Examples and Counterexamples 2 (2022) 100051

Contents lists available at ScienceDirect

Examples and Counterexamples

journal homepage: www.elsevier.com/locate/exco

The maximum cardinality of trifferent codes with lengths 5 and 6

Stefano Della Fiore [a,*], Alessandro Gnutti [a], Sven Polak [b]

[a] Department of Information Engineering, University of Brescia, Via Branze 38, Brescia, Italy
[b] Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

ARTICLE INFO

MSC:
68R05
68Q17

Keywords:
Perfect k-hashing
Trifferent codes

ABSTRACT

A code $C \subseteq \{0,1,2\}^n$ is said to be trifferent with length n when for any three distinct elements of C there exists a coordinate in which they all differ. Defining $\mathcal{T}(n)$ as the maximum cardinality of trifferent codes with length n, $\mathcal{T}(n)$ is unknown for $n \geq 5$. In this note, we use an optimized search algorithm to show that $\mathcal{T}(5) = 10$ and $\mathcal{T}(6) = 13$.

1. Introduction

Let $k \geq 3$ and $n \geq 1$ be integers, and let C be a subset of $\{0,1,\dots,k-1\}^n$ with the property that for any k distinct elements there exists a coordinate in which they all differ. A subset C with this property is called perfect k-hash code with length n (perfect 3-hash codes are called trifferent codes). The problem of finding upper bounds for the maximum size of perfect k-hash codes is a fundamental problem in theoretical computer science. An elementary double counting argument, as shown in [1], gives the following bound on the cardinality of k-hash codes:

$$|C| \leq (k-1)\cdot\left(\frac{k}{k-1}\right)^n \textit{ for every } k \geq 3\,. \tag{1}$$

In 1984 Fredman and Komlós [2] improved the bound in (1) for every $k \geq 4$ and sufficiently large n, obtaining the following result:

$$|C| \leq \left(2^{k!/k^{k-1}}\right)^n. \tag{2}$$

Additional refinements of this bound have been progressively achieved over the years. See for example [3–5] for the case $k = 4$, [6] for the cases $k = 5$, 6, and [1,7–9] for $k \geq 5$. For the sake of completeness, we mention that some improvements on the asymptotic probabilistic lower bounds on the maximum size of perfect k-hash codes have been recently obtained in [10] for both small values of k and k sufficiently large.

In contrast, no recent progress has been made to improve the simple bound given in (1) for $k = 3$. This bound has not been outperformed by any algebraic technique, including the recent slice-rank method by Tao [11]. Indeed, Costa and Dalai showed in [12] that the slice-rank method cannot be applied in a *simple way* in order to improve the bound in (1). It is worth to mention that an improvement has been recently obtained in [13], however the authors restrict the codes to be linear, i.e., $C \subset \mathbb{F}_3^n$ and C is a subspace of $\mathbb{F}_3^n$.

As a consequence, particular attention is given to the case $k = 3$. Defining $\mathcal{T}(n)$ as the maximum cardinality of trifferent codes with length n, it is easy to verify that $\mathcal{T}(1) = 3$, $\mathcal{T}(2) = 4$ and $\mathcal{T}(3) = 6$. In addition, the authors in [1] showed that the so called *tetra-code* is a trifferent code with length 4 and cardinality 9: this result leads to $\mathcal{T}(4) = 9$. To the best of our knowledge, $\mathcal{T}(n)$ is currently unknown for $n \geq 5$. In this note, we show that $\mathcal{T}(5) = 10$ and $\mathcal{T}(6) = 13$ and we use these results to refine the current best known upper bound on the cardinality of trifferent codes with length $n \geq 5$ (Section 2). The exact value is achieved by implementing an optimized algorithm in GAP which exhibits the non-existence of trifferent codes with lengths 5 and 6 and cardinalities 11 and 14, respectively (the algorithm description is given in Section 3).

