A Genetic Approach for Generating Good Linear Block Error-Correcting Codes

Alan Barbieri, Stefano Cagnoni, and Giulio Colavolpe

Università di Parma
Dipartimento di Ingegneria dell'Informazione
Parco Area delle Scienze 181/A, 43100 Parma - ITALY
http://www.dii.unipr.it/

Abstract. In this paper we describe a method, based on a genetic algorithm, for generating good (in terms of minimum distance) linear block error-correcting codes. Preliminary experimental results indicate that the method can be very effective, especially in terms of fast generation of good sub-optimal codes.

The search space for the problem of finding good codes is too large for many of the standard search algorithms. In such cases, genetic algorithms can be very effective in finding good solutions in a relatively short time. A genetic approach to code generation was followed in [1]. In this case, however, the target was to find the entire codebook. That requirement makes such an approach useful, in practice, only for codes whose parameters are no larger than few units. In [2] a table of minimum-distance bounds for binary linear codes has been presented. These bounds may be used as references in evaluating the effectiveness of new search algorithms.

A binary error-correcting code can be used in digital transmission to reduce the probability of word errors. The coding operation consists in the transformation of a k-bits word to a n-bits one, where $n > k$. This transformation defines a codebook $\mathcal{C}$ of 2^k n-bits codewords, taken from a set of 2^n possible words. An important parameter which determines the asymptotic performance of a block code for low error probabilities is the code *minimum distance*. It is defined as the Hamming distance between the two nearest codewords, i.e.,

$$d_{min} = \min\{d(\mathbf{c}_i, \mathbf{c}_j) | \mathbf{c}_i \neq \mathbf{c}_j \in \mathcal{C}\}$$

where $d(\mathbf{c}_i, \mathbf{c}_j)$ represents the Hamming distance between codewords $\mathbf{c}_i$ and $\mathbf{c}_j$ and is defined as the number of elements by which the two codewords differ.

The algorithm we propose is aimed at finding good codes, i.e., codes with given k and n, and maximum d_{min}. In our implementation, we co-evolve more than one fixed-size population at one time. We use a classical genetic algorithm, with some adaptations to the problem under consideration. In particular, before applying crossover, the columns of every individual are sorted from best to worst, thus ensuring higher probability of a fitness increase. At the end of every iteration, the degree of similarity between individuals in each population is checked. If it is too high, a population crossover is made, i.e. the degenerated population swaps some individuals with another randomly-chosen one. Fitness is evaluated

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 1301–1302, 2004.

so that the integer part is the minimum distance, while the fractional part is related with the distance spectrum. Exact evaluation of d_{min} and of the distance spectrum is a computationally demanding problem: in [3] authors prove that it is impossible to find algorithms that approximate d_{min} in polynomial time. For this reason, fitness evaluation is the most time-consuming task of our genetic algorithm.

Some simulation results are reported in Table 1. We have used two different kinds of code: (34,15,9) (using populations of 100 individuals) and (21,11,6) (1000 individuals for each population), where the notation (n, k, d^*_{min}) indicates a $n \times k$ code with upper bound d^*_{min} on the minimum distance (see [2]). In the first case the best codes found have $d_{min} = 8$, while for a (34,15) code $d^*_{min} = 9$. However, it should be also noticed that only very few codes with $d_{min} = 9$ have been reported in literature, and that the algorithm has found several different codes with $d_{min} = 8$ in relatively few iterations. Instead in the second case the algorithm has been able to reach the theoretical bound on the minimum distance ($d_{min} = 6$) within the first 172 iterations, in all experiments but one.

Table 1. Convergence and best code found in each run for the (34,15) and the (21,11) codes: n is the experiment, a the iteration in which the first code with d_{min}=8 and 6 respectively has been found and b the iteration in which the best-fitness code has been found. The last column reports the number of codes with best fitness that have been found in each run.

n	a	b	best fitness	codes found
1	35	730	8.999603	1
2	28	463	8.999573	1
3	5	1309	8.999359	1
4	28	403	8.999664	2
5	34	2039	8.999481	1
6	27	295	8.999573	1
7	42	304	8.999451	2
8	22	1716	8.999237	1
9	25	250	8.999359	2
10	19	143	8.999390	1

n	a	b	best fitness	codes found
1	39	104	6.936035	1
2	62	62	6.935547	1
3	34	34	6.933594	1
4	N.A.	55	5.995583	1
5	172	172	6.935059	1

In conclusion, this approach is able to produce good sub-optimal codes in very few iterations. Future developments of our research will be aimed, in first place, at improving effectiveness of the search strategy by further adapting the genetic operators to the challenges posed by the problem under consideration.

References

1. Dontas, K., Jong, K.D.: Discovery of maximal distance codes using genetic algorithms. In: Proceedings of the 2nd International IEEE Conference on tools for Artificial Intelligence. (1990) 805–811
2. Brouwer, A.E., Verhoeff, T.: An updated table of minimum-distance bounds for binary linear codes. IEEE Transactions on Information Theory **39** (1993) 662–677
3. Dumer, I., Micciancio, D., Sudan, M.: Hardness of approximating the minimum distance of a linear code. IEEE Transactions on Information Theory **49** (2003) 22–37