Discrete Applied Mathematics 360 (2025) 181–187

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Note

An efficient algorithm for group testing with runlength constraints

Marco Dalai [a], Stefano Della Fiore [b,*], Adele A. Rescigno [b], Ugo Vaccaro [b]

[a] DII, Università degli Studi di Brescia, Via Branze 38, I-25123 Brescia, Italy
[b] DI, Università degli Studi di Salerno, Via Giovanni Paolo II 132, 84084 Fisciano, Italy

ARTICLE INFO

Article history:
Received 10 July 2023
Received in revised form 28 June 2024
Accepted 2 September 2024
Available online 13 September 2024

Keywords:
Lovász Local Lemma
Group Testing
Superimposed codes
Runlength-constrained codes

ABSTRACT

In this paper, we provide an efficient algorithm to construct almost optimal (k, n, d)-superimposed codes with runlength constraints. A (k, n, d)-superimposed code of length t is a $t \times n$ binary matrix such that any two 1's in each column are separated by a run of at least d 0's, and such that for any column $\mathbf{c}$ and any other $k-1$ columns, there exists a row where $\mathbf{c}$ has 1 and all the remaining $k-1$ columns have 0. These combinatorial structures were introduced by Agarwal et al. (2020), in the context of Non-Adaptive Group Testing algorithms with runlength constraints.

By using Moser and Tardos' constructive version of the Lovász Local Lemma, we provide an efficient randomized Las Vegas algorithm of complexity $\Theta(tn^2)$ for the construction of (k, n, d)-superimposed codes of length $t = O(dk \log n + k^2 \log n)$. We also show that the length of our codes is shorter, for n sufficiently large, than that of the codes whose existence was proved in Agarwal et al. (2020).

1. Introduction

In this paper, we devise efficient construction algorithms for (k, n, d)-*superimposed* codes recently introduced by Agarwal et al. in [1] and defined as follows:

Definition 1.1 ([1]). Let k, n, d be positive integers, $k \leq n$. A (k, n, d)-*superimposed* code is a $t \times n$ binary matrix M such that

(1) any two 1's in each column of M are separated by a run of at least d 0's,
(2) for any k-tuple of the columns of M and for any column $\mathbf{c}$ of the *given* k-tuple, it holds that there exists a row $i \in \{1, \ldots, t\}$ such that $\mathbf{c}$ has symbol 1 in row i and all the remaining $k-1$ columns of the k-tuple have symbol 0 in row i.

The number of rows t of M is called the length of the (k, n, d)-superimposed code.

(k, n, d)-superimposed codes were introduced within the context of Non-Adaptive Group Testing algorithms for topological DNA-based data storage, and represent one of the main instruments to derive the strong results obtained therein [1] (see also [17]). The parameter of (k, n, d)-superimposed codes that one wants to optimize (i.e., minimize)

* Corresponding author.
E-mail addresses: marco.dalai@unibs.it (M. Dalai), sdellafiore@unisa.it (S. Della Fiore), arescigno@unisa.it (A.A. Rescigno), uvaccaro@unisa.it (U. Vaccaro).

https://doi.org/10.1016/j.dam.2024.09.001
0166-218X/

is the length t of the code. Indeed, this is the parameter that mostly affects the DNA-based data storage algorithms considered in [1]. Using the probabilistic method the authors of [1] proved that (k, n, d)-superimposed codes of length $t = O(dk \log n + k^2 \log n)$ exists and they provided a randomized Montecarlo algorithm, in the sense that it gives, with high probability, a (k, n, d)-superimposed code whose length is upper bounded by this quantity. They also proved that any (k, n, d)-superimposed code must have length

$$t \geq \min\left(n, \Omega\left(\frac{dk}{\log(dk)} \log n + \frac{k^2}{\log k} \log n\right)\right). \tag{1}$$