2. Improved upper bound on $\mathcal{T}(n)$ for $n \geq 5$

The simple recursion used to obtain the bound in (1) for $k = 3$ is:

$$\mathcal{T}(n) \leq \left\lfloor \frac{3}{2}\cdot\mathcal{T}(n-1)\right\rfloor, \tag{3}$$

for every $n \geq 2$, with $\mathcal{T}(1) = 3$. Since $\mathcal{T}(4) = 9$, then $10 \leq \mathcal{T}(5) \leq \left\lfloor \frac{3}{2}\cdot 9\right\rfloor = 13$. The upper bound is obtained using (3), while the lower bound comes easily from the fact that $\mathcal{T}(n) \geq \mathcal{T}(n-1)+1$, for every $n \geq 2$. Indeed, when a construction of a trifferent code with length $n-1$ is known, then it is always possible to trivially add an element of $\{0,1,2\}^n$ preserving the trifference property. In Example 2.1 we give a construction of a trifferent code with length 5 and cardinality 10 that is built using the tetra-code, see [1] for the definition. The 10 elements of $\{0,1,2\}^5$ are represented in columns. For $n = 6$, we have

* Corresponding author.
E-mail addresses: s.dellafiore001@unibs.it (S.D. Fiore), alessandro.gnutti@unibs.it (A. Gnutti), sven.polak@cwi.nl (S. Polak).

https://doi.org/10.1016/j.exco.2022.100051
Received 12 October 2021; Received in revised form 23 November 2021; Accepted 17 January 2022

that $13 \le \mathcal{T}(6) \le 19$. A trifferent code with length 6 and cardinality 13 is given in Example 2.2.

Example 2.1 ($\mathcal{T}(5) \ge 10$).

$$\left|\begin{array}{cccccccccc} 0&0&0&0&1&1&1&2&2&2\\ 0&0&1&2&0&1&2&0&1&2\\ 0&0&1&2&2&0&1&1&2&0\\ 0&0&1&2&1&2&0&2&0&1\\ 0&1&2&2&2&2&2&2&2&2 \end{array}\right|$$

Example 2.2 ($\mathcal{T}(6) \ge 13$).

$$\left|\begin{array}{ccccccccccccc} 0&0&2&2&2&2&2&0&1&1&1&1&1\\ 0&1&0&2&2&1&2&2&0&1&1&2&1\\ 0&1&1&0&2&2&1&2&2&0&1&1&2\\ 0&1&1&1&0&2&2&2&2&2&0&1&1\\ 0&1&2&1&1&0&2&2&1&2&2&0&1\\ 0&1&1&2&1&1&0&2&2&1&2&2&0 \end{array}\right|$$

We have designed an algorithm for searching trifferent codes with lengths 5 and 6 and cardinalities 11 and 14, respectively (see Section 3 for the description of the algorithm). The search ended without returning any trifferent codes, thus proving that $\mathcal{T}(5) \le 10$ and $\mathcal{T}(6) \le 13$. Hence, the following theorem holds:

Theorem 2.3. *$\mathcal{T}(5) = 10$ and $\mathcal{T}(6) = 13$.*

This result allows us to focus on the current best known bounds on the maximum cardinality of trifferent codes, which can be expressed as $\mathcal{T}(n) \le c \cdot (3/2)^n$, where c is a constant and n is sufficiently large. Since finding a better upper bound on the $\limsup_{n\to\infty} \sqrt[n]{\mathcal{T}(n)}$ is a very hard task, it becomes interesting to improve the constant c. The bound shown in (1) gives us $c = 2$, but a better constant can be obtained using (3) and the fact that $\mathcal{T}(4) = 9$, that is $c = 9/(3/2)^4 \approx 1.78$. We are able to improve this constant using Theorem 2.3 and (3). These statements directly imply:

Corollary 2.4.

$$\mathcal{T}(n) \le \frac{10}{(3/2)^5} \cdot \left(\frac{3}{2}\right)^n \approx 1.32 \cdot \left(\frac{3}{2}\right)^n \text{ for every } n \ge 5,$$

$$\mathcal{T}(n) \le \frac{13}{(3/2)^6} \cdot \left(\frac{3}{2}\right)^n \approx 1.15 \cdot \left(\frac{3}{2}\right)^n \text{ for every } n \ge 6.$$

Since the floor function is involved in the recursive formula (3), we can improve the constant c by iterating (3) m times starting from a fixed n_0 and a known upper bound on $\mathcal{T}(n_0)$. This results in the following theorem.

