Electronic Colloquium on Computational Complexity, Report No. 18 (2024)

# The more the merrier! On the complexity of finding multicollisions, with connections to codes and lattices

Huck Bennett* Surendra Ghentiyala† Noah Stephens-Davidowitz‡

**Abstract**

We study the problem of finding *multicollisions*, that is, the total search problem in which the input is a function $\mathcal{C} : [A] \to [B]$ (represented as a circuit) and the goal is to find $L \leq \lceil A/B \rceil$ *distinct* elements $x_1, \ldots, x_L \in A$ such that $\mathcal{C}(x_1) = \cdots = \mathcal{C}(x_L)$. The associated complexity classes Polynomial Multi-Pigeonhole Principle ($(A, B)$-$\mathsf{PMPP}^L$) consist of all problems that reduce to this problem.

We show close connections between $(A, B)$-$\mathsf{PMPP}^L$ and many celebrated upper bounds on the minimum distance of a code or lattice (and on the list-decoding radius). In particular, we show that the associated computational problems (i.e., the problem of finding distinct codewords or lattice points that lie in a certain small ball) are in $(A, B)$-$\mathsf{PMPP}^L$, with a more-or-less smooth tradeoff between the distance and the parameters $A$, $B$, and $L$. These connections are particularly rich in the case of codes, in which case we show that multiple incomparable bounds on the minimum distance lie in seemingly incomparable complexity classes. Surprisingly, we also show that some bounds on the minimum distance are actually *complete* for these classes (for codes represented by arbitrary circuits).

We go on to study $(A, B)$-$\mathsf{PMPP}^L$ as an interesting family of complexity classes in their own right, and we uncover a rich structure. We first show that techniques that were recently developed in the cryptographic literature on multicollision-resistant hash functions can be applied in our setting. Specifically, we show inclusions of the form $(A, B)$-$\mathsf{PMPP}^L \subseteq (A', B')$-$\mathsf{PMPP}^{L'}$ for certain non-trivial parameters, black-box separations between such classes in different parameter regimes, and a non-black-box proof that $(A, B)$-$\mathsf{PMPP}^L \in \mathsf{FP}$ if $(A', B')$-$\mathsf{PMPP}^{L'} \in \mathsf{FP}$ for yet another parameter regime. We also show that $(A, B)$-$\mathsf{PMPP}^L$ lies in the recently introduced complexity class Polynomial Long Choice for some parameters.

*University of Colorado Boulder. huckbennett@gmail.com. This work is supported in part by NSF Grant No. 2312297. Most of this work was done while the author was at Oregon State University.

†Cornell University. sg974@cornell.com. This work is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

‡Cornell University. noahsd@gmail.com. This work is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

ISSN 1433-8092

# Contents


**1 Introduction** **1**
- 1.1 Our results . . . 1
  - 1.1.1 Connections between coding and lattice problems and PMPP . . . 1
  - 1.1.2 Containments between different classes . . . 4
  - 1.1.3 Black-box separations (and a non-black-box non-separation) . . . 5
- 1.2 Comparison with Jain, Li, Robere, and Xun . . . 6
- 1.3 Other related work . . . 6
- 1.4 A note on "codes" represented by circuits, injectivity, and systematic form . . . 8
- 1.5 Open problems . . . 8

**2 Preliminaries** **9**
- 2.1 Coding basics . . . 9
- 2.2 Computational problems and complexity classes . . . 10
- 2.3 Some computational coding problems . . . 11
- 2.4 Lattices, Minkowski's theorem, and $\ell_p$ norms . . . 12
- 2.5 A list-decoding bound on Reed-Solomon codes . . . 13

**3 Coding Problems** **14**
- 3.1 The (list) Singleton and Plotkin bounds . . . 14
- 3.2 The (list) Hamming and Elias-Bassalygo bounds . . . 16
  - 3.2.1 The (list) Hamming bound . . . 16
  - 3.2.2 E pluribus duo . . . 17
  - 3.2.3 The Elias-Bassalygo bound . . . 17
- 3.3 Hardness of SDP and DenseBall . . . 18
  - 3.3.1 PWPP-hardness of SDP . . . 18
  - 3.3.2 PMPP-hardness of DenseBall . . . 19

**4 Finding short lattice vectors is in PMPP** **21**

**5 Inclusions** **22**
- 5.1 Merkle–Damgård construction for multicollisions . . . 22
- 5.2 A reduction using Merkle trees and list-recoverable codes . . . 23
- 5.3 Polynomial Long Choice . . . 25

**6 Black-box separations** **25**
- 6.1 Simulation approach . . . 25
- 6.2 KNY approach . . . 27

**7 A non-black-box non-separation** **29**

**A On efficient injections from $[N]$ to sets of size roughly $N$** **33**
- A.1 Injections into $\ell_p$ balls . . . 34
- A.2 An injection into the Hamming ball . . . 35

# 1 Introduction

We are interested in the following computational search problem: given as input a circuit $\mathcal{C} : [A] \to [B]$, the goal is to find $L \geq 2$ distinct input values $x_1, x_2, \ldots, x_L$ such that $\mathcal{C}(x_1) = \mathcal{C}(x_2) = \cdots = \mathcal{C}(x_L)$. Notice that, by the (generalized) pigeonhole principle, this problem is total if (and only if) the size $A$ of the domain of $\mathcal{C}$ and the size $B$ of its range satisfy $L \leq \lceil A/B \rceil$. We will focus on the regime in which the problem is total.

In the special case when $L = 2$ and $A = B + 1$,[1] this is the canonical complete problem for the Polynomial Pigeonhole Principle complexity class (PPP).[2] This class was introduced by Papadimitriou in his celebrated work studying problems in TFNP (i.e., total search problems in FNP) [Pap94]. Since then, the class PPP has been of great interest because it is known to contain many important computational problems, such as the problem of breaking a cryptographic collision-resistant hash function, the problem of breaking a one-way permutation, factoring (under randomized reductions) [Jeř16], the problem of finding a vector in a lattice within Minkowski's bound (in the $\ell_\infty$ norm, though see below) [BJP+19], and many more problems of interest [SZZ18, BFH+23].

Because of its relationship with the pigeonhole principle, we call this problem Pigeon. (Papadimitriou originally used this name for the special case when $L = 2$ [Pap94].) In fact, we get a family of problems $(A, B)$-Pigeon$^L$, parameterized by the input size $A$, the output size $B$, and the number of colliding inputs $L \leq \lceil A/B \rceil$ that we must find. Accordingly, we define the complexity class $(A, B)$-PMPP$^L$ as the set of all search problems that have a polynomial-time (Karp) reduction to $(A, B)$-Pigeon$^L$.[3]

In the complexity-theoretic literature, there is very little work on Pigeon for $L > 2$ (although we note Sotikari's thesis [Sot20, Section 4.5] as foundational work in this direction; and see also Section 1.2). On the other hand, there is an exciting line of work in the cryptographic literature studying *multicollision-resistant hash functions* [Jou04, NS07, YW07, BDRV18, BKP18, KNY18, Sot20, RV22]. From our perspective, one can think of such works as studying the *average-case* complexity of the Pigeon problem (under efficiently sampleable distributions of circuits $\mathcal{C}$). However, even these works have been rather limited, primarily focusing on the case when $L$ is constant and when $\log A \gtrsim 2 \log B$.

## 1.1 Our results

In this work, we study the family of complexity classes $(A, B)$-PMPP$^L$ for a wide range of parameters $A$, $B$, and $L$. We show a number of fundamental results about these classes, which are as follows.

### 1.1.1 Connections between coding and lattice problems and PMPP

Our first set of results is a number of connections between PMPP and computational search problems related to error-correcting codes and lattices.[4]

**Coding problems and PMPP.** Recall that a $q$-ary code with messages of length $k$, block length $n$, and distance $d$ is a function $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ such that

$$d = \min_{\boldsymbol{x}_1 \neq \boldsymbol{x}_2} \Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2)) \ ,$$

where $\Delta$ is the Hamming distance, i.e., the number of entries in which two elements in $\mathbb{F}_q^n$ differ. For our purposes, we think of $\mathcal{C}$ as an arbitrary circuit with size $\mathrm{poly}(n, k, \log q)$. (Much of the literature is concerned

[1] See Section 2 for discussion of what we mean by a circuit with input size $A$ and output size $B$ when $A$ and $B$ are not necessarily powers of 2.

[2] When $L = 2$ and $A/B \geq 1 + 1/\mathrm{poly}(n)$, one obtains the canonical complete problem for Polynomial Weak Pigeonhole Principle complexity class (PWPP). We will say much more about this distinction later. Here, we are deliberately conflating the two classes.

[3] To define PMPP formally, $A$ and $B$ should be functions of some asymptotic parameter $n$, and $L$ might also be a function of $n$. But, we mostly ignore this issue in the introduction for simplicity.

[4] In fact, the authors were not originally interested in Pigeon or PMPP. Rather, the authors were interested in the complexity of these coding and lattice problems, and were quite surprised to discover so many links to multicollisions.

with the special case when $\mathcal{C}$ is a *linear* function, in which case the code is called a *linear* code. We discuss our choice of definition more in Section 1.4.)

The most fundamental question in coding theory is to find the largest possible value of $d$ for fixed $n$, $k$, and $q$. Many beautiful upper bounds on $d$ are known in terms of $n$, $k$, and $q$.

We define the $(n, k, d)_q$-Short Distance Problem ($(n, k, d)_q$-SDP) as the computational search problem in which the input is a circuit $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ and the goal is to find $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ such that $\Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2)) \leq d$.[5] Notice that $(n, k, d)_q$-SDP is total if and only if all $q$-ary codes with message length $k$ and block length $n$ have distance at most $d$. So, upper bounds on the minimum distance of a code correspond to proofs of totality of SDP. It is therefore natural to ask about the complexity of $(n, k, d)_q$-SDP for a given upper bound $d = d(n, k, q)$ on the minimum distance of a $q$-ary code with block length $n$ and message length $k$.