A preliminary study of the questions treated in the present paper was done in [5] where we improved some of the existential bounds of [1] by showing the existence of (k, n, d)-superimposed codes having shorter lengths than the codes of [1]. In [1,5], the authors left open the problem of devising an efficient polynomial time algorithm to construct (k, n, d)-superimposed codes of length $t = O(dk \log n + k^2 \log n)$. More precisely, the results of [1,5] only imply the existence of $\Theta(n^k)$-time algorithms for constructing (k, n, d)-superimposed codes of length $t = O(dk \log n + k^2 \log n)$. We note that such algorithms achieve a time-complexity of $\Theta(n^k)$ since in order to see if a randomized constructed matrix is a (k, n, d)-superimposed code they need to check conditions that involved k-tuple of columns. It is clear that already for moderate values of k, those algorithms are impractical. In view of the relevance of the application scenario considered in [1], it is quite important to have an efficient algorithm for constructing (k, n, d)-superimposed codes of length $t = O(dk \log n + k^2 \log n)$. The purpose of this paper is to provide a randomized Las Vegas $\Theta(k(k + d)n^2 \ln n)$-time algorithm to construct such codes that is polynomial both in n and k. We remark that our algorithm produces almost optimal codes (in the asymptotic sense) because of the lower bound (1).

In the same spirit of this work, in [3,18], the authors provided using probabilistic methods, such as the Lovász Local Lemma, new bounds on the length of $(k + 1, n, 0)$-superimposed codes (also known as k-disjunct matrices) with fixed-weight columns. Their approaches can be adapted to derive $\Theta(n^k)$-time algorithms and bounds on the length of (k, n, d)-superimposed codes. This further motivates our work in studying algorithms to construct (k, n, d)-superimposed codes that are polynomial both in n and k.

Before going into the technical details, we would like to recall that $(k, n, 0)$-superimposed codes correspond to the classical superimposed codes (a.k.a. cover free families) introduced in [9,13], and extensively studied since then. We refer to the excellent survey papers [8,11] for a broad discussion of the relevant literature, and to the monographs [6,12] for an account of the applications of superimposed codes to group testing, multi-access communication, data security, data compression, and several other different areas. It is likely that also (k, n, d)-superimposed codes will find applications outside the original scenario considered in [1].

2. Preliminaries

Throughout the paper, the logarithms without subscripts are in base two, and we denote with $\ln(\cdot)$ the natural logarithm. We denote by $[a, b]$ the set $\{a, a + 1, \ldots, b\}$. Given integers w and d, a binary (w, d)-vector $\mathbf{x}$ is a vector of Hamming weight w (that is, the number of 1's in $\mathbf{x}$ is equal to w), such that any two 1's in $\mathbf{x}$ are separated by a run of at least d 0's.

We recall, for positive integers $c \leq b \leq a$, the following well-known properties of binomial coefficients:

$$\left(\frac{a}{b}\right)^b \leq \binom{a}{b} \leq \frac{a^b}{b!} \leq \left(\frac{ea}{b}\right)^b, \tag{2}$$

$$\binom{a}{b}\binom{b}{c} = \binom{a}{c}\binom{a-c}{b-c}. \tag{3}$$

We shall also need the following technical lemma from [10].

Lemma 2.1 (*[10]*). *Let a, b, c be positive integers such that $c \leq a \leq b$. We have that*

$$\frac{\binom{a}{c}}{\binom{b}{c}} \leq \left(\frac{a - \frac{c-1}{2}}{b - \frac{c-1}{2}}\right)^c. \tag{4}$$

Finally, we recall here the celebrated algorithmic version of the Lovász Local Lemma for the symmetric case, due to Moser and Tardos [15]. It represents one of the main tools to derive the results of this paper. We first recall the setting for the Lovász Local Lemma in the random-variable scenario. The relevant probability space Ω is defined by n mutually independent random variables $X_1, \ldots, X_n$, taking values in a finite set $\mathcal{X}$. One is interested in a set of events $\mathcal{E}$ in the probability space Ω, (generally called "bad events", that is, events one wants to avoid), where each event $E_i \in \mathcal{E}$ only depends on $\{X_j : j \in S_i\}$ for some subset $S_i \subseteq [1, n]$, for $i = 1, \ldots, |\mathcal{E}|$. Note that two events E_i, E_j are independent if $S_i \cap S_j = \emptyset$. A *configuration* in the context of the Lovász Local Lemma is a specific assignment of values to the set of random variables involved in defining the events. Sampling a random variable X_i means generating a value $x \in \mathcal{X}$ in such a way that the probability of generating x is in accordance with the probability distribution of the random variable X_i.