Theorem 2.5. *$\mathcal{T}(n) \le 1.09 \cdot \left(\frac{3}{2}\right)^n$ for every $n \ge 12$.*

Proof. Fix an integer $n_0 \ge 1$ and consider the following recursive formula that describes a sequence of achievable constants for $\mathcal{T}(n) \le l(m) \cdot (3/2)^n$ when $n \ge n_0 + m$:

$$l(m) = \left\lfloor l(m-1) \cdot \left(\frac{3}{2}\right)^{n_0+m} \right\rfloor \cdot \left(\frac{3}{2}\right)^{-n_0-m} \text{ for } m \ge 1, \tag{4}$$

where $l(0) = \mathcal{T}(n_0) \cdot (3/2)^{-n_0}$. Taking $n_0 = 6$ and $m = 6$, we obtain the thesis. □

Since the sequence $l(m)$ is non-increasing, we are interested in the $\lim_{m\to\infty} l(m)$. Computing that limit is not trivial, so we use the following recursive relation to obtain a lower bound:

$$d(m) = d(m-1) - \frac{1}{2} \cdot \left(\frac{3}{2}\right)^{-n_0-m} \text{ for } m \ge 1,$$

where $d(0) = \mathcal{T}(n_0) \cdot (3/2)^{-n_0}$. It is easy to see that $l(m) \ge d(m)$ for every $m \ge 0$. Then we have:

$$\lim_{m\to\infty} d(m) = \lim_{m\to\infty} \left(\mathcal{T}(n_0) - \frac{1}{2} \cdot \sum_{i=1}^{m} \left(\frac{2}{3}\right)^i\right) \cdot \left(\frac{3}{2}\right)^{-n_0} = \left(\mathcal{T}(n_0) - 1\right) \cdot \left(\frac{3}{2}\right)^{-n_0}. \tag{5}$$

Remark 2.1. Since $\mathcal{T}(4) = 9$, if we fix $n_0 = 4$ then we can substitute them into (5) to get that $\lim_{m\to\infty} l(m) \ge \lim_{m\to\infty} d(m) = 8 \cdot (3/2)^{-4} \approx 1.59$. This lower bound is, in any case, greater than the constant that we have found in Theorem 2.5.

3. Proof of Theorem 2.3—The algorithm

Computing a brute-force search for finding a trifferent code with length n and cardinality M would require to test $\binom{3^n}{M}$ subsets, and for each of them compare $\binom{M}{3}$ triplets: overall, for $(n, M) = (5, 11)$ one would test $\approx 10^{20}$ triplets while for $(n, M) = (6, 14)$ one would test approximately 10^{30} triplets. These numbers are prohibitively large.

Our algorithm dramatically reduces the number of operations, without missing any potential trifferent code. First, we list the elements of $C_n = \{0, 1, 2\}^n$ in lexicographic order and fix $(i_1, i_2, \dots, i_M)$ as the indices representing the M elements to test, requiring that $i_1 < i_2 < \dots < i_M$. Then, let C_n^m be the code containing the elements associated to the first m indices. Starting from $m = 3$, we check if C_n^m is trifferent: based on the output, the variable m and the indices are updated accordingly to the pseudocode reported in Algorithm 1.

Algorithm 1 Check if $\mathcal{T}(n) \ge M$.

