Generic Decoding in the Sum-Rank Metric

Sven Puchinger, IEEE Member, Julian Renner IEEE Student Member, Johan Rosenkilde

Abstract

We propose the first non-trivial generic decoding algorithm for codes in the sum-rank metric. The new method combines ideas of well-known generic decoders in the Hamming and rank metric. For the same code parameters and number of errors, the new generic decoder has a larger expected complexity than the known generic decoders for the Hamming metric and smaller than the known rank-metric decoders. Furthermore, we give a formal hardness reduction, providing evidence that generic sum-rank decoding is computationally hard. As a by-product of the above, we solve some fundamental coding problems in the sum-rank metric: we give an algorithm to compute the exact size of a sphere of a given sum-rank radius, and also give an upper bound as a closed formula; and we study erasure decoding with respect to two different notions of support.

Index Terms

Decisional Sum-Rank Syndrome Decoding Problem, Erasure Decoding, Generic Decoding, Probabilistic Hardness Reduction, Sum-Rank-Metric Codes

I. INTRODUCTION

The sum-rank metric is a family of metrics which contains both Hamming and rank metric as special cases and in general can be seen as a mix of the two. It was introduced under the name “extended rank metric” as a suitable distance measure for multi-shot network coding in 2010 [2]. Since then, several code constructions and efficient decoders have been proposed for the metric [3]–[13]. The codes have also been studied in the context of distributed storage [14], further aspects of network coding [10], and space-time codes [15]. Recently, the authors of [16] derived several fundamental results on sum-rank-metric codes, including various bounds, MacWilliams identities, and new code constructions.

A generic decoder is an algorithm that takes a code and a received word as input and outputs a codeword that is close to the received word, without any restriction on or knowledge about the structure of the code. Designing such algorithms has a long tradition in coding theory, both for theoretical and practical reasons: studying the complexity of generic decoding is essential to evaluate the practical security level of code-based cryptosystems such as the McEliece [17], Niederreiter [18] and Gabidulin–Paramonov–Tretjakov [19] cryptosystems, or the numerous variants thereof. A trivial generic decoding algorithm is to simply tabulate the input code and compare each codeword with the received word, but there are much more efficient approaches. For the Hamming metric, the related decision problem is NP-hard [20], and there is also a hardness reduction for the rank metric [21], so it is not surprising that all known general decoding algorithms have exponential running time in the code parameters.

Prange [22] presented in 1962 a generic decoder for the Hamming metric whose type is now known as information-set decoding. The basic idea is to repeatedly choose $n-k$ random positions, where $n$ is the length and $k$ the dimension of the code, until the chosen positions contain all the errors and the complementary positions form an information set. This event can be detected by re-encoding on the remaining $k$ positions, obtaining a codeword, and seeing that this is close to the received word. There have been at least 27 papers improving Prange’s algorithm (see the list in [23, Section 4.1]), which have significantly reduced the exponent of the complexity expression.

In the rank metric, the first generic decoder was proposed in 1996 [24] and since then, there have also been several improvements [25]–[28]. One idea here is to repeatedly choose a sub row space (or column space) of the received word until this contains the error row space (resp. column space), and when it does use rank-erasure decoding techniques to decode using linear algebra. The complexity of generic decoding in the rank metric remains significantly higher than in the Hamming metric, which results in a substantial advantage of rank-metric-based cryptosystems over their Hamming-metric analogs.

A. Contributions

In this paper, we propose the first non-trivial generic decoding algorithm for arbitrary $\mathbb{F}_{q^m}$-linear codes in the sum-rank metric, where $\mathbb{F}_{q^m}$ denotes the field over which the code is defined. The algorithm takes as input parameters which specify the metric, a parity-check matrix of the code, the received word, and the sum-rank weight of the additive error $t$. The algorithm outputs a vector with weight at most $t$ such that the difference of this vector and the received word is a codeword. If $t$ is at most half the minimum distance of the code, the obtained vector is equal to the error of the received word. For this purpose,

This work was partly presented at the IEEE International Symposium on Information Theory (ISIT), 2020 [1]. S. Puchinger and J. Rosenkilde are with the Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Denmark. J. Renner is with the Institute for Communications Engineering, Technical University of Munich (TUM), Germany. S. Puchinger has been supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 713683. J. Renner has been supported by the European Union’s Horizon 2020 research and innovation programme under the European Research Council (ERC) grant agreement no. 801434.
the algorithm combines the sketched ideas for the Hamming and rank metric: we first randomly choose a rank in each block according to a carefully crafted distribution, and then for each block choose a random row or column space of the given rank. The process succeeds when the error row or column space in each block is covered, whence decoding is performed using sum-rank erasure decoding using linear algebra.

The most involved part is to design a suitable distribution from which to draw random vectors of a given sum-rank. In fact, we first observe that even counting the number of such vectors is non-trivial, and so drawing uniformly at random is also non-trivial. Our distribution is more involved than this, since it turns out that the probability of successful decoding depends on how the rank errors are distributed across blocks. Roughly, the complexity of our decoding algorithm smoothly interpolates between the basic generic decoders in the two “extremal” cases of the sum-rank metric: Hamming and rank metric.

Our work can be seen as a proof-of-concept that known methods of generic decoding can be adapted to the sum-rank metric. Though out of scope of this paper, it seems reasonable that many improvements for generic decoding in Hamming and rank metric can also be applied, which might further reduce the complexity.

As related results, we study several fundamental problems related to the sum-rank metric:

- We propose an efficient algorithm to compute the number of vectors of a given sum-rank weight. Apart from the use in our work, this can e.g. be used to efficiently compute the sphere-packing and Gilbert–Varshamov bounds in [16].
- We give a simple upper bound on the size of a sum-rank-metric sphere.
- Besides the existing notion of row support [29] and an associated row-erasure decoder [14], we introduce a “transposed” notion of column support and an associated column-erasure decoder. We analyze the computational complexity of both erasure decoders.

Finally, we generalize the formal hardness proof of [21] from the rank metric to the sum-rank metric. We show that if, for sufficiently large base field, the decisional sum-rank syndrome decoding problem is in the complexity class \( ZPP \), then \( NP = ZPP \). Loosely, \( ZPP \) is the set of problems which are computationally easy if one is allowed to use randomness, and includes the problems which are easy to solve deterministically, i.e. \( P \). Our result means that sum-rank syndrome decoding is either hard (i.e. not in \( ZPP \)), or that all \( NP \) problems are easy.

B. Reader’s Guide

After giving some preliminaires in Section II, we study the problem of counting vectors of a given sum-rank weight in Section III. This gives a first comparative line for the generic decoder and is also required for the formal hardness proof. In Section IV, we introduce two notions of support in the sum-rank metric and show how to efficiently erasure-decode w.r.t. these types of support. Erasure decoding is an essential ingredient of the new generic decoder. Section V presents the generic decoder. We explain how to randomly find a super-support of the error and show how to efficiently implement and bound the complexity of both erasure decoders.

Section VI presents the formal hardness proof. In Section VII, we compare the generic decoder to other (naive) generic decoders, as well as existing algorithms for the Hamming and rank metric. Section VIII presents the formal hardness proof.

II. Preliminaries

A. Notation

Let \( q \) be a prime power and \( m \) be a positive integer; the codes we consider are over \( \mathbb{F}_{q^m} \), the finite field with \( q^m \) elements, whose elements we often expand into \( \mathbb{F}_q^m \) vectors. For \( r \in \mathbb{Z}_{>0} \) and a fixed basis of \( \mathbb{F}_{q^m} \) over \( \mathbb{F}_q \), we define the mapping

\[
\text{ext}_{q,m}^r : \mathbb{F}_{q^m}^r \mapsto \mathbb{F}_q^{rn},
\]

where the \( i \)-th column of \( X \) is the expansion of \( x_i \) in the fixed basis over \( \mathbb{F}_q \). We use the big-O notation family to state asymptotic costs of algorithms, and \( O^\sim(\cdot) \), which neglects logarithmic terms in the input parameter. For a finite set \( S \), we denote by \( s \sim \leftarrow S \) the operation of drawing uniformly at random an element \( s \) from \( S \).

B. Sum-Rank Metric

Throughout the paper, \( n \) is the length of the studied codes, and \( \ell \) is a blocking parameter satisfying \( \ell \mid n \). The length of each block is \( \eta := n/\ell \), and we let \( \mu := \min \{ \eta, m \} \). For a vector \( x \in \mathbb{F}_{q^m}^n \), we define \( \text{rk}_{\ell,q}(x) := \dim_{\mathbb{F}_q} \langle x_1, \ldots, x_\eta \rangle_{\mathbb{F}_q} = \text{rk}_{\ell,q}(\text{ext}_{q,m}^\ell(x)) \). Obviously, \( \text{rk}_{\ell,q}(x) \leq \mu \). The sum-rank metric is defined as follows.

**Definition 1.** The \((\ell,\mu,\eta)-\text{sum-rank weight} is defined as**

\[
\text{wt}_{\text{SR},\ell} : \mathbb{F}_{q^m}^n \rightarrow \mathbb{Z}_{\geq 0},
\]

\[
x \mapsto \sum_{i=1}^\ell \text{rk}_{\ell,q}(x_i),
\]
where we write \( \mathbf{x} = [x_1 | x_2 | \ldots | x_t] \) with \( x_i \in \mathbb{F}_{q^n} \). We call
\[
[rk_{q^n}(x_1), \ldots, rk_{q^n}(x_t)] \in \{0, \ldots, \mu\}^t
\]
the weight decomposition of \( \mathbf{x} \). Furthermore, the \((t, \mu)\)-sum-rank distance is defined as
\[
d_{SR,t} : \mathbb{F}_{q^n}^n \times \mathbb{F}_{q^n}^m \rightarrow \mathbb{Z}_{\geq 0}, \quad [\mathbf{x}, \mathbf{x}'] \mapsto wt_{SR,t}(\mathbf{x} - \mathbf{x}')
\]
The family of sum-rank metrics includes two well-known metrics as extremal cases: For \( t = 1 \), it coincides with the rank metric, \( wt_R \), and for \( t = n \), it is the Hamming metric, \( wt_H \). In between, we have \( wt_R(\mathbf{x}) \leq wt_{SR,t}(\mathbf{x}) \leq \min \{ \mu t, wt_H(\mathbf{x}) \} \) for \( \mathbf{x} \in \mathbb{F}_{q^n}^n \).

**Remark 1.** Some results in this paper can be generalized in a relatively straightforward way to the sum-rank metric with varying block size (i.e., subblocks of \( \mathbf{x} \) are of the form \( x_i \in \mathbb{F}_{q^n} \) for positive integers \( \eta_1, \ldots, \eta_i \) with \( \sum_{i=1}^t \eta_i = n \)). We decided to present only the constant block size case \( (\eta_i = \eta \text{ for all } i) \) to avoid an even more technical presentation.

**C. Gaussian Binomial and Number of Matrices**

For non-negative integers \( a \) and \( b \), the Gaussian binomial \( \begin{bmatrix} a \\ b \end{bmatrix}_q \) is defined by the number of \( b \)-dimensional subspaces of \( \mathbb{F}_q^n \). We have
\[
\begin{bmatrix} a \\ b \end{bmatrix}_q = \prod_{i=1}^b \frac{q^{a-b+1} - 1}{q^i - 1}
\]
and the bounds \([30]\)
\[
q^{(a-b)b} \leq \begin{bmatrix} a \\ b \end{bmatrix}_q \leq \gamma_q q^{a-bb},
\]
where
\[
\gamma_q := \prod_{i=1}^{\infty} (1 - q^{-i})^{-1}.
\]
Note that \( \gamma_q \) is monotonically decreasing in \( q \) with a limit of 1, and e.g. \( \gamma_2 \approx 3.463, \gamma_3 \approx 1.785, \) and \( \gamma_4 \approx 1.452 \). We let \( \text{NM}_q(a,b,i) \) denote the number of \( a \times b \) matrices over \( \mathbb{F}_q \) of rank exactly \( i \), for \( 0 \leq i \leq \min \{a,b\} \). We have \([31]\):
\[
\text{NM}_q(a,b,i) = \prod_{j=0}^{i-1} \frac{(q^a - q^b)(q^a - q^b)}{q^{j+1} - q^j} \leq 4q^{(a+b)-i^2}.
\]

**D. Weight Decompositions and Partitions**

For a non-negative integer \( t \leq \ell \mu \), we define the set
\[
\mathcal{T}_{t, \ell, \mu} := \left\{ t \in \{0, \ldots, \mu\}^\ell : \sum_{j=1}^\ell t_j = t \right\},
\]
which contains all possible weight decompositions of a vector with \( \ell \mu \)-sum-rank weight \( t \).

The set \( \mathcal{T}_{t, \ell, \mu} \) has also a combinatorial interpretation: its elements correspond exactly to the ordered partitions of the integer \( t \) with part size at most \( \mu \) and number of parts at most \( \ell \). Hence, its cardinality is the \( t \)-th coefficient of the generating polynomial\(^1\)
\[
P^{(\ell, \mu)}(X) = \left( \sum_{i=0}^{\mu} X^i \right)^\ell,
\]
i.e.,
\[
|\mathcal{T}_{t, \ell, \mu}| = \left| P^{(\ell, \mu)} \right| = \sum_{i=0}^{\ell+1} (-1)^i \binom{\ell}{i} \binom{t + \ell - 1 - (\mu + 1)i}{\ell - 1}.
\]
In particular, \( |\mathcal{T}_{t, \ell, \mu}| \) can be computed efficiently, and we have the upper bound
\[
|\mathcal{T}_{t, \ell, \mu}| \leq \binom{\ell + t - 1}{\ell - 1}.
\]
Depending on the relative size of \( \ell \) and \( \mu \), the cardinality \( |\mathcal{T}_{t, \ell, \mu}| \) may grow super-polynomially in \( t \).

\(^1\)We would like to thank Cornelia Ott for deriving this closed-form expression for \( |\mathcal{T}_{t, \ell, \mu}| \).
E. Linear Codes

Throughout this paper, we consider $\mathbb{F}_{q^m}$-linear codes. An $\mathbb{F}_{q^m}$-linear code $C$ over $\mathbb{F}_{q^m}$ of dimension $k$ and length $n$ is an $\mathbb{F}_{q^m}$-linear $k$-dimensional subspace of $\mathbb{F}_{q^m}^n$, and we write $C_{[n,k]}_{\mathbb{F}_{q^m}}$. The minimum ($\ell$-)sum-rank distance of $C$ is given by

$$d = \min_{c,d \in C_{\mathbb{F}_{q^m}}} \{d_{SR,\ell}(c,d)\}.$$  

If $d$ is known, we call the code $C$ an $[n,k,d]_{\mathbb{F}_{q^m}}$ code. A matrix $G \in \mathbb{F}_{q^m}^{k \times n}$ is a generator matrix of $C$ if and only if its rows form a basis of $C$. Furthermore, a parity-check matrix $H \in \mathbb{F}_{q^m}^{(n-k) \times n}$ of $C$ is matrix whose rows form a basis of the right kernel of $G$.

In this paper, we aim at solving the following problem for any given code $C$:

**Problem 2 (Generic Sum-Rank-Metric Decoding).**

**Given:**

- Parameters $q, m, k, n, \ell, t$ with $n \leq t \leq \min\{n, m\} \ell$
- Parity-check matrix $H \in \mathbb{F}_{q^m}^{(n-k) \times n}$ of an $\mathbb{F}_{q^m}$-linear $[n,k]_{\mathbb{F}_{q^m}}$ code $C$
- Received vector $r = c + e \in \mathbb{F}_{q^m}^n$, where $c \in C$ and $wt_{SR,\ell}(e) = t$

**Objective:** Find a vector $e'$ with $wt_{SR,\ell}(e') \leq t$ such that $r - e' \in C$.

**Remark 3.** We formulate Problem 2 such that the sum-rank weight of the additive error is known and at least one solution to the problem exists. This results from the fact that this is true for most of the applications of generic decoding algorithms. For instance in the code-based encryption schemes BIKE [32], HQC [33], ROLLO [34], RQC [35], and ClassicMcEliece [36], whose security relies on generic decoding in either the Hamming or the rank metric (all systems reached at least the second round of the NIST post-quantum standardization process [37]).

III. Counting Error Vectors

As the generic decoding problem can be solved by brute-forcing through all vectors of a given sum-rank weight, we are interested in finding the number of such vectors. The question of counting is also related to explicitly writing down a list of such vectors (hence, how to realize this naive generic decoder) and provides a comparative line for the complexity of our new generic decoder that we present in the remainder of the paper. In the extreme cases of the Hamming and rank metric, simple closed-form expressions are easy to obtain. The question seems more involved for the general sum-rank metric.

