mathematics

MDPI

Article

An Extension of the Brouwer–Zimmermann Algorithm for Calculating the Minimum Weight of a Linear Code

Stefka Bouyuklieva [1,*] and Iliya Bouyukliev [2]

[1] Faculty of Mathematics and Informatics, St. Cyril and St. Methodius University of Veliko Tarnovo, 5000 Veliko Tarnovo, Bulgaria
[2] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, 5000 Veliko Tarnovo, Bulgaria; iliyab@math.bas.bg
* Correspondence: stefka@ts.uni-vt.bg

Abstract: A modification of the Brouwer–Zimmermann algorithm for calculating the minimum weight of a linear code over a finite field is presented. The aim was to reduce the number of codewords for consideration. The reduction is significant in cases where the length of a code is not divisible by its dimensions. The proposed algorithm can also be used to find all codewords of weight less than a given constant. The algorithm is implemented in the software package QEXTNEWEDITION.

Keywords: linear code; Hamming weight; Brouwer–Zimmermann algorithm

MSC: 94B05; 05A18

Citation: Bouyuklieva, S.; Bouyukliev, I. An Extension of the Brouwer–Zimmermann Algorithm for Calculating the Minimum Weight of a Linear Code. *Mathematics* **2021**, *9*, 2354. https://doi.org/10.3390/math9192354

Academic Editor: Patrick Solé

Received: 27 August 2021
Accepted: 18 September 2021
Published: 22 September 2021

1. Introduction

In 1997, Alexander Vardy proved that for general binary linear code, computing the minimum weight is an NP-hard problem, and the corresponding decision problem is NP-complete [1]. The popular practical algorithms are based on a maximum number of generating matrices $G_1, G_2, \ldots, G_s$ with disjoint sets of systematic coordinates. Some are the Brouwer–Zimmermann algorithm and its various modifications for cyclic codes, quasi-cyclic codes, divisible codes, etc. (see [2–5]). Such algorithms are implemented in the software package MAGMA [6].

The basic idea is that after taking $1, 2, \ldots, l$ linear combinations of the rows of all matrices, all codewords with weight $\leq w$ (depending on l) will be generated, provided there are any. Additionally, if so far the lightest generated codeword has a weight $w+1$, then the minimum distance $d(C)$ is equal to $w+1$. The problem we are considering here is which matrices among $G_1, G_2, \ldots, G_s$ to use and which linear combinations of their rows to take so that the number of the codewords generated is minimal. We propose a new modification of the Brouwer–Zimmermann algorithm that allows the generation of fewer codewords than the algorithm implemented in MAGMA. The MAGMA algorithm is described by M. Grassl in [4] and is implemented in the function MINIMUMWEIGHT(C). The advantages of the proposed approach are very visible in code with length n, close but less than $2k$, where k is the dimension. This can be seen in the table of experimental results. The number of required vector operations in some cases is many times smaller in our algorithm compared to the algorithms we know.

In Section 2, we present some important properties of the minimum weight of linear code and a short description of the Brouwer–Zimmermann algorithm. We give the theoretical basis of our method in Section 3. Section 4 is devoted to the new algorithm that we propose. The first difference between our algorithm and the BZ algorithm is that the Brouwer–Zimmermann algorithm uses overlapping information sets, but we partition the set of coordinate positions into disjoint sets called systematic sets (full and reduced). For each of the considered sets, the algorithm constructs a corresponding generator matrix of

the code (this holds for both algorithms). Another important difference is in the number of considered linear combinations of rows of generator matrices needed to calculate the minimum weight. If the length n is not divisible by the dimension k, our algorithm needs fewer codewords to obtain the value of $d(C)$. We list and explain some computational results in Section 5.

2. Preliminaries

Let $\mathbb{F}_q$ be a finite field with q elements and $\mathbb{F}_q^n$ be the n-dimensional vector space over $\mathbb{F}_q$. A linear $[n,k]_q$ code C is a k-dimensional subspace of the vector space $\mathbb{F}_q^n$. A matrix whose rows form a basis of C is called a generator matrix of this code.

The (Hamming) *distance* $d(x,y)$ between two vectors $x, y \in \mathbb{F}_q^n$ is the number of coordinate positions in which they differ, and the (Hamming) *weight* $\mathrm{wt}(x)$ of a vector $x \in \mathbb{F}_q^n$ is the number of its nonzero coordinates. The *minimum distance* of a linear code is the smallest distance between two different codewords, and the *minimum weight* is the smallest weight among all non-zero codewords of the code. If C is a linear code, then its minimum weight and minimum distance are equal. An $[n,k,d]_q$ code is a linear code over $\mathbb{F}_q$ with minimum distance d. Then $d \leq n$, but there are much better upper bounds for the minimum weight. In our algorithm, we use the Singleton bound, namely, $d \leq n - k + 1$.