Require: $(c(1), \dots, c(3^n)) = \{0, 1, 2\}^n$ ordered lexicographically, $(i_1, \dots, i_M) \leftarrow (1, \dots, M)$, $m \leftarrow 3$
repeat
  if $\{c(i_1), \dots, c(i_m)\}$ **is trifferent or** $m < 3$ **then**
    if $m = M$ **then return** True **end if**
    $m \leftarrow m + 1$
  else
    $m' \leftarrow \min\{m'' : i_{m''} \ge 3^n - M + m''\}$
    if m' **exists then**
      $m \leftarrow m' - 1$, $i_m \leftarrow i_m + 1$
      $i_{t+1} \leftarrow i_t + 1$, *for every* $m \le t \le M - 1$
    else
      $i_t \leftarrow i_t + 1$, *for every* $m \le t \le M$
    end if
  end if
until $i_1 \ge 2$
return False

At each update, C_n^m is tested: however, only the triplets containing the i_m-th element have to be examined, since all the other triplets have been already verified by construction. This is the first key point of our algorithm.

In addition, we are able to force some restrictions on the set of the indices. Two codes $C, D \subseteq \mathbb{F}_3^n$ are called *equivalent* if D be obtained from C by subsequently applying permutations to the coordinate positions and to the symbols $\{0, 1, 2\}$ in each coordinate. Given a trifferent code, by symmetry we can find an equivalent code containing the zero vector and a vector of the form $(0, \dots, 0, 1, \dots, 1)$, and not containing nonzero words lexicographically smaller than this vector. As a consequence, our algorithm stops the search of trifferent codes immediately when $i_1 = 2$, and limits the set of values that the second index can assume, namely, $i_2 \in \{\frac{3^i+1}{2} : i = 1, \dots, n\}$. Furthermore, suppose there exists a trifferent code C with length n and cardinality M. Let s_0, s_1, s_2 be the number of elements in C with symbols 0, 1 and 2, respectively, at the first coordinate. It is easy to see that $s_i + s_j \leq \mathcal{T}(n-1)$ for $i \neq j$, so $s_0, s_1, s_2 \geq M - \mathcal{T}(n-1)$. It means that we should have for each symbol 0, 1 and 2 at least $M - \mathcal{T}(n-1)$ elements in C with that symbol in the first coordinate. As a consequence, recalling that we list the elements of C_n in lexicographic order, we can force:

- $i_{M-\mathcal{T}(n-1)} \leq 3^{n-1}$ (first coordinate equal to 0);
- $i_{2(M-\mathcal{T}(n-1))} \leq 2 \cdot 3^{n-1}$ (first coordinate equal to 1);
- $i_{2\mathcal{T}(n-1)-M+1} > 3^{n-1}$ (first coordinate equal to 1);
- $i_{\mathcal{T}(n-1)+1} > 2 \cdot 3^{n-1}$ (first coordinate equal to 2).

For the sake of readability, the pseudocode reported in Algorithm 1 does not include the restrictions on the set of the indices. However, the code associated to the final version of the algorithm is publicly available and can be found at [14].

We have executed our program for $(n, M) = (5, 11)$ and $(n, M) = (6, 14)$, and no trifferent code has been found. The total number of tested triplets is $\approx 10^7$ for $(n, M) = (5, 11)$ and $\approx 10^{11}$ for $(n, M) = (6, 14)$, thus saving a factor of $\approx 10^{13}$ and $\approx 10^{19}$, respectively, compared to the full brute-force strategy.

As a side note: inspired by a semidefinite programming upper bound for cap sets [15], we could alternatively obtain the upper bound $\mathcal{T}(5) \leq 10$ using the method from [16], in which all extra constraints from Eq. (3) of [16] were included to obtain the bound.