We denote by $N_{q,\eta,m}(t,\ell)$ the number of vectors in $\mathbb{F}_{q^m}^d$, of $\ell$-sum-rank weight exactly $t \leq \mu \ell$. It is easy to see that we have

$$N_{q,\eta,m}(t,\ell) = \sum_{t \in T_{\ell,\eta,\mu}} \prod_{i=1}^\ell NM_q(m,\eta,t_i).$$

However, the number of terms in this formula is $|T_{\ell,\eta,\mu}|$ and it is not obvious how the sum can be computed efficiently. For this reason, we propose an efficient dynamic programming routine to compute the number. The method is based on the following lemma and outlined in Algorithm 1 below; note that $q, \eta,$ and $m$ remain constant throughout the recursion.

**Lemma 4.** $N_{q,\eta,m}(t,\ell) = 0$ for $t > \mu \ell$. Otherwise:

$$N_{q,\eta,m}(t,\ell) = \begin{cases} NM_q(m,\eta,t), & \text{if } \ell = 1, \\ \min_{\{\eta,m,t\}} \sum_{t' = 0}^{t} NM_q(m,\eta,t') \cdot N_{q,\eta,m}(t-t',\ell-1), & \text{if } \ell > 1. \end{cases}$$

**Proof.** The first claim is obvious since each of the $\ell$ blocks can have at most rank weight $\mu$. For $\ell = 1$, the formula is simply the number of $m \times \eta$ matrices of rank $t$. For larger $\ell$, we sum up over the number of possibilities to choose the rank weight $t'$ of the first block multiplied with the number of sum-rank weight words in the remaining $\ell - 1$ blocks.  

We also give a simple upper bound on $N_{q,\eta,m}(t,\ell)$, which we use for bounding the complexity of Algorithm 1, as well as for proving the formal hardness of generic decoding in Section VII.

**Theorem 5.** For $\ell > 1$ and $t \leq \mu \ell$, the number of vectors in $\mathbb{F}_{q^m}^d$ of $\ell$-sum rank weight $t$ can be bounded by

$$N_{q,\eta,m}(t,\ell) \leq \gamma_q^\ell \left( \frac{\ell + t - 1}{\ell - 1} \right)^{\ell} \left( \frac{1}{q^{m+\eta}} \right)^t,$$

where $\gamma_q \leq 3.5$ is given in (2).
Proof. By definition,
\[ N_{q,\eta,m}(t,\ell) = \sum_{t \in T_{t,\ell,\mu}} \prod_{i=1}^{\ell} N_{M_{q}}(m,\eta,t_i) \]
\[ \leq |T_{t,\ell,\mu}| \max_{t \in T_{t,\ell,\mu}} \left\{ \prod_{i=1}^{\ell} N_{M_{q}}(m,\eta,t_i) \right\} \]
\[ \leq \left( \frac{\ell + t - 1}{\ell - 1} \right)^{\gamma_q q t (m + \eta - t)} \]
where the latter inequality follows from \(|T_{t,\ell,\mu}| \leq \left( \frac{\ell + t - 1}{\ell - 1} \right)^{\ell} \) and \( N_{M_{q}}(m,\eta,t_i) \leq \gamma_q q t (m + \eta - t) \). Thus we should upper-bound \( \max_{t \in T_{t,\ell,\mu}} \left\{ \sum_{i=1}^{\ell} t_i(m + \eta - t_i) \right\} \) subject to \( \sum_{i=1}^{\ell} t_i = t \), which simplifies to maximising
\[ t(m + \eta) - \sum_{i=1}^{\ell} t_i^2. \]
By Jensen’s inequality, this is upper-bounded by choosing \( t_i = t/\ell \) for all \( i \), i.e.
\[ \max_{t \in T_{t,\ell,\mu}} \left\{ \sum_{i=1}^{\ell} t_i(m + \eta - t_i) \right\} \leq t(m + \eta) - t^2/\ell. \]

Fig. 1 shows example values of \( N_{q,\eta,m}(t,\ell) \) and the bound in Theorem 5 for different divisors \( \ell \) of a fixed length \( n \). It seems that the bound is quite tight for most values of \( \ell \), and only significantly differs for \( \ell \) close to \( n \). This deviation is due to the factor \( \gamma_q q t (m + \eta - t) \), which is large for these values of \( \ell \), and which is due to a relatively bad bound on the number of matrices. Note that for \( \ell = n \), we know better bounds on \( N_{q,\eta,m}(t,\ell) \) from the Hamming metric.

\[ \text{Fig. 1. Comparison of the exact number of vectors of sum-rank weight } t = 10 \text{ and the derived upper bound for } q = 2, m = 40, n = 60 \text{ as a function of } \ell. \]

**Theorem 6.** Algorithm 1 is correct and has bit complexity
\[ O^\sim(\ell t^2 + \ell t^3(m + \eta) \log(q)). \]

**Proof.** The algorithm computes a table that fulfills \( N(t',\ell') = N_{q,\eta,m}(t',\ell') \) for all \( t' = 0,\ldots,t \) and \( \ell' = 1,\ldots,\ell \) using the recursive formula in Lemma 4. This implies the correctness.

Complexity-wise, the algorithm performs \( \ell t^2 \) integer multiplications, where the size of the integers are such that they impact performance. An upper bound is given by
\[ N_{q,\eta,m}(t,\ell) \leq \left( \frac{\ell + t - 1}{\ell - 1} \right)^{\gamma_q q t (m + \eta - t)} \]
\[ \leq \left( \frac{\ell + t - 1}{\ell - 1} \right)^{\ell - 1} \gamma_q q t (m + \eta - t) \]
(4)
Algorithm 1: Compute \( N_{t,\ell}(m,\eta) \)

Input : Prime power \( q \) and \( \eta, m, \ell, t \in \mathbb{Z}_{\geq 0} \) such that \( 0 < t \leq \mu \ell \) and \( \mu := \min\{\eta, m\} \)

Output : Number \( N_{t,\ell}(m,\eta) \) of vectors in \( F_{q}^{\ell} \) of \( t \)-sum-rank weight \( \ell \)

1. Initialize table of integers \( \{N(t',\ell') = 0\}^{\ell'} \)
2. for \( \ell' = 0, \ldots, \ell \) do
3. \[ N(t', 1) \leftarrow N_{M}(m, \eta, t') \]
4. for \( \ell' = 2, \ldots, \ell \) do
5. \[ N(t', \ell') \leftarrow \sum_{t''=0}^{t} N_{M}(m, \eta, t'') N(t - t'', \ell' - 1) \]
6. return \( N(t, \ell) \)

where the first inequality follows from Theorem 5, the second inequality follows from an upper bound on binomial coefficients, and \( e \) is Euler’s constant. Since integer multiplication can be implemented with quasi-linear bit operations in the bit size of the involved integers [38], each multiplication costs at most

\[
O_{\sim} \left( (\ell - 1) \log \left( e^{-\gamma} q^{-\frac{\ell + t - 1}{\ell - 1}} + t(m + \eta - \frac{1}{2}) \log(q) \right) \right)
\]

\[
\leq O_{\sim} (\ell + t(m + \eta - \frac{1}{2}) \log(q))
\]

\[
\leq O_{\sim} (\ell + t(m + \eta) \log(q)).
\]

Corollary 7. There is a deterministic algorithm that solves Problem 2 using at most \( W_{\text{errors}} \) operations in \( F_{q} \), where

\[
W_{\text{errors}} \in O \left( n(n-k)m^{2(\ell+t-1)}q_{\ell}^{(m+\eta-1)} \right) .
\] (5)

Proof. Algorithm 1 can be easily adapted to create a list of all errors of sum-rank weight \( t \): instead of storing the number of vectors in the table \( N(\cdot, \cdot) \), we store lists of the respective vectors. By brute-forcing the overall list and checking whether the received word minus each error is a codeword (this costs at most \( O(n(n-k)m^{2}) \) operations over \( F_{q} \). Notably, the constant in the \( O \) notation is small.), we obtain a generic decoder with complexity

\[
O(n(n-k)m^{2}N_{q,\ell}(t,\ell)) \leq O(n(n-k)m^{2(\ell+t-1)}q_{\ell}^{(m+\eta-1)}),
\]

using (4) in the proof of Theorem 6.

The binomial in the expression can be simplified, depending on the relation between \( t \) and \( \ell \): for instance, since \( t \leq \ell \mu \), then \( \frac{t}{\ell} \leq 2\mu \), and therefore

\[
\left( \frac{t}{\ell} - 1 \right) ^{\ell - 1} \leq \left( \frac{\ell + t - 1}{\ell - 1} \right) ^{\ell - 1} \in O \left( \left[ e(2\mu + 1) \right] ^{\ell} \right) ,
\]

where \( e \) is Euler’s constant.

Remark 8. The recursion in Lemma 4 can be turned into an efficient algorithm to draw uniformly at random from the set of vectors of sum-rank weight \( t \), see Appendix A.

Remark 9. In [16], several fundamental bounds of sum-rank-metric codes are derived. To evaluate two of their bounds, the sphere-packing and Gilbert–Varshamov bound, one needs to efficiently compute the volume of a ball of given sum-rank radius, but this is not addressed in [16]. Algorithm 1 (and a straightforward variant thereof for variable block size and extension degree in each block) provides an efficient method to do this. Furthermore, the upper bound in Theorem 5 allows a significant simplification of their Gilbert–Varshamov bound, though we have not investigated how much weaker it becomes.

IV. Erasure Decoding and Support in the Sum-Rank Metric

In Section V, we will present a new generic decoding algorithm for the sum-rank metric. The idea is similar to the generic decoders in the Hamming and rank metric: first we find the “support” of an error (e.g., the error positions in the Hamming metric) in a randomized fashion and second we compute the full error by erasure decoding (e.g., computing the error values after having found the error positions).

In this section, we therefore study two notions of support in the sum-rank metric: row and column support. We describe erasure decoding w.r.t. these two notions, i.e., we explain under which conditions and in which complexity we can uniquely
Furthermore, the decomposition is unique up to elementary $F$-row operations follows directly from the analogous results in the rank metric.

Let $\Xi_q := \{ \mathbf{e} \in \mathbb{F}_q^m \mid \mathbf{e} \text{ has rank } \ell \}$ be its weight decomposition. Then there are vectors $a_i \in \mathbb{F}_q^{\ell_i}$, for $i = 1, \ldots, \ell$, as well as matrices over the sub-field $\mathbb{F}_q$:
\[
B_i \in \mathbb{F}_q^{t_i \times \eta}, \quad \text{ranks}(B_i) = t_i, \quad \text{for } i = 1, \ldots, \ell,
\]

such that
\[
e = [a_1 \ a_2 \ a_3 \ \ldots \ a_\ell], \quad \begin{bmatrix} 0 & \ & \ & 0 \\
0 & B_1 & \ & \ & \ & \ & \ & B_{\ell_i} \\
0 & \ & \ & 0 \end{bmatrix}.
\]

Furthermore, the decomposition is unique up to elementary $\mathbb{F}_q$-row operations on the matrices $B_i$. In particular, the $\mathbb{F}_q$-row spaces of the matrices $B_i$, as well as the $\mathbb{F}_q$-column space of $\text{ext}_{q,m}(a_i)$, are uniquely determined by $e$.

Proof. By basic linear algebra, see e.g. [39], there is an $a_i \in \mathbb{F}_q^{t_i}$ and $B_1 \in \mathbb{F}_q^{t_i \times \eta}$ such that $e_i = a_i B_i$. Also the uniqueness up to row operations follows directly from the analogous results in the rank metric.

**Definition 2.** Let $e \in \mathbb{F}_q^m$ be of sum-rank weight $t$.

- **Row Support:** The row support of $e$ is defined as the product of subspaces
\[
\mathcal{E}_e^{(R)} := \mathcal{E}_1^{(R)} \times \mathcal{E}_2^{(R)} \times \cdots \times \mathcal{E}_\ell^{(R)},
\]

where $\mathcal{E}_i^{(R)} \subseteq \mathbb{F}_q^t$ is the $\mathbb{F}_q$-row space of $B_i \subseteq \mathbb{F}_q^{t_i \times \eta}$ as in Lemma 10. A product
\[
\mathcal{F}^{(R)} := \mathcal{F}_1^{(R)} \times \mathcal{F}_2^{(R)} \times \cdots \times \mathcal{F}_\ell^{(R)}
\]

of subspaces $\mathcal{F}_i^{(R)} \subseteq \mathbb{F}_q^t$ is called a row super-support of $e$, denoted by $\mathcal{E}_e^{(R)} \subseteq \mathcal{F}^{(R)}$, if $\mathcal{E}_i^{(R)} \subseteq \mathcal{F}_i^{(R)}$ for all $i$.

- **Column Support:** The column support of $e$ is defined by
\[
\mathcal{E}_e^{(C)} := \mathcal{E}_1^{(C)} \times \mathcal{E}_2^{(C)} \times \cdots \times \mathcal{E}_\ell^{(C)},
\]

where $\mathcal{E}_i^{(C)} \subseteq \mathbb{F}_q^m$ is the column space of $\text{ext}_{q,m}(a_i) \subseteq \mathbb{F}_q^{m \times t_i}$ as in Lemma 10. A column super-support $\mathcal{F}^{(C)} \supseteq \mathcal{E}_e^{(C)}$ is defined analogously to the row case.

If it is clear from the context that we mean the row or column support, we will simply write $\mathcal{E}_e$, $\mathcal{F}$, and $\mathcal{E}_e \subseteq \mathcal{F}$, and omit the prefixes “row” and “column” to simplify notation.

**Remark 11.** It is easily seen that Definition 2 specializes the usual notions of support for Hamming metric when $\ell = n$, and the row and column support, respectively, in the rank metric for $\ell = 1$.

The following notation will be useful in the next section.

**Definition 3.** Let $\zeta$ be a positive integer and $0 \leq s \leq \ell \zeta$. For $s \in \mathcal{T}_{s,\ell,\zeta}$, we define the set
\[
\Xi_{q,\zeta}(s) := \left\{ \mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_\ell : \mathcal{F}_i \text{ is an } s_i \text{-dimensional subspace of } \mathbb{F}_q^t \right\}.
\]

For any $\mathcal{F} \in \Xi_{q,\zeta}(s)$, we say that its weight decomposition is $s$ and its weight is $s$. 

B. Erasure Decoding

The following theorem generalizes the classical Hamming metric statement that \( d - 1 \) is the maximal number of linearly independent columns, as well as the analogous statement in rank metric [39, Theorem 1]:

**Lemma 12.** Let \( \mathbf{H} \in \mathbb{F}_q^{(n-k) \times n} \) be a parity-check matrix of a code \( C[n,k] \). Define for any integer \( 0 \leq t \leq n \) the set

\[
\mathcal{B}_{t,t} := \left\{ \mathbf{B} = \begin{bmatrix} B_1 & 0 & 0 & \ldots & 0 \\ 0 & B_2 & 0 & \ldots & 0 \\ 0 & 0 & B_3 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & B_t \end{bmatrix} \in \mathbb{F}_q^{t \times n} : B_1 \in \mathbb{F}_q^{t \times (n/t)}, \ \text{rk}(B_i) = t_i, \ \sum_{i=1}^{t} t_i = t \right\}
\]

Then, \( C \) has minimum \( t \)-sum-rank distance \( d \) if and only if

- we have \( \text{rk}_{\mathbb{F}_q}(\mathbf{HB}^\top) = d - 1 \) for any \( \mathbf{B} \in \mathcal{B}_{t,d-1} \) and
- we have \( \text{rk}_{\mathbb{F}_q}(\mathbf{HB}^\top) < d \) for at least one \( \mathbf{B} \in \mathcal{B}_{t,d} \).

**Proof.** The proof follows by the decomposition of words of a given \( t \)-sum-rank weight in Lemma 10, together with the definition of the minimum sum-rank distance, i.e., that \( \mathbf{Hx}^\top \neq 0 \) for any word of \( \text{wt}_{\text{SR},t}(\mathbf{x}) = d - 1 \) and there is at least one \( \mathbf{x} \in \mathbb{F}_q^n \) with \( \text{wt}_{\text{SR},t}(\mathbf{x}) = d \) and \( \mathbf{Hx}^\top = 0 \).

Lemma 12 implies the following statement about erasure decoding w.r.t. the row support in the sum-rank metric. The uniqueness of the recovered codeword can also be derived from [14, Corollary 1].

**Theorem 13** (Column Erasure Decoding). Let \( \mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^n \) be a received word, where \( \mathbf{c} \) is an unknown codeword of a code \( C \) with minimum sum-rank distance \( d \) and \( \mathbf{e} \) is an unknown error of sum-rank weight at most \( d - 1 \). If we know a row super-support \( \mathcal{S} = \mathcal{C}[i] \) of \( \mathbf{e} \) of weight at most \( d - 1 \), then we can uniquely recover \( \mathbf{c} \) from \( \mathbf{r} \) with complexity \( O((n-k)^3m^2) \) operations over \( \mathbb{F}_q \).