We show that many of the celebrated bounds of this form yield versions of SDP that are in different versions of PMPP, including the Singleton bound [Sin64], the Hamming bound [Ham50], the Plotkin bound [Plo60], and the Elias-Bassalygo bound [Bas65]. In fact, we show a smooth tradeoff. Specifically, we show that one can obtain larger values of $d$ by increasing $L$ *or* decreasing $A$ (while holding $B$ fixed and maintaining totality). In particular, we show that SDP is in PWPP for distances $d$ above the Hamming bound.

Furthermore, we show that finding any pair of codewords within distance $(1/2 - \varepsilon)n$ is PWPP-hard for arbitrary codes. Note that the problem with relative distance larger than $1/2$ is trivial, so this hardness is essentially tight. For codes in systematic form, we show PWPP-hardness for all distances below the Plotkin bound. (See Section 1.4 for a discussion about the distinction.) Indeed, there is a large overlap in these parameter regimes, so that in a large range of parameters we show that SDP is PWPP-complete! We summarize these results for binary codes in Figure 1.

We also show analogous results for the problem of finding *many* codewords that all lie in a small ball, which corresponds to bounds on list decoding. Indeed, we show that list-decoding generalizations of the Hamming bound and the Singleton bound lie in PMPP *and* are actually hard for this problem as well in certain parameter regimes. (These results are significantly more subtle than the $L = 2$ case, since for $L > 2$, the complexity of $(A, B)$-Pigeon$^L$ seems to vary quite a bit with the parameters, whereas for $L = 2$, the parameters $A$ and $B$ matter much less.)

In fact, even for $L = 2$, our results (shown in Figure 1) are novel. Our completeness results in particular add SDP with various parameters to the rather short list of problems that are known be hard for PWPP. (This might even be viewed as resolving a coding-theoretic analogue of the conjecture that Minkowski's bound is PPP-complete [BJP+19].)

**Lattice problems and PMPP.** We next show similar results for computational lattice problems. Recall that a lattice is the set of all integer linear combinations of $n$ linearly independent basis vectors $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{Z}^n$, i.e.,

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \{z_1\boldsymbol{b}_1 + \cdots + z_n\boldsymbol{b}_n \ : \ z_i \in \mathbb{Z}\} \ .$$

A fundamental question about lattices asks when there must exist a non-zero lattice point $\boldsymbol{y} \in \mathcal{L}_{\neq \mathbf{0}}$ such that $\|\boldsymbol{y}\|_K \leq r$, where $\|\cdot\|_K$ is some norm of interest and $r$ is some bound.

Minkowski's celebrated theorem [Min10] tells us that such a $\boldsymbol{y}$ is guaranteed to exist when

$$r \leq 2\det(\mathcal{L})^{1/n}/\operatorname{vol}(K)^{1/n} \ ,$$

where $\det(\mathcal{L}) := |\det(\mathbf{B})|$ is the lattice determinant and $K$ is the unit ball of the norm $\|\cdot\|_K$. The corresponding computational problem Minkowski$_K$ is the search problem in which the input is a basis $\mathbf{B}$ for a lattice, and the goal is to output a non-zero vector $\boldsymbol{y} \in \mathcal{L}_{\neq \mathbf{0}}$ such that

$$\|\boldsymbol{y}\|_K \leq 2\det(\mathcal{L})^{1/n}/\operatorname{vol}(K)^{1/n} \ .$$

[5] The problem of finding $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ that minimizes $\Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2))$ for the given input $\mathcal{C}$ is called the Minimum Distance Problem (MDP), and it is known to be NP-hard. One is often interested only in the important special case in which $\mathcal{C}$ computes a linear function over $\mathbb{F}_q$, but we stick to this more general setting because many of our results still apply even for arbitrary codes.

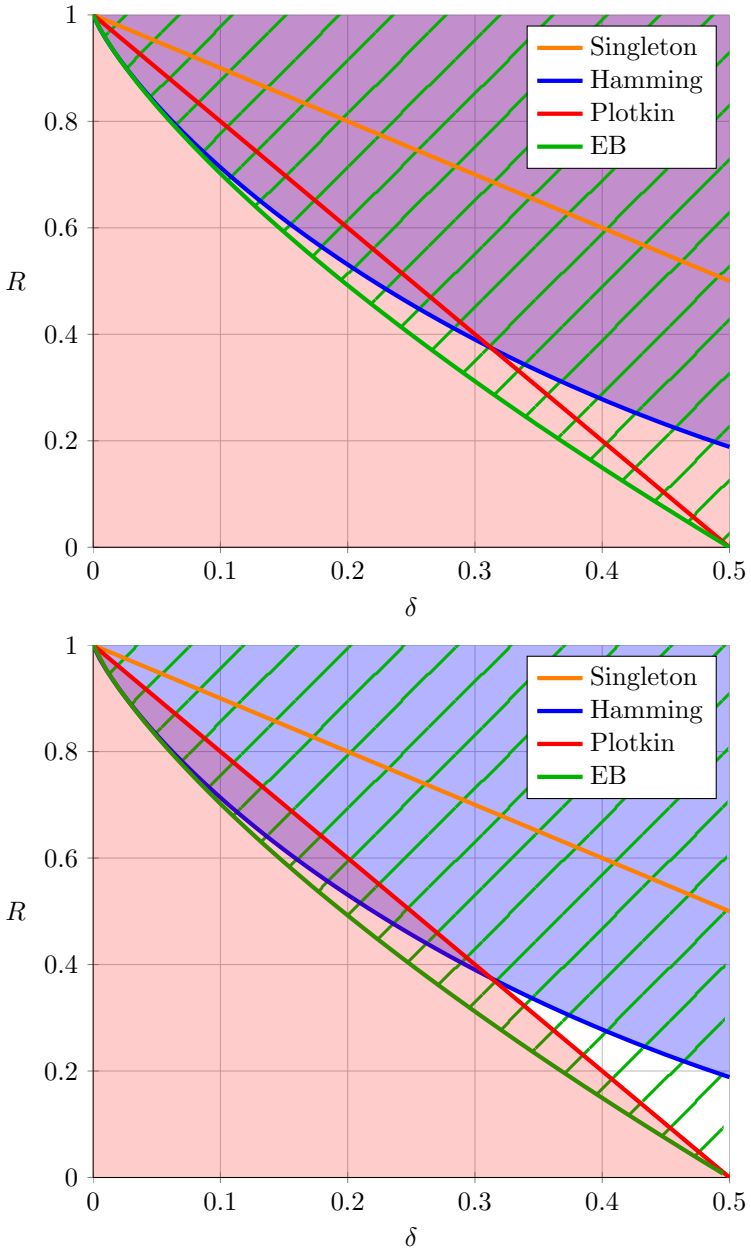


Figure 1: Two plots showing the complexity of the problem of finding two codewords within relative distance $\delta := d/n$ in a code with rate $R := k/n$ for binary codes (represented by circuits). The shaded red region represents PWPP-hardness. (The entire top figure is shaded red.) The shaded blue region represents containment in PWPP. The region of overlap therefore represents regimes where the problem is PWPP-complete. The area covered with green dashed diagonal lines represents problems in $\mathsf{PMPP}^L$ for some $L \geq 2$. The top figure is for arbitrary (binary) codes represented by arbitrary circuits, while the bottom figure is for (binary) codes in systematic form, for which our hardness results are a bit weaker. (See Section 1.4 for discussion of systematic form and how we define codes in this context.)

This problem is quite important in cryptography, particularly in the $\ell_2$ norm. In the special case of the $\ell_\infty$ norm, it is known that Minkowski$_\infty \in$ PPP, and Ban, Jain, Papadimitriou, Psomas, and Rubinstein [BJP+19] conjectured that Minkowski$_\infty$ is actually PPP-complete. (This conjecture remains open.)

We study the more general problem of finding $\boldsymbol{y} \in \mathcal{L}_{\neq \mathbf{0}}$ with

$$\|\boldsymbol{y}\|_p \leq \gamma \det(\mathcal{L})^{1/n} ,$$

where

$$\|\boldsymbol{y}\|_p := (|y_1|^p + \cdots + |y_n|^p)^{1/p}$$

is the $\ell_p$ norm. This problem is known as the $\gamma$-Hermite Shortest Vector Problem ($\gamma$-HSVP$_p$), and it is very important for cryptography and very well studied in a wide range of parameters $\gamma$ [LLL82, Sch87, GN08, MW16, ALS21], particularly for $p = 2$. The case $\gamma = 2/\operatorname{vol}(\mathcal{B}_p^n)^{1/n}$ corresponds to Minkowski$_p$, where $\mathcal{B}_p^n$ is the unit $\ell_p$ ball. However, Minkowski's bound is not tight (except in the $\ell_\infty$ norm). For example, Blichfeldt improved on Minkowski's celebrated theorem in the $\ell_2$ norm, proving that $\gamma$-HSVP$_2$ is total when $\gamma \lesssim \sqrt{2}/\operatorname{vol}(\mathcal{B}_2^n)^{1/n}$ [Bli29].[6]

Perhaps surprisingly, we show that $\gamma$-HSVP$_2 \in (A, B)$-PMPP$^L$ for $\gamma \approx \sqrt{2}/\operatorname{vol}(\mathcal{B}_2^n)^{1/n}$ and appropriate choices of $A$, $B$, and $L$. In fact, we show a smooth tradeoff between the Hermite factor $\gamma$ and the parameters $A$, $B$, and $L$, showing that one can obtain shorter vectors by either increasing $L$ or decreasing the ratio between $A$ and $B$ (while maintaining totality). A similar story holds for $\ell_p$ norms more generally. (The result below makes sense when $\alpha$ and $L$ are constants. Our more general result works for superconstant $\alpha$ and $L$, but the statement is rather technical because of the subtle relationship between the dimension $n$ and the parameters $\alpha$, $B$, and $L$.)