The most widely used algorithm for computing the minimum weight of a linear $[n,k]_q$ code was designed by A. Brouwer and subsequently extended by K.-H. Zimmermann. The literature about this problem refers to it as the Brouwer–Zimmermann algorithm, abbreviated the BZ algorithm (see [2,4] for its description). The BZ algorithm was outlined in [5], where the authors proposed its extension, which consists of a good (sometimes best possible) sequence of information sets for a given code. We propose another extension to the BZ algorithm which is related to the linear combinations needed to compute the minimum weight of the given code.

The main idea of the BZ algorithm is to enumerate the codewords in such a way that one not only obtains an upper bound on the minimum weight of the code via the minimum of the weights of the words that have been encountered, but also a lower bound on the minimum weight. For this, the concept of information sets is needed [4].

Definition 1 ((Information Set) [4]). *Let C be a linear $[n,k,d]_q$ code of length n and dimension k over $\mathbb{F}_q$. A subset $T \subseteq \{1,2,\ldots,n\}$ of size $|T| = k$ is called an information set if the corresponding columns in a generator matrix G of C are linearly independent. Then there also exists a systematic generator matrix G_T of C such that the columns of G_T specified by T form an identity matrix.*

The BZ algorithm uses a family of information sets $T_1, \ldots, T_l$ for the code C such that $T_1 \cup \cdots \cup T_l = \{1,2,\ldots,n\}$. These information sets are not necessarily disjoint, and therefore a sequence $r_1, \ldots, r_l$ of nonnegative integers, called relative ranks, is defined, where $r_i = |T_i \setminus (T_1 \cup \cdots \cup T_{i-1})|$. The methods used in [2,4] for constructing sets T_i and their corresponding systematic matrices G_i produce them in such a way that the sequence of relative ranks is non-increasing.

The Brouwer–Zimmermann algorithm goes in the following way: In the initial step, the upper bound on the minimum weight is $\overline{d} = n - k + 1$ (Singleton bound), and the lower bound is $\underline{d} = 1$. Each step depends on the integers w and j so that the algorithm enumerates all codewords uG_j such that $\mathrm{wt}(u) = w$. During this process, if a codeword x with $\mathrm{wt}(x) < \overline{d}$ is generated, then the algorithm updates $\overline{d}$ according to

$$\overline{d} = \min\{\overline{d}, \min\{\mathrm{wt}(uG_j) : u \in \mathbb{F}_q^k, \mathrm{wt}(u) = w\}\}.$$

The value of $\underline{d}$ is also updated (see [4,5] for more details). The algorithm then tests whether $\underline{d} \geq \overline{d}$, and if so, then it terminates with the conclusion that $d(C) = \overline{d}$.

3. Systematic Sets

We define the term *systematic set*, which is similar to the *information set*, but the difference is in the size of the set.

Definition 2. Let C be a linear $[n,k]_q$ code with a generator matrix G. A subset $T \subseteq \{1,2,\ldots,n\}$ of size $|T| \leq k$ is called a *systematic set* for C if the corresponding columns in G are linearly independent. If $|T| = k$. then T is a *full systematic set*; otherwise it is a *reduced systematic set*.

The terms *information set* and *full systematic set* coincide. If T is a systematic set, then C has a generator matrix G_T such that the columns corresponding to the set T form a submatrix of G_T which is equivalent to the identity matrix $I_{|T|}$ extended with $k - |T|$ zero rows.

The following notation is very important for this research. Let $T_1, \ldots, T_s$ be disjoint systematic sets for the linear $[n,k]_q$ code C, such that $T_1 \cup \cdots \cup T_s = \{1,\ldots,n\}$. Suppose that $T_1, \ldots, T_t$, and $t \leq s$, are full systematic sets, and the other $s - t$ systematic sets are reduced. Denote by $U_i^{(a_i)}$ the set of all nonzero codewords with $j \leq a_i$ nonzero coordinates in the systematic set T_i, $a_i \geq 0$, and $i = 1,\ldots,s$. If $v|_{T_i}$ is the restriction of the vector $v \in C$ on the set T_i, then $v \in U_i^{(a_i)}$ if and only if $v \neq 0$ and $\mathrm{wt}(v|_{T_i}) \leq a_i$. In other words,

$$U_i^{(a_i)} = \{v \in C \setminus \{0\} \mid \mathrm{wt}(v|_{T_i}) \leq a_i\}.$$