**Proof.** It follows from Lemma 10 that \( \mathbf{c} \) can be written as \( \mathbf{aB} \), where \( \mathbf{B} \) is a block-diagonal matrix containing bases of the super-support entries \( \mathcal{C}_i \). Let \( \mathbf{H} \) be a parity-check matrix of the given code \( C \) of minimum sum-rank distance \( d \). Since \( \mathcal{S} \) has weight \( t \leq d - 1 \), by Lemma 12, the matrix \( \mathbf{HB}^\top \in \mathbb{F}_q^{(n-k) \times t} \) has \( \text{SR}_t \)-rank \( t \). Hence, the linear system

\[
\mathbf{Hr}^\top = \mathbf{He}^\top = (\mathbf{HB}^\top)\mathbf{a}^\top,
\]

where \( \mathbf{a} \) is unknown, and \( \mathbf{r}, \mathbf{H}, \) and \( \mathbf{B} \) are known, has a unique solution \( \mathbf{a} \) and we can uniquely determine \( \mathbf{a}, \mathbf{e} \), and thus \( \mathbf{c} \) using linear-algebraic operations. Using elementary matrix multiplication, Gaussian elimination, and polynomial multiplication algorithms, the involved operations have the following complexities: Multiplying \( \mathbf{HB}^\top \) costs \( O((n-k)sm) \) operations in \( \mathbb{F}_q \) since each row of \( \mathbf{B} \) has at most \( s \) non-zero entries. The only remaining step is solving the linear system \((\mathbf{HB}^\top)\mathbf{a}^\top = s^\top \), where \( s \) is the syndrome of the received word. This costs \( O(s^2(n-k)) \) operations over \( \mathbb{F}_q \), and any operation in \( \mathbb{F}_q \) costs again \( O(m^2) \) operations in \( \mathbb{F}_q \).

Similarly, we can recover a codeword from the received word and a column super-support of the error.

**Theorem 14** (Row Erasure Decoding). Let \( \mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^n \) be a received word, where \( \mathbf{c} \) is an unknown codeword of a code \( C \) with minimum sum-rank distance \( d \) and parity check matrix \( \mathbf{H} \in \mathbb{F}_q^{(n-k) \times n} \). Further \( \mathbf{e} \) is an unknown error of sum-rank weight \( t < d \). If we know a column super-support of dimension \( t' \leq d - 1 \), then we can uniquely recover \( \mathbf{c} \) from \( \mathbf{r} \) with complexity \( O((n-k)^3m^2) \) operations over \( \mathbb{F}_q \).

**Proof.** Let \( \mathbf{H} = [\mathbf{H}_1, \ldots, \mathbf{H}_t] \), where \( \mathbf{H}_i \in \mathbb{F}_q^{(n-k) \times \eta} \). Then, using the same notation as in Theorem 13, the syndrome is equal to

\[
\mathbf{s}^\top = \mathbf{HB}^\top \mathbf{a}^\top = \sum_{i=1}^{t} \mathbf{H}_i \mathbf{B}_i^\top \mathbf{a}_i^\top = \sum_{i=1}^{t} \mathbf{H}_i \mathbf{B}_i^\top \hat{\mathbf{a}}_i^\top,
\]

where \( \hat{\mathbf{a}} = [\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_t] \in \mathbb{F}_q^{t \times \eta} \) is a basis of the known column super-support (more precisely, the columns of \( \text{ext}_{\eta,m}^t(\hat{\mathbf{a}}_i) \) form a basis of the \( i \)-th constituent subspace of the super-support) of the error and \( \mathbf{B}_i \in \mathbb{F}_q^{\ell_i \times \eta} \). To perform erasure decoding, we solve the latter system of equations for the \( \eta t' \) unknown entries of \( \mathbf{B}_1, \ldots, \mathbf{B}_t \) over \( \mathbb{F}_q \). The system over \( \mathbb{F}_q \) can be written as

\[
\mathbf{s}_{\text{ext}}^\top = \mathbf{H}_{\text{ext}} \mathbf{b}^\top,
\]

where \( \mathbf{s}_{\text{ext}} \in \mathbb{F}_q^{(n-k)m} \) is the expanded syndrome and the matrix \( \mathbf{H}_{\text{ext}} \in \mathbb{F}_q^{m(n-k) \times \eta t'} \) depends only on \( \mathbf{H} \) and \( \hat{\mathbf{a}} \). Further, the vector \( \mathbf{b} \) is defined as

\[
\mathbf{b} := [\hat{\mathbf{B}}_{111}, \ldots, \hat{\mathbf{B}}_{tt\eta}],
\]

where \( \hat{\mathbf{B}}_{ijr} \) denotes the entry in the \( j \)-th row and \( r \)-th column of the matrix \( \hat{\mathbf{B}}_i \).
The system has a unique solution if and only if \( \text{rk}(\hat{H}_{\text{ext}}) = \eta t' \). To see that this is always the case, suppose \( s_{\text{ext}}^\top = \hat{H}_{\text{ext}} \hat{b}^\top = 0 \) and \( \text{rk}(\hat{H}_{\text{ext}}) < \eta t' \). Then, there exists a vector \( \hat{b} \neq 0 \) such that \( \hat{H}_{\text{ext}} \hat{b}^\top = \hat{H}(\alpha \hat{B})^\top = 0 \) which means \( \alpha \hat{B} \in \mathcal{C} \setminus \{0\} \). Since \( \text{wt}_{\text{SR}, \ell}(\alpha \hat{B}) = t' < d \), this is a contradiction.

The highest step is to solve an \( m(n-k) \times \eta t' \) linear system over \( \mathbb{F}_q \), where \( \eta t' \leq m(n-k) \). This can be done in \( O(m^3(n-k)^3) \) operations over \( \mathbb{F}_q \).

**Remark 15.** As we consider \( \mathbb{F}_q^n \)-linear codes in this paper, it is necessary to treat row and column sum-rank supports separately. However, in case \( \mathbb{F}_q \)-linear or non-linear codes are considered, this distinction can be neglected since transposition preserves \( \mathbb{F}_q \)-linearity, and therefore, the column support can be thought of as the row support, and vice versa. Note that the presented algorithm can be adapted to \( \mathbb{F}_q \)-linear codes if an erasure decoder of this code is known. However, deriving an erasure decoder for \( \mathbb{F}_q \)-linear codes is outside the scope of this paper.

**V. THE GENERIC DECODER**

We have seen in the previous section that we can uniquely recover an error \( e \) if we find a row or column super-support \( \mathcal{F} \supseteq \mathcal{E}_s \) of sum-rank weight \( s \) with \( t \leq s < d \). In this section, we describe a Las Vegas-type algorithm (Algorithm 2 below) that chooses row or column supports \( \mathcal{F} \) of weight \( s \) at random according to a designed probability mass function (here denoted by \( \text{DrawRandomSupport}(s, t, \zeta) \), see Algorithm 4 in Section V-B). Notation-wise, there is no difference between drawing random row or column supports if we allow the ambient space dimension \( \dim F_i s \) of a constituent support subspace to be arbitrary. We denote this dimension by \( \zeta \) and set it \( \zeta = \eta \) (i.e., \( \mathcal{F}_i = \mathcal{F}_i^{(B)} \subseteq \mathbb{F}_q^s \)) in the row support case and \( \zeta = m \) (i.e., \( \mathcal{F}_i = \mathcal{F}_i^{(C)} \subseteq \mathbb{F}_q^m \)) in the column support case. We also omit the prefixes “row” or “column” in this section. This allows us to treat both cases in a unified manner.

**Algorithm 2: Generic Sum-Rank Decoder**

```plaintext
Input : Parameters \( q, m, k, n, \ell, t \) Parity-check matrix \( H \in \mathbb{F}_q^{(n-k) \times n} \) of an \( \mathbb{F}_q \)-linear \( [n,k]_{\mathbb{F}_q^m} \) code \( \mathcal{C} \) Received vector \( r \in \mathbb{F}_q^n \) Integer \( s \) with \( t \leq s \leq n-k \)

Output : Vector \( e' \in \mathbb{F}_q^n \) such that \( \text{wt}_{\text{SR}, \ell}(e') \leq t \) and \( r - e' \in \mathcal{C} \)

1. \( e' \leftarrow 0 \)
2. \( \eta \leftarrow n/\ell \)
3. \( \zeta \leftarrow \min\{m, \eta\} \)
4. while \( H(r - e')^\top \neq 0 \) or \( \text{wt}_{\text{SR}, \ell}(e') > t \) do
5.   \( \mathcal{F} \leftarrow \text{DrawRandomSupport}(s, t, \zeta) \) (Algorithm 4 in Section V-B)
6.   if \( \zeta = \eta \) then
7.     \( e' \leftarrow \text{column erasure decoding w.r.t. } \mathcal{F}, H, r \) (cf. Theorem 13)
8.   else
9.     \( e' \leftarrow \text{row erasure decoding w.r.t. } \mathcal{F}, H, r \) (cf. Theorem 14)
10. return \( e' \)
```

The main statement of this section is Theorem 16, which bounds the expected runtime of Algorithm 2. Note that by ignoring the cost of one iteration (i.e., setting \( W_{\text{iter}} = 1 \) in Theorem 16), one obtains lower and upper bounds on the expected number of iterations that the algorithm takes until a suitable support is found. Since the proof is rather technical, we prove it in the course of this section. In the statement, we use the notation \( Q_{t, \ell, \mu} \), which is defined in (15) below.

**Theorem 16.** Let \( e \) be a codeword of a sum-rank metric code \( \mathcal{C} \) of minimum sum-rank distance \( d \). Further, let \( e \) be an error of sum-rank weight \( t < d \). Then, Algorithm 2 with input \( r = e + e \) and parameter \( s \) with \( t \leq s < d \) returns an error \( e' \) of sum-rank weight \( t \) such that \( r - e' \) is a codeword.

Each iteration (Lines 5–9) of Algorithm 2 costs \( W_{\text{iter}} \in O^\sim(n^3m^3 \log_2(q)) \) bit operations. By including also the expected number of iterations, we can bound the overall expected runtime (in bit operations) \( W_{\text{new}} \) of Algorithm 2 by

\[
W_{\text{new}}^{(\text{LB})} \leq W_{\text{new}} \leq W_{\text{new}}^{(\text{UB})} \leq W_{\text{new}}^{(\text{UB, simple})},
\]

where, for \( \zeta = \mu = \min\{\eta, m\} \), we define (see (15) for \( Q_{t, \ell, \mu} \))

\[
W_{\text{new}}^{(\text{LB})} := |T_{t, \ell, \mu}|^{-1} Q_{t, \ell, \mu}, \quad W_{\text{new}}^{(\text{UB})} := W_{\text{iter}} Q_{t, \ell, \mu} \quad \text{and} \quad W_{\text{new}}^{(\text{UB, simple})} := W_{\text{iter}} (\ell + t - 1) \gamma_q^\ell (\zeta - \frac{\zeta}{2}),
\]
Furthermore, the more precise bounds (6) and (7) can be computed in bit complexity $O^\sim(tsn^3\mu^2\log_2(q))$.

Proof. See Section V-F.

Remark 17. We can guarantee uniqueness of erasure decoding in Algorithm 2 only for $s < d$, but it might work up to $s = \min\{n - k, \lfloor m/\eta(n - k)\rfloor\}$, depending on the chosen super-support. Most generic Hamming- and rank-metric decoding papers use $s = n - k$ without analyzing the erasure decoding success probability. Since in practice, the latter probability is high for many codes, $s = \min\{n - k, \lfloor m/\eta(n - k)\rfloor\}$ is indeed a good heuristic choice for a practical generic decoder.

A. Aim and Design of the Support Drawing Algorithm

Our aim in designing the probability distribution for drawing a random support $F$ of weight $s$ is to minimize the worst-case expected number of iterations until we find a super-support of $E_e$. Since we draw random supports $F$ until one of them is a super-support of $E_e$, the expected number of required draws is equal to the inverse of the probability that $F$ contains $E_e$. To achieve this, we draw a support $F$ per iteration, we have

$$\max_{e \in \mathbb{P}^n_q} \mathbb{E}[^\# \text{iterations}] = \max_{e \in \mathbb{P}^n_q} \left\{ \frac{1}{\Pr(E_e \subseteq F)} \right\}.$$

Our algorithm draws $F$ in two steps: First, we choose at random a weight decomposition $s \in T_{s, \ell, \mu}$ of weight $s$, according to a designed probability distribution $\hat{p}_s$. Then, we draw the support $F$ uniformly at random from the set $\Xi_{q, \zeta}(s)$ of supports with weight decomposition $s$. The following lemma states that the success probability of this decoder, conditioned on a specific weight decomposition $s$, only depends on $s$ and the weight decomposition $t_e$ of the error.

Lemma 18. Let $e$ be of $\ell$-sum-rank weight $t$. Further, let $s \in T_{s, \ell, \mu}$ and choose $F$ uniformly at random from $\Xi_{q, \zeta}(s)$. Then,

$$\Pr(E_e \subseteq F \mid s) = \varrho_{q, \zeta}(s, t_e),$$

where we define

$$\varrho_{q, \zeta}(s, t) := \prod_{i=1}^{\ell} \left\lfloor \frac{s_i}{t_{i, q}} \right\rfloor \left\lceil \frac{t_{i, q}}{t_{i, q}} \right\rceil.$$

In particular, $\Pr(E_e \subseteq F \mid s)$ only depends on the decompositions $s$ and $t_e$, and we have $\Pr(E_e \subseteq F \mid s) > 0$ if and only if $s \succeq t_e$, where $\succeq$ is the partial order given by coordinate-wise comparisons.

Furthermore, we have (with $1 \leq \gamma_q \leq 3.5$ as defined in (2))

$$\gamma_q^{-\ell}q^{-\sum_{i=1}^{\ell}t_i(\xi - s_i)} \leq \varrho_{q, \zeta}(s, t) \leq \gamma_q^{-\ell}q^{-\sum_{i=1}^{\ell}t_i(\xi - s_i)}.$$

Proof. Since $F$ is drawn uniformly, the subspaces $F_i$ are drawn independently and uniformly from the set of $s_i$-dimensional subspaces of $\mathbb{P}^n_q$. Hence, $\Pr(E_e \subseteq F \mid s)$ equals the product of the probabilities that the $i$-th subspace $F_i$ is a subspace of $E_i$. This probability is given by $\left[ \frac{s_i - t_{i, q}}{s_i - t_{i, q}} \right]^{-1}$, where the numerator counts the number of possibilities to expand the $t_{i, q}$-dimensional subspace $E_i$ into an $s_i$-dimensional space and the denominator gives the total number of $s_i$-dimensional subspaces of $\mathbb{P}^n_q$. By properties of the Gaussian binomial coefficient, we get $\left[ \frac{s_i - t_{i, q}}{s_i - t_{i, q}} \right]^{-1} = \left[ \frac{s_i}{t_{i, q}} \right]^{-1} \left[ \frac{t_{i, q}}{t_{i, q}} \right]^{-1}$. The bounds immediately follow from (1).

Lemma 18 allows us to compute the worst-case number of iterations of the algorithm for a given probability mass function $\hat{p}_s$ of $s$ by

$$\max_{e \in \mathbb{P}^n_q : \text{wt}(e) = t} \mathbb{E}[^\# \text{iterations}] = \max_{e \in \mathbb{P}^n_q : \text{wt}(e) = t} \left( \sum_{s \in T_{s, \ell, \mu}} \hat{p}_s \varrho_{q, \zeta}(s, t) \right)^{-1}.$$  

The problem of minimizing (11) over all valid distributions $\hat{p}_s$ on $T_{s, \ell, \mu}$ can be formulated as a linear program and solved numerically for small parameters $\ell, \zeta, s$ using standard methods. Note that the unknowns are the $\hat{p}_s \in [0, 1]$, and the number of unknowns, $|T_{s, \ell, \mu}|$, grows fast in $\ell, \zeta$, and $s$. Due to this limitation, we present a formal discussion in Appendix B of this “optimal” choice of $\hat{p}_s$, and continue with a more scalable solution.

We relax the problem of maximizing (11) as follows.

- We give a randomized mapping $\text{scmp}_{q} : T_{s, \ell, \mu} \rightarrow T_{s, \ell, \mu}$ that maximizes $\varrho_{q, \zeta}(\text{scmp}_{q}(t, s), t)$ for a given $t \in T_{s, \ell, \mu}$ (see Algorithm 18 and Lemma 19 below). This mapping is randomized, i.e. for each input there are multiple possible outputs and one is selected at random; we discuss this further below.
• Instead of choosing a vector \( s \in T_{s,\ell,\mu} \) directly, we first choose a vector \( t \in T_{s,\ell,\mu} \) at random according to a designed distribution \( p_t \) on \( T_{s,\ell,\mu} \), and set \( s \leftarrow \text{scomp}_\zeta(t, s) \). This means that for a fixed error \( e \), we can bound

\[
\Pr(\mathcal{E}_e \subseteq \mathcal{F}) = \sum_{s \in T_{s,\ell,\mu}} \hat{p}_s \cdot g_{\zeta,\zeta}(s, t_e) \
\geq p_{t_e} \cdot g_{\zeta,\zeta}(\text{scomp}_\zeta(t_e, s), t_e).
\]

This bound is relatively tight for this choice of \( \zeta \) (see Proposition 20 below).