As said before, in the applications of the Lovász Local Lemma the events E_i's are bad-events that one wants to avoid, that is, one seeks a configuration such that all the events E_i's do not hold. In the seminal paper [15] Moser and Tardos introduced a simple randomized algorithm that produces such a configuration, under the same hypothesis of the classical Lovász Local Lemma. The algorithm is the following:

Algorithm 1: The MT algorithm

```
Sample the random variables X_1, ..., X_n from their distributions in Ω
while some event is true on X_1, ..., X_n do
    Arbitrarily select some true event E_i
    For each j ∈ S_i, sample X_j from its distribution in Ω
```

Moser and Tardos [15] proved the following important result (see also [14], p. 266).

Lemma 2.2 ([15]). *Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space and let $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ be a set of m events where each E_i is determined by a subset S_i of the random variables and, for each i, $S_j \cap S_i \neq \emptyset$ for at most D values of $j \neq i$. Suppose that $\Pr(E_i) \leq P$ for all $1 \leq i \leq m$. If $ePD \leq 1$, then $\Pr(\cap_{i=1}^{m} \overline{E_i}) > 0$. Moreover, **Algorithm 1** finds a configuration avoiding all events E_i by using an average number of resampling of at most m/D.*

3. New algorithms for (k, n, d)-superimposed codes

We aim to efficiently construct (k, n, d)-superimposed codes with a small length. The difficulty faced in [1,5] was essentially due to the fact that the constraints a $t \times n$ binary matrix has to satisfy in order to be a (k, n, d)-superimposed code involve all the $\binom{n}{k}$ k-tuples of columns of M. Checking whether those conditions are satisfied or not requires time $\Theta(n^k)$. To overcome this difficulty, we use an idea of [9,13]. That is, we first introduce an auxiliary class of binary matrices where the constraints involve *only* pairs of columns. Successively, we show that for suitably chosen parameters such a class of matrices give rise to (k, n, d)-superimposed codes with small length t. This implies that we need to check the validity of the constraints only for the $\Theta(n^2)$ pairs of columns. This observation and Lemma 2.2, will allow us to provide an efficient algorithm to construct (k, n, d)-superimposed codes.

It is convenient to first consider the following class of (k, n, d)-superimposed codes.

Definition 3.1. A (k, n, d, w)-*superimposed* code is a (k, n, d)-superimposed code with the additional constraint that each column has Hamming weight w, that is, each column of the code is a binary (w, d)-vector.

We now introduce the auxiliary class of matrices mentioned above.

Definition 3.2. Let n, d, w, λ be positive integers. A $t \times n$ binary matrix M is a (n, d, w, λ)-matrix if the following properties hold true:

(1) each column of M is a binary (w, d)-vector;
(2) any pair of columns $\mathbf{c}, \mathbf{d}$ of M have at most λ 1's in common, that is, there are at most λ rows among the t's where columns $\mathbf{c}$ and $\mathbf{d}$ both have symbol 1.

(n, d, w, λ)-matrices are related to (k, n, d, w)-superimposed codes by way of the following easy result.

Lemma 3.3. *A binary (n, d, w, λ)-matrix M of dimension $t \times n$, with parameter $\lambda = \lfloor (w-1)/(k-1) \rfloor$, is a (k, n, d, w)-superimposed code of length t.*

Proof. The bitwise OR of any set C of $k-1$ columns of M can have at most $(k-1)\lambda = (k-1)\lfloor (w-1)/(k-1) \rfloor < w$ symbols equal to 1 in the same w rows where an arbitrary column $\mathbf{c} \notin C$ has a 1. □

We now show how to efficiently construct binary (n, d, w, λ)-matrices with a small number of rows. This fact, by virtue of Lemma 3.3, will give us an upper bound on the minimum length of (k, n, d, w)-superimposed codes.