**Theorem 1.1** (Informal; see Corollary 4.2)**.** *For any constant integer $p \geq 1$, $\gamma$-HSVP$_p \in (\alpha B, B)$-PMPP$^L$ for*

$$\gamma \lesssim d_p(L, n)\alpha^{1/n} \cdot \operatorname{vol}(\mathcal{B}_p^n)^{-1} ,$$

*where $1 \leq d_p(L, n) \leq 2$ is a particular function that is decreasing in the parameter $L$.*

### 1.1.2 Containments between different classes

We next study containments between these PMPP classes with different parameters $A$, $B$, and $L$, and other classes of interest in TFNP.

Recall that the celebrated Merkle-Damgård construction [Mer89, Dam89] shows that the ratio of the input size $A$ to the output size $B$ of a circuit essentially does not matter in the special case when $L = 2$, since one can efficiently reduce from the problem of finding a single collision (i.e., $L = 2$) in a barely compressing circuit $\mathcal{C} : A \to B$ with $A = (1 + 1/\operatorname{poly}(\log B))B$ to the (seemingly much easier) problem of finding a single collision in a much more compressing circuit $\mathcal{C}' : A' \to B$ with $\log A' = \log(B)^C$ for any constant $C > 1$. In our terminology, $((1 + 1/\operatorname{poly}(\log B)), B)$-PMPP$^2 = (2^{\log^C B}, B)$-PMPP$^2$. This surprising collapse of complexity classes is known as *domain extension*, and it has innumerable applications in cryptography and complexity theory.[7]

Already in 2004, it was noticed by Joux that Merkle and Damgård's elegant domain extension technique does not seem to work for $L > 2$ [Jou04]. So, it appears that for $L > 2$, the relationship between $A$ and $B$

[6] We note that Blichfeldt proved two distinct relevant theorems in this context, which might easily confuse the reader. So, we endeavor to clarify here. The theorem commonly referred to as "Blichfeldt's theorem" says that any measurable set $S \subset \mathbb{R}^n$ with $\operatorname{vol}(S) > \det(\mathcal{L})$ is guaranteed to contain two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ with $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ such that $\boldsymbol{x}_1 - \boldsymbol{x}_2 \in \mathcal{L}$. This theorem is often discussed in the context of total lattice problems because it can be used to prove Minkowski's theorem. In fact, Sotikari, Zampetakis, and Zirdelis introduced a related computational problem that they called BLICHFELDT (in which the set $S$ is represented implicitly by a circuit), and they showed that BLICHFELDT is actually PPP-complete. It is not clear how BLICHFELDT is related to $\gamma$-HSVP for $\gamma \approx \sqrt{2}/\operatorname{vol}(B_2^n)^{1/n}$.

[7] Admittedly, we are deliberately conflating the distinction between the problem of breaking a cryptographic hash function (which is what Merkle and Damgård actually studied) and the problem of finding a collision in an arbitrary worst-case circuit. All of the above statements hold in both cases.

(i.e., how "compressing" the circuit $\mathcal{C}$ is) might matter quite a bit. This suggests a surprising fundamental difference between the case $L = 2$ and the case when $L > 2$.

However, we show two (rather weak) notions of domain extension that still work in the setting of multicollisions. The first such result follows by analyzing the Merkle-Damgård construction in this setting and showing that it does achieve *something*, albeit with a large loss in parameters. The second result shows a more sophisticated reduction (using Merkle trees together with list-recoverable codes) that is better than the Merkle-Damgård-based reduction in the regime where $A$ is very large compared to $B$. The latter result can be thought of as a translation into our setting of a beautiful result due to Bitanski, Kalai, and Paneth [BKP18] in the setting of multicollision-resistant hash functions. (In fact, the proof is substantially simpler in our setting than in that of [BKP18], since we do not have to worry about the many additional issues that arise in the average-case setting.)

Together, these results show that it is possible to reduce the problem of finding multicollisions in a less compressing function to the problem of finding multicollisions in a more compressing function, but at the expense of a large loss in the number of collisions found.

**Theorem 1.2** (Informal; see Section 5)**.** *For $m > a > b$,*

$$(2^a, 2^b)\text{-PMPP}^{L'} \subseteq (2^m, 2^b)\text{-PMPP}^{L}$$

*for $L' \approx L^{(a-b)/(m-b)}$.*

*For any $r \geq 2$, any $k \geq 1$, and any $L$,*

$$(2^{vr}, 2^v)\text{-PMPP}^{M'} \subseteq (2^{vrk}, 2^v)\text{-PMPP}^{M}$$

*under randomized reductions, for*

$$M' \approx M^{\log r/(2\log r + \log k + \log(M)/2)}.$$

We also show that the class PMPP is contained in the complexity class Polynomial Long Choice (PLC), recently introduced in [PPY23], for choices of the parameters $A$, $B$, and $L$. In fact, we show a reduction to the Unary Long Choice problem. (This result was recently independently discovered in [JLRX24]. See Section 1.2.) This strengthens a result of Pasarkar, Yannakakis, and Papadimitriou [PPY23], who showed that PWPP $\subseteq$ PLC.

**Theorem 1.3** (Informal; see Section 5.3)**.** *For any $L < n$,*

$$(2^n, 2^{n-L})\text{-PMPP}^{L} \subseteq \text{PLC}$$

### 1.1.3 Black-box separations (and a non-black-box non-separation)

Our final set of results concerns black-box separations between $(A, B)$-PMPP$^L$ for different values of $A$, $B$, and $L$, which suggest that it might be hard to prove stronger containments than what we show above. We note that recent independent work of Jain, Li, Robere, and Xun [JLRX24] showed exciting black-box separations of this form. While our results are formally incomparable to theirs, we believe that the results of [JLRX24] are more interesting than our own. (See Section 1.2.)

The starting point for our results is a beautiful idea due to Komargodski, Naor, and Yogev for separating $(2^n, 2^{n/2})$-PMPP$^L$ from $(2^n, 2^{n/2})$-PMPP$^{L'}$ for any constants $L \neq L'$ (particularly in the average-case setting relevant to cryptography). Unfortunately, however, their proof had a subtle bug that does not seem easy to fix [KY23].

We show two different black-box separations, which can be seen as partial evidence that domain extension and range compression are not possible when $L > 2$. However, we note that our black-box separations are rather weak, since they only rule out rather fine-grained black-box reductions between the classes.

Finally, we show that a very clever non-black-box proof due to Rothblum and Vasudevan [RV22] extends to our setting. In particular, Rothblum and Vasudevan show, using non-black-box techniques, that the existence of a certain sufficiently strong multicollision-resistant hash function implies the existence of a collision-resistant hash function. We prove an analogue of their result in our different (worst-case) setting.

As these results are rather technical, we refer the reader to Sections 6 and 7 for the details.

## 1.2 Comparison with Jain, Li, Robere, and Xun

Recent work by Jain, Li, Robere, and Xun [JLRX24] also defines and studies the computational problem $(A, B)$-Pigeon$^L$ and the associated complexity class $(A, B)$-PMPP$^L$ (with slightly different notation). (They also define additional classes that correspond to the union of $(A, B)$-PMPP$^L$ over different parameters $A$, $B$, and $L$.) [JLRX24] is concurrent with and independent of this work. Here, we provide a brief comparison of their work with ours.

Both the present work and [JLRX24] define multi-collision classes and study relationships between them. At a high level, [JLRX24] has morally stronger (although formally incomparable) results on the structural complexity of these classes, whereas a large part of the present work focuses on showing containment and hardness of coding and lattice problems with respect to these classes (which [JLRX24] does not study at all). Not surprisingly, the two works also have many results with no analogue in the other.

In terms of structural complexity, [JLRX24] contains exciting black-box separation results, which, although formally incomparable to ours, we think of as stronger. In particular, [JLRX24] are essentially able to black-box separate $(A, B)$-PMPP$^L$ from $(A', B')$-PMPP$^{L'}$ for *any* constants $L \neq L'$ and *any* (reasonable) $A$, $B$, $A'$, and $B'$. They also show black-box separations between PMPP and other interesting complexity classes in TFNP, and in particular show a black-box separation between the Ramsey problem and PMPP. We refer the reader to [JLRX24] for the technical details.

[JLRX24] also studies the relationship between PLC and PMPP. Indeed, they prove a result that is essentially identical to our Theorem 1.3, which shows that PMPP $\subseteq$ PLC for certain parameters. (Formally, our technical result in Theorem 5.8 is more general than the analogous result in [JLRX24], but it is clear that the proof in [JLRX24] yields the more general result as well.) In addition, they show a containment in the other direction, that PLC $\subseteq$ PMPP, albeit for different parameters. Finally, [JLRX24] shows that an interesting problem in TFBQP is in PMPP.

In this work, we study the relationship of PMPP with coding and lattice problems (Sections 3 and 4), Merkle-Damgård-style reductions between PMPP with different parameters (Section 5), and a non-block-box non-separation in the style of [RV22] (Section 7). These topics are not studied in [JLRX24].

## 1.3 Other related work

Our work lies at the intersection of a number of different areas, and there is therefore much related work to discuss in addition to [JLRX24]. Here, we focus on how this prior work relates to our work.