• Instead of minimizing (11), we minimize the following upper bound on the worst-case expected number of iterations

\[
\max_{s \in T_{s,\ell,\mu} : w_{\text{SNR}}(s) = t} \mathbb{E}[\#\text{iterations}] \leq \max_{t \in T_{s,\ell,\mu}} \left[ p_t \cdot g_{\zeta,\zeta}(\text{scomp}_\zeta(t, s), t) \right]^{-1},
\]

over all valid probability mass functions \( p_t \) on \( T_{s,\ell,\mu} \).

This comes at the cost of a slightly smaller success probability than the optimal choice of \( \tilde{p}_\mu \) (cf. Section VI for a numerical comparison), but allows us to give a support drawing strategy that can be practically implemented and whose running time we can bound.

Algorithm 3 formally defines the randomized mapping \( \text{scomp}_\zeta \) and Lemma 19 proves that \( s = \text{scomp}_\zeta(t, s) \) maximizes \( g_{\zeta,\zeta}(s, t) \) among all \( s \in T_{s,\ell,\mu} \). The randomization in Line 6 prevents a bias in preferring certain positions (compared to some deterministic choice), and seems to be practically advantageous, especially for large \( \ell \); in fact, for the Hamming case with \( \mu = 1 \) and \( n = t \), then such a randomization is essential for the efficacy of Prange’s generic decoder (cf. Section VI-A). Our analysis, however, is not able to take the randomness properly into account, and will depend merely on \( g_{\zeta,\zeta,s}(t) \), which is defined as

\[
g_{\zeta,\zeta,s}(t) := g_{\zeta,\zeta}(\text{scomp}_\zeta(t, s), t)
\]

for all \( t \in T_{s,\ell,\mu} \) and a fixed \( s \geq t \). Note that though \( \text{scomp}_\zeta \) is randomized, then \( g_{\zeta,\zeta,s}(t) \) is not.

**Algorithm 3: \text{scomp}_\zeta(t, s)**

\begin{algorithm}
\begin{algorithmic}[1]
\Input \( t \in T_{s,\ell,\mu} \) and \( s \in \mathbb{Z} \) with \( t \leq s \leq \ell \mu \).
\Output \( s \in T_{s,\ell,\mu} \).
\State \( s = [s_1, \ldots, s_t] \leftarrow t \); \quad \delta \leftarrow s - t \)
\While {\( \delta > 0 \)}
\State \( J_1 \leftarrow \{ i \in \{1, \ldots, n\} : s_i \neq \zeta \} \)
\State \( J_2 \leftarrow \{ i \in J_1 : t_i = \max_{j \in J_1} t_j \} \)
\State \( J_3 \leftarrow \{ i \in J_2 : s_i = \min_{j \in J_2} s_j \} \)
\State \( h \leftarrow \# J_3 \)
\State \( s_h \leftarrow s_h + 1 \); \quad \delta \leftarrow \delta - 1 \)
\EndWhile
\State return \( s \)
\end{algorithmic}
\end{algorithm}

**Lemma 19.** Let \( t \in T_{s,\ell,\mu} \) and let \( t \leq s \leq \ell \mu \). Then, \( s = \text{scomp}_\zeta(t, s) \), with \text{scomp}_\zeta \) as in Algorithm 3, maximizes \( g_{\zeta,\zeta}(s, t) \), i.e.,

\[
g_{\zeta,\zeta}(\text{scomp}_\zeta(t, s), t) = \max_{s \in T_{s,\ell,\mu}} g_{\zeta,\zeta}(s, t).
\]

**Proof.** As the denominator of (9) is independent of \( s \), it suffices to show that \( s = \text{scomp}_\zeta(t, s) \) maximizes

\[
\prod_{i=1}^\ell \left[ \frac{s_i}{\binom{t_i}{q}} \right].
\]

for a given \( t \). Say that we start with \( s = t \) and increase entries of \( s \) by one until we have \( \sum_{i=1}^\ell s_i = s \) (note that we can assume this since (14) is zero if \( s_i < t_i \) for some \( i \)). We observe that (14) is increased by a factor

\[
\left[ \frac{s_i+1}{t_i} \right]_q \left[ \frac{s_i}{t_i} \right]_q.
\]
if we increase position \( i \) of \( s \). For \( s_i \geq t_i \), we have
\[
\begin{align*}
\frac{[s_i+1]}{t_i} &= \prod_{\mu = 1}^{s_i} \frac{\left(q^{s_i+2-\nu} - 1\right)}{q^n - 1} \\
\frac{[s_i]}{t_i} &= q^{s_i+1} - 1 \quad \frac{[t_i+1]}{t_i} = q^n - t_i+1 - 1
\end{align*}
\]
For a fixed \( t_i \), the quantity \( q^{s_i+1} - 1 \) is monotonically decreasing in \( s_i \), and we have
\[
q^{s_i} < q^{s_i+1} - 1 < q^{s_i+1}.
\]
It follows that the largest increase of (14) is achieved by increasing a position \( i \) with smallest \( s_i \) among those positions with largest \( t_i \). Increasing such a position in a greedy fashion attains a global maximum since this choice will also maximize the possible increase in the following steps. Hence, (14) is maximized by iteratively increasing \( s_i \) by one such that \( s_i \leq \zeta \) and \( \sum_{i=1}^{\ell} s_i \leq s \) for some \( i \) with smallest \( s_i < \zeta \) among those positions that have a maximal \( t_i \). This is exactly what \( \text{scomp}_\zeta(\cdot, \cdot) \) does.

**B. The Support-Drawing Algorithm**

Based on the ideas presented above, Algorithm 4 outlines the support-drawing algorithm that we propose. The probability distribution \( p_t \) is chosen to minimize the bound on the worst-case expected number of iterations in (12). The following proposition presents bounds on the expected number of iterations. Note that the lower and upper bound are independent of the error and differ by only a factor \(|T_{t, \ell, \mu}| \), which is relatively small compared to the absolute values of the bounds for not too large \( \ell \). For notational convenience, we define the following value:
\[
Q_{t, \ell, \mu} := \sum_{t \in T_{t, \ell, \mu}} q_{q, \zeta, s}(t)^{-1}.
\]  

**Algorithm 4: DrawRandomSupport(\( s, t, \zeta \))**  

**Input** : Integers \( t, s, \zeta \) with \( \mu \leq \zeta \) and \( t \leq s \leq \ell \mu \)  

**Output** : \( \mathcal{F} \) of weight \( s \)  

1. Draw \( t \in T_{t, \ell, \mu} \) according to the distribution \( p_t := q_{q, \zeta, s}(t)^{-1}Q_{t, \ell, \mu}^{-1} \forall t \in T_{t, \ell, \mu} \), where \( Q \) is defined as in (15)

2. \( s \leftarrow \text{scomp}_\zeta(t, s) \)

3. \( \mathcal{F} \leftarrow \text{scomp}_\zeta(s) \)

4. return \( \mathcal{F} \)

**Proposition 20.** Let \( e \) be an error of sum-rank weight \( t \) and let \( s \) be an integer with \( t \leq s \leq \ell \mu \). If \( \mathcal{F} \) is a super-support that is drawn by Algorithm 4 with input \( t \) and \( s \), then we have
\[
|T_{t, \ell, \mu}|^{-1}Q_{t, \ell, \mu} \leq \frac{1}{\Pr(\mathcal{E}_e \subseteq \mathcal{F})} \leq Q_{t, \ell, \mu},
\]
where \( Q_{t, \ell, \mu} \) is defined as in (15).

**Proof.** Denote by \( \tilde{p}_s \) the distribution of \( s = \text{scomp}_\zeta(t, s) \), where \( t \) is a random variable with probability mass function \( p_t \). By (11), we have
\[
\Pr(\mathcal{E}_e \subseteq \mathcal{F}) = \sum_{t \in T_{t, \ell, \mu}} p_t q_{q, \zeta, s}(\text{scomp}_\zeta(t, s), t_e) \geq p_t q_{q, \zeta, s}(\text{scomp}_\zeta(t_e, s), t_e) = Q_{t_e \mu}^{-1}.
\]
This proves the upper bound on \( \Pr(\mathcal{E}_e \subseteq \mathcal{F})^{-1} \). For the lower bound, we first observe that for all \( t \in T_{t, \ell, \mu} \), Lemma 19 implies
\[
q_{q, \zeta, s}(\text{scomp}_\zeta(t, s), t_e) \leq q_{q, \zeta, s}(\text{scomp}_\zeta(t, s), t) = q_{q, \zeta, s}(t).
\]
This yields
\[
\Pr(\mathcal{E}_e \subseteq \mathcal{F}) = \sum_{t \in T_{t,\ell,\mu}} p_t \theta_{q,\zeta}(s_{\text{comp}}(t, s), t_e)
\]
\[
\leq \sum_{t \in T_{t,\ell,\mu}} p_t \theta_{q,\zeta}(t)
\]
\[
= \sum_{t \in T_{t,\ell,\mu}} Q_{t,\ell,\mu}^{-1} = |\mathcal{T}_{t,\ell,\mu}| Q_{t,\ell,\mu}^{-1},
\]
which proves the claim.

At first glance, the lower and upper bounds in Proposition 20 appear infeasible to compute since the number of summands, \(|\mathcal{T}_{t,\ell,\mu}|\), may grow super-polynomially in \(t\) (depending on \(\ell\) and \(\mu\)). Furthermore, it is at this point unclear how to efficiently implement Line 1 of Algorithm 4. Below, we answer these two questions, and also give a simple upper bound on \(Q_{t,\ell,\mu}\).

### C. A Simple Bound on the Success Probability

We start with a simple bound on \(Q_{t,\ell,\mu}\) from (15).

**Proposition 21.** For any \(t \leq s \leq t\mu\), we have
\[
\max_{t \in T_{t,\ell,\mu}} \theta_{q,\zeta}(s)^{-1} \leq \gamma_q q^{t(\zeta-\bar{\zeta})},
\]
In particular,
\[
Q_{t,\ell,\mu} \leq \left(\frac{t}{t-1}\right) \gamma_q q^{t(\zeta-\bar{\zeta})},
\]
where \(\gamma_q \leq 3.5\) is defined as in (2).

**Proof.** By (10) in Lemma 18, we have
\[
\max_{t \in T_{t,\ell,\mu}} \theta_{q,\zeta}(s)^{-1} \leq \gamma_q \max_{t \in T_{t,\ell,\mu}} \left\{ q^{\sum_{i=1}^t t_i (\zeta - s_i) \mid s = s_{\text{comp}}(t, s)} \right\}
\]
\[
= \gamma_q \frac{q}{t} q^{t(\zeta-\bar{\zeta})}. \quad \text{for any } s = s_{\text{comp}}(t, s)
\]
We claim that the last exponent satisfies:
\[
\min_{t \in T_{t,\ell,\mu}} \left\{ \left(\frac{t}{t-1}\right) \gamma_q q^{t(\zeta-\bar{\zeta})} \mid s = s_{\text{comp}}(t, s) \right\} \geq \frac{t\delta}{\ell}
\]
We will prove this by relaxing the variables to reals, and consider only the ordered vectors \(t\), so define the set:
\[
\mathcal{T}_{t,\ell,\mu}^{(\text{ord})} := \left\{ t \in \mathbb{R}_{\geq 0}^\ell : \sum_{i=1}^\ell t_i = t, t_i \leq \mu, t_1 \geq t_2 \geq \cdots \geq t_\ell \right\}
\]
and the mapping
\[
s_{\text{comp}}^{(\text{R})} : \mathcal{T}_{t,\ell,\mu}^{(\text{ord})} \to \mathbb{R}_{\geq 0},
\]
\[
t \mapsto \left[ \underbrace{\zeta, \ldots, \zeta}_{h \text{ times}}, t_{h+1} + \underbrace{\xi + 1, \ldots, t_{h+g} + \xi + 1}_{g \text{ times}}, t_{h+g+1} + \underbrace{\delta, \ldots, t_{h+f} + \xi + \delta}_{f-g \text{ times}}, \ldots, t_\ell \right],
\]
where
\[
h := \max \left\{ h' \in \{0, 1, \ldots, \ell\} : \sum_{i=1}^{h'} (\zeta - t_i) \leq s - t, t_{h'+1} > t_{h'+1} \right\}
\]
\[
f := \max \{ f' \in \{1, \ldots, \ell\} : t_{f'} = t_{h+1} \} - h,
\]
\[
s_{\text{rem}} := s - t - \sum_{i=1}^h (\zeta - t_i),
\]
\[
\xi := \frac{s_{\text{rem}}}{f},
\]
\[
g := \frac{s_{\text{rem}}}{f} - \xi f,
\]
\[
\delta := \frac{s_{\text{rem}}}{f-g}.
\]
Note that $\text{scomp}_\zeta^{(R)}$ agrees with a deterministic variant of $\text{scomp}_\zeta$ on $\mathcal{T}_{t,\ell,\mu}^{(R,\text{ord})} \cap \mathbb{Z}^{\ell}$. Since $\sum_{i=1}^{\ell} t_i s_i | s = \text{scomp}_\zeta^{(R)}(t, s)$ is independent of the ordering of the entries of $t$ and the set of sorted elements (vectors) of $\mathcal{T}_{t,\ell,\mu}$ are subset of $\mathcal{T}_{t,\ell,\mu}^{(R,\text{ord})}$, we have

$$\min_{t \in \mathcal{T}_{t,\ell,\mu}} \left\{ \sum_{i=1}^{\ell} t_i s_i \mid s = \text{scomp}_\zeta^{(R)}(t, s) \right\} \geq \min_{t \in \mathcal{T}_{t,\ell,\mu}} \left\{ \sum_{i=1}^{\ell} t_i s_i \mid s = \text{scomp}_\zeta(t, s) \right\}.$$  

For $t \in \mathcal{T}_{t,\ell,\mu}^{(R,\text{ord})}$ and $s = \text{scomp}_\zeta^{(R)}(t, s)$, we have

$$\sum_{i=1}^{\ell} t_i s_i = h \sum_{i=1}^{h} (t_i + \xi + 1) + h+g \sum_{i=h+1}^{h+g} (t_i + \xi + \delta) + h+f \sum_{i=h+f+1}^{\ell} t_i^2.$$  

(16)

$$= \zeta \sum_{i=1}^{h} t_i + g(t_{h+1} + \xi + 1)t_{h+1} + (f-g)(t_{h+1} + \xi + \delta)t_{h+1} + \sum_{i=h+f+1}^{\ell} t_i^2.$$  

$$= \zeta \sum_{i=1}^{h} t_i + g(\xi + 1)t_{h+1} + (f-g)(\xi + \delta)t_{h+1} + \sum_{i=h+1}^{\ell} t_i^2$$  

$$= \zeta \sum_{i=1}^{h} t_i + \sum_{i=h+1}^{\ell} t_i^2.$$  

(17)

Since $t_i \leq t_{i+1} + \xi + \delta \leq t_{i+1} + \xi + 1 \leq \zeta$, it follows that (16) is minimized by a sequence in $\mathcal{T}_{t,\ell,\mu}^{(R,\text{ord})}$ with smallest-possible $h$. Among these sequences with minimal $h$, it is minimized by sequence with largest $f$. Since $t_i$ are non-increasing, these requirements directly imply that (17) is minimized for

$$t = \left[ \frac{\ell}{\ell}, \ldots, \frac{\ell}{\ell} \right],$$

for which we have

$$\sum_{i=1}^{\ell} t_i s_i = \frac{\ell}{\ell} \sum_{i=1}^{\ell} s_i = \frac{\ell}{\ell}.$$  

This proves the first claim.