We need the following enumerative lemma from [5]. We include here the short proof to keep the paper self-contained. Since (w, d)-vectors of length t have necessarily $t \geq (w-1)d + w$, we use this inequality throughout the paper.

Lemma 3.4. *Let $V \subseteq \{0, 1\}^t$ be the set of all binary (w, d)-vectors of length t. Then*

$$|V| = \binom{t-(w-1)d}{w}.$$

Proof. Let A be the set of all distinct binary vectors of length $t-(w-1)d$ and weight w. One can see that $|V| = |A|$ since each vector of V can be obtained from an element $a \in A$ by inserting a run of exactly d 0's between each pair of 1's in a. Conversely, each element of A can be obtained from an element $s \in V$ by removing exactly d consecutive 0's in between each pair of consecutive 1's in s. □

We are ready to state one of the main results of this paper.

Theorem 3.5. *There exists a $t \times n$ (n, d, w, λ)-matrix with*

$$t = \left\lceil (w-1)d + \frac{\lambda}{2} + \frac{ew}{\lambda+1}\left(w - \frac{\lambda}{2}\right)(e(2n-4))^{\frac{1}{\lambda+1}} \right\rceil . \tag{5}$$

Proof. Let M be a $t \times n$ binary matrix, $t \geq (w-1)d + w$, where each column $\boldsymbol{c}$ is sampled uniformly at random among the set of all distinct binary (w, d)-vectors of length t. Since we are assuming that $t \geq (w-1)d + w$, by Lemma 3.4 we have that

$$\Pr(\mathbf{c}) = \binom{t-(w-1)d}{w}^{-1}.$$

Let $i, j \in [1, n]$, $i \neq j$ and let us consider the event $\overline{E}_{i,j}$ that there exists *at most* λ rows such that both the ith column and the jth column of M have the symbol 1 in *each* of these rows. We evaluate the probability of the complementary "*bad*" event $E_{i,j}$. Hence $E_{i,j}$ is the event that the random ith and jth columns $\mathbf{c}_i$ and $\mathbf{c}_j$ have 1 in at least $\lambda+1$ coordinates. We bound $\Pr(E_{i,j})$ by conditioning on the event that $\mathbf{c}_i$ is equal to a c, where c is a binary (w, d)-vector.

For a subset $S \subset [1, t]$ of coordinates, let $E^S_{i,j}$ be the event that in each coordinate of S the ith and jth column have the symbol 1. For a fixed column $\mathbf{c}_i = c$, let A be the set of coordinates where c has 1's. Note that for $S \in \binom{A}{\lambda+1}$, i.e., for a subset S of A of size $\lambda+1$, it holds that

$$\Pr(E^S_{i,j} | \mathbf{c}_i = c) \leq \frac{\binom{t-(w-1)d-(\lambda+1)}{w-(\lambda+1)}}{\binom{t-(w-1)d}{w}}. \tag{6}$$

We justify (6). Since by assumption c already contains a 1 in each coordinate of $S \subset A$, given that $\mathbf{c}_i = c$ we have the event $E^S_{i,j}$ conditionally reduces to the event that $\mathbf{c}_j$ also has 1's in all the $\lambda+1$ coordinates in S. Therefore, we only need to upper bound the number of (w, d)-vectors of length t with 1's in the $\lambda+1$ coordinates of S. Note that each such t-long vector, upon removing exactly d 0's in between each pair of consecutive 1's and the coordinates in S, reduces to a distinct binary vector of length $t-(w-1)d-(\lambda+1)$ and weight $w-(\lambda+1)$. It follows that the number of choices for $\mathbf{c}_j$ (such that it has 1's in all the $\lambda+1$ coordinates in S) is upper bounded by $\binom{t-(w-1)d-(\lambda+1)}{w-(\lambda+1)}$. Then, formula (6) holds.