According to this definition, $U_i^{(a_i)} \subseteq U_i^{(a_i+1)}$ for any $a_i \geq 0$. These sets of codewords underlie our approach and make it different from known methods. We use a union of such sets (denoted as usual by $\bigcup U_i^{(a_i)}$), and a multiset sum (denoted by $\biguplus U_i^{(a_i)}$), which may contain some codewords repeated several times.

Consider first the case when $n = tk$ for an integer t and C has t disjoint information sets $T_1, \ldots, T_t$, so $s = t$. Such is, for example, the self-dual code or the t-CIS (complementary information set) code. Obviously, the minimum distance of such code is at least t. This is because for each codeword corresponding to a non-trivial linear combination of rows of any generating matrix, there will be at least one nonzero coordinate for each information set. If we compute the weights of all rows of G_1, we will determine all codewords with weight t, but not those with weight $t+1$, because the code may contain a codeword with two nonzero coordinates from T_1. If we continue with computing the weights on all rows of G_2, we will determine all codewords with weight $t+1$ (if any), etc.

The following proposition is basic for both algorithms (Algorithm 1 and the BZ algorithm) for which all systematic sets are full. In this case, $U_i^{(a_i)}$ consists of the linear combinations of up to a_i rows of the matrix G_i.

Algorithm 1: Minimum Weight

INPUT: Linear code C with disjoint systematic sets $T_1, \ldots, T_s$, where
$T_1 \cup \cdots \cup T_s = \{1,2\ldots,n\}$, $|T_1| = \cdots = |T_t| = k > |T_{t+1}| \geq \cdots \geq |T_s|$,
$G_1, \ldots, G_s$—the corresponding systematic matrices.
OUTPUT: The minimum weight $d(C)$.
$\delta := t - 1$;
$ub := n - k + 1$;
while $(\delta + 1 < ub)$ do
$\delta := \delta + 1$;
Generate Ω_δ;
$ub := \min\{ub, \min\{\mathrm{wt}(v) | v \in \Omega_\delta\}\}$;
end while;
return $d(C) = ub$.

Proposition 1. *Let $a_1, \ldots, a_t$ be integers, such that $0 \leq a_i \leq k$ and $a_1 + \cdots + a_t \geq 1$, and $G_1, \ldots, G_t$ be the generator matrices of C, which correspond to the sets $T_1, \ldots, T_t$, respectively. If all T_i are full systematic sets, then the set $U = U_1^{(a_1)} \cup \cdots \cup U_t^{(a_t)}$ contains all codewords with weight $w \leq a_1 + a_2 + \cdots + a_t + t - 1$.*

Proof. Let $m = a_1 + a_2 + \cdots + a_t + t - 1$ and $v \in C$ be a nonzero codeword of weight $w \leq m$. Let $b_i = \mathrm{wt}(v|_{T_i})$, where $v|_{T_i}$ is the restriction of v on the systematic set T_i, $i = 1, \ldots, t$. It turns out that $w = b_1 + \cdots + b_t$ and $b_i > 0$ for all $i = 1, \ldots, t$. Suppose that $b_i > a_i$ for each $i \in \{1, \ldots, t\}$. Hence, $b_i \geq a_i + 1$ and $m \geq w \geq a_1 + a_2 + \cdots + a_t + t = m + 1$ constitute a contradiction. Hence, there is at least one index i, $1 \leq i \leq t$, such that $b_i \leq a_i$, which means that $v \in U_i^{(a_i)} \subset U$. □

We would like to mention here that a similar technique has been applied in the study of zero-divisor graph structure in [7].

