Des. Codes Cryptogr. (2007) 45:213–217
DOI 10.1007/s10623-007-9113-2

A search algorithm for linear codes: progressive dimension growth

Tsvetan Asamov · Nuh Aydin

Received: 15 March 2007 / Revised: 6 July 2007 / Accepted: 9 July 2007 /
Published online: 10 August 2007

Abstract This paper presents an algorithm, called progressive dimension growth (PDG), for the construction of linear codes with a pre-specified length and a minimum distance. A number of new linear codes over $GF(5)$ that have been discovered via this algorithm are also presented.

Keywords New codes · Minimum distance bounds · Search algorithm · Progressive dimension Growth (PDG)

AMS Classification 94B05

1 Introduction

A linear code over $GF(q)$, the finite field with q elements, of length n, dimension k, and minimum distance (weight) d is referred to as an $[n, k, d]_q$-code. One of the most important and challenging problems in coding theory is to find optimal values of the parameters of a linear code and to explicitly construct codes whose parameters attain, or get as close as possible to, those optimal values. One formulation of the problem is: Given a q-ary linear code C with parameters $[n, k]$, find $d_q[n, k]$, the largest value of the minimum distance. Another formulation would be: Given a length n, and a minimum distance d, what is the largest value of k, the dimension of the code? We denote this value by $k_q[n, d]$. Similarly, one can fix the dimension and the minimum distance and ask for the smallest possible value of n.

Communicated by V. D. Tonchev.

T. Asamov · N. Aydin (✉)
Department of Mathematics, Kenyon College, Gambier, OH 43022, USA
e-mail: aydinn@kenyon.edu

There are tables for the bounds on the minimum distances of linear codes over small finite fields (up to certain lengths and dimensions) available online at [3][1] and [7]. The computer algebra system MAGMA [2,11] also has such a database. The values of $d_q[n,k]$ (or $k_q[n,d]$) are determined only for relatively small values of the parameters, or when $n-k$ is small. In most cases, there are gaps in the tables which means that the lower bounds on $d_q[n,k]$ (or $k_q[n,d]$) could potentially be improved to approach or attain the known upper bounds. In fact, researchers continually update the bounds on the tables by constructing new codes. However, the rate of discovery of new codes seems to be slow compared to the number of gaps present in the tables. It is also the case that as the gaps in the tables decrease, it becomes more difficult to find new codes.

It is well-known that almost all linear codes attain the Gilbert-Varshamov bound [13, p. 77]. So, if there was an efficient algorithm to compute the minimum distance of an arbitrary linear code, then randomized algorithms could be used to construct codes with optimal or good (near optimal) parameters. However, it is proven [14] that it is unlikely that such an algorithm exists: computing the minimum distance of an arbitrary linear code is NP-hard (and the corresponding decision problem is NP-complete).

Currently, there is no general purpose, deterministic, polynomial-time algorithm to search for new linear codes. There are various specialized search methods that give good results in particular cases. Recently, an intensively utilized class of such search methods has been related to the class of quasi-twisted (QT) (including quasi-cyclic (QC)) codes. A large number of new codes have been discovered by these methods (e.g. [1,4–6,8,9,12]).

The purpose of this article is to contribute to the effort of finding new codes with a new search algorithm. We call this algorithm “progressive dimension growth" (PDG). Given a length n, and a minimum distance d, the algorithm attempts to construct a linear code of dimension K over a specified finite field, where K is greater than the current lower bound for $k_q[n,d]$ (largest known dimension) and less than or equal to the known upper bound for $k_q[n,d]$. A major advantage of this algorithm is that, unlike some other search methods, it can be used for any set of parameter values without any restrictions. Recently, a heavily used search method is related to the class of QC/QT codes. Since such codes are constructed using cyclic/constayclic codes which do not necessarily exist for every possible dimension, there is an inherent restriction on the dimension. (For instance, possible dimensions of constacyclic codes depend on the degree distribution of the factors of the polynomial $x^n - a$). Also, typically one requires that the length of a constacyclic code be relatively prime with the characteristic of the field. We refer the reader to [1,6,8] for the details of these search methods. With this new algorithm, on the other hand, one can choose the parameters freely, without any inherent restrictions. The algorithm, however, does not introduce any improvements to the NP-hard problem of computing minimum distance. We make use of MAGMA's algorithms [2] to compute minimum distances. (In some special cases, MAGMA has particularly fast implementation of computing minimum distances of linear codes, e.g. [15].) Another indication of the usefulness of the algorithm is shown by the large number codes it produced that tie the parameters of currently best-known codes. We have dozens of instances of this situation over various small fields.