Therefore, it holds that

$$\begin{aligned}\Pr(E_{i,j} | \mathbf{c}_i = c) &\leq \sum_{S \in \binom{A}{\lambda+1}} \Pr(E^S_{i,j} | \mathbf{c}_i = c) \\ &= \binom{w}{\lambda+1} \frac{\binom{t-(w-1)d-(\lambda+1)}{w-(\lambda+1)}}{\binom{t-(w-1)d}{w}}.\end{aligned} \tag{7}$$

Since the right-hand side of (7) does not depend on the fixed column c, it also holds unconditionally. Hence

$$\Pr(E_{i,j}) \leq \binom{w}{\lambda+1} \frac{\binom{t-(w-1)d-(\lambda+1)}{w-(\lambda+1)}}{\binom{t-(w-1)d}{w}}. \tag{8}$$

Hence, by (8) we have

$$\begin{aligned}\Pr(E_{i,j}) &\leq \binom{w}{\lambda+1}\binom{t-(w-1)d-(\lambda+1)}{w-(\lambda+1)} \Big/ \binom{t-(w-1)d}{w} \\ &\overset{(i)}{=} \binom{w}{\lambda+1}\binom{w}{\lambda+1} \Big/ \binom{t-(w-1)d}{\lambda+1} \\ &\overset{(ii)}{\leq} \binom{w}{\lambda+1}\left(\frac{w-\frac{\lambda}{2}}{t-(w-1)d-\frac{\lambda}{2}}\right)^{\lambda+1} \\ &\overset{(iii)}{\leq} \left(\frac{ew}{\lambda+1}\right)^{\lambda+1}\left(\frac{w-\frac{\lambda}{2}}{t-(w-1)d-\frac{\lambda}{2}}\right)^{\lambda+1} = P,\end{aligned} \tag{9}$$

where (i) holds due to equality (3) (since $t \geq (w-1)d + w$), (ii) is true due to Lemma 2.1, and finally (iii) holds thanks to inequalities (2).

The number of events $E_{i,j}$ is equal to $n(n-1)/2$. Let us fix an event $E_{i,j}$. Then the number of events $E_{i',j'}$ with $\{i,j\}\cap\{i',j'\}\neq\emptyset$ and $\{i,j\}\neq\{i',j'\}$ is equal to $D=2n-4$. Hence, according to Lemma 2.2, if we take $\mathcal{P}=\{\mathbf{c}_1,\mathbf{c}_2,\dots,\mathbf{c}_n\}$ to be the set of n mutually independent random variables that represent the columns of the matrix M, $\mathcal{E}=\{E_{i,j}\}$ to be the set of events defined earlier that are associated to these random variables (where each $E_{i,j}$ is only determined by $\mathbf{c}_i$ and $\mathbf{c}_j$), P (as defined in (9)) and $D=2n-4$ that satisfies $ePD\leq 1$, then the probability that *none* of the "bad" events $E_{i,j}$ occurs is strictly positive. By solving the following inequality for t

$$ePD = e(2n-4)\left(\frac{ew}{\lambda+1}\right)^{\lambda+1}\left(\frac{w-\frac{\lambda}{2}}{t-(w-1)d-\frac{\lambda}{2}}\right)^{\lambda+1} < 1,$$

one can see that by setting t as in (5) we are indeed satisfying this inequality. We also note that the initial condition $t\geq(w-1)d+w$ is satisfied for this value of t, as given in (5).

Hence, from Lemma 2.2 one can construct a binary (n,d,w,λ)-matrix M whose number of rows t satisfies equality (5). □

Now, thanks to Lemma 3.3 and Theorem 3.5, we can prove the following result.

Theorem 3.6. *There exists a randomized algorithm to construct a (k,n,d,w)-superimposed code with length*

$$t \leq 1+(w-1)d+\frac{w-1}{2(k-1)}+\frac{ew(k-1)}{w-1}\left(w-\frac{w-1}{2(k-1)}+\frac{1}{2}\right)(e(2n-4))^{\frac{k-1}{w-1}}. \tag{10}$$

The algorithm requires, on average, time $O(tn^2)$ to construct the code.