We get the bound on $Q_{t,\ell,\mu}$ by

$$Q_{t,\ell,\mu} = \sum_{t \in \mathcal{T}_{t,\ell,\mu}} q_{t,\xi,\zeta} (t)^{-1} \leq |\mathcal{T}_{t,\ell,\mu}| \max_{t \in \mathcal{T}_{t,\ell,\mu}} q_{t,\xi,\zeta} (t)^{-1} \leq \left( \frac{\ell}{\ell-1} \right)^{\ell} q^{\ell} (\xi + \zeta).$$

D. Computing Bounds on the Success Probability Efficiently

We turn to the question of computing $Q_{t,\ell,\mu}$, as in (15), exactly. Below we give a dynamic-programming algorithm that computes this sum efficiently using a recursion formula. The algorithm is similar to the counting algorithm for the number of vectors of a given sum-rank weight (Algorithm 1), with a major complication: we have

$$q_{t,\xi,\zeta} (t)^{-1} = \prod_{i=1}^{\ell} \frac{\lceil \frac{\zeta}{t_i} \rceil}{s_{\text{comp}}^{(t, s_i)}} q_i,$$

where $s_{\text{comp}}^{(t, s_i)}$ (the $i$-th entry of the vector $s_{\text{comp}}^{(t, s)}$) depends on the entire vector $t$ and not only on $t_i$. Hence, we cannot easily split the product into a part depending only on $t_1$ and one depending only on $t_2, \ldots, t_\ell$. Note, however, that if $t$ is ordered in non-decreasing order and $j$ is such that $t_j > t_{j+1}$, then

$$\prod_{i=1}^{j} \frac{s_{\text{comp}}^{(t, s_i)}}{t_i} q_i.$$  

The outputs are equal if we choose $j \leftarrow \min \{ j : s_j = \max_{i,j \neq j} \{ s_i \} \}$ instead of a random choice in Line 6 of Algorithm 3. Note that in what follows here, the choice of $j$ is irrelevant, so we may w.l.o.g. assume that $j$ is chosen like this.
Furthermore, we define the following two notions:

Lemma 22. For all \( t \in T_{t, \ell, \mu} \), we define

\[
Q_{t, \ell, \mu} = \ell ! \cdot M(t, \ell, \mu, s),
\]

where for any \( t', \ell', \mu', s' \in \mathbb{Z}_{\geq 0} \), we define

\[
M(t', \ell', \mu', s') := \begin{cases} 
\sum_{t \in T_{t', \ell', \mu', s'}} \left( \prod_{i=1}^{\ell'} \delta_i(t) \right) ! \prod_{i=1}^{\ell'} \frac{\left[ \frac{\zeta}{t_i} \right]_q}{t_i} \cdot \delta \in \text{comp}_{\text{ord}(t, s)}(t', s'), & 0 \leq \ell' \leq \min\{s', \ell' \}, \\
1, & \ell' = t' = s' = 0, \\
0, & \text{else.}
\end{cases}
\]

Furthermore, \( M(t', \ell', \mu', s') \) fulfills the following recursive relation.

\[
M(t', \ell', \mu', s') = \min\{\mu', \ell' \} \cdot \max\{\delta : \delta \leq \ell', t_i \delta \leq \ell' \} \left( \prod_{i=1}^{\ell} \frac{\delta}{t_i} \prod_{i=1}^{\ell} \frac{\left[ \frac{\zeta}{t_i} \right]_q}{t_i} \cdot \delta \in \text{comp}_{\text{ord}(t, s)}(t', s'), \{t_1, \ldots, t_{\ell - \ell'} - 1, \delta \} \right) \cdot M(t' - \delta t_1, \ell' - \delta t_1 - 1, s' - \min\{s' - (t' - \delta t_1), \delta \zeta \}).
\]

Proof. Equation (18) holds since, by definition, we have

\[
Q_{t, \ell, \mu} = \sum_{\ell \in T_{t, \ell, \mu}} \theta_{t, \ell, \mu}(t) = \sum_{t \in T_{t, \ell, \mu}} \prod_{i=1}^{\ell} \frac{[\zeta t_i]_q}{t_i}. \quad (18)
\]

Furthermore, the term \( \prod_{i=1}^{\ell} \frac{[\zeta t_i]_q}{t_i} \) is invariant under permutations of \( t \), so we can group these summands into those that belong to a unique sorted vector \( t \in T_{t, \ell, \mu} \). The number of these summands belonging to the same sorted \( t \) equals the number of permutations of \( t \), which is \( \ell ! \cdot \text{ord}(t) \). This proves (18).

The recursion formula is correct by the following argument. The restrictions on the choice of \( t_1 \) and \( \delta \) are as follows (which directly yield the limits of the sums):

- \( t_1 \leq \min\{\mu', \ell' \} \) by definition of \( T_{t', \ell', \mu', s'} \).
- \( t_1 \geq \frac{\ell'}{\ell} \) since for a given \( t_1 \), the entire vector \( t \) may only sum up to at most \( t_1 \ell' \) (since \( \delta \leq \ell' \) and \( t_1 \leq t_1 \)). On the other hand, the entries of the vector must sum to \( \ell' \), which is impossible for \( t_1 \ell' < \ell' \).
- \( 1 \leq \delta \leq \ell' \) since \( t_1 \) may appear between 1 and \( \ell' \) times.
- \( t_1 \delta \leq \ell' \) since \( t \) sums to \( \ell' \) and thus we must have \( t_1 \delta \leq \ell' \).
- \( \delta \geq \ell' - \ell'(t_1 - 1) \) since the remaining entries of \( t \) have values \( \leq t_1 - 1 \) and must nevertheless sum to \( \ell' \). This is only possible for \( (\ell' - \delta)(t_1 - 1) \geq \ell' - \ell'(t_1 - 1) \).

For fixed \( t_1 \) and \( \delta < \ell' \), a vector \( t \in T_{t', \ell', \mu', s'} \) whose first \( \delta \) positions equal \( t_1 \) and whose remaining positions are \( \leq t_1 - 1 \) can be split into two parts \( t = [t^{(1)} \mid t^{(2)}] \), where

\[
t^{(1)} := [t_1, \ldots, t_{\delta}], \quad t^{(2)} := [t_{\delta + 1}, \ldots, t_{\ell'}] \in \{0, \ldots, t_1 - 1\}^{\ell' - \delta}.
\]

In particular, we have

\[
t^{(2)} \in T_{t_{\delta - t_1}, \ell' - \delta, t_1 - 1}.
\]

Hence, we can split up the product

\[
\prod_{i=0}^{\ell'} \delta_i(t) ! = \delta ! \cdot \prod_{i=0}^{t_1 - 1} \delta_i(t^{(2)}) !.
\]
Furthermore, note that by definition of \( \text{scomp} \), we have that \([s^{(1)}, s^{(2)}]\) is a valid output of \( \text{scomp} \), where
\[
s^{(1)} = \text{scomp}_q(\ell^{(1)}), \min\{s' - (t' - \delta t_1), \delta \zeta\}
\]
\[
s^{(2)} = \text{scomp}_q(\ell^{(2)}), s' - \min\{s' - (t' - \delta t_1), \delta \zeta\}.
\]
In particular, \( s^{(1)} \) and \( s^{(2)} \) only depend on \( \ell^{(1)} \) and \( \ell^{(2)} \), respectively, and on the parameters \( s', t_1, \) and \( \delta \). Hence, we can also split the product
\[
\prod_{i=1}^{\ell'} \left[ \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell, s')_{t_1 q}} \right] = \left( \prod_{i=1}^{\ell} \left[ \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell, s')_{t_1 q}} \right] \right) \cdot \left( \prod_{i=1}^{\ell'} \left[ \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell^{(2)}, s' - (t' - \delta t_1), \delta \zeta)_{t_1 q}} \right] \right)
\]
For \( \delta = \ell' \), we have \( \ell = [t_1, \ldots, t_1] \). Hence, we get
\[
\left( \prod_{i=0}^{\ell} \frac{\mu}{\delta_i(t_i)!} \right) \cdot \prod_{i=1}^{\ell'} \left[ \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell, s')_{t_1 q}} \right] = \begin{cases} \frac{1}{\delta_1} \prod_{i=1}^{\ell} \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell, s')_{t_1 q}}, & \text{if } \ell' = \delta t_1 \text{ and } s' \leq \delta \zeta, \\ 0, & \text{else.} \end{cases}
\]
By definition of the base case,
\[
M(t', 0, \mu', s') := \begin{cases} 1, & \text{if } \ell' = 0 \text{ and } s' = 0, \\ 0, & \text{else,} \end{cases}
\]
we get exactly this summand for \( \delta = \ell' \). This proves the recursion.

\[\square\]

**Algorithm 5:** Fill table \( \{M(t', \ell', \mu', s')\}_{1 \leq \mu s' \leq s} \)

**Input:** Integers \( t' \leq t, \ell' \leq \ell, \mu' \leq \mu, s' \leq s \), global table \( \{M(t', \ell', \mu', s')\}_{1 \leq \mu s' \leq s} \), global parameters \( q, \zeta, t, \mu, s \)

**Output:** \( M(t', \ell', \mu', s') \)

1. if \( M(t', \ell', \mu', s') = -1 \) then
2. else if \( \ell' = t' = s' = 0 \) then
3. \( \text{res} \leftarrow 1 \)
4. else if \( \ell' \geq 1 \) and \( 0 \leq \ell' \leq \min\{s', \ell' \mu'\} \) and \( s' \leq \ell' \zeta \) then
5. \( \text{res} \leftarrow 0 \)
6. for \( t_1 = [t'/\ell'], \ldots, \min\{\mu', \ell'\} \) do
7. \( \text{for } \delta = \max\{\ell' - (t_1 + 1), 1\}, \ldots, \max\{\delta : \delta \leq \ell', t_1 \delta \leq t_1\} \) do
8. \( \text{res} \leftarrow \text{res} + M(\ell' - \delta t_1, \ell' - \delta, t_1 - 1, s' - \min\{s' - (t' - \delta t_1), \delta \zeta\}) \cdot \delta^{t_1 - 1} \cdot \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell, s')_{t_1 q}}, \delta \zeta, t_1 \delta \leq s' \leq \zeta \delta. \]
9. else
10. \( \text{res} \leftarrow 0 \)
11. \( \text{M}(t', \ell', \mu', s') \leftarrow \text{res} \)
12. return \( M(t', \ell', \mu', s') \)

**Proposition 23.** If we initialize a table \( \{M(t', \ell', \mu', s')\}_{1 \leq \mu s' \leq s} \) with \( M(t', \ell', \mu', s') = -1 \) for all entries and call Algorithm 5 with input \( t, \ell, \mu, s \), then the algorithm computes the entry \( M(t, \ell, \mu, s) \) in
\[
O^*(t s n^3 \mu^2 \log_2(q))
\]
bit operations. In particular, we can compute \( Q_{\ell, \ell, \mu} \) from (15) in this bit complexity.

**Proof.** The correctness of the algorithm follows from Lemma 22. For the complexity, we observe the following: Lines 2–13 of the algorithm are only once called for each table index \( [t', \ell', \mu', s'] \). The number of table entries, and thus the calls of these expensive lines, is in \( O(ts \mu) \subseteq O(t s n) \). It is negligible compared to the entire recursive call of the algorithm to pre-compute the products \( \delta^{t_1 - 1} \cdot \frac{[\xi]_{t_1 q}}{\text{scomp}_q(\ell, s')_{t_1 q}} \) for all \( 0 \leq \delta \leq \ell, 0 \leq t_1 \leq \min\{t, \mu\}, \) and \( t_1 \delta \leq s' \leq \zeta \delta. \)
The bottleneck of the algorithm is Line 10, where we multiply two rational numbers and add the result to another rational number. All these rational numbers are in $\ell^{-1}\mathbb{Z}$, where

$$ t = \ell! \left( \prod_{t'=0}^{\mu} \prod_{s'=1}^{\zeta} \frac{\left[ s' \right]}{q} \right)^{t} \leq \ell! \mu^\mu \zeta^\zeta \sum_{s'=0}^{\zeta} \sum_{t'=0}^{\mu} \ell^{s'+t'} \leq 2^{\ell \log_2(t+2\mu \zeta^2 \zeta^2 \log_2(q)}.
$$

Hence, we can implement all operations in $\ell^{-1}\mathbb{Z}$, and operations have a quasi-linear cost [38] in the size of the numerators plus the size of $\ell$. Furthermore, the numerator is upper bounded by $\ell Q_{t', \ell, \mu}$, which is again upper bounded by the bound in Proposition 21. Thus, Line 10 costs

$$ O^{\alpha} \left( (\ell - \frac{t}{q}) \log_2(q) + (\ell - 1) \log_2(t + \ell - 1) + \ell \mu^2 \zeta^2 \log_2(q) \right) \leq O^{\alpha} \left( n \mu^2 \zeta^2 \log_2(q) \right)
$$

bit operations.

Since Line 10 is called $O(\ell t)$ $O(n)$ times for each table entry, the overall bit complexity of the entire recursion is

$$ O^{\alpha} \left( t n^2 \mu \zeta^2 \log_2(q) \right),$$

which proves the claim.

### E. Efficiently Drawing Decomposition Vectors

With the help of Algorithm 5 and a bit of extra work, we can draw efficiently from the distribution $p_{t'}$ as in Algorithm 4. The idea of the method (see Algorithm 6 below) is based on enumerative encoding [40]. To formalize the idea, we need the following notation. We denote by $t \leq t'$ for $t, t' \in \mathbb{Z}^\ell$ the lexicographical (total) ordering on $\mathbb{Z}^\ell$. For $t \in \mathbb{Z}^\ell$, $t' \in \mathbb{Z}^\ell$, and $i \leq \min(\ell, \ell')$, we define the preorder $t \leq_i t'$ as

$$ [t_1, \ldots, t_i] \leq [t'_1, \ldots, t'_i].$$

Further, we write $t =_i t'$ if $t \leq_i t'$ and $t' \leq_i t$, as well as $t <_i t'$ if $t \leq_i t'$, but not $t =_i t'$. The following lemma shows how to compute the sum

$$ \sum_{t \in T_{t', \ell, q}} \prod_{t' \in \delta_i(t, t')} P_{q, c, s}(t) = \prod_{t' \in \delta_i(t, t')}^{e_i} \left( \frac{\left[ \zeta \right]}{q} \right) \left( \frac{\left[ 1 \right]}{q} \right) \left( \frac{\left[ \delta \right]}{q} \right)
$$

efficiently, where we only sum over those vectors $t \in T_{t', \ell, q}$ who have a given prefix $t$ of length $\ell'$. This is a key ingredient for the enumerative-coding-based drawing method presented below.

**Lemma 24.** Let $1 \leq \ell' \leq \ell$ and $t \in T_{t', \ell, q}$. Denote by $t_{i'}$ the $i'$-th entry of $t'$ and by $1 \leq \delta \leq \ell'$ the number of times $t_{i'}$ occurs in $t$. Thus, we can split $t$ into

$$ t := \left[ t(1), t(2) \right],$$

where $t(1) \in \{t_{i'}, 1, \ldots, \mu\}^{\ell' - \delta}$ and $t(2) = [t_{\ell'}, \ldots, t_{\ell'}] \in \mathbb{Z}^\delta$. Write $\ell(1) := \sum_{t' \in \delta_i(t, t')}^{e_i} t_i$ and $s(1) := \min\{s - t + 1, \ell' - \delta \zeta\}$. Then,

$$ \sum_{t \in T_{t', \ell, q}} \prod_{t' \in \delta_i(t, t')} P_{q, c, s}(t) = \prod_{t' \in \delta_i(t, t')}^{e_i} \left( \frac{\left[ \zeta \right]}{q} \right) \left( \frac{\left[ 1 \right]}{q} \right) \left( \frac{\left[ \delta \right]}{q} \right)
$$

$$ - \max(\delta': \delta' \leq \ell' + \delta, t_{i'} \delta' \leq t - 1) \left( \frac{1}{\delta'} \prod_{t' \in \ell(1) - (t_{i'} - 1)(\ell' - \delta') \delta', q} \left[ \frac{\left[ \delta \right]}{t_{i'}} \right] \left[ \frac{\left[ \zeta \right]}{t_{i'}} \right] \right)$$

$$ \cdot M \left( t - \delta't_{i'} - t(1), \ell - (\ell' - \delta + \delta'), t_{i'} - 1, s - s(1) - s(2) \left( \delta' \right), q \right),$$

where $s(2)(\delta') := \min\{s - s(1) - (t - \delta't_{i'} - t(1)), \delta' \zeta\}$. and $M(\delta', \ell', \mu', s')$ is defined as in Lemma 22.

In particular, if the table $\{M(\ell', \ell', \mu', s')\}_{t \leq \ell' \leq \ell}$ is pre-computed, we can compute $\sum_{t \in T_{t', \ell, q}} \prod_{t' \in \delta_i(t, t')} P_{q, c, s}(t) = \prod_{t' \in \delta_i(t, t')}^{e_i} \left( \frac{\left[ \zeta \right]}{q} \right) \left( \frac{\left[ 1 \right]}{q} \right) \left( \frac{\left[ \delta \right]}{q} \right)
$$

$$ \cdot M \left( t - \delta't_{i'} - t(1), \ell - (\ell' - \delta + \delta'), t_{i'} - 1, s - s(1) - s(2) \left( \delta' \right), q \right),$$

where $s(2)(\delta') := \min\{s - s(1) - (t - \delta't_{i'} - t(1)), \delta' \zeta\}$ and $M(\delta', \ell', \mu', s')$ is defined as in Lemma 22.
bit operations.