Now consider the general case where not all systematic sets are full. Let $\{1, \ldots, n\} = T_1 \cup \cdots \cup T_t \cup T_{t+1} \cup \cdots \cup T_s$, where $T_1, \ldots, T_t$ are full systematic sets for the code C; $T_{t+1}, \ldots, T_s$ are reduced systematic sets; and $T_i \cap T_j = \emptyset$ for $1 \leq i < j \leq s$. Without loss of generality we can suppose that $|T_1| = \cdots = |T_t| = k > |T_{t+1}| \geq \cdots \geq |T_s|$, and

$$G_1 = (I_k | A_1'), \ldots, G_t = (A_t | I_k | A_t'),$$

$$G_{t+1} = \left(A_{t+1} \middle| \begin{matrix} I_{|T_{t+1}|} \\ 0 \end{matrix} \middle| \begin{matrix} A_{t+1}' \\ 0 \end{matrix} \right), \ldots, G_s = \left(A_s \middle| \begin{matrix} I_{|T_s|} \\ 0 \end{matrix} \right).$$

Theorem 1. *Let C be a linear $[n, k]_q$ code with disjoint systematic sets $T_1, \ldots, T_s$, such that $T_1 \cup \cdots \cup T_s = \{1, \ldots, n\}$. Suppose that the systematic sets $T_1, \ldots, T_t$ are full, and the other $s - t$ sets are reduced. Let $r \leq s - t$ and $a_1, \ldots, a_{t+r}$ be nonnegative integers, such that $a_i \leq k$ and $a_1 + \cdots + a_{t+r} \geq 1$. Then the set $U = U_1^{(a_1)} \cup \cdots \cup U_{t+r}^{(a_{t+r})}$ contains all codewords with weight $w \leq a_1 + a_2 + \cdots + a_{t+r} + t + r - 1$.*

Proof. The proof is similar to the proof of Proposition 1. Let $m = a_1 + a_2 + \cdots + a_{t+r} + t + r - 1 \geq t + r$ and $v \in C$ be a codeword of weight $w \leq m$. Let $b_i = \mathrm{wt}(v|_{T_i})$, where $v|_{T_i}$ is the restriction of v on the systematic set T_i, $i = 1, \ldots, s$. It turns out that $w = b_1 + \cdots + b_s$ and $b_i > 0$ for all $i = 1, \ldots, t$. Suppose that $b_i \geq a_i + 1$ for all $i = 1, \ldots, t + r$. Then

$$\begin{aligned} m \geq w &\geq a_1 + a_2 + \cdots + a_{t+r} + t + r + b_{t+r+1} + \cdots + b_s \\ &= m + 1 + b_{t+r+1} + \cdots + b_s \geq m + 1, \end{aligned}$$

which is not possible. Hence, there is at least one index i, $1 \leq i \leq t + r$, such that $b_i \leq a_i$, which means that $v \in U_i^{(a_i)} \subseteq U$. □

If $r_i = |T_i|$, $i = 1, \ldots, s$, then $r_1 = \ldots = r_t = k > r_{t+1} \geq \cdots \geq r_s \geq 0$. For the set $U_i^{(a_i)}$ we have

$$|U_i^{(0)}| = \begin{cases} 0 & \text{if } i \leq t \\ q^{k-r_i} - 1 & \text{if } i > t \end{cases}$$

If $a > 0$ then

$$|U_i^{(a)}| = \begin{cases} \sum_{j=1}^{a} (q-1)^j \binom{k}{j} & \text{if } i \leq t \\ q^{k-r_i}(1 + \sum_{j=1}^{a} (q-1)^j \binom{|T_i|}{j}) - 1 & \text{if } i > t \end{cases}$$

These formulae show why we use the parameter r in Theorem 1 and then in the algorithm. Let $G_1, \ldots, G_s$ be the generator matrices that correspond to the sets $T_1, \ldots, T_s$, respectively. If the matrix G_i is not required in a current step of the algorithm for $1 \leq i \leq t$, then we take $a_i = 0$ and then $U_i^{(a_i)} = \emptyset$. However, if $t < i \leq s$ then the set $U_i^{(0)}$ contains

$q^{k-|T_i|} - 1 \geq q - 1$ codewords. Therefore, when the matrix G_i is not required, we reduce the number of the considered nonnegative integers a_i.

4. The Algorithm

We are looking for the minimum weight of the linear $[n,k]_q$ code C. Let the set of the coordinate positions $\{1,\ldots,n\}$ for the code C is partitioned into s systematic sets $T_1,\ldots,T_s$, such that $|T_1| = \cdots = |T_t| = k > |T_{t+1}| \geq \cdots \geq |T_s|$, $t \leq s$.