Remark 3.1. For $(n, M) = (6, 13)$, the search returned a set S of 1046 trifferent codes up to symmetry choices explained above. For any code in S, we generated all equivalent codes and deleted the ones contained in S from S. We had to repeat this 3 times until the set was empty. So there are 3 distinct trifferent $(n, M) = (6, 13)$-codes up to equivalence. These are:

$$\begin{vmatrix} 0&0&2&2&2&2&2&0&1&1&1&1&1 \\ 0&1&0&2&2&2&1&2&0&1&1&1&2 \\ 0&1&1&0&2&1&2&2&2&0&1&2&1 \\ 0&1&1&1&0&2&2&2&2&2&0&1&1 \\ 0&1&1&2&1&0&2&2&2&1&2&0&1 \\ 0&1&2&1&1&1&0&2&1&2&2&2&0 \end{vmatrix},$$

$$\begin{vmatrix} 0&0&2&2&2&2&2&0&1&1&1&1&1 \\ 0&1&0&2&2&1&2&2&0&1&1&2&1 \\ 0&1&1&0&2&2&1&2&2&0&1&1&2 \\ 0&1&1&1&0&2&2&2&2&2&0&1&1 \\ 0&1&2&1&1&0&2&2&1&2&2&0&1 \\ 0&1&1&2&1&1&0&2&2&1&2&2&0 \end{vmatrix},$$

$$\begin{vmatrix} 0&0&2&2&2&2&2&0&1&1&1&1&1 \\ 0&1&0&2&2&1&2&2&0&1&1&2&1 \\ 0&1&1&0&2&2&2&2&2&0&1&1&1 \\ 0&1&1&1&0&2&1&2&2&2&0&1&2 \\ 0&1&2&1&1&0&2&2&1&2&2&0&1 \\ 0&1&1&1&2&1&0&2&2&2&1&2&0 \end{vmatrix}.$$

For each of these codes, in each coordinate position two symbols occur 5 times and one symbol occurs 3 times.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Kőrner, K. Marton, New bounds for perfect hashing via information theory, European J. Combin. 9 (1988) 523–530.
[2] L. Michael, Fredman and jános komlós, on the size of separating systems and families of perfect hash functions, SIAM J. Algebraic Discret. Methods 5 (1984) 61–68.
[3] E. Arikan, An upper bound on the zero-error list-coding capacity, IEEE Trans. Inform. Theory 40 (1994) 1237–1240.
[4] E. Arikan, An improved graph-entropy bound for perfect hashing, in: IEEE International Symposium on Information Theory, 1994.
[5] M. Dalai, V. Guruswami, J. Radhakrishnan, An improved bound on the zero-error listdecoding capacity of the 4/3 channel, IEEE Trans. Inform. Theory 66 (2020) 749–756.
[6] S. Costa, M. Dalai, New bounds for perfect k-hashing, Discrete Appl. Math. 289 (2021) 374–382.
[7] V. Guruswami, A. Riazanov, Beating fredman-komlos for perfect k-hashing, in: Leibniz International Proceedings in Informatics, 2019.
[8] S. Della Fiore, S. Costa, M. Dalai, Improved Bounds for (b, k)-hashing, arXiv: 2012.00620.
[9] S. Della Fiore, S. Costa, M. Dalai, New upper bounds for (b, k)-hashing, in: IEEE International Symposium on Information Theory, 2021.
[10] C. Xing, C. Yuan, Towards beating the probabilistic lower bound on q-perfect hashing for all q, arXiv: https://arxiv.org/abs/1908.08792.
[11] T. Tao, A symmetric formulation of the Croot–Lev-Pach-Ellenberg-Gijswijt capset bound, 2016, Blog post.
[12] S. Costa, M. Dalai, A gap in the slice-rank of k-tensors, J. Comb. Theory Ser. A 177 (2021).
[13] C. Pohoata, D. Zakharov, On the trifference problem for linear codes, arXiv: 2105.00122.
[14] S. Della Fiore, A. Gnutti, S. Polak, GAP Code, available at: http://stefano-dellafiore.me/trifference_gap.g.
[15] D.C. Gijswijt, Matrix Algebras and Semidefinite Programming Techniques for Codes (PhD-thesis), University of Amsterdam, 2005.
[16] B.M. Litjens, A. Schrijver, S.C. Polak, Semidefinite bounds for nonbinary codes based on quadruples, Des. Codes Cryptogr. 84 (2017) 87–100.