Proof. The statement follows by the same arguments as the recursive formula for \( M(\cdot, \cdot, \cdot) \) in Lemma 22. The only difference is that we split the sum (only) into those subsets of \( \{ \tilde{t} \in \mathcal{T}_{t, \ell, \mu}^{(ord)} : \tilde{t} \equiv_{t'} \tilde{t} \} \) in which the value \( t_{\ell'} \) occurs exactly the same number of times \( \delta' \). Since we know that \( t_{\ell'} \) is contained \( \delta \) times in the last positions of the prefix vector, it must occur \( \delta' \geq \delta \) times in \( \tilde{t} \). Furthermore, \( \delta' \) must be chosen large enough such that

\[
t - t^{(1)} \leq t_{\ell'} \delta' + (t_{\ell'} - 1)(\ell - (\ell' - \delta + \delta'))
\]

which gives the other lower bound (and sum limit) on \( \delta' \). On the other hand, we must have \( \delta' \leq \ell - \ell' + \delta \) since the length of \( \tilde{t} \) is \( \ell \) and the length of \( t^{(1)} \) is \( \ell' - \delta \). After subtracting the sum of the entries \( > t_{\ell'} \) of the prefix vector, the remaining part of the vector \( \tilde{t} \) can only sum up to \( t - t^{(1)} \). In particular, we must have \( t_{\ell'} \delta' \leq t - t^{(1)} \). This gives the upper bound (and sum limit) on \( \delta' \).

The formula follows by splitting the product (w.r.t. \( j \)) in

\[
\prod_{t_{\ell'} = 0}^{t_{\ell'}} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} = \prod_{t_{\ell'} = 0}^{t_{\ell'}} \prod_{j=1}^{\delta'} \left[ \prod_{i=1}^{t_{\ell'}} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \right]_{q}
\]

into the following subsets of positions \( j \):

- the positions of the prefix vector with values \( t_{j} > t_{\ell'} \),
- the part of \( \tilde{t} \) in which \( t_{\ell'} \) is repeated \( \delta' \) times, and
- the remaining part of \( \tilde{t} \).

The sum over the latter part is given by \( M\left(t - \delta t_{\ell'} - t^{(1)}, \ell - \ell' + \delta' - \delta, t_{\ell'} - 1, s - s^{(1)} - s^{(2)}(\delta') \right) \) since this part of \( \tilde{t} \) must sum up to \( t - \delta t_{\ell'} - t^{(1)} \), it is a vector of length \( \ell - (\ell' - \delta + \delta') \), we have \( \tilde{t}_{i} < t_{\ell'} \) for these entries of \( \tilde{t} \). The choices of \( s^{(1)}, s^{(2)}(\delta') \), and \( s - s^{(1)} - s^{(2)}(\delta') \) are to ensure that

\[
\left[ \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \right]_{q} = \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1}
\]

is a valid output of \( \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t}) \) (i.e., independent of \( \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t}) \)'s randomness, we can split the product \( \prod_{i=1}^{t_{\ell'}} \left[ \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \right]_{q} \) into the given three parts).

Complexity-wise, the bottleneck are at most \( \ell \) multiplications and additions of rational numbers in \( \mathbb{Z}_{\ell} \), where \( \ell \) is the same as in the proof of Proposition 23. Hence, computing \( \sum_{t \in \mathcal{T}_{t, \ell, \mu}}^{(ord)} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \) costs \( \mathcal{O}(n \mu \zeta^{2} \log_{2}(q)) \leq \mathcal{O}(n^{2} \zeta^{2} \log_{2}(q)) \) bit operations.

Proposition 25. Algorithm 6 is correct and has complexity

\[
\mathcal{O}(n^{3} \zeta^{2} \log_{2}(q))
\]

bit operations. In particular, Line 1 of Algorithm 4 can be implemented with this complexity.

Proof. Since \( p_{t'} = p_{t''} \) for two vectors \( t' \) and \( t'' \) that are permutationally equivalent, we can simply draw a sorted vector from \( \mathcal{T}_{t, \ell, \mu}^{(ord)} \) using the probability mass function

\[
\tilde{p}_{t'} := \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} = \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t'})^{-1}
\]

for all \( t' \in \mathcal{T}_{t, \ell, \mu}^{(ord)} \) (recall that \( \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \) is the number of permutations of the vector \( t' \)). This is done in Lines 1–7. Subsequently, we randomly permute this vector and obtain a vector that is drawn according to the distribution \( p_{t'} \) (see Lines 8 and 9).

The idea of Lines 1–7 is to partition the interval

\[
\mathcal{I} := \left[ 0, \sum_{t \in \mathcal{T}_{t, \ell, \mu}}^{(ord)} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \right] = \left[ 0, \sum_{t \in \mathcal{T}_{t, \ell, \mu}}^{(ord)} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \right]
\]

into the intervals

\[
\mathcal{I}_{t} := \left[ \sum_{t \in \mathcal{T}_{t, \ell, \mu}}^{(ord)} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1}, \sum_{t \in \mathcal{T}_{t, \ell, \mu}}^{(ord)} \mathcal{D}_{t, \ell, \mu} \mathcal{Q}_{\mathcal{T}, \mathcal{S}}(\tilde{t})^{-1} \right]
\]
Furthermore, we have

Equations (19) and (20) combined prove that Lines 1–7 indeed compute the “index” \( t \) for which \( x \in I_t \). This concludes the correctness proof.
The complexity follows since we need to compute \( \sum_{\mathbf{t} \in \mathcal{T}_{t,\ell,\mu}} \mathbb{E}_{\mathbf{t} \sim \mathbb{P}} \prod_{i=1}^{\mu} \theta_{q,\zeta,s}(\mathbf{t})^{-1} \) for at most \( \ell \mu \leq n \) different vectors \( \mathbf{t} \), and the cost to compute each of these sums from the precomputed table \( \{M_i(t', t', \mu', s')\}_{t' \leq t, \mu' \leq \mu, s' \leq s} \) as derived in Lemma 24. The cost of drawing \( x \) corresponds to drawing uniformly at random a non-negative integer smaller than \( \sum_{\mathbf{t} \in \mathcal{T}_{t,\ell,\mu}} \theta_{q,\zeta,s}(\mathbf{t})^{-1} \), i.e., of bit size \( \in \mathcal{O}(\eta \mu \zeta^2 \log_2(q)) \). This cost, as well as the cost of drawing a random permutation of \( \mathbf{t} \), is negligible.

F. Proof of the Main Statement

The following proof summarizes the statements shown in this section, which all together imply the main statement, Theorem 16.

**Proof of Theorem 16.** First note that for \( \eta \leq m \), the algorithm sets \( \zeta = \eta \) and draws a random row support. For \( \eta > m \), we do the same for the column case. Correctness follows since if a suitable \( e \) exists, there is a non-zero probability that a (row or column) super-support of \( e \) is drawn, and erasure decoding has a unique result for a super-support of weight \( s < d \) (cf. Theorem 13 and Theorem 14).

The expected complexity is given by the product of the cost of one iteration \( W_{\text{iter}} \) (erasure decoding plus random support drawing) and the expected number of iterations. The first value can be lower-bounded by 1 and upper-bounded by \( O(\mu^3 m^3 \log_2(q)) \) due to Theorem 13, Theorem 14, and Proposition 25. The bounds on the expected number of iterations directly follow from Proposition 20 and Proposition 21.

The claim that the bounds (6) and (7) can be computed efficiently follows directly from Proposition 23. \( \square \)

VI. COMPARISON TO OTHER GENERIC DECODERS

We compare the new generic decoder to other (naive) generic decoding strategies, as well as to existing generic decoders in the extreme cases \( \ell = 1 \) (rank metric) and \( \ell = n \) (Hamming metric).

A. Comparison to Extreme Cases: Hamming and Rank Metric

In the Hamming-metric case (\( \ell = n \)), the set \( \mathcal{T}_{t,\ell,\mu} \) consists of all permutations of the vector \( [1, \ldots, 1, 0, \ldots, 0] \), where the number of ones equals \( t \). In particular, we have \( |\mathcal{T}_{t,\ell,\mu}| = \binom{n}{t} \). For \( \mathbf{t} \in \mathcal{T}_{t,\ell,\mu} \) and \( t \leq s \leq n - k \), the function \( \text{scomp}_q(\mathbf{t}, s) \) (Algorithm 3) returns a random vector \( s \in [0, 1]^n \) with exactly \( s \) ones and whose support contains the support of \( \mathbf{t} \). In particular, \( \theta_{q,\zeta,s}(\mathbf{t}) = 1 \) for all \( t \in \mathcal{T}_{t,\ell,\mu} \). Hence, Algorithm 2 uniformly at random selects a subset of \( s \) positions in a vector of length \( n \), and succeeds if and only if these \( s \) positions contain the error positions of an error corresponding to an error \( e' \) with \( r - e' \in C \), where \( r \) is the received word. Although the bounds in Theorem 16 are—as expected—quite bad for this case (we get \( 0 \leq W_{\text{new}} \leq W_{\text{iter}}(\binom{n}{t}) \)), this algorithm equals exactly Prange’s information-set decoder [22], which has expected runtime

\[
W_{\text{Prange}} = W_{\text{iter}}(\frac{\binom{n}{t}}{\binom{n}{s}})
\]

where \( W_{\text{iter}} \) denotes the (polynomial-time) cost of one iteration.

In the rank-metric case (\( \ell = 1 \)), the set \( \mathcal{T}_{t,\ell,\mu} \) contains only one element: \([t] \in \mathbb{Z}^1 \). Algorithm 2 thus chooses uniformly at random a row or column space of dimension \( s \), and row- or column-erasure decodes in the rank metric. This method is exactly the rank-syndrome decoder by Gaborit, Ruatta, and Schrek [26]. The complexity bound (8) in Theorem 16 simplifies to

\[
W_{\text{GRS}} = W_{\text{iter}} q^{(\min(n,m) - s)}
\]

where \( t \leq s \leq \min\{n - k, \lfloor \frac{n}{\mu}(n - k) \rfloor \} \) and \( W_{\text{iter}} \) denotes the (polynomial-time) cost of one iteration. This coincides exactly with Gaborit, Ruatta, and Schrek’s complexity bound.

For arbitrary \( \ell \) and \( t \leq s \leq \min\{n - k, \lfloor \frac{n}{\mu}(n - k) \rfloor \} \), the simple upper complexity bound (8) in Theorem 16 is

\[
W_{\text{new, simple}}^{(\text{UB})} = W_{\text{iter}}(\frac{\ell t + 1 - 1}{\ell - 1}) q^{s(\zeta - 1)} \leq W_{\text{iter}}(\frac{\ell t + 1 - 1}{\ell - 1}) q^{\min(n,\ell m) - s}.
\]

For constant \( \ell \), the factor \((\frac{\ell t + 1 - 1}{\ell - 1}) q^{s(\zeta - 1)} \) is polynomial in the code length, and can be neglected compared to the exponential term. Hence, the exponent of the sum-rank-metric generic decoder is roughly a factor \( \ell \) smaller than in the rank-metric case (\( \ell = 1 \)). Note that the bound \( W_{\text{new}}^{(\text{UB, simple})} \) appears to be a loose approximation of the actual work factor for large \( \ell \) (cf. Figure 2, 3, and 4). Therefore, we refrain from a discussion of \( W_{\text{new}}^{(\text{UB, simple})} \) for \( \ell \in \Omega(n) \) as this does not necessarily give a good intuition about the work factor.

Overall, the new generic decoding algorithm smoothly interpolates two generic decoding principles known for the extreme cases: Prange’s information-set decoder [22] for the Hamming metric and Gaborit, Ruatta, and Schrek’s decoder [26] for the rank metric. The bounds on the work factor in Theorem 16 are, in a rough sense, good for \( \ell \) not too large. For constant \( \ell \), the logarithm of the work factor of our generic \( \ell \)-sum-rank decoder is roughly a factor \( \ell \) smaller than Gaborit, Ruatta, and Schrek’s rank-metric decoder.
B. Comparison to Naive Generic Sum-Rank Decoders

We compare the new generic decoder to other possible generic decoding strategies. One naive strategy for generic decoding is given by brute-forcing the codewords, which has a complexity $W_C = q^{m^2 kn}$, where $m^2 kn$ is the cost of encoding. Another naive approach is brute-forcing the errors with complexity $W_{\text{errors}}$ as in (5) (see Corollary 7 in Section III). For the extreme cases $\ell = 1$ and $\ell = n$, we compare the bounds on the work factor of the new decoder with the Gaborit–Ruatta–Schrek decoder ($W_{\text{GRS}}$) and Prange’s information-set decoder ($W_{\text{Prange}}$), respectively, cf. Section VI-A.

In Figures 2, 3, and 4, we compare the expected complexities of these generic decoding algorithms with the algorithm that we propose. We plot all bounds on the work factor of the new generic decoder that we present in the main statement, Theorem 16, as well as the work factor of the “optimal choice” for $\tilde{p}_s$ as derived in Appendix B. We show the $\log_2$ of the number of operations required.

For all considered parameters, we observe that the difference of the derived upper and lower bound $W_{\text{new}}(\text{LB})$ and $W_{\text{new}}(\text{UB})$ is small, which indicates that the bounds must be tight on the true work factor. Furthermore, for small values of $\ell$, the simplified upper bound $W_{\text{new}}(\text{UB, simple})$ is very close to $W_{\text{new}}(\text{UB})$ and becomes loose only for large values of $\ell$. We note that the optimal solution derived in Appendix B is almost exactly on the accurate upper bound $W_{\text{new}}(\text{UB})$, for all cases in which we can compute it. Further, the work factor of Prange’s algorithm (case $\ell = n$) and the generic rank-metric decoder (case $\ell = 1$) are close to the upper bound $W_{\text{new}}(\text{UB})$.

![Figure 2. Comparison of different generic decoding strategies for $q = 2$, $m = 20$, $n = 60$, $k = 30$, $t = 9$, $s = 10$. The work factor $W_C$ is $2^{620}$ for all values of $\ell$ and $W_{\text{errors}}$ is equal to $2^{661}$ for $\ell = 1$.](image)

VII. FORMAL HARDNESS PROOF

In this section, we formally prove the hardness of the decisional version of the generic decoding problem in the sum-rank metric. We adapt the approach of Gaborit and Zémor [21], who probabilistically reduced the decisional Hamming syndrome decoding problem to the decisional rank syndrome decoding problem over a sufficiently large field extension. We generalize the method from $\ell = 1$ to arbitrary $\ell$, where the size of the extension field can be chosen smaller than in [21] for $\ell > 1$.

A. Complexity Classes

Let $\mathcal{A}$ be an algorithm that gets as input a sequence of random bits $r$ and the input $x$ of a problem. Then $\mathcal{A}$ is called probabilistic polynomial time (PPT) algorithm if the size of the random sequence $|r|$ (number of bits) is polynomial in the input $|x|$ and $\mathcal{A}$ runs in time polynomial in $|x|$.

We make use of the following complexity classes (see, e.g., [41]). Here, $L$ is a decision problem, $0 \leq \Delta < 1$ is any constant:

- $L \in \text{P}$ (polynomial time): there is a PPT algorithm $\mathcal{A}^P$ with output true, false such that $\forall x \in L$ we have $\forall r \; \mathcal{A}^P(x, r) = \text{true}$; and $\forall x \not\in L$ we have $\forall r \; \mathcal{A}^P(x, r) = \text{false}$.
Figure 3. Comparison of different generic decoding strategies for $q = 2$, $m = 60$, $n = 60$, $k = 30$, $t = 10$, $s = 30$, where we chose the row support for all values of $ℓ$ in the proposed algorithm. The work factor $W_C$ is $2^{1823}$ for all values of $ℓ$ and $W_{\text{errors}}$ is equal to $2^{1125}$ for $ℓ = 1$.

Figure 4. Comparison of different generic decoding strategies for $q = 2$, $m = 25$, $n = 60$, $k = 20$, $t = 30$, $s = 30$. The work factor $W_{\text{errors}}$ is equal to $2^{1225}$ for $ℓ = 2$. The case $ℓ = 1$ is not feasible since the condition $t \leq s \leq \min\left\{ n-k, \left\lfloor \frac{m}{n} (n-k) \right\rfloor \right\}$ is not fulfilled.