The algorithm uses an integer δ which increases by one from the integer t until it reaches $d(C)-1$ or $d(C)$. For a particular value of δ, we use consistently $r = 0, 1, \ldots, s-t$ of the reduced systematic sets. For a fixed r, let $\delta = a_1 + \cdots + a_{t+r} + t + r - 1$ for some nonnegative integers $a_1,\ldots,a_{t+r}$. We define multisets Ω_δ recursively in the following way:

- For $\delta = t$ we set $\Omega_\delta = U_1^{(1)} \uplus U_2^{(0)} \uplus \cdots \uplus U_t^{(0)}$.
- Let $\delta = m \geq t$ and $\Omega_m = U_1^{(a_1)} \uplus U_2^{(a_2)} \uplus \cdots \uplus U_{t+r}^{(a_{t+r})}$, where $a_1,\ldots,a_{t+r}$ are nonnegative integers such that $m = a_1 + \cdots + a_{t+r} + t + r - 1$, $0 \leq r \leq s-t$.
- Take $\delta = m+1$. The nonnegative integers $r, a_1,\ldots,a_{t+r}$ are the same as above so that $\Omega_m = U_1^{(a_1)} \uplus U_2^{(a_2)} \uplus \cdots \uplus U_{t+r}^{(a_{t+r})}$. Let

$$S_{m+1} = \begin{cases} \min\{|U_j^{(a_j+1)}|,\ 1 \leq j \leq t+r\} & \text{if } t+r = s \\ \min\{|U_1^{(a_1+1)}|,\ldots,|U_{t+r}^{(a_{t+r}+1)}|,|U_{t+r+1}^{(0)}|\}, & \text{if } t+r < s. \end{cases}$$

and $j = l$ is the smallest value of the parameter j for which the sum is minimal (is equal to S_{m+1}). If $l \leq t+r$, we set

$$\Omega_{m+1} = U_1^{(a_1)} \uplus \cdots \uplus U_{l-1}^{(a_{l-1})} \uplus U_l^{(a_l+1)} \uplus U_{l+1}^{(a_{l+1})} \uplus \cdots \uplus U_{t+r}^{(a_{t+r})} = \Omega_m \uplus U_l^{(a_l+1)}.$$

If $r < s-t$ and $l = t+r+1$, then $\Omega_{m+1} = \Omega_m \uplus U_{t+r+1}^{(0)}$. We write briefly

$$\Omega_{m+1} = \begin{cases} \Omega_m \uplus U_l^{(a_l+1)}, & \text{if } l \leq t+r \\ \Omega_m \uplus U_{t+r+1}^{(0)}, & \text{otherwise.} \end{cases}$$

According to Theorem 1, all codewords in C of weight $\leq \delta$ belong to the set Ω_δ. If the code does not contain nonzero codewords of weight $\leq \delta$, then δ increases by one, and this value is taken as a lower bound lb for the minimum weight of the code. The lightest codeword in Ω_δ defines an upper bound for the minimum weight of C. The exact value of $d(C)$ is obtained when $lb + 1 = ub$, or during the generation process of Ω_δ, a codeword of weight $lb = \delta$ is obtained.

The reason we take Ω_δ this way is to generate as few codewords as possible, while making sure we get a codeword with a minimum weight. The summands $a_1,\ldots,a_{t+r}$ in the expression $\delta = a_1 + \cdots + a_{t+r} + t + r - 1$ were chosen so that the number of the codewords in the set Ω_δ should be as small as possible.

The pseudocode is presented as Algorithm 1.

The correctness of the algorithm follows from Theorem 1. If all considered systematic sets are full ($s = t$), our algorithm is similar to the other variants of the Brouwer–Zimmermann algorithm [4]. The number of elementary operations performed by the algorithm depends on k, d, and the sizes of the systematic sets. For given n, k, q, and sizes of the disjoint systematic sets, it is theoretically possible to determine both the minimum and maximum numbers of codewords to be generated to prove that the minimum weight of the corresponding code is equal to d. If $\Omega_{d-1} = U_1^{(a_1)} \uplus \cdots \uplus U_{t+r}^{(a_{t+r})}$, then the minimum number of the generated codewords is

$$\frac{1}{q-1} \sum_{i=1}^{t+r} \sum_{j=0}^{a_i} |U_j^{(j)}|, \tag{1}$$

and it is attained when there is a codeword $v \in \Omega_{d-1}$ of weight d. If all vectors in Ω_{d-1} have weights larger than d, then the algorithm generates the set Ω_d. The experimental results show that the upper bound ub reaches the minimum weight at an early stage of the algorithm, and this can be used to solve some problems faster, for example, when we want to prove that the minimum weight is $\leq d$ for a given integer d.