[1] Just before the submission of this manuscript, it has been announced that this online database is discontinued due to existence of [7] which has more explicit information for constructions.

We describe the algorithm in detail in Sect. 2 and give results we have obtained in Sect. 3.

2 The algorithm

```
**Initialize** Set the input parameters and initialize sets and variables.
Set $q, N, D, T$;
Use the record tables to determine $K$;
$BitShifts = GF(q) - \{0\}$;
$S = \{K + 1, ..., N\}$;
$G = [0]; t = 1; k = 1$;
```

The input parameters comprise the desired field characteristic q, the length N, the minimum distance D and a small positive integer T, usually between 3 and 7 but not more than $N - K$. Higher T implies a greater computational time but also a better chance for the construction of a new code. Given these initial parameters, we fix the value of K as the integer one greater than the dimension of the best-known q-ary linear code of length N and minimum distance D (obtained from the tables or the MAGMA database). The main idea of the PDG algorithm is to start with an empty code and then attempt to build new, larger codes by enlarging the current code in a dimension by dimension fashion. The whole process terminates either when a new record code is constructed or the prescribed level of T has been reached.

```
**General PDG Algorith**
Initialize
while (($t \le T$) and ($k \le K$)) do
 $\overrightarrow{v_{old}} = \vec{0}$; $\overrightarrow{v_{new}} = \vec{0}$;
 $\overrightarrow{v_{temp}} = \vec{0}$; $\overrightarrow{v_{old}}(k) = 1$;
 $d = 1$;
 Search for a suitable vector $\overrightarrow{v_{old}}$
 $G[k] = \vec{0}$;
 if ($d \ge D$) then
  $G[k] = \overrightarrow{v_{old}}$;
  $k = k + 1$;
 end if
end while
```

The search for a new suitable vector to be added to the code presents the main computational load. A key to a practical implementation of the search is the *VerifyMinimumDistance UpperBound* function. We used the procedure implemented in MAGMA. It executes the minimum weight algorithm until it determines whether or not d is an upper bound for the minimum distance of the code C, and returns *true* or *false* accordingly. This way we are able to quickly discard a large number of vectors without running the entire *MinimumDistance* procedure. (One could alternatively use the *VerifyMinimumDistanceLowerBound* function too.) Note that the generator matrices produced are in the standard form.

```
Search for a suitable vector $\overrightarrow{v_{old}}$
while $((t \leq T)$ and $(d < D))$ do
 $increment\_t = true$;
 $WordShifts =$ The set of all sequences of $BitShifts$ of length $t$;
 $Positions =$ The set of all subsets of $S$ of size $t$;
 for $p$ in $Positions$ do
  $\overrightarrow{v_{new}} = \overrightarrow{v_{old}}$;
  for $w$ in $WordShifts$ do
   for $i = 1$ to $t$ do
    $\overrightarrow{v_{new}}(p(i)) = \overrightarrow{v_{old}}(p(i)) + w(i)$;
   end for
   $G[k] = \overrightarrow{v_{new}}$; $C =< G >$;
   if $Rank(G) = k$ then
    $IsUB = VerifyMinimumDistanceUpperBound(C, d)$;
    if $IsUB = false$ then
     $d = MinimumDistance(C)$;
     $\overrightarrow{v_{temp}} = \overrightarrow{v_{new}}$;
     $increment\_t = false$;
     break $p$;
    end if
   end if
  end for
 end for
 $\overrightarrow{v_{old}} = \overrightarrow{v_{temp}}$;
 if $increment\_t$ then
  $t = t + 1$;
 end if
end while
```