Proof. The upper bound (10) on t is derived by substituting the value of $\lambda=\lfloor(w-1)/(k-1)\rfloor$ from Lemma 3.3 into Eq. (5) of Theorem 3.5, and by using the inequalities $\frac{w-1}{k-1}-1\leq\left\lfloor\frac{w-1}{k-1}\right\rfloor\leq\frac{w-1}{k-1}$. The time complexity $O(tn^2)$ comes from Lemma 2.2 by first noticing that $m/D=n(n-1)/(4n-8)\leq n/3$, for $n\geq 5$. Moreover, **Algorithm** 1 requires to randomly generate a matrix, checking if the $\Theta(n^2)$ events $\overline{E}_{i,j}$ are satisfied, and resampling *only on non-satisfied* events. The generation of each matrix-column can be done by first generating an integer in the interval $[0,\binom{t-(w-1)d}{w}-1]$ uniformly at random, and encoding it with a different binary vector of length $t-(w-1)d$ containing w 1's. This one-to-one encoding can be performed in time $O(t)$ for each column, by using the enumeration encoding technique by Cover [4]. Successively, one inserts a run of exactly d 0's between each pair of 1's so that each matrix-column is a (w,d)-vector. All together, this requires $O(t)$ operations per column.

In order to check whether an arbitrary event $\overline{E}_{i,j}$ is satisfied, we need to check whether the ith column and the jth column of the matrix have at most $\left\lfloor\frac{w-1}{k-1}\right\rfloor$ 1's in common; this can be done with at most $O(t)$ operations. Successively, we resample only over non-satisfied events. Then, we need to check only the events that involve columns that have been resampled. Altogether, by Lemma 2.2 this procedure requires $O(tn^2+n\cdot m/D\cdot t)=O(tn^2)$ elementary operations. □

Now, we optimize the parameter w in Eq. (10) to obtain a randomized algorithm for (weight-unconstrained) (k,n,d)-superimposed codes.

Theorem 3.7. *There exists a randomized algorithm to construct a (k,n,d)-superimposed code with length*

$$t \leq d(k-1)\ln(2en)+\frac{\ln(n)}{2}+e^2(k-1)^2\ln(2en)+\frac{7e^2(k-1)}{2}+d+O(1).$$

The algorithm requires, on average, time $O(tn^2)$ to construct the code.

Proof. Let $w=\lceil 1+(k-1)\ln(2en)\rceil$. The algorithm described in Theorem 3.6 constructs a (k,n,d,w)-superimposed code which is, clearly, a (k,n,d)-superimposed code. Using the inequalities

$$1+(k-1)\ln(2en)\leq\lceil 1+(k-1)\ln(2en)\rceil\leq 2+(k-1)\ln(2en)$$

we get

$$\ln(2en)\leq\frac{w-1}{k-1}\leq\frac{1+(k-1)\ln(2en)}{k-1}.$$

Therefore, by (10) of Theorem 3.6 we have that

$$\begin{aligned} t &\leq 1+((k-1)\ln(2en)+1)d+\frac{1+(k-1)\ln(2en)}{2(k-1)}+\frac{e}{\ln(2en)}\cdot\\ &\quad\cdot(2+(k-1)\ln(2en))\left(2+(k-1)\ln(2en)-\frac{\ln(2en)}{2}+\frac{1}{2}\right)(2en)^{\frac{1}{\ln(2en)}}\\ &\overset{(i)}{\leq} 1+d(k-1)\ln(2en)+d+\frac{1}{2(k-1)}+\frac{\ln(2en)}{2}+\frac{e}{\ln(2en)}\cdot \end{aligned}$$

$$\cdot (2 + (k-1)\ln(2en)) \left((k-1)\ln(2en) + \frac{3}{2} \right) (2en)^{\frac{1}{\ln(2en)}}$$

$$\overset{(ii)}{=} 1 + d(k-1)\ln(2en) + d + \frac{1}{2(k-1)} + \frac{\ln(2en)}{2} + \frac{e^2}{\ln(2en)} \cdot$$