In the `while-loop`, to construct Ω_δ we use its subset $\Omega_{\delta-1}$ because its codewords have already been generated. To get each vector in Ω_δ with only one vector addition, we generate some codewords more than once and the formula (1) actually gives the number of these additions. More details on this process are given in [8] for prime fields and in [9] for composite fields. Moreover, instead of the sets $U_i^{(a)}$ we take $\widehat{U}_i^{(a)} \subseteq U_i^{(a)}$, where $\widehat{U}_i^{(a)}$ consists of nonproportional vectors and any vector from $U_i^{(a)}$ is proportional to a vector in $\widehat{U}_i^{(a)}$ (therefore we divide by $q-1$ in the formula (1)). Then we generate the multiset Ω_δ as a multiset sum of $\widehat{U}_i^{(a_i)}$ in the same way, as is explained in the beginning of this section.

The whole algorithm consists of several subproblems. The time complexity for finding an optimal solution for constructing systematic sets is given in detail in [5]. The complexity of the rest is difficult to be estimated because it depends on two parameters that are not known at the beginning. One of them includes the number and cardinality of the systematic sets, which depend on the structure of the code. The second one is the minimum weight we are looking for. While the problem with the second parameter can be solved by considering the corresponding decision problem [5], the first parameter is difficult to estimate. The time complexity of the algorithm for generating the codewords in a set $U_i^{(a)}$ is given in [8].

The same approach can also be used to obtain the set of codewords with minimum weight, and the set of codewords with a given weight $w > d$. This is necessary, for example, in equivalence tests and finding automorphism groups of linear code. This approach is more effective than the one given in [8]. The procedure is the following:

1. Finding the nonnegative integers $r, a_1, \ldots, a_{t+r}$, for which $\Omega_w = U_1^{(a_1)} \uplus \cdots \uplus U_{t+r}^{(a_{t+r})}$.
2. Generating the set $U_1^{(a_1)}$ and save its vectors with weight w.
3. For $2 \leq i \leq t+r$, generating the set $U_i^{(a_i)}$ and save the codewords $v \in U_i^{(a_i)}$ with weight w, for which $\mathrm{wt}(v|_{T_j}) < a_j$ for $j = 1, \ldots, i-1$.

With this procedure we obtain all codewords of weight w without repetitions. If we want to have a maximal set of nonproportional codewords of weight w, we apply the same procedure, but using the sets $\widehat{U}_i^{(a_i)}$ for $i = 1, \ldots, s$.

The following example illustrates Algorithm 1. In fact, the code itself is not important to see how the algorithm works; the parameters and the sizes of the systematic sets are sufficient.

Example 1. *Let C be a $[20, 8, 7]$ binary code with three systematic sets T_1, T_2, T_3 such that $|T_1| = |T_2| = 8$, $|T_3| = 4$. Obviously, $t = 2 \leq d(C) \leq n - k + 1 = 13$, and so in the beginning $lb = \delta = 2$ and $ub = 13$. Next we follow the steps of the algorithm:*

- *$\delta = 2$) We take $\Omega_2 = U_1^{(1)} \uplus U_2^{(0)}$ and $ub = \min\{\mathrm{wt}(v) | v \in \Omega_2\} \geq 7 > \delta$.*
- *$\delta = 3$) Since $\min\{|U_1^{(2)}|, |U_2^{(1)}|, |U_3^{(0)}|\} = |U_2^{(1)}| = 8$, we get $\Omega_3 = U_1^{(1)} \uplus U_2^{(1)}$ and $ub = \min\{ub, \min\{\mathrm{wt}(v) | v \in \Omega_3\}\} \geq 7 > \delta$.*
- *$\delta = 4$) In this case $\min\{|U_1^{(2)}|, |U_2^{(2)}|, |U_3^{(0)}|\} = |U_3^{(0)}| = 15$; hence, $r = 1$ and $\Omega_4 = U_1^{(1)} \uplus U_2^{(1)} \uplus U_3^{(0)}$.*
- *$\delta = 5$) In this case $\Omega_5 = U_1^{(2)} \uplus U_2^{(1)} \uplus U_3^{(0)}$, $|\Omega_5| = 60$.*
- *$\delta = 6$) Now $\Omega_6 = U_1^{(2)} \uplus U_2^{(2)} \uplus U_3^{(0)}$, $|\Omega_6| = 88$, $ub \geq 7$. If $ub = 7$, after this step the algorithm returns the answer $d(C) = 7$. Otherwise we need also the next step $\delta = 7$.*
- *$\delta = 7$) Then $\Omega_7 = U_1^{(3)} \cup U_2^{(2)} \cup U_3^{(0)}$, $|\Omega_7| = 144$, and $ub = 7$, and so $d(C) = 7$.*