- $L \in \text{RP}$ (randomized polynomial-time): there is a PPT algorithm $\mathcal{A}^{\text{RP}}$ with output true, false such that $∀x \in L$ then $\Pr[\mathcal{A}^{\text{RP}}(x, r) = \text{true}] \geq Δ$; and $∀x \notin L$ we have $∀r \mathcal{A}^{\text{RP}}(x, r) = \text{false}$. Note that the probability is over the randomness of the bits $r$, and the input $x$ is considered fixed.
- $L \in \text{coRP}$ (co-randomized polynomial-time): there is a PPT algorithm $\mathcal{A}^{\text{coRP}}$ with output true, false such that $∀x \in L$ then $∀r \mathcal{A}^{\text{coRP}}(x, r) = \text{true}$; and $∀x \notin L$ then $\Pr[\mathcal{A}^{\text{coRP}}(x, r) = \text{false}] \geq Δ$.
- $L \in \text{ZPP}$ (zero-error probabilistic polynomial time): there is a PPT algorithm $\mathcal{A}^{\text{ZPP}}$ with output true, false or fail such that the following two are satisfied: 1) For all $x$ then $\Pr[\mathcal{A}^{\text{ZPP}}(x, r) = \text{fail}] \leq Δ$; and 2) for all $x$ and $r$ then $\mathcal{A}^{\text{ZPP}}(x, r) = \text{true} \implies x \in L$ and $\mathcal{A}^{\text{ZPP}}(x, r) = \text{false} \implies x \notin L$. Note that ZPP = RP $∩$ coRP.
- $L \in \text{NP}$ (non-deterministic polynomial time): there is a PPT algorithm $\mathcal{A}^{\text{NP}}$ such that $x \in L$ exactly when there exists an
Then, the probability that 

\[ \text{wt}_X C. \]

\[ \text{Hardness Reduction} \]

\[ (\text{Decisional Hamming Syndrome Decoding} (\text{SynDec}_{\text{h}})) \text{ Problem).} \]

\[ \text{Given:} \]

\[ \cdot \text{Parity-check matrix } H \in \mathbb{F}_q^{(n-k) \times n} \text{ of a code } C \]

\[ \cdot \text{Syndrome } s \in \mathbb{F}_q^{n-k} \]

\[ \cdot \text{Integer } 0 \leq t \leq n \]

\[ \text{Question: Is there an } e \in \mathbb{F}_q^n \text{ with } \text{wt}_H(e) \leq t \text{ such that } s = eH^T? \]

The SynDec\textsubscript{h} problem was proven to be NP-complete in [20].

\[ \text{Problem 27 (Decisional } \ell\text{-Sum-Rank Syndrome Decoding (SynDec}_{\ell-\text{SR}})) \text{ Problem).} \]

\[ \text{Given:} \]

\[ \cdot \text{Parameter } \ell \mid n \]

\[ \cdot \text{Parity-check matrix } H \in \mathbb{F}_q^{(n-k) \times n} \text{ of a code } C \]

\[ \cdot \text{Syndrome } s \in \mathbb{F}_q^{n-k} \]

\[ \cdot \text{Integer } 0 \leq t \leq n \]

\[ \text{Question: Is there an } e \in \mathbb{F}_q^n \text{ with } \text{wt}_{\text{SR},\ell}(e) \leq t \text{ such that } s = eH^T? \]

Note that the SynDec\textsubscript{n-\text{SR}} problem and the SynDec\textsubscript{h} problem are the same. The SynDec\textsubscript{1-\text{SR}} problem is the decisional rank-syndrome decoding problem, which was shown to be hard in the following way [21]: If the SynDec\textsubscript{1-\text{SR}} problem is in \( \text{ZPP} = \text{RP} \cap \text{coRP} \), then NP = ZPP. The next subsection generalizes this statement to arbitrary \( \ell \).

\[ \text{C. Hardness Reduction} \]

The following statements constitute the formal hardness proof, which is summarized in Theorem 31. The proof strategy is similar to the proof of the probabilistic reduction of the “decisional minimum rank weight problem” in [21] (note that Gaborit and Zémor prove the reduction for the Sum-Rank Syndrome Decoding problem and Zémor prove the reduction for the “decisional minimum rank weight problem” in [21]). Compared to the original statement in the case \( \ell = 1 \), we can improve the tightness of the reduction (for \( \ell > 1 \)) using the bound on the sum-rank-metric sphere size derived in Theorem 5 in Section III. We start with a technical lemma, which we will use to bound some probabilities in our probabilistic reductions.

\[ \text{Lemma 28. Let } \varepsilon > 0 \text{ be fixed and } m, n, \ell \text{ be positive integers with } m \geq n^2 + n \log_q(8n) + \log_q(2\varepsilon^{-1}). \text{ Let } H \in \mathbb{F}_q^{(n-k) \times n}, \]

\[ s \in \mathbb{F}_q^{n-k} \text{ and } x \in \mathbb{F}_q^n, \text{ where } x \text{ is a vector of minimum Hamming weight } \text{wt}_H \text{ such that } xH^T = s. \text{ Further let } \beta \text{ be chosen uniformly at random from } (\mathbb{F}_q^{n-k})^n \text{ and let then } x' \in \mathbb{F}_q^n \text{ be a vector of minimum sum-rank weight such that } x'(H \text{ diag}(\beta))^T = s. \text{ Then, the probability that } \text{wt}_{\text{SR},\ell}(x') < t_H \text{ is at most } \varepsilon.} \]

\[ \text{Proof: Let } H, s \text{ and } t_H \text{ be fixed. We define } P \text{ as the probability (randomness in } \beta) \]

\[ P := \Pr \left\{ \exists x' \in \mathbb{F}_q^n : x'(H \text{ diag}(\beta))^T = s \land \text{wt}_{\text{SR},\ell}(x') < t_H \right\}. \]

For randomly chosen \( \beta \overset{\$}{\leftarrow} (\mathbb{F}_q^{n-k})^n \), let \( \mathcal{E}_a \) be the event that for a fixed vector \( a \in \mathbb{F}_q^n \), the equality \( a(H \text{ diag}(\beta))^T = s \) holds. Further define the set \( \mathcal{X}(t_H - 1) := \{ a \in \mathbb{F}_q^n : \text{wt}_{\text{SR},\ell}(a) < t_H \} \). Then,

\[ \Pr \left\{ \exists x' \in \mathbb{F}_q^n : x'(H \text{ diag}(\beta))^T = s \land \text{wt}_{\text{SR},\ell}(x') < t_H \right\} = \Pr \left[ \bigcup_{x' \in \mathcal{X}(t_H)} \mathcal{E}_{x'}^c \right] \leq \sum_{x' \in \mathcal{X}(t_H)} \Pr[\mathcal{E}_{x'}]. \]

Next, we bound \( \Pr[\mathcal{E}_{x'}] \) for a given \( x' \in \mathbb{F}_q^n \). If there exists a \( \beta \in (\mathbb{F}_q^{n-k})^n \) such that \( x'(H \text{ diag}(\beta))^T = s \), then, by [21, Lemma 4], there exists a subset \( \mathcal{W} \subseteq \{ i : x'_i \neq 0 \} \) of cardinality \( |\mathcal{W}| = t_H \) such that the columns \( h_i \) of \( H \) indexed by \( i \in \mathcal{W} \) are linearly independent. Hence, if we fix \( \beta_i \) for all \( i \notin \mathcal{W} \), then the set of vectors \( x'(H \text{ diag}(\beta))^T \) obtained by choosing the remaining \( \beta_i \in \mathbb{F}_q \) for \( i \in \mathcal{W} \) has cardinality \( (q^m - 1)^{|\mathcal{W}|} \). Hence, for \( \beta \overset{\$}{\leftarrow} (\mathbb{F}_q^{n-k})^n \), we have

\[ \Pr[\mathcal{E}_{x'}] \leq \frac{1}{(q^m - 1)^{|\mathcal{W}|}}. \]
Otherwise, if there is no \( \beta \in (\mathbb{F}_q^m)^n \) such that \( x' (H \text{ diag} (\beta))' = s \), then we have \( \Pr [\mathcal{E}_{x'}] = 0 \), which is obviously \( \leq \frac{1}{(q^m - 1)^n} \).

Define \( \Gamma(q, m, t_H) := \frac{mq^m}{(q^m - 1)^m} = \frac{1}{(1 - q^{-m})^m} \). Since \( m \geq t_H \) by assumption, we have

\[
\Gamma(q, m, t_H) \leq \frac{1}{(1 - q^{-m})^m} = \frac{1}{\sum_{i=0}^{m-1} \left( \frac{m}{i} \right) (-q^{-m})^i} \leq \frac{1}{1 - mq^{-m}} \leq 2,
\]

where we use \( mq^{-m} \leq \frac{1}{2} \), and \((*)\) follows from the fact that the terms in the sum in the second line are alternating and their absolute values are strictly monotonically decreasing, i.e.,

\[
\left( \frac{m}{i} \right) q^{-(i+1)m} = \frac{q^{-m} m^{-1}}{i+1} \left( \frac{m}{i} \right) q^{-im} \leq \frac{1}{2} \left( \frac{m}{i} \right) q^{-im}.
\]

Combining the arguments above, we get

\[
P \leq \frac{1}{(q^m - 1)^n} |X(t_H - 1)|
= \Gamma(q, m, t_H) \frac{1}{q^m} \sum_{i=1}^{t_H-1} \mathcal{N}_{q,m}(i, \ell)
\leq 2 \frac{1}{q^m} (t_H - 1) \max_{i \in [1, t_H-1]} \mathcal{N}_{q,m}(i, \ell)
\leq 2 \frac{1}{q^m} (t_H - 1) \left( \frac{\ell + t_H - 2}{\ell - 1} \right)^{\ell} q^{-(t_H - 1)q^{-m}(\ell - 1)^2}
\leq 2 \left( \frac{t_H - 1}{\ell + t_H - 2} \right)^{\ell} q^{-(t_H - 1)q^{-m}(\ell - 1)^2}
\leq 2 \left( \frac{t_H - 1}{\ell + t_H - 2} \right)^{\ell-1} 4^{\ell} q^{-(t_H - 1)q^{-m}(\ell - 1)^2}
\leq 2 \left( \frac{t_H - 1}{\ell + t_H - 2} \right)^{\ell-1} 4^{\ell} q^{-(t_H - 1)q^{-m}(\ell - 1)^2}
\leq 2 [4(\ell + t_H - 2)]^{\ell} q^{-(t_H - 1)q^{-m}(\ell - 1)^2}
\leq 2q^{m + \frac{\ell}{r} - \frac{(t_H - 1)^2}{r^2} + 4 log_q (4(\ell + t_H - 2))}
\leq 2q^{m + \frac{\ell}{r} - \frac{(t_H - 1)^2}{r^2} + \ell log_q (4(\ell + t_H - 2))}
\leq 2q^{m + \frac{\ell}{r} + n log_q (8n)}
\leq e.
\]

We first show, that if there is a coRP-algorithm for SynDec_{\ell-SR}, then we can make a coRP-algorithm for SynDec_{H} (Algorithm 7 below). The idea is simple: The algorithm transforms the input into an instance of the SynDec_{\ell-SR} problem via a random linear map, and simply calls the coRP-algorithm for SynDec_{\ell-SR}. Using Lemma 28, we can show that the solution to this problem will usually project back to a solution to the SynDec_{\ell-SR} instance.

**Lemma 29.** For any \( \ell < n \) and \( m > \frac{n^2}{\ell^2} + n \log_q (\ell + 8n) + \log_q (2) \), if the SynDec_{\ell-SR} problem is in coRP, then the SynDec_{H} problem is in coRP.

**Proof:** Let \( \mathcal{A}_{\text{coRP}}^{\text{SR}} \) be a hypothesised coRP-algorithm for the SynDec_{\ell-SR} problem, i.e. it inputs an instance \( (H' \in \mathbb{F}_{q^m}^{(n-k) \times n}, s \in \mathbb{F}_{q^m}^n, t \in \mathbb{Z}_{\geq 0}) \) and outputs true whenever \( t_{SR} \leq t \), while it outputs false with probability at least \( 1 - \varepsilon \) if \( t_{SR} > t \), where \( t_{SR} \) is the minimum sum-rank weight of the vectors \( x' \in \mathbb{F}_{q^m}^n \) such that \( x' H'^T = s \), and \( \varepsilon \geq 0 \) is some fixed constant. Then Algorithm 7 details an coRP-algorithm \( \mathcal{A}_{\text{coRP}}^{\text{H}} \) for the SynDec_{H} problem that inputs an instance \( (H \in \mathbb{F}_{q^m}^{(n-k) \times n}, s \in \mathbb{F}_{q^m}^n, t \in \mathbb{Z}_{\geq 0}) \). We should show that \( \mathcal{A}_{\text{coRP}}^{\text{H}} \) outputs true whenever \( t_H \leq t \), while it outputs false with at least some non-zero constant probability if \( t_H > t \), where \( t_H \) denotes the minimum Hamming weight of the vectors \( x \in \mathbb{F}_{q^m}^n \) such that \( x H^T = s \).

Observe first that if \( t_{SR} \leq t \), it follows that \( t_{SR} \leq t_{H} \), so \( \mathcal{A}_{\text{coRP}}^{\text{H}} \) outputs true. Consider now the case \( t_{SR} > t \). By the definition of \( m \), we may choose a non-negative constant \( \varepsilon < 1 \) such that \( m \geq \frac{n^2}{\ell^2} + n \log_q (\ell + 8n) + \log_q (2\varepsilon^{-1}) \). Hence by Lemma 28, with probability \( \geq 1 - \varepsilon \), we have \( t_{SR} = t_H > t \), and so \( \mathcal{A}_{\text{coRP}}^{\text{H}} \) outputs false with probability at least \( (1 - \varepsilon)(1 - \varepsilon) \), which is again a constant.
Algorithm 7: $A_{\text{SynDec}}^{\text{RP}}$

Input: $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, integer $t$
Output: true or false
1. $\beta \leftarrow (\mathbb{F}_q^n)^n$
2. $H' \leftarrow H \text{diag}(\beta) \in \mathbb{F}_q^{(n-k) \times n}$
3. return $A_{\text{SynDec}}^{\text{RP}}(H', s)$

The RP reduction is more involved: in order to solve the SynDec$_H$ problem, we solve its related search problem by calling a (hypothetical) RP-algorithm for SynDec$_{\ell-SR}$ at most $n$ times on certain punctured and randomly transformed parity-check matrices.

Lemma 30. For any $\ell < n$ and $m \geq \frac{n^2}{\ell} + n \log_q(8n) + \log_q(4n)$, if the SynDec$_{\ell-SR}$ problem is in RP, then the SynDec$_H$ problem is in RP.

Proof: Let $A_{\text{SynDec}}^{\text{RP}}$ be a hypothesised RP-algorithm for SynDec$_{\ell-SR}$, i.e., it inputs an instance $(H' \in \mathbb{F}_q^{(n-k) \times n'}, s \in \mathbb{F}_q^{n-k}, t \in \mathbb{Z}_{>0})$, and outputs false whenever $t_{SR} > t$ and outputs true with probability $1 - \varepsilon$ if $t_{SR} \leq t$, where $t_{SR}$ is the minimum sum-rank weight of the vectors $x' \in \mathbb{F}_q^{n'}$ such that $x'H'^\top = s$, and $\varepsilon > 0$ is some constant smaller than 1. By iterating $A_{\text{SynDec}}^{\text{RP}}$ at most $O(\log n)$ times, we may assume $\varepsilon < \frac{1}{2n}$.

Then Algorithm 8 details an RP-algorithm $A_{\text{H}}^{\text{RP}}$ for the SynDec$_H$ problem that inputs an instance $(H \in \mathbb{F}_q^{(n-k) \times n}, s \in \mathbb{F}_q^{n-k}, t \in \mathbb{Z}_{>0})$. We should show that $A_{\text{H}}^{\text{RP}}$ outputs false whenever $t_{H} > t$, while it outputs true with at least some constant non-zero probability if $t_{H} \leq t$, where $t_{H}$ denotes the minimum Hamming weight of the vectors $x \in \mathbb{F}_q^n$ such that $xH^\top = s$.

The idea of the algorithm is to determine a Hamming super-support $S$ of cardinality at most $t$ of a vector $x \in \mathbb{F}_q^n$ such that $xH^\top = s$. The function $\text{KeepCols}(H, T)$ returns the sub-matrix of $H$ consisting of the columns indexed by the index set $T$. Note that each line runs in polynomial time: in particular, Line 9 is simply solving a linear system. From Lines 9–12, we observe that the algorithm outputs true whenever a super-support is found, and outputs false otherwise. Hence $A_{\text{H}}^{\text{RP}}$ outputs false whenever $t_{H} > t$, and we need to show that if $t_{H} \leq t$ then $A_{\text{H}}^{\text{RP}}$ returns true with some non-zero constant probability.

So assume $t_{H} \leq t$. The purpose of Lines 3–7 is to answer the following question:

(Q) Is $S \setminus \{i\}$ a super-support of a vector $x \in \mathbb{F}_q$ with Hamming weight $w_t(x) \leq t$ and syndrome $s = xH^\top$?

Since we start with $S = \{1, \ldots, n\}$, it is clear that if we always get the correct answer to this question, at termination, the set $S$ will be the support of a vector $x$ of Hamming weight $w_t(x) \leq t$ and syndrome $s = xH^\top$. If we get an incorrect answer in only one of the $\leq n$ loops, then we are not guaranteed that $S$ has this property, but we can detect this event by Lines 9–12.

We show that the probability that Lines 3–7 answer the question (Q) correctly in all iterations of the loop is at least a constant. Note that there are two types of randomness in these lines, which both can influence the answer that we get: the choice of $\beta$ and the randomness in the algorithm $A_{\text{H}}^{\text{RP}}$. We distinguish two cases and denote for given $s \in \mathbb{F}_q^{n-k}$, $i, \beta \in \mathbb{F}_q^{[m]}$, the smallest Hamming weight of a vector $\tilde{x} \in \mathbb{F}_q^{[m]}$ such that $\tilde{x}H = s$ by $t_{H}$ and the smallest $t$-sum-rank weight of a vector $\tilde{x} \in \mathbb{F}_q^{[m]}$ with $\tilde{x}H' = s$ as $t_{SR}$.