**Literature on multicollisions.** There is quite a bit of prior work in the cryptography literature on *multicollision-resistant hash functions*. In our terminology, a multicollision-resistant hash function is some efficiently sampleable distribution of instances of $(A, B)$-Pigeon$^L$ (i.e., circuits $\mathcal{C} : [A] \to [B]$) that are actually hard (i.e., that cannot be solved by polynomial-time algorithms with non-negligible probability). A survey of this literature is beyond the scope of this work. Broadly speaking, these works have been concerned with (1) understanding the relationship between collision resistance and multicollision resistance; (2) finding applications of multicollision-resistant hash functions; and (3) building multicollision-resistant hash functions from various cryptographic hardness assumptions.

Many of the techniques that we use to understand the relationship between $(A, B)$-Pigeon$^L$ in different parameter regimes are directly inspired by this cryptographic literature. In particular, our inclusion based on Merkle trees and list-recoverable codes is a direct adaption of Bitanski, Kalai, and Paneth's construction from the cryptographic setting to our setting [BKP18]; our black-box separations are inspired by [KNY18]; and our non-black-box non-separation is a direct adaptation of Rothblum and Vasudevan's proof from the cryptographic setting to our setting [RV22].

In contrast, there is very little prior work in the worst-case setting. To our knowledge, the only works that consider the worst-case complexity of finding multicollisions are Komargodski, Naor, and Yogev [KNY18] and Sotikari [Sot20] (though see Section 1.2). In both of these works, the worst-case complexity of $(A, B)$-Pigeon$^L$ is not the primary focus, but both do define the special cases of the complexity class $(A, B)$-PMPP$^L$, when

$A = B^2$ (specifically, $A = 2^{2n}$ and $B = 2^n$) and $L$ is a constant. This is a natural setting of parameters, but we show interesting results in other parameter regimes as well.

Sotikari in particular shows a complete problem for PMPP that is similar to the PPP-complete and PWPP-complete problems from [SZZ18]. In particular, Sotikari's PMPP-complete problem bears some resemblance to certain lattice and coding problems, though the relationship is unclear. We refer the reader to [SZZ18, Sot20] for more.

Komargodski, Naor, and Yogev claimed a black-box separation between $(2^{2n}, 2^n)$-PMPP$^L$ and $(2^{2n}, 2^n)$-PMPP$^{L'}$ for any constants $L' < L$, but their elegant proof contained a subtle bug in their proof that has not been fixed [KY23]. Our black-box separations use similar ideas but are weaker than what they originally claimed.

**Literature on PPP and PWPP.** In contrast, there is much literature studying the complexity classes PPP and PWPP, which correspond to the special case of PMPP when $L = 2$ (in different parameter regimes). These classes were introduced by Papadimitriou in his seminal work [Pap94]. Since then, many problems of interest have been shown to be contained in either PWPP or PPP [Jeř16, BJP+19, SZZ18, BFH+23]. Until recently, only a small handful of problems are known to be complete for PPP or PWPP. However, Bourneuf, Folwarczný, Hubáček, Rosen, and Schwartzbach recently showed a number of problems arising from extremal combinatorics that are complete for either PPP or PWPP [BFH+23].

There has also been some literature concerned with generalizing PPP and PWPP to classes other than PMPP. In particular, Pasarkar, Papadimitriou, and Yannakakis [PPY23] recently introduced the class Polynomial Long Choice (PLC), which can be thought of as corresponding to the generalization of the pigeonhole principle obtained by iterating the pigeonhole principle many times.

**The complexity of total lattice problems.** The complexity of Minkowski's bound and HSVP more generally is quite well studied, particularly in the $\ell_\infty$ norm and the $\ell_2$ norm. In particular, algorithms for $\text{HSVP}_2$ play a very important role in lattice-based cryptography, and algorithms for HSVP with different approximation factors are a very active area of research [LLL82, Sch87, GN08, MW16, ALS21]. Some of these algorithms can be viewed as constructive proofs of classical results about the minimum distance of a lattice relative to the determinant. (For example, the celebrated LLL algorithm gives a constructive proof of Hermite's bound [LLL82], and the slide reduction algorithm gives a constructive proof of Mordell's inequality [GN08].)

Finding a vector within Minkowski's bound in the $\ell_\infty$ norm is considered one of the most important problems in the complexity class PPP. In particular, Ban, Jain, Papadimitriou, Psomas, and Rubinstein showed that some of the most important problems in PPP can be reduced to this problem, and they conjectured that it is PPP-complete [BJP+19]. Sotikari, Zampetakis, and Zirdelis further investigated the relationship between lattice problems, PPP, and PWPP, showing two problems related to lattices that are PPP-complete and PWPP-complete respectively [SZZ18].

**The complexity of total coding problems.** To our knowledge, much less is known about the complexity of total problems that arise in coding theory. Instead, much work has focused on the $\gamma$-*approximate Minimum Distance Problem* ($\gamma$-MDP), in which the input is a linear code and the goal is to output a pair of distinct codewords $\boldsymbol{c}_1, \boldsymbol{c}_2$ such that the distance between them is within a factor $\gamma$ of the minimum distance of the input code (or, since the code is linear, one can equivalently output a non-zero codeword with nearly minimal Hamming weight). This problem is known to be NP-hard [Var97], even to approximate [DMS03]. In contrast, we are interested in the problem of finding distinct codewords $\boldsymbol{c}_1, \boldsymbol{c}_2$ that are within distance $d$, where $d$ depends only on the message length $k$ and block size $n$ of the code (and *not* on the minimum distance). We are particularly interested in the total regime, where such problems are unlikely to be NP-hard.

To our knowledge, the only direct work in this area is a beautiful paper by Debris-Alazard, Ducas, and van Woerden [DDv22], which showed an LLL-like algorithm for linear codes that efficiently finds codewords within the Griesmer bound [Gri60] (which only applies to linear codes).

## 1.4 A note on "codes" represented by circuits, injectivity, and systematic form

The coding theory results in this paper are concerned with the problem of finding close codewords (or many codewords that lie in a relative small ball) in a "code" *represented by an arbitrary circuit* $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with size $\mathrm{poly}(n, k, \log q)$. It is far more common in the literature to consider *linear* codes represented by a generator matrix (or, equivalently, an invertible circuit with linear gates). (Sometimes when arbitrary codes are considered in the literature, the code is simply represented by listing all $q^k$ points, while we represent our codes succinctly.)

Whether one should really think of a generic circuit $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ as representing a "code" is not so clear. In particular, such a circuit might not be *injective*, i.e., there might be distinct "messages" $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{F}_q^k$ that map to the same "codeword" $\mathcal{C}(\boldsymbol{x}_1) = \mathcal{C}(\boldsymbol{x}_2)$. (And, there is likely no way to efficiently determine whether such a circuit is injective. Indeed, determining this is coNP-complete.) But, we find the coding-theoretic perspective to be quite useful. In particular, the notion of the distance of a code still makes sense with this slightly more general definition, and the bounds on the distance that we discuss in this paper still apply. Indeed, much of coding theory still makes sense if we treat such degenerate, non-injective codes simply as "codes with distance 0," and standard bounds in coding theory, such as the Singleton and Hamming bounds, still apply. (These coding theory bounds guarantee *near collisions* in such general circuits $\mathcal{C}$.)

Of course, it is not an issue, and actually a strength that our upper bounds apply to such general "codes." Indeed, any upper bound that applies in the more general case when "codes" are represented by arbitrary circuits certainly applies to the special case of injective circuits or the even more special (and quite important) case of linear codes.

For our lower bounds, however, this can be viewed as a major flaw in our model. Therefore, we prove our hardness results in two different settings.

In the first "generic" setting, we show hardness for "codes" represented by arbitrary circuits $\mathcal{C} : [q]^k \to [q]^n$, which are not necessarily injective. One may argue that these reductions are rather artificial, in that the reductions produce "codes" $\mathcal{C}$ such that $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j))$ is either zero or strictly larger than $d$, where $d$ is the bound on the distance needed to solve the associated coding problem. So, these reductions rely quite heavily on this rather strange definition of a code.

In the "systematic" setting, our codes $\mathcal{C} : [q]^k \to [q]^n$ are in *systematic* form, in the sense that the first $k$ characters of $\mathcal{C}(\boldsymbol{x})$ are simply $\boldsymbol{x}$ itself. Such circuits are clearly injective, and therefore this setting is much less artificial. However, in this setting, we achieve weaker parameters. In particular, we show completeness for a very wide range of parameters in the "generic" setting, but for a more narrow range in the "systematic" setting. (See Figure 1.)

## 1.5 Open problems

We leave a number of interesting open problems. Here, we mention some of them.

**Complexity of coding and lattice problems.** We place the computational problems associated with a number of fundamental bounds on the minimum distance of a code in PMPP. However, we were unable to say anything non-trivial about the complexity of Delsarte's linear programming bound [Del73] and the closely related MRRW bound(s) [MRRW77], which are the best known bounds in a number of interested parameter regimes. The associated computational problems are, by definition, in TFNP, but we do not know any natural subclass of TFNP that contains these problems.

Similarly, the best known bound on the minimum distance in the $\ell_2$ norm of a lattice relative to the determinant is the celebrated bound due to Kabatjanskiĭ and Levenšteĭn [KL78] (which is better than Blichfeldt's [Bli29] by a factor of roughly $2^{1/10}$). The problem of finding a vector within the Kabatjanskiĭ and Levenšteĭn is again, by definition, in TFNP, but we do not know any natural subclass of TFNP that contains this problem.

In a different direction, we show hardness of various total coding problems (and even completeness in some regimes), but only for codes represented by circuits. It would be very exciting to show hardness results

of total problems on *linear* codes (represented, e.g., by generator matrices), since these are the more standard problems. (The analogous question for lattices was already asked in [BJP+19].)