3 Results

In the table below we list the parameters of the new codes we have been able to construct using the algorithm PDG. All of these codes are over $GF(5)$, and they improve the lower bounds on minimum distances of best-known linear codes. (The code in the last row does not improve the bound, so it is not numbered. It is however used as an ingredient in the construction of another record breaking code). We have executed the algorithm over the finite fields $GF(q)$, for $q = 2, 3, 4, 5, 7, 8, 9$. Although we have found many record-tiers over all fields, all record-breakers came from the field $GF(5)$. The first column in the table contains the codes we constructed using PDG only. Since the algorithm generates a sequence of nested codes, it is particularly well suited to an application of the well-known construction, Construction X [10, p. 581]. Combining PDG with Construction X, where the trivial code [1, 1, 1] along with two nested codes obtained by the algorithm are used as ingredients, we have been able to construct three additional new codes listed in the last column. The generator matrices of the new codes are available from the authors.

New codes over $GF(5)$				
1. [40, 22, 11]				
2. [41, 22, 11]				
3. [42, 23, 11]				
4. [43, 24, 11]				
5. [44, 25, 11]				
6. [45, 26, 11]	$\xrightarrow{PDG}$	(45, 27, 10)	$\xrightarrow{ConstructionX}$	14.[46, 27, 11]
7. [41, 24, 10]				
8. [42, 25, 10]				
9. [43, 26, 10]	$\xrightarrow{PDG}$	[43, 27, 9]	$\xrightarrow{ConstructionX}$	15.[44, 27, 10]
10. [39, 24, 9]				
11. [40, 25, 9]				
12. [41, 26, 9]				
13. [42, 27, 9]				
[44, 27, 10]	$\xrightarrow{PDG}$	[44, 28, 9]	$\xrightarrow{ConstructionX}$	16.[45, 28, 10]

Remarks

1. Some of the codes in the table can be obtained from others by the standard shortening construction but we have found all of these codes by the execution of PDG.
2. Some of the codes in the table were discovered by Markus Grassl and the authors independently and almost simultaneously. These are the codes numbered 6, 8, 9, 12-16.

References

1. Aydin N, Siap I, Ray-Chaudhuri DK (2001) The structure of 1-generator quasi-twisted codes and new linear codes. Des Codes Cryptogr 24(3): 313–326
2. Bosma W, Cannon J (eds) (2006) Discovering mathematics with magma: reducing the abstract to concrete. Springer, Berlin
3. Brouwer AE. Bounds (Bounds on the minimum distance of linear codes). http://www.win.tue.nl/aeb/voorlincod.html
4. Chen EZ (2007) New quasi-cyclic codes from simpex codes. IEEE Trans Inform Theory 53(1):1193
5. Daskalov R, Gulliver TA, Metodieva E (1999) New ternary linear codes. IEEE Trans Inform Theory 45(5):1687–1688
6. Daskalov R, Hristov P (2003) New quasi-twisted degenerate ternary linear codes. IEEE Trans Inform Theory 49(9):2259–2263
7. Grassl M. Bounds on the minimum distance of linear codes. http://www.codetables.de
8. Grassl M, White G (2005) New codes from chains of quasi-cyclic codes. In: Proceedings of IEEE international symposium on information theory (ISIT 2005), Adelaide, Australia, September 2005, pp 2095–2099
9. Gulliver TA, Östergard PRJ (2000) New binary linear codes. Ars Combinatoria 56: 105–112
10. MacWilliams FJ, Sloane NJA (1977) The theory of error correcting codes. North Holland, Amsterdam
11. MAGMA computer algebra system. http://magma.maths.usyd.edu.au/magma/MagmaInfo.html
12. Siap I, Aydin N, Ray-Chaudhuri DK (2000) New ternary quasi-cyclic codes with better minimum distances. IEEE Trans Inform Theory 46(4):1554–1558
13. Tsfasman MA, Vlădut SG (1991) Algebraic geometry codes. Kluwert, Dordrecht
14. Vardy A (1997) The intractability of computing the minimum distance of a code. IEEE Trans Inform Theory 43(6):1757–1766
15. White G (preprint) An improved minimum weight algorithm for quasi-cyclic and quasi-twisted codes