Note that the answer to (Q) is true if and only if $t_{H} \leq t$.

- Case 1: The answer to (Q) is true (i.e., $t_{H} \leq t$): Independent of how $\beta$ is chosen, we always have $t_{SR} \leq t_{H} \leq t$, so $A_{\text{SynDec}}^{\text{RP}}(H', s, t)$ returns true (the correct answer) with probability at least $1 - \varepsilon > 1 - \frac{1}{2n}$ (randomness in $A_{\text{H}}^{\text{RP}}$) and false (the incorrect answer) with probability at most $\varepsilon < \frac{1}{2n}$.

- Case 2: The answer to (Q) is false (i.e., $t_{H} > t$):

  - With probability $1 - \frac{1}{2n}$ (randomness in the choice of $\beta$), the vector $\beta$ is chosen such that $t_{SR} = t_{H}$ due to Lemma 28 so where we set $\varepsilon = \frac{1}{2n}$, which is permissible with our restriction on $m$. In this case, we thus have $t_{SR} > t$, and $A_{\text{SynDec}}^{\text{RP}}(H', s, t)$ returns false (the true answer).

  - The counter-event of the above occurs with probability $< \frac{1}{2n}$: the vector $\beta$ is chosen such that $t_{SR} \leq t_{H}$. In this case, $A_{\text{SynDec}}^{\text{RP}}(H', s, t)$ may return true (the wrong answer) or false (the correct answer).

Hence, in both cases, Lines 3–7 answer the question (Q) correctly with probability greater than $1 - \frac{1}{2n}$. Since the question is asked at most $n$ times, we get the correct answer to (Q) in all iterations with probability at least $1 - \frac{n}{2n} = \frac{1}{2}$ by the union bound.

The lemmas above imply the main statement of this section.

Theorem 31. For $\ell < n$ and $m \geq \frac{n^2}{\ell} + n \log_q(8n) + \log_q(4n)$, if the SynDec$_{\ell-SR}$ problem is in ZPP, then we have NP = ZPP.

Proof. It is well-known that ZPP $\subseteq$ NP. The other inclusion, ZPP $\supseteq$ NP, follows from the NP-hardness of SynDec$_H$, Lemma 29, and Lemma 30.

\qed
Algorithm 8: $A_{RP}^\ell$

Input : $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, integer $t$
Output : true or false
1 $S = \{1, \ldots, n\}$
2 for $i = 1, \ldots, n$ do
3     $H \leftarrow \text{KeepCols}(H, S \setminus \{i\}) \in \mathbb{F}_q^{(n-k)\times |S|-1}$
4     $\beta \leftarrow \bar{\text{diag}}(\beta) \in \mathbb{F}_q^{(n-k)\times |S|-1}$
5     $H' \leftarrow H \beta \in \mathbb{F}_q^{(n-k)\times |S|-1}$
6     if $A_{RP}^\ell(H', s, t) = \text{true}$ then
7         $S \leftarrow S \setminus \{i\}$
8     $H \leftarrow \text{KeepCols}(H, S) \in \mathbb{F}_q^{(n-k)\times |S|}$
9     if $1 \leq |S| \leq t$ and $\exists x \in \mathbb{F}_q^{|S|}$ s.t. $xH^\top = s$ then
10        return true
11     else
12        return false

Remark 32. In the special case of Theorem 31 for the rank metric $(\ell = 1)$, which was shown in [21], the restriction on the extension degree is $m > n^2$. It can be seen that our assumption, $m \geq \frac{n^2}{\ell} + n \log_q(8n) + \log_q(4n)$, is less restrictive for $\ell > 1$. An interesting special case is $\ell \in \Omega(n)$, i.e. a sum-rank metric close to the Hamming metric, for which we can choose $m \in O(n \log(n))$.

VIII. Conclusion

We have proposed the first generic decoder in the $(\ell)$-sum-rank metric, which combines known generic decoding algorithms in the Hamming metric $(\ell = n)$ and rank metric $(\ell = 1)$. For $\ell = n$, the algorithm resembles the information-set decoder by Prange [22] and for $\ell = 1$, it coincides with the generic decoder for the rank metric by Gaborit, Ruatta, and Schrek [26].

We have derived lower and upper bounds on the runtime of our generic decoding algorithm, which can be computed in small-degree polynomial time in the code parameters. Furthermore, we derived a simple upper bound on the complexity of the new generic decoding algorithm. For a constant number of blocks $\ell$, the bound shows that the exponent of our algorithm’s work factor is roughly a factor $\ell$ smaller than for the generic rank-metric decoder by Gaborit, Ruatta, and Schrek [26]. Our formal hardness proof in Section VII extends a result by Gaborit and Zémor [21] from the rank metric, and provides evidence that generic decoding in the sum-rank metric is a hard problem.

Besides being of theoretical interest, the results open up the possibility to study sum-rank-metric codes in code-based cryptosystems. We have also derived results on the cardinality of sum-rank-metric spheres, which can, among others, be used to efficiently compute bounds on code parameters (cf. Remark 9). Furthermore, the new notion of column support and the erasure decoding algorithms can be of more general interest.

The article can be seen as a proof-of-concept that ideas for generic decoding in the extreme cases, Hamming and rank metric, can be adapted to the family of sum-rank metrics. An obvious open problem is the study of the many improvements of [22] in the Hamming and [26] in the rank metric. In particular, it would be interesting to adapt the very recent significant improvement of generic decoding in the rank metric based on algebraic methods [28] to the sum-rank metric. As for the rank metric, it is an open problem whether there is a deterministic reduction from an NP-hard problem to the decisional sum-rank syndrome decoding problem.

References

[1] S. Puchinger, J. Renner, and J. Rosenkilde, “Generic decoding in the sum-rank metric,” in IEEE International Symposium on Information Theory (ISIT), 2020.
[2] R. W. Nóbrega and B. F. Uchoa-Filho, “Multishot codes for network coding using rank-metric codes,” in IEEE International Workshop on Wireless Network Coding, 2010.
[3] A. Wachter, V. R. Sidorenko, M. Bossert, and V. V. Zyablov, “On (partial) unit memory codes based on gabidulin codes,” Problems of Information Transmission, vol. 47, no. 2, pp. 117–129, 2011.
[4] A. Wachter-Zeh and V. Sidorenko, “Rank metric convolutional codes for random linear network coding,” in IEEE International Symposium on Network Coding (NetCod), 2012.
[5] A. Wachter-Zeh, M. Stinner, and V. Sidorenko, “Convolutional codes in rank metric with application to random network coding,” IEEE Transactions on Information Theory, vol. 61, no. 6, pp. 3199–3213, 2015.
[6] D. Napp, R. Pinto, J. Rosenthal, and P. Vettori, “MRD rank metric convolutional codes,” in IEEE International Symposium on Information Theory (ISIT), 2017.
[7] ——, “Faster decoding of rank metric convolutional codes,” in 23rd International Symposium on Mathematical Theory of Networks and Systems, 2018.
Proposition 33. Let \( q, m, k, n, t \) be integers such that \( t \mid n \) and \( t \leq \mu \ell \). Then, Algorithm 9 outputs a vector \( e \in \mathbb{F}_q^n \) drawn uniformly at random from \( \{ e' \in \mathbb{F}_q^n : \text{wt}_S(e') = t \} \).

Proof. The set \( \{ e' \in \mathbb{F}_q^n : \text{wt}_S(e') = t \} \) has cardinality \( \mathcal{N}_{q,n,m}(t, \ell) \). Let \( \varphi : \{1, \ldots, \mathcal{N}_{q,n,m}(t, \ell)\} \to \{ e' \in \mathbb{F}_q^n : \text{wt}_S(e') = t \} \) be a bijective mapping. If we know an efficient algorithm to realize the mapping \( \varphi \), then the drawing could
be simply realized by choosing uniformly at random \( D^{(1)} \) from \( \{1, \ldots, \mathcal{N}_{q,n,m}(t, \ell)\} \) and outputting \( \varphi(D^{(1)}) \). However, the drawing algorithm can also be realized with a different method.

Let \( \phi : \{e \in \mathbb{F}_q^n : \text{wt}_{\text{SR}, \ell}(e) = t\} \mapsto T_{t,\ell,\mu} \), \( e \mapsto [\text{rk}_\varphi(e_1), \ldots, \text{rk}_\varphi(e_t)] \). Then, the drawing can be conducted by computing \( t = (\phi \circ \varphi)(D^{(1)}) \) and sampling \( e_j \sim \{a \in \mathbb{F}_q^n :\text{rk}_\varphi(a) = t_j\} \) and \( B_j \sim \{B \in \mathbb{F}_q^{j \times \eta} : \text{rk}_\varphi(B) = t_j\} \), for \( j \in \{1, \ldots, \ell\} \). Since \( e_j = a_j B_j \in \mathbb{F}_q^n \) is a vector drawn uniformly at random from \( \{e' \in \mathbb{F}_q^n : \text{rk}_\varphi(e') = t_j\} \), it follows that \( e = [a_1 B_1 \mid \cdots \mid a_\ell B_\ell] \) is a vector drawn uniformly at random from \( \{e' \in \mathbb{F}_q^n : \text{ws}_{\text{SR}}(e') = t\} \).

To derive the theorem reformulates the optimization problem into a linear programming instance. The following section presents a scalable method to design an optimal support-drawing algorithm. In Lines 7 to 10, the vectors \( e_j \in \mathbb{F}_q^n \) are drawn uniformly at random from the set of vectors of rank weight \( t_j \), and the vector \( [e_1 | \ldots | e_\ell] \) is returned.

**B. Optimal Support-Drawing Algorithm**

In Section V, we saw that the worst-case expected number of iterations of a super-support drawing algorithm that first draws a vector \( s \in T_{s,\ell,\mu} \), according to a probability distribution \( \tilde{\varphi}_s \) and then \( F \sim \Xi_{q,\xi}(s) \), can be given as (cf. (11))

\[
\max_{e \in \mathbb{F}_q^n} E[\text{#iterations}] = \max_{t \in T_{t,\ell,\mu}} \left( \sum_{s \in T_{s,\ell,\mu}} \tilde{\varphi}_s \theta_{q,\xi}(s, t) \right)^{-1}.
\]

Section V presented a scalable method to design \( \tilde{\varphi}_s \) that can be implemented in polynomial time, but does not guarantee to minimize (11). Here, we show how to achieve an optimal solution, at the cost of a super-polynomial complexity. The following theorem reformulates the optimization problem into a linear programming instance.
Figure 5. Illustration of the mapping $\phi \circ \varphi : \{1, \ldots, N_{\eta, \eta, m}(t, \ell)\} \rightarrow T_{t, t, \mu}, D^{(1)} \mapsto t$. The variables are defined as in Algorithm 9, and the function $D''(t', j) := \sum_{t=0}^{t''-1} NM_T (m, \eta, t') N_{\eta, \eta, m}(t'' - t', \ell - j)$.

**Theorem 34.** Fix arbitrary orders $s_1, \ldots, s_{|T_{s, t, \mu}|}$ and $t_1, \ldots, t_{|T_{t, s, \mu}|}$ of elements in $T_{s, t, \mu}$ and $T_{t, s, \mu}$, respectively. Let

$$
A = \begin{bmatrix}
-\varrho_{q, \zeta}(s_1, t_1) & -\varrho_{q, \zeta}(s_2, t_1) & \cdots & -\varrho_{q, \zeta}(s_{|T_{t, s, \mu}|}, t_1) & 1 \\
-\varrho_{q, \zeta}(s_1, t_2) & -\varrho_{q, \zeta}(s_2, t_2) & \cdots & -\varrho_{q, \zeta}(s_{|T_{t, s, \mu}|}, t_2) & 1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
-\varrho_{q, \zeta}(s_1, t_{|T_{t, s, \mu}|}) & -\varrho_{q, \zeta}(s_2, t_{|T_{t, s, \mu}|}) & \cdots & -\varrho_{q, \zeta}(s_{|T_{t, s, \mu}|}, t_{|T_{t, s, \mu}|}) & 1 \\
1 & 1 & \cdots & 1 & 0
\end{bmatrix} \in \mathbb{R}^{(|T_{t, s, \mu}|+2) \times (|T_{s, t, \mu}|+1)}.
$$

If $x \in \mathbb{R}^{(|T_{s, t, \mu}|+1) \times 1}$ is a solution to the linear program

Maximize $c^\top x$

subject to $Ax \leq b$

and $x \geq 0$,

then $\bar{p}_i = x_i$, for all $i = 1, \ldots, |T_{s, t, \mu}|$, is a distribution that maximizes (11), and we have

$$
x_i^{-1} = \min \left\{ \max_{s \in T_{s, t, \mu}} \left( \sum_{a \in T_{t, s, \mu}} \bar{p}_a \varrho_{q, \zeta}(s, t) \right) \right\}^{-1} : \bar{p}_a \in [0, 1] \forall s \in T_{s, t, \mu}, \sum_{a \in T_{t, s, \mu}} \bar{p}_a = 1.
$$

**Proof.** We write $\bar{p}_{s_i} = x_i$ and $\xi = x_i^{-1}$ for a solution $x$ of the linear program. The last two rows of $A$ are equivalent to

$$
\sum_{i=1}^{\left| T_{t, s, \mu} \right|} \bar{p}_{s_i} = 1,
$$
Together with $x \geq 0$, we get that the $\tilde{p}_{s_i}$ form a valid discrete probability mass function. The first $|T_{t,\ell,\mu}|$ rows of $A$ correspond to the constraints

$$\sum_{i=1}^{T_{t,\ell,\mu}} \tilde{p}_{s_i,0,q,\zeta} (s_i, t_j) \geq \xi \quad \forall j = 1, \ldots, |T_{t,\ell,\mu}|.$$  

Since $\xi$ is the maximal positive value for which this constraint is fulfilled for all $j = 1, \ldots, |T_{t,\ell,\mu}|$ and solutions $\tilde{p}_{s_i}$, we have

$$\xi = \max \left\{ \min_{j=1,\ldots,|T_{t,\ell,\mu}|} \left\{ \sum_{i=1}^{T_{t,\ell,\mu}} \tilde{p}_{s_i,0,q,\zeta} (s_i, t_j) : \tilde{p}_{s_i} \in [0,1] \forall i = 1, \ldots, |T_{t,\ell,\mu}|, \sum_{i=1}^{T_{t,\ell,\mu}} \tilde{p}_{s_i} = 1 \right\} \right\}$$

which is equivalent to (22). This proves the claim.

Using standard methods, the linear program (21) in Theorem 34 can be solved in polynomial time in the number of variables, $|T_{s,\ell,\mu}| + 1$ (note that the number of linear constraints is in $O(|T_{s,\ell,\mu}|)$). As, depending on the relative growth of $s, \mu$, and $\ell$, this number may grow super-polynomially in $s$, it is usually not possible to solve the linear program efficiently for large code parameters. Furthermore, even if a solution $x$ is found or pre-computed, it is not apparent how to draw efficiently from the distribution $\tilde{p}_{s_i} = x_i$ for all $i = 1, \ldots, |T_{s,\ell,\mu}|$.

Nevertheless, we include this “optimal” solution to the design of $\tilde{p}_{s_i}$ in the discussion in Section VI for all values of $\ell, \mu, s$ for which we can retrieve a solution in short time (and ignore the issue of efficient drawing). For these computations, we apply a trick that reduces the number of variables and constraints: We assume that the restriction to those solutions $x$ such that $x_i = x_j$ for all $i, j$ with permutationally equivalent $s_i \sim s_j$. Hence, we can reduce the number of variables to $|T_{s,\ell,\mu}| + 1$ (which may still be super-polynomially in $s$, though) and the number of constraints to $|T_{s,\ell,\mu}^{(\text{ord})}| + 2 \leq |T_{s,\ell,\mu}^{(\text{ord})}| + 2$. The complexity of this generic decoding approach is roughly given by

$$W_{\text{optimal}} := W_{\text{iter}} x_{|T_{s,\ell,\mu}|+1}^{-1},$$

where $x_1, \ldots, x_{|T_{s,\ell,\mu}|+1}$ is a solution vector to the optimization problem in Theorem 34 and $W_{\text{iter}}$ is the cost of one iteration. The latter value is at least the cost of erasure decoding, which is in $O^n(r^3m^3 \log(q))$, but the real cost might be larger since we need to be able to efficiently draw from the distribution $\tilde{p}_{s_i}$. In the plots in Section VI, we use the same $W_{\text{iter}}$ as for the other algorithms, which is an optimistic estimate.

It can be seen that in all cases in which we can compute the expected runtime of a generic decoder that draws $s$ according to such an optimal distribution $\tilde{p}_{s_i}$, the “optimal” runtime is only insignificantly smaller than the practical solution presented in Section V.