**Better understanding of PMPP across different regimes.** The first question that comes to mind about the complexity classes $(A, B)$-$\mathsf{PMPP}^L$ is, of course, "how do these classes relate to each other across different parameter regimes?" In the case when $L = 2$, it has long been known that the relationship between $A$ and $B$ does not matter much, with, e.g.,

$$((1 + 1/\operatorname{poly}(n)) \cdot 2^n, 2^n)\text{-}\mathsf{PMPP}^2 = ((2^{n+\operatorname{poly}(n)}, 2^n)\text{-}\mathsf{PMPP}^2 = \mathsf{PWPP} .$$

For $L > 2$, it seems unlikely that a similar result holds. We instead describe a rich and rather complicated set of inclusions (and non-black-box relationships) among these classes, as well as black-box separations.

However, we have no evidence that these results are tight. One would ideally like to show black-box separations and inclusions that are tight with one another. (Or, alternatively, one might hope to prove non-trivial equalities $(A, B)$-$\mathsf{PMPP}^L = (A', B')$-$\mathsf{PMPP}^{L'}$ like what we know in the case of $L = L' = 2$.) The analogous average-case question has been the topic of much research in the cryptography literature [Jou04, NS07, YW07, BDRV18, BKP18, KNY18, Sot20, RV22], but to the authors' knowledge the worst-case setting was barely explored before this work and the (concurrent and independent) work of Jain, Li, Robere, and Xun [JLRX24].


## Acknowledgements

The authors would like to thank Atri Rudra for very helpful discussions.


# 2 Preliminaries

We will denote vectors by boldface letters like $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$. For an $n$-coordinate vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $1 \le i \le j \le n$, we define $\boldsymbol{x}_{[i,j]} := (x_i, \ldots, x_j)$ to denote the restriction of $\boldsymbol{x}$ to its $i$th coordinate through $j$th coordinate. For a positive integer $q$, we define $[q] := \{1, \ldots, q\}$.

For convenience, we define a circuit $\mathcal{C} : [A] \to [B]$ for $A, B \in \mathbb{Z}^+$ (i.e., a circuit with domain and range sizes that are not necessarily powers of two) as a Boolean circuit $\mathcal{C}' : \{0,1\}^a \to \{0,1\}^b$ with $a := \lceil \log_2(A) \rceil$ input bits and $b := \lceil \log_2(B) \rceil$ outputs bits, where inputs $x$ corresponding to integers greater than $A$ and outputs $y$ corresponding to integers greater than $B$ are ignored. We similarly define circuits with $[q]^n$, $\mathbb{F}_q^n$, $[A_1] \times [A_2]$, $\mathbb{F}_q^n \times [A]$, etc., as their domain or range, noting that in each case there is a simple efficiently computable bijection between such domains and ranges and sets $[A]$ and $[B]$.

We will need to use the following quantity in many different places.

**Definition 2.1.** *Let $e > 0$ and $N \ge 2$ be an integer and let $\mathcal{X} = (A, \Delta)$ be a metric space on the set $A$ with metric $\Delta$. We define*

$$d_{\mathcal{X}}(e, N) := \sup_{\substack{y, v_1, \ldots, v_N \in A \\ \Delta(y, v_i) \le e}} \min_{i \ne j} \Delta(v_i, v_j)$$

*to be the smallest distance $d$ so that any $N$ points in a ball of radius $e$ must contain a pair within distance $d$.*

## 2.1 Coding basics

We use $\|\boldsymbol{x}\|_0$ to denote the Hamming weight (i.e., number of non-zeroes) in a vector $\boldsymbol{x}$, and

$$\mathcal{H}_q^n(\boldsymbol{y}, r) := \{\boldsymbol{x} \in \mathbb{F}_q^n : \|\boldsymbol{x}\|_0 \le r\}$$

to denote the *Hamming ball* in $\mathbb{F}_q^n$ of radius $r$ centered at $\boldsymbol{y}$. We define

$$V_q^n(r) := \left|\mathcal{H}_q^n(\boldsymbol{0}, r)\right| = \sum_{i=0}^{r} (q-1)^i \cdot \binom{n}{i}$$

to be the *volume* (i.e., cardinality) of such a Hamming ball. (The volume of Hamming balls is shift-invariant, and so does not depend on the center $\boldsymbol{y}$ of the ball.)

We next define list decodability.

**Definition 2.2.** *For $k, n, r, N \in \mathbb{Z}^+$, a code specified as a circuit $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$, is $(r, N)$-*list decodable *if for all $\boldsymbol{y} \in \mathbb{F}_q^n$, $\left|\{\boldsymbol{x} : \mathcal{C}(x) \in \mathcal{H}_q^n(\boldsymbol{y}, r)\}\right| \leq N$.*

That is, a code $(r, N)$-list decodable if there are at most $N$ code words in $\mathcal{C}$ in any radius-$r$ Hamming ball in $\mathbb{F}_q^n$.

## 2.2 Computational problems and complexity classes

A *search problem* is specified by a binary relation $P : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, where $P(\boldsymbol{x}, \boldsymbol{y}) = 1$ if and only if "$\boldsymbol{y}$ is a valid output on input $\boldsymbol{x}$." A search problem $P$ is *total* if for every $\boldsymbol{x} \in \{0,1\}^*$ there exists $\boldsymbol{y} \in \{0,1\}^*$ such that $P(\boldsymbol{x}, \boldsymbol{y}) = 1$. An algorithm solves a total search problem $P$ if on input $\boldsymbol{x}$ it (always) outputs $\boldsymbol{y}$ such that $P(\boldsymbol{x}, \boldsymbol{y}) = 1$.

We will use the notion of Karp reductions between search problems.

**Definition 2.3.** *A deterministic Karp reduction from a search problem $P$ to a search problem $Q$ is a pair of deterministic polynomial-time algorithms $R, S$ with the following properties.*

1. *Given an instance $I_P$ of $P$, $R(I_P)$ outputs an instance of $I_Q$ of $Q$.*
2. *Given any valid solution $s_Q$ to $I_Q := R(I_P)$, $S(I_P, s_Q)$ outputs a valid solution $s_P$ for $I_P$.*

For convenience, we often describe our Karp reductions simply as algorithms $\mathcal{A}^B$ for problem $P$ that work with an oracle for problem $Q$ and make a single oracle call. Notice that this is equivalent, as one can always define $R(I_P)$ to be the oracle call made by $\mathcal{A}^Q$ and $S(I_P, s_Q)$ to be the output of $\mathcal{A}^B$ when the response of the oracle is replaced by $s_Q$.

We now define the central computational problem that we study in this paper.

**Definition 2.4** (Pigeon)**.** *For integers $A \geq B \geq 1$ and $L \geq 2$, the problem $(A, B)$-Pigeon$^L$ is defined as follows. The input is a circuit $\mathcal{C} : [A] \to [B]$. The goal is to output distinct $x_1, \ldots, x_L \in [A]$ such that $\mathcal{C}(x_1) = \cdots = \mathcal{C}(x_L)$.*

Notice that $(A, B)$-Pigeon$^L$ is a total search problem if (and only if) $L \leq \lceil A/B \rceil$. And, in this work, we are only interested in this case. The problem Pigeon$ABL$ is naturally associated with a family of complexity classes, which we now define.

**Definition 2.5** (PMPP)**.** *For functions $A := A(v)$, $B := B(v)$, and $L := L(v)$, the complexity class $(A, B, L)$-Polynomial Multi-Pigeonhole Principle ($(A, B)$-PMPP$^L$) is defined as the set of all search problems $P$ such that there is a Karp reduction from $P$ to $(A(v), B(v))$-Pigeon$^{L(v)}$ for some $v$.*

*We often abuse notation by leaving out the parameter $v$, by writing two of the parameters as functions of the other (e.g., $(A, \sqrt{A})$-PMPP$^{\log A}$), or by using asymptotic notation to implicitly quantify over functions $A(v)$, $B(v)$, and $L(v)$ (e.g., $(\mathrm{poly}(B), B)$-PMPP$^{\mathrm{poly}(\log B)}$).*

The special case of $L = 2$ is particularly important, and it is well studied. In particular, in [Pap94], Papadimitriou defined the following two classes.

**Definition 2.6** (PWPP and PPP)**.** *Polynomial Pigeonhole Principle (PPP) is the set of all search problems $P$ such that there is a Karp reduction from $P$ to $(2^v, 2^v - 1)$-Pigeon$^2$ for some $v$. Equivalently, PPP $= (2^v, 2^v - 1)$-PMPP$^2$.*[8]

*Polynomial Weak Pigeonhole Principle (PWPP) is the set of all search problems $P$ such that there is a Karp reduction from $P$ to $(2^{v+1}, 2^v)$-Pigeon$^2$ for some $v$. Equivalently, PWPP $= (2^{v+1}, 2^v)$-PMPP$^2$.*

[8] PPP is typically defined slightly differently. In particular, PPP is typically defined as the class of problems that are reducible to the problem in which the input is a circuit $\mathcal{C} : [2^v] \to [2^v]$ and the goal is *either* to find $x \in [2^v]$ such that $\mathcal{C}(x) = 0$, *or* to find two distinct elements $x_1, x_2 \in [2^v]$ such that $\mathcal{C}(x_1) = \mathcal{C}(x_2)$. However, there are simple Karp reductions between this problem and $(2^v, 2^v - 1)$-Pigeon$^2$. So, this definition is equivalent.

We note that because of Merkle-Damgård domain extension [Mer89, Dam89], the class PWPP is actually quite flexible. In particular, we obtain an equivalent definition if we replace $A = 2^{v+1}$ and $B = 2^v$ in the above definition by any of a wide range of parameters.

**Theorem 2.7.** *For any $A := A(B)$ with $(1 + 1/\operatorname{poly}(\log B)) \cdot B \leq A \leq B^{\operatorname{poly}(\log B)}$,*

$$(A, B)\text{-PMPP}^2 = \text{PWPP} .$$

A roughly similar result holds for PPP.