5. Computational Results

For a given $[n,k,d]_q$ linear code C, we computed the number of codewords that have to be generated, required to prove that $\mathrm{wt}(C) = d$ by the BZ algorithm, and by our algorithm. For the partitioning of the set of coordinate positions $\{1,\ldots,n\}$ we used the same strategy as the BZ algorithm. The difference is that the BZ algorithm uses overlapping information sets. As is described in [4], for a general linear code with systematic generator matrix $G_1 = (I|A_1)$, the rank of the matrix A_1 can be less than k, which implies that there is no information set T_2 with $T_1 \cap T_2 = \emptyset$. In this situation, we can obtain a reduced systematic set T_2 of size $|T_2| = \mathrm{rank}A_1$ (this reduced systematic set is called a partial information set in [4]). Our algorithm uses this reduced systematic set, unlike the BZ algorithm, which uses an information set $I_2 \supseteq T_2$ obtained from T_2 and $k - r_2$ elements of T_1.

We performed experiments with some $[n,k,d]_q$ codes with MAGMA, and our program included in the package QEXTNEWEDITION. We have used MAGMA V2.25-2 via online Magma Calculator run in a virtual machine on an Intel Xeon Processor E3-1220, 3.10 GHz. Our implementation was executed on Intel Core i7-6700hq 2.60 GHz processor. We present the number of the enumerated vectors with both programs in Table 1. Moreover, we give the amounts of time for computing the minimum weights of the corresponding codes with MAGMA and QEXTNEWEDITION. To compare the algorithms, only the numbers of codewords generated are important, because we ran the programs on different computers.

Table 1. Minimum weights of $[n,k,d]_q$ linear codes.

Random Codes	MAGMA # Codewords	s	QextNewEdition # Codewords	s
$[114,60,13]_2$	20,784,896,304	100.91	310,623,009	2.62
$[115,60,13]_2$	6,001,753,644	28.56	198,461,377	1.52
$[116,60,14]_2$	6,001,753,644	28.46	579,155,882	4.69
$[117,60,14]_2$	3,443,132,799	16.26	430,378,776	2.80
$[118,60,14]_2$	884,511,954	3.98	266,773,648	1.84
$[119,60,14]_2$	498,305,034	2.31	175,745,024	1.05
$[145,50,25]_2$	9,481,372,155	61.53	1,943,266,923	21.11
$[45,15,16]_5$	1,053,993	0.030	1,173,563	0.026
$[44,15,16]_5$	1,822,761	0.037	1,454,628	0.020
$[43,15,16]_5$	2,591,529	0.047	2,065,165	0.024
$[42,15,15]_5$	2,591,529	0.054	1,580,947	0.015
$[41,15,14]_5$	2,591,529	0.040	2,152,729	0.025
$[40,15,14]_5$	7,716,649	0.124	1,697,666	0.017
$[39,15,12]_5$	1,727,701	0.030	354,917	0.009

In fact, in the case when C has disjoint full systematic sets and $n = tk$, our algorithm and the BZ algorithm are similar. Therefore, in this case the numbers of codewords generated were almost the same, but there was a difference when $n = (t-1)k + r$ for $1 \le r < k$. Take, for example, the $[115,60,13]_2$ code in Table 1. To calculate the minimum weight of this code, Magma uses two overlapping information sets I_1 and I_2, their corresponding generator matrices G_{I_1} and G_{I_2}, and generates $2\sum_{i=1}^{8}\binom{60}{i} = 6{,}001{,}753{,}644$ codewords. Our algorithm operates with two systematic sets T_1 and T_2, $|T_1| = 60$, and $|T_2| = 55$, so $\Omega_{12} = U_1^{(6)} \uplus U_2^{(5)}$. The program generated only $\sum_{i=1}^{6}|U_1^{(i)}| + \sum_{i=0}^{5}|U_2^{(i)}| = \sum_{i=1}^{6}(7-i)\binom{60}{i} + 2^5\sum_{i=1}^{5}(6-i)\binom{55}{i} + 6(2^5-1) = 198{,}461{,}377$ codewords. Actually, as we see in this example, when we know the minimum distance in advance, we can compute the number of the codewords to be generated by a formula independently before running the algorithm.