$$\cdot \left(3 + \frac{7(k-1)\ln(2en)}{2} + (k-1)^2(\ln(2en))^2 \right)$$

$$= 1 + d(k-1)\ln(2en) + d + \frac{1}{2(k-1)} + \frac{\ln(2en)}{2} + \frac{3e^2}{\ln(2en)} +$$

$$+ e^2(k-1)^2\ln(2en) + \frac{7e^2(k-1)}{2}$$

$$\overset{(iii)}{\leq} d(k-1)\ln(2en) + \frac{\ln(n)}{2} + e^2(k-1)^2\ln(2en) + \frac{7e^2(k-1)}{2} + d + O(1),$$

where (i) holds due to the fact that $\ln(2en) \geq 2$ for $n \geq 2$, (ii) holds since $(2en)^{\frac{1}{\ln(2en)}} = e$, and (iii) is since $k \geq 2$ and $\ln(2en) \geq 2$. □

We notice that a widely believed conjecture of Erdős, Frankl and Füredi [9] says that for $k \geq \sqrt{n}$ one has that minimum-length $(k, n, 0)$-superimposed codes (i.e., classical superimposed codes) have length t equal to n. The current best-known result has been proved in [16] which shows that if $k \geq 1.157\sqrt{n}$ then the minimum length of $(k, n, 0)$-superimposed codes is equal to n. This last result clearly holds also for arbitrary (k, n, d)-superimposed codes. We also recall the following result obtained in [1].

Remark 3.8 ([1]). Every (k, n, d)-superimposed codes of length t must satisfy

$$t \geq \min\{n, 1 + (k-1)(d+1)\}.$$

This implies that if $k \geq \frac{n-1}{d+1} + 1$ then $t = n$, so we cannot construct a (k, n, d)-superimposed code of length t that is better than the identity matrix of size $n \times n$.

To properly appraise the value of Theorem 3.7, we recall the following result presented in [1] that provides a lower bound on the minimum length of any (k, n, d)-superimposed codes.

Theorem 3.9 ([1]). *Given positive integers k and n, with $2 \leq k \leq \min\{1.157\sqrt{n}, \frac{n-1}{d+1} + 1\}$, the length t of any (k, n, d)-superimposed code satisfies*

$$t \geq \Omega\left(\frac{kd}{\log(kd)} \log n + \frac{k^2}{\log k} \log n \right).$$

Therefore, one can see that the construction method provided by our Theorem 3.7, besides being quite efficient, produces codes of almost optimal length.

In [1], the authors provide the following upper bound on the length of (k, n, d, w)-superimposed codes.

Theorem 3.10 ([1]). *There exists a (k, n, d, w)-superimposed code of length t, provided that t satisfies the inequality*

$$n\binom{n-1}{k-1}\left(\frac{w(k-1)}{t - (2d+1)(w-1)} \right)^w < 1.$$

From Theorem 3.10 one can derive an explicit upper bound on the length of the codes whose existence was showed in [1] when $w = k\ln(n)$ by upper bounding $n\binom{n-1}{k-1}$ with $k\left(\frac{en}{k}\right)^k$. We report here the obtained result.

Theorem 3.11 ([1]). *There exists a randomized algorithm to construct a (k, n, d)-superimposed code with length*

$$t \leq 2dk\ln(n) + k\ln(n) + e^2k(k-1)\ln(n) - 2d + O(1). \tag{11}$$

It can be seen that for n sufficiently large, the upper bound on the length t of (k, n, d)-superimposed codes given in our Theorem 3.7 improves on the upper bound given in Theorem 3.10 of [1]. As observed in Section 1, the algorithm provided in [1] is a Montecarlo randomized algorithm that constructs a (k, n, d)-superimposed code whose length is upper bounded by (11). In order to transform the algorithm given in [1] into a Las Vegas randomized algorithm that *always* outputs a correct (k, n, d)-superimposed code, one can perform the following steps: (1) Generate a random matrix in accordance with the probabilities specified in Theorem 2 of [1], (2) check whether the matrix satisfies the properties of Definition 1.1 and, if not, repeat the experiment till one obtains a matrix with the desired property. However, it is known that the problem of checking whether a matrix satisfies superimposed-like properties is considered computationally infeasible (see, e.g., [2,7]) and no algorithm of complexity less that $\Theta(n^k)$ is known. On the other hand, our result provides a randomized algorithm of average time complexity $\Theta(k(k+d)n^2\ln n)$, that is, polynomial both in n and k, to construct (k, n, d)-superimposed codes of length not greater than that of [1].