**Theorem 2.8.** *For any $A := A(B)$ with $B < A \leq B + \operatorname{poly}(\log B)$,*

$$(A, B)\text{-PMPP}^2 = \text{PPP} .$$

We will also be interested in the following relatively new total search problems and the associated complexity class defined in [PPY23]. In particular, [PPY23] shows that this class contains PPP as well as many important problems related to extremal combinatorics.

**Definition 2.9** (LongChoice and UnaryLongChoice)**.** *The* LongChoice *problem is the following: There is a universe $U$ of $2^n$ objects, represented by $2^n$ $n$-bit strings. We are given a sequence of $n-1$ circuits $P_0, \ldots, P_{n-2}$, each of $\operatorname{poly}(n)$ size, such that $P_i$ has $(i+2)n$ inputs bits and one output bit; circuit $P_i$ represents a predicate on $i+2$ objects. We are asked to find a sequence of $n+1$ distinct objects $a_0, a_1, \ldots, a_n$ with the following property: for each $i$ in $[0, \ldots, n-2]$, $P_i(a_0, \ldots, a_i, a_j)$ is the same for all $j > i$.*

UnaryLongChoice *is the the version of* LongChoice *where every predicate $P_i$ depends only on its last argument, i.e., $P_i(a_0, \ldots, a_i, x) = P_i(x)$.*

**Definition 2.10** (PLC)**.** *Polynomial Long Choice (*PLC*) is the set of all search problems $P$ such that there is a Karp reduction from $P$ to* LongChoice*.*

## 2.3 Some computational coding problems

We next define the main coding problems that we study, which is to find distinct inputs $\boldsymbol{x}_1, \boldsymbol{x}_2$ to a circuit $\mathcal{C}$ such that $\mathcal{C}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is small (or possibly 0).

**Definition 2.11.** *For $n, k, d, q \in \mathbb{Z}^+$, the* Short Distance Problem *$(n, k, d)$-SDP$_q$ is the search problem defined as follows. The input is a circuit $\mathcal{C} : [q]^k \to [q]^n$. The goal is to output $\boldsymbol{x}_1, \boldsymbol{x}_2 \in [q]^k$ such that $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and $\Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2)) \leq d$.*

A central goal in coding theory is to characterize when a code with a given input length (dimension) $k$, block length $n$, and alphabet size $q$ must have distance (at most) $d$ (i.e., a pair of distinct code words at distance at most $d$). (In Section 3, we describe many such bounds.) One may alternatively view this goal as trying to characterize when SDP is *total*, i.e., for what values of $k, n, q, d$ the problem is guaranteed to have a solution.

Since the circuit $\mathcal{C}$ describing a "code" as defined above might not be injective, we also define the following special case of SDP in which the code is required to be in systematic form. See Section 1.4.

**Definition 2.12.** *For $n, k, d, q \in \mathbb{Z}^+$, the* systematic Short Distance Problem *$(n, k, d)$-sysSDP$_q$ is the search problem defined as follows. The input is a circuit $\mathcal{C} : [q]^k \to [q]^{n-k}$. The goal is to output $\boldsymbol{x}_1, \boldsymbol{x}_2 \in [q]^k$ such that $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and $\Delta(\boldsymbol{x}_1 \circ \mathcal{C}(\boldsymbol{x}_1), \boldsymbol{x}_2 \circ \mathcal{C}(\boldsymbol{x}_2)) \leq d$.*

We will need the following simple reduction from SDP on arbitrary codes to sysSDP.

**Lemma 2.13.** *For all $n, k, d, q \in \mathbb{Z}^+$, there is a Karp reduction from $(n, k, d)$-SDP$_q$ to $(n + k, k, d + 1)$-sysSDP$_q$.*

*Proof.* On input an instance $\mathcal{C} : [q]^k \to [q]^n$ of $(n,k,d)$-SDP$_q$ on input $\mathcal{C}$, The reduction simply calls its $(n+k,k,d+1)$-SDP$_q$, receiving as output distinct $\boldsymbol{x}_1, \boldsymbol{x}_2 \in [q]^k$ such that $\Delta((\boldsymbol{x}_1, \mathcal{C}(\boldsymbol{x}_1)), (\boldsymbol{x}_2, \mathcal{C}(\boldsymbol{x}_2))) \leq d+1$. It then simply outputs $\boldsymbol{x}_1, \boldsymbol{x}_2$.

To see that this reduction is correct, it suffices to notice that

$$d+1 \geq \Delta((\boldsymbol{x}_1, \mathcal{C}(\boldsymbol{x}_1)), (\boldsymbol{x}_2, \mathcal{C}(\boldsymbol{x}_2))) = \Delta(\boldsymbol{x}_1, \boldsymbol{x}_2) + \Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2)) \geq 1 + \Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2)) \,,$$

because $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$. Therefore, $\Delta(\mathcal{C}(\boldsymbol{x}_1), \mathcal{C}(\boldsymbol{x}_2)) \leq d$, as needed. □

We will also be interested in the following problem on codes, which we view as an interesting problem in its own right, and which also interests us because of its relationship with SDP.

**Definition 2.14** (DenseBall)**.** *For $n,k,r,L,q \in \mathbb{Z}^+$, $(n,k,r)$-DenseBall$_q^L$ is the search problem defined as follows. The input is a circuit $\mathcal{C} : [q]^k \to [q]^n$. The goal is to find distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L \in [q]^k$ and $\boldsymbol{y} \in [q]^n$ such that for all all $i \in [L]$, $\Delta(\boldsymbol{y}, \mathcal{C}(\boldsymbol{x}_i)) \leq r$.*

*Similarly, $(n,k,r)$-sysDenseBall$_q^L$ is the search problem defined as follows. The input is a circuit $\mathcal{C} : [q]^k \to [q]^{n-k}$. The goal is to find distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L \in [q]^k$ and $\boldsymbol{y} \in [q]^n$ such that for all all $i \in [L]$, $\Delta(\boldsymbol{y}, \boldsymbol{x}_i \circ \mathcal{C}(\boldsymbol{x}_i)) \leq r$.*

As we did above for SDP, we show a simple reduction from DenseBall (on arbitrary codes) to sysDenseBall.

**Lemma 2.15.** *For all $n,k,t,q,L \in \mathbb{Z}^+$, there is a Karp reduction from $(n,k,t)$-DenseBall$_q^L$ to $(n+k,k,t)$-sysDenseBall$_q^L$.*

*Proof.* On input an instance $\mathcal{C} : [q]^k \to [q]^n$ of $(n,k,t)$-DenseBall$_q^L$ the reduction then calls its $(n+k,k,t)$-sysDenseBall$_q^L$ oracle on $\mathcal{C}$, receiving as output distinct $\boldsymbol{x}, \ldots, \boldsymbol{x}_L \in [q]^k$ and a center $\boldsymbol{y} = (\boldsymbol{y}', \boldsymbol{y}'') \in [q]^{k+n}$ satisfying $\Delta(\boldsymbol{y}, (\boldsymbol{x}_i, \mathcal{C}(\boldsymbol{x}_i))) \leq t$ for all $i \in [L]$. It then outputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L \in [q]^k$ and $\boldsymbol{y}'' \in [q]^n$. The reduction clearly runs in polynomial time, and correctness follows because $\Delta(\boldsymbol{y}'', \mathcal{C}(\boldsymbol{x}_i)) \leq \Delta(\boldsymbol{y}, (\boldsymbol{x}_i, \mathcal{C}'(\boldsymbol{x}_i))) \leq t$ for all $i \in [L]$. □

We note that $(n,k,2r)$-SDP$_q$ efficiently reduces to $(n,k,r)$-DenseBall$_q^2$, and vice-versa. (For the reduction from $(n,k,r)$-DenseBall$_q^2$ to $(n,k,2r)$-SDP$_q$, take $\boldsymbol{y}$ to be the midpoint between $\mathcal{C}(\boldsymbol{x}_1)$ and $\mathcal{C}(\boldsymbol{x}_2)$, where $\boldsymbol{x}_1, \boldsymbol{x}_2$ are the vectors output by the SDP oracle.)

**Lemma 2.16.** *For $n,k,r,L,q \in \mathbb{Z}^+$, the problems $(n,k,2r)$-SDP$_q$ and $(n,k,r)$-DenseBall$_q^2$ are Karp-reducible to each other.*

However, we note that SDP is not *quite* equivalent to the $L = 2$ case of DenseBall because of the case of $(n,k,d)$-SDP$_q$ when $d$ is odd.

## 2.4 Lattices, Minkowski's theorem, and $\ell_p$ norms

A lattice $\mathcal{L} \subset \mathbb{Z}^n$ is the set of all integer linear combinations of $n$ linearly independent basis vectors $\mathbf{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, i.e.,

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \{z_1 \boldsymbol{b}_1 + \cdots + z_n \boldsymbol{b}_n \ : \ z_i \in \mathbb{Z}\} \,.$$

The *determinant* of a lattice is

$$\det(\mathcal{L}) := |\det(\mathbf{B})| \,,$$

and the minimum distance in some norm $\|\cdot\|_K$ is

$$\lambda_1^{(K)}(\mathcal{L}) := \min_{\boldsymbol{y} \in \mathcal{L}_{\neq \mathbf{0}}} \|\boldsymbol{y}\|_K \,.$$

Perhaps the most important theorem in the study of lattices is Minkowski's theorem.

