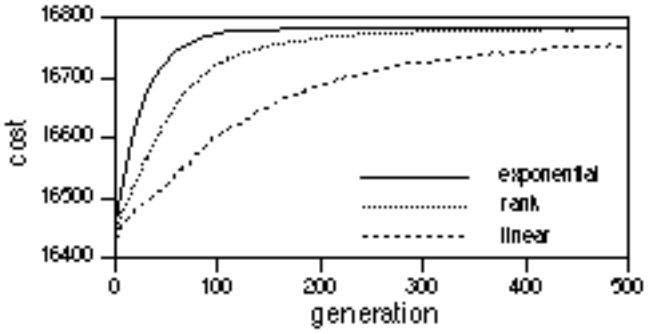

Figure 2: Exponential, rank and linear scalings compared.

The exponential scaling facilitates the control on the evolution of the algorithm, may deal with negative values of the cost function and is easy to implement. This new scaling might be also considered in other applications of genetic algorithms.

Crossover and mutation

The crossover and mutation operators were designed to deal with the characteristics of the problem. One important point is that all codewords must have the same constant weight and therefore, the genetic operators must preserve it.

The crossover on pairs of solutions $x^{par} = x_1 x_2 \ldots x_M$ and $y^{par} = y_1 y_2 \ldots y_M$, with $x_i, y_i \in \{0, 1\}$, is done by randomly choosing a cutting point $r < M$, $r \bmod L = 0$, and creating the child lists as follows: $x^{chld} = x_1 x_2 \ldots x_r y_{r+1} \ldots y_M$ and $y^{chld} = y_1 y_2 \ldots y_r x_{r+1} \ldots x_M$. The crossover operator keeps full codewords from both codes but mixes the codes. The weights of the codewords are not modified. The fact that this crossover does not create new codewords may be seen as a drawback for the algorithm. To ensure a more extensive exploration of the state space, a special mutation operator was designed. Mutation is done by complementing the value of two different list elements, chosen at random, from positions cor-

responding to the same codeword. This operator preserves also the weights of the codewords.

3 Results

For our research we implemented a small program in C (less than 500 lines) based on the Pascal code presented in the book of Goldberg [3]. All experiments were performed on Sun SPARC2 and HP Apollo 730 workstations. After initial trials, the population size was set to 300, the crossover rate to 0.9 and the mutation rate to 0.6.

With these parameters we found the results shown in Table 1. Each code

Code	Simulated Annealing	Genetic Algorithm
A(21,10,9)	22 words	22 words
A(22,10,9)	23 words	**25** words
A(23,10,7)	18 words	*17* words
A(23,10,8)	28 words	*24* words
A(23,10,9)	24 words	**26** words
A(23,10,10)	39 words	39 words
A(23,10,11)	39 words	**40** words
A(24,10,8)	33 words	*30* words
A(24,10,9)	24 words	**27** words
A(24,10,11)	57 words	**58** words

Table 1: Codes found using Simulated Annealing and a Genetic Algorithm.

was obtained in the following way: Once fixed the code size, word length, minimum distance and constant weight the program was let running until the code was found (usually in less than 300 generations) or generation 1000 was attained.

As an example, Table 2 shows a code with 26 words, length 23, minimum distance 10 and weight 9.

The genetic algorithm found an equivalent result or a better code than the simulated annealing technique in most of the cases we studied. It was not possible to find three of the codes from El Gamal *et al.* paper. They have reported that these codes were found with "more computational effort" in order to improve some new results from Conway and Sloane, see [2]. The use of a more elaborated genetic algorithm would permit to obtain these codes and other codes that at present may be only constructed by special techniques as those cited in [1].

1.	11111101100000000000100	14.	00010001101000001101101
2.	01010001110000101010010	15.	00101011010110000100100
3.	00101110101001010001000	16.	11100001000100001101010
4.	10111000010000011100001	17.	00100000100011010110011
5.	10000000110101001110100	18.	11000110101000001010001
6.	00000100000110001011111	19.	00110110010101000000011
7.	00010100111010010010100	20.	00100101101100100110000
8.	01000001111101010000001	21.	10110000001000100011011
9.	00001100001000110100111	22.	01110010000110010011000
10.	00001010100110111000001	23.	01100101010010100001001
11.	10000110110000100001110	24.	00011001000101011000110
12.	00001000011011100111000	25.	10011011000011000001001
13.	01010110000011001100100	26.	11100000001001111000100

Table 2: A constant weight code A(23,10,9).

4 Conclusions

The results show that the genetic algorithm presented in this work perform well, and that in general, with less computational effort than in simulated annealing it is possible to obtain the desired codes. The main advantage of the method is its simplicity. On the other hand, the method is robust and may be easily implemented on a parallel computer.

Acknowledgement

This work was supported by the *CICYT*, Spain, under grant TIC-92-1228-E.

References

[1] A.E. Brower, J.B. Shearer, N.J.A. Sloane and W.D. Smith. "A new table of constant weight codes", *IEEE Trans. Inform. Theory*, vol. 36, pp. 134–138, 1990.

[2] A.A.El Gamal, L.A. Hemachandra, I. Shperling and V.K. Wei. "Using simulated annealing to design good codes", *IEEE Trans. Inform. Theory*, vol. 33, pp. 116–123, 1987.

[3] D. E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[4] R.L. Graham and N.J.A. Sloane. " Lower bounds for constant weight codes", *IEEE Trans. Inform. Theory*, vol. 26, pp. 37–43, 1980.