Data availability

No data was used for the research described in the article.

Acknowledgment

The third and fourth authors were partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NGEU.

References

[1] A. Agarwal, O. Milenkovic, S. Pattabiraman, J. Ribeiro, Group testing with runlength constraints for topological molecular storage, in: Proceedings of the 2020 IEEE International Symposium on Information Theory, 2020, pp. 132–137.
[2] Y. Cheng, D.-Z. Du, K.-I. Ko, Guohui Lin, On the parameterized complexity of pooling design, J. Comput. Biol. 16 (11) (2009) 1529–1537.
[3] Y. Cheng, D.-Z. Du, G. Lin, On the upper bounds of the minimum number of rows of disjunct matrices, Optim. Lett. 3 (2009) 297–302.
[4] T. Cover, Enumerative source encoding, IEEE Trans. Inform. Theory 19 (1) (1973) 73–77.
[5] M. Dalai, S. Della Fiore, U. Vaccaro, Achievable rates and algorithms for group testing with runlength constraints, in: Proceedings of the 2022 IEEE Information Theory Workshop, 2022, pp. 576–581.
[6] D.-Z. Du, F.K. Hwang, Combinatorial Group Testing and Its Applications, World Scientific, 2000.
[7] D.-Z. Du, K.-I. Ko, Some completeness results on decision trees and group testing, SIAM J. Algebra. Discr. 8 (1987) 762–777.
[8] A. D'yachkov, V. Rykov, C. Deppe, V. Lebedev, Superimposed codes and threshold group testing, in: H. Aydinian, F. Cicalese, C. Deppe (Eds.), Information Theory, Combinatorics, and Search Theory, in: Lecture Notes in Computer Science, vol. 7777, Springer, Berlin, Heidelberg, 2013.
[9] P. Erdös, P. Frankl, Z. Füredi, Families of finite sets in which no set is covered by the union of r others, Israel J. Math. 51 (1985) 79–89.
[10] L. Gargano, A.A. Rescigno, U. Vaccaro, Low-weight superimposed codes and related combinatorial structures: Bounds and applications, Theoret. Comput. Sci. 806 (2020) 655–672.
[11] T.B. Idalino, L. Moura, A survey of cover-free families: Constructions, applications and generalizations, in: C. Colbourn, J. Dinitz (Eds.), Stinson66 - New Advances in Designs, Codes and Cryptography, in: Fields Institute Communications, Springer, 2023, pp. 195–239.
[12] O. Johnson, J. Scarlett, M. Aldridge, Group Testing: An Information Theory Perspective, Now Publishers, 2019.
[13] W. Kautz, R. Singleton, Nonrandom binary superimposed codes, IEEE Trans. Inform. Theory 10 (1964) 363–377.
[14] D.E. Knuth, The Art of Computer Programming: Combinatorial Algorithms, Part 2, vol. 4B, Addison-Wesley Professional, 2022.
[15] R.A. Moser, G. Tardos, A constructive proof of the general Lovász local lemma, J. ACM 57 (2010) 1–15.
[16] C. Shangguan, G. Ge, New bounds on the number of tests for disjunct matrices, IEEE Trans. Inform. Theory 61 (12) (2016) 7518–7521.
[17] S.K. Tabatabaei, B. Wang, N.B.M. Athreya, B. Enghiad, A.G. Hernandez, C.J. Fields, J.-P. Leburton, D. Soloveichik, H. Zhao, O. Milenkovic, DNA punch cards for storing data on native DNA sequences via enzymatic nicking, Nature Commun. 11 (2020) 1742.
[18] H.G. Yeh, d-Disjunct matrices: Bounds and Lovász local lemma, Discrete Math. 253 (1–3) (2002) 97–107.