**Theorem 2.17** (Minkowski's theorem [Min10])**.** *For any lattice $\mathcal{L} \subset \mathbb{Z}^n$ and any norm $\|\cdot\|_K$ with unit ball $K$,*

$$\lambda_1^{(K)}(\mathcal{L}) \leq 2\,\mathrm{vol}(K)^{-1/n}\det(\mathcal{L})^{1/n}\ .$$

We are particularly interested in $\ell_p$ norms, which we write as

$$\|\boldsymbol{x}\|_p := (|x_1|^p + \cdots + |x_n|^p)^{1/p}$$

for $\boldsymbol{x} \in \mathbb{R}^n$ and $1 \leq p \leq \infty$ (where we interpret this as $\max|x_i|$ in the case when $p = \infty$). We write

$$\mathcal{B}_p^n(r) := \{\boldsymbol{x} \in \mathbb{R}^n \ :\ \|\boldsymbol{x}\|_p \leq r\}$$

for the $\ell_p$ ball with radius $r$. In the special case when $r = 1$, we sometimes elide it and simply write $\mathcal{B}_p^n$.

For finite $p$, the volume of the $\ell_p$ ball is given by

$$\mathrm{vol}(\mathcal{B}_p^n(r)) = (2r)^n \cdot \frac{\Gamma(1+1/p)^n}{\Gamma(1+n/p)} \approx \Big(\frac{2e^{1/p}p^{1/p}\cdot\Gamma(1+1/p)\cdot r}{n^{1/p}}\Big)^n\ .$$

Minkowski's theorem is easily seen to be tight in the $\ell_\infty$ norm (by, e.g., taking $\mathcal{L} = \mathbb{Z}^n$). However, in other norms, one can hope to do better. Particularly relevant to us is the following improvement of Minkowski's bound for the $\ell_2$ norm, due to Blichfeldt.

**Theorem 2.18** ([Bli29])**.** *For any lattice $\mathcal{L} \subset \mathbb{Z}^n$,*

$$\lambda_1^{(\ell_2)}(\mathcal{L}) \leq \sqrt{2}(n/2+1)^{1/n}\,\mathrm{vol}(\mathcal{B}_2^n)^{-1/n}\det(\mathcal{L})^{1/n}\ .$$

We are primarily interested in the following closely related computational problem.

**Definition 2.19.** *For any $p \geq 1$ and $\gamma := \gamma(n)$, $\gamma$-HSVP$_p$ (the Hermite Shortest Vector Problem) is the search problem in which the input is a basis $\mathbf{B} \in \mathbb{Z}^{n\times n}$ for a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, and the goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y}\|_p \leq \gamma\det(\mathcal{L})^{1/n}$.*

For technical reasons, we will also need the following variant of HSVP$_p$. Here, we add an additional parameter that one can think of as effectively bounding the bit length of the input.

**Definition 2.20.** *For any $p \geq 1$, $\ell := \ell(n)$ and $\gamma := \gamma(n)$, $(\gamma,\ell)$-HSVP$_p$ (the Hermite Shortest Vector Problem) is the search problem in which the input is a basis $\mathbf{B} \in \mathbb{Z}^{n\times n}$ for a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ with $\det(\mathcal{L}) \leq 2^{\ell(n)}$, and the goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y}\|_p \leq \gamma\det(\mathcal{L})^{1/n}$.*

**Lemma 2.21.** *For any $p \geq 1$, $\ell := \ell(n) \leq \mathrm{poly}(n)$ and $\gamma := \gamma(n)$, there is a Karp reduction from $\gamma'$-HSVP$_p$ to $(\gamma,\ell)$-HSVP$_p$, where $\gamma' := (1 + 2^{n^3-\ell/n})\gamma$.*

## 2.5 A list-decoding bound on Reed-Solomon codes

**Lemma 2.22** ([Sud97, GRS00])**.** *Let $\mathbb{F}$ be a finite field, and let $\alpha_1,\ldots,\alpha_L \in \mathbb{F}$ be* distinct *field elements. Then, for any $\beta_1,\ldots,\beta_L \in \mathbb{F}$ and any degree bound $d \geq 1$, there are at most $\sqrt{2L/d}$ polynomials $p_i \in \mathbb{F}[x]$ such that for all $i$,*

$$|\{j \ :\ p_i(\alpha_j) = \beta_j\}| \geq \sqrt{2dL}\ .$$

**Corollary 2.23.** *Let $\mathbb{F}$ be a finite field, and let $\alpha_1,\ldots,\alpha_{L_2} \in \mathbb{F}$ be distinct field elements. Let $p_{i,j}$ be polynomials with degree at most $d \leq L_2/2$ for all $1 \leq i \leq L_2$ and $1 \leq j \leq L_1$ such that $p_{i,j} \neq p_{i,j'}$ for any $i$ and any $j \neq j'$, and such that*

$$p_{i,j}(\alpha_i) = p_{i',j'}(\alpha_{i'})$$

*for all $i,j,i',j'$. Then, there are at least*

$$L' := L_1\cdot\sqrt{L_2/(2d)} - L_2 + \sqrt{2dL_2}$$

*distinct polynomials $p_{i,j}$ (i.e., $|\{p_{i,j}\}| \geq L'$).*

*Proof.* For any $i$, let $P_i := \{p_{i',j'} \ : \ p_{i',j'}(\alpha_i) = p_{i,1}(\alpha_i)\}$. (The choice of the index 1 here is arbitrary because $p_{i,j}(\alpha_i) = p_{i,1}(\alpha_i)$ for all $j$.) Notice that $P_i$ contains $p_{i,j}$ for all $j$, and in particular, since $p_{i,j} \neq p_{i,j'}$ for $j \neq j'$, we have $|P_i| \geq L_1$. (We stress that $P_i$ is a set of polynomials, not a set of indices. So, it does not count duplicate polynomials.) We wish to show that $P := \bigcup_i P_i$ is large.

For any polynomial $p$, let $C(p)$ be the number of distinct values of $i$ such that $p \in P_i$. Let $P^*$ be the set of all polynomials $p$ such that $C(p) \geq \sqrt{2dL_2}$. By Lemma 2.22, $|P^*| \leq \sqrt{2L_2/d}$. Therefore, we have

$$L_1 L_2 \leq \sum_{i=1}^{L_2} |P_i| = \sum_{p \in P} C(p) \leq L_2 |P^*| + \sqrt{2dL_2} \cdot |P \setminus P^*| = L_2 |P^*| + \sqrt{2dL_2}(|P| - |P^*|) \ .$$

Rearranging, we see that

$$|P| \geq L_1 \cdot \sqrt{L_2/(2d)} - |P^*| \cdot \left(\sqrt{L_2/(2d)} - 1\right) \geq L_1 \cdot \sqrt{L_2/(2d)} - L_2 + \sqrt{2dL_2} = L' \ ,$$

as needed. □

# 3 Coding Problems

## 3.1 The (list) Singleton and Plotkin bounds

We start by giving a fairly general reduction from SDP (the problem of finding a pair of close codewords) to Pigeon$^L$. On input an instance $\mathcal{C}$ of SDP, the reduction works by defining a compressing circuit $\mathcal{C}'$ that truncates the output of $\mathcal{C}$. It then finds $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ such that $\mathcal{C}'(\boldsymbol{x}_1) = \cdots = \mathcal{C}'(\boldsymbol{x}_L)$ using its Pigeon$^L$ oracle, and finally outputs a pair $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ that minimizes $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j))$.

**Theorem 3.1.** *Let $k, m, n, q, L \in \mathbb{Z}^+$ be such that $m \leq k \leq n$ and $2 \leq L \leq q^{k-m}$. Then there is a Karp reduction from $(n, k, d)$-SDP$_q$ to $(q^k, q^m)$-Pigeon$^L$ where $d := d_{[q]^{n-m},\Delta}(n-m, L)$.*[9]

*Proof.* On input a circuit $\mathcal{C} : [q]^k \to [q]^n$, the reduction first constructs $\mathcal{C}' : [q]^k \to [q]^m, \mathcal{C}'(x) = \mathcal{C}(x)_{[1,m]}$. It then calls its $(q^k, q^m)$-Pigeon$^L$ oracle on $\mathcal{C}'$, receiving as output distinct vectors $\mathbf{x}_1, \ldots, \mathbf{x}_L \in [q]^k$. Finally, the reduction iterates through all pairs $\boldsymbol{x}_i, \boldsymbol{x}_j$ with $i \neq j$, and outputs a pair that minimizes $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j)) = \Delta(\mathcal{C}(\boldsymbol{x}_i)_{[m+1,n]}, \mathcal{C}(\boldsymbol{x}_j)_{[m+1,n]})$.

The reduction clearly runs in polynomial time, and by construction $\mathcal{C}' : [q]^k \to [q]^m$ is a valid instance of $(q^k, q^m)$-Pigeon$^L$. We now show that $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j)) \leq d$. Indeed,

$$\begin{aligned}
\Delta(\mathcal{C}(\boldsymbol{x}_i), C(\boldsymbol{x}_j)) &= \Delta(\mathcal{C}(\boldsymbol{x}_i)_{[1,m]}, C(\boldsymbol{x}_j)_{[1,m]}) + \Delta(\mathcal{C}(\boldsymbol{x}_i)_{[m+1,n]}, C(\boldsymbol{x}_j)_{[m+1,n]}) \\
&= \Delta(\mathcal{C}'(\boldsymbol{x}_i), \mathcal{C}'(\boldsymbol{x}_j)) + \Delta(\mathcal{C}(\boldsymbol{x}_i)_{[m+1,n]}, C(\boldsymbol{x}_j)_{[m+1,n]}) \\
&= \Delta(\mathcal{C}(\boldsymbol{x}_i)_{[m+1,n]}, C(\boldsymbol{x}_j)_{[m+1,n]}) \\
&\leq d_{[q]^{n-m},\Delta}(n-m, L) \ ,
\end{aligned}$$

where the second equality follows from the definition of $\mathcal{C}'$, the third equality follows from the fact that $\mathcal{C}'(\boldsymbol{x}_i) = \mathcal{C}'(\boldsymbol{x}_j)$, and the inequality follows by the behavior of the reduction and applying the definition of $d_{[q]^{n-m},\Delta}(n-m, L)$ to the $L$ vectors $\mathcal{C}(\boldsymbol{x}_1)_{[m+1,n]}, \ldots, C(\boldsymbol{x}_L)_{[m+1,n]} \in [q]^{n-m}$ (see Definition 2.1). □

We get useful corollaries from Theorem 3.1 that show how to compute vectors achieving Singleton bound (the $L = 2$ case) and the Plotkin bound (for larger $L$) using a Pigeon$^L$ oracle.