6. Conclusions

In conclusion, we would like to add that Algorithm 1 is a competitive version of the classical Brouwer–Zimmermann algorithm. This approach can be further developed for

special types of codes, such as cyclic, quasi-cyclic, and divisible codes. The algorithm is implemented in the software package QEXTNEWEDITION [10].

At the time of preparing this paper, three modules of QEXTNEWEDITION are publicly available and can be freely downloaded. These are the programs GENERATION for classifying linear codes over small finite fields; LCEQUIVALENCE, designed to obtain the inequivalent codes in a set of linear codes over a finite field with $q < 65$ elements and compute their automorphism groups; and WDHV, which calculates the weight distribution of linear code. As a stand-alone program, the implementation of the presented algorithm is not finalized. The current LINUX or WINDOWS version will be sent upon request by the authors.

We would like to mention some open problems related to the algorithms for computing the minimum weight of linear code. The standard way to find the minimum distance is through the weight spectrum of the code. In practice, the Brouwer–Zimmermann type algorithms are effective for codes with small numbers of disjoint systematic sets. One of the open questions is to determine for which parameters each of the two approaches is more effective.

Our experimental results show that when a matrix with a reduced systematic set is used, the proposed algorithm is more efficient than the one implemented in MAGMA. The question arises as to whether this is true for all such cases.

A parallel implementation of the Brouwer–Zimmermann algorithm is presented in [11]. The codes that the authors considered are very suitable for our algorithm. Therefore, the natural question arises about the parallel implementation of Algorithm 1.

Author Contributions: Conceptualization, I.B.; methodology, I.B.; software, I.B.; validation, I.B. and S.B.; formal analysis, S.B.; resources, I.B. and S.B; data curation, I.B.; writing—original draft preparation, S.B.; project administration, I.B. and S.B.; funding acquisition, I.B. and S.B. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Bulgarian National Science Fund grant number KP-06-N32/2-2019, and the Bulgarian Ministry of Education and Science, Grant No. D01-387/18.12.2020 for NCHDC, a part of the Bulgarian National Roadmap on RIs.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The current LINUX or WINDOWS version of the software will be sent upon request by the authors.

Acknowledgments: We are greatly indebted to the unknown referees for their useful suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vardy, A. The intractability of Computing the Minimum distance of a Code. *IEEE Trans. Inform. Theory* **1997**, *43*, 1757–1766. [CrossRef]
2. Betten, A.; Braun, M.; Fripertinger, H.; Kerber, A.; Kohnert, A.; Wassermann, A. *Error-Correcting Linear Codes: Classification by Isometry and Applications*; Springer: Berlin/Heidelberg, Germany, 2006.
3. Canteaut, A.; Chabaud, F. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Trans. Inform. Theory* **1998**, *44*, 367–378. [CrossRef]
4. Grassl, M. Searching for linear codes with large minimum distance. In *Discovering Mathematics with Magma. Algorithms and Computation in Mathematics*; Bosma, W., Cannon, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 19, pp. 287–313.
5. Lisoněk, P.; Trummer, L. Algorithms for the Minimum Weight Of Linear Codes. *Adv. Math. Commun.* **2016**, *10*, 195–207. [CrossRef]
6. Bosma, W.; Cannon, J.; Playoust, C. The Magma algebra system I: The user language. *J. Symb. Comput.* **1997**, *24*, 235–265. [CrossRef]
7. Rather, B.A.; Pirzada, S.; Naikoo, T.A.; Shang, Y. On Laplacian Eigenvalues of the Zero-Divisor Graph Associated to the Ring of Integers Modulo n. *Mathematics* **2021**, *9*, 482. [CrossRef]
8. Bouyukliev, I.; Bakoev, V. A method for efficiently computing the number of codewords of fixed weights in linear codes. *Discret. Appl. Math.* **2008**, *156*, 2986–3004. [CrossRef]
9. Gulliver, T.; Bhargava, V.; Stein, J. Q-ary gray codes and weight distribution. *Appl. Math. Comput.* **1999**, *103*, 97–109.

10. Bouyukliev, I. QEXTNEWEDITION-GENERATION Module. Available online: http://www.moi.math.bas.bg/moiuser/~data/Software/QextNewEdition.html (accessed on 26 August 2021).
11. Hernando, F.; Igual, F.D.; Quintana-Ortí, G. Algorithm 994: Fast Implementations of the Brouwer-Zimmermann Algorithm for the Computation of the Minimum Distance of a Random Linear Code. *ACM Trans. Math. Softw.* **2019**, *45*, 28. [CrossRef]