We start by stating the Singleton bound (see [GRS23, Theorem 4.3.1]).

[9] Here $d_{[q]^{n-m},\Delta}(n-m, L)$ uses Definition 2.1 and corresponds to the maximal value of the minimum distance of a code with size $L$ in $[q]^{n-m}$. (More generally, $d_{[q]^\ell,\Delta}(e, L)$ is the maximal value of the minimum distance of a code in $[q]^\ell$ with $L$ points *that all lie in a Hamming ball of radius $e$.*)

**Theorem 3.2** (Singleton bound; [Sin64])**.** *Let $k, n, d \in \mathbb{Z}^+$ and let $q$ be a prime power. Then every $(n, k, d)_q$ code satisfies $d \leq n - k + 1$.*

We now show that the problem of finding a pair of codewords whose distance is at most the Singleton bound is in PWPP.

**Corollary 3.3** (Singleton bound in PWPP)**.** *Let $k, n, q \in \mathbb{Z}^+$ with $k \leq n$. Then there is a Karp reduction from $(n, k, n-k+1)$-SDP$_q$ to $(q^k, q^{k-1})$-Pigeon$^2$. In particular, $(n, k, n-k+1)$-SDP$_q$ is in PWPP.*

*Proof.* The claim follows from Theorem 3.1 by setting $m := k - 1$ and noting that, trivially, $d_{[q]^{n-m},\Delta}(n - m, L) = d_{[q]^{n-k+1},\Delta}(n - k + 1, L) \leq n - k + 1$ for any $L \geq 2$. □

We now move to the Plotkin bound. We start by stating (the contrapositive of) the Plotkin bound. See [GRS23, Theorem 4.4.1].

**Theorem 3.4** (Plotkin bound; [Plo60])**.** *Let $n, k, d, q \in \mathbb{Z}^+$ and let $\mathcal{C}$ be an $(n, k, d)_q$ code. If $q^k > qd/(qd - (q-1)n)$ then $d \leq (1 - 1/q)n$.*

We now show that the problem of computing a pair of codewords whose distance satisfies the Plotkin bound is in PMPP.

**Corollary 3.5** (Plotkin bound in PMPP)**.** *Let $k, n, d, m, q, L \in \mathbb{Z}^+$ be such that $k \leq n$ and*

$$\frac{qd}{qd - (q-1)(n-m)} < L \leq q^{k-m} \ . \tag{1}$$

*Then $d \leq (1-1/q)(n-m)$ and there is a Karp reduction from $(n, k, (1-1/q)(n-m))$-SDP$_q$ to $(q^k, q^m)$-Pigeon$^L$. In particular, any $m \leq n - \lfloor qd/(q-1) \rfloor + 1$ satisfies Equation (1).*

*Proof.* By combining the lower bound in Equation (1) and Theorem 3.4, $d_{[q]^{n-m},\Delta}(n-m, L) \leq (1-1/q) \cdot (n-m)$. The claim then follows from Theorem 3.1. One can check the sufficient condition for $m$ to satisfy Equation (1), which appears in [GRS23, Corollary 4.4.2].[10] □

Next, we give a reduction from DenseBall to Pigeon similar to the reduction from SDP to Pigeon in Theorem 3.1.

**Theorem 3.6.** *Let $k, n, m, q, L \in \mathbb{Z}^+$ be such that $m < n$ and $2 \leq L \leq q^{k-m}$. Then there is a Karp reduction from $(n, k, n - m - \lfloor (n-m)/L \rfloor)$-DenseBall$_q^L$ to $(q^k, q^m)$-Pigeon$^L$.*

*Proof.* On input a circuit $\mathcal{C} : [q]^k \to [q]^n$, the reduction first computes $\mathcal{C}' : [q]^k \to [q]^m$ such that $\mathcal{C}'(\boldsymbol{x}) := \mathcal{C}(\boldsymbol{x})_{[1,m]}$. It then calls its $(q^k, q^m)$-Pigeon$^L$ oracle on $\mathcal{C}'$, receiving as output $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ such that $\mathcal{C}'(\boldsymbol{x}_1) = \cdots = \mathcal{C}'(\boldsymbol{x}_L)$. After that, it computes $\mathcal{C}(\boldsymbol{x}_1), \ldots, \mathcal{C}(\boldsymbol{x}_L)$, and constructs a center $\boldsymbol{y}$ as follows. It sets $\boldsymbol{y}_{[1,m]} := \mathcal{C}'(\boldsymbol{x}_1)$ and sets the remaining $n - m$ coordinates of $\boldsymbol{y}$ so that at least $\lfloor (n-m)/L \rfloor$ of them agree with each codeword $\mathcal{C}(\boldsymbol{x}_i)$ for $i \in [L]$. Finally, it outputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L, \boldsymbol{y}$.

It is clear that the reduction runs in polynomial time, and by construction $\mathcal{C}' : [q]^k \to [q]^m$ is a valid instance of $(q^k, q^m)$-Pigeon$^L$. Moreover,

$$\Delta(\boldsymbol{y}, \mathcal{C}(\boldsymbol{x}_i)) = \Delta(\boldsymbol{y}_{[m+1,n]}, \mathcal{C}(\boldsymbol{x}_i)_{[m+1,n]}) \leq n - m - \lfloor (n-m)/L \rfloor$$

for all $i \in [L]$, as needed. □

Now, we state the list Singleton bound, which was first stated relatively recently in [ST20, Theorem 1.2].[11]

**Theorem 3.7** (List Singleton bound; [ST20])**.** *Let $n, k, t, q, L \in \mathbb{Z}^+$. Then for every $(t, L)$-list-decodable code $\mathcal{C} : [q]^k \to [q]^n$,*

$$q^k \leq L q^{n - \lfloor (L+1)t/L \rfloor} \ .$$

[10] The parameterization in [GRS23] is different: $m$ here corresponds to $n - n'$ there.

[11] The bound is called the "generalized Singleton bound" in [ST20].

Finally, we show that the problem of computing a set of codewords lying in a ball whose radius *almost* satisfies the list Singleton bound is in PMPP.

**Corollary 3.8** (List Singleton bound in PMPP)**.** *Let $n, k, q, L \in \mathbb{Z}^+$ with $k > \lfloor \log_q(L) \rfloor$, let $m := k - \lfloor \log_q L \rfloor$, and let*

$$t := n - m + \lfloor (n-m)/L \rfloor = n - k + \lfloor \log_q L \rfloor - \lfloor (n - k + \lfloor \log_q L \rfloor)/L \rfloor \ . \tag{2}$$

*Then $(n, k, t)$-DenseBall$_q^L$ is in $(q^k, q^m)$-PMPP$^L$.*

*Proof.* Apply Theorem 3.6 with $m := k - \lfloor \log_q L \rfloor$. □

We remark that the radius $t$ in Equation (2) is almost as small the smallest $t$ satisfying Theorem 3.7. Indeed, one can check that any $t$ satisfying Theorem 3.7 must be such that

$$t > (1 - 1/L) \cdot (n - k + \log_q(L-1)) \ ,$$

and that the radius $t$ in Equation (2) satisfies

$$t \le (1 - 1/L) \cdot (n - k + \lfloor \log_q(L) \rfloor) + 1 \ .$$

## 3.2 The (list) Hamming and Elias-Bassalygo bounds

### 3.2.1 The (list) Hamming bound

We will use the following "list Hamming bound," which is a direct consequence of the pigeonhole principle. See [Ham50] and see [GRS23, Theorem 8.1.1] for a very similar result.

**Theorem 3.9** (List Hamming bound)**.** *Let $n, k, t, q \in \mathbb{Z}^+$ with $t \le n$, and let $\mathcal{C} : [q]^k \to [q]^n$. Then there exists a center $\boldsymbol{y} \in [q]^n$ such that*

$$\left| \{ \boldsymbol{x} \in [q]^k : \mathcal{C}(\boldsymbol{x}) \in \mathcal{H}_q^n(\boldsymbol{y}, t) \} \right| \ge \left\lceil \frac{q^k \cdot V_q^n(t)}{q^n} \right\rceil \ .$$

*In particular, $(n, k, t)$-DenseBall$_q^L$ is a total search problem for any $L \le \lceil q^{k-n} \cdot V_q^n(t) \rceil$.*

We now give a reduction from DenseBall to Pigeon$^L$, the key to which is the circuit $\mathcal{C}_H^{n,V,q}$ giving an injection from $[V]$ into a Hamming ball for sufficiently large $V$ defined in Lemma A.9. (In fact, we will use such circuits $\mathcal{C}_H^{n,V,q}$ that are *bijective*.)

**Theorem 3.10.** *Let $n, k, t, L \in \mathbb{Z}^+$, and let $q$ be a prime power satisfying*

$$L \le \left\lceil \frac{q^k \cdot V_q^n(t)}{q^n} \right\rceil \ .$$

*Then there is a Karp reduction from $(n, k, t)$-DenseBall$_q^L$ to $(q^k \cdot V_q^n(t), q^n)$-Pigeon$^L$.*

*Proof.* Let $V := V_q^n(t)$. On input an instance $\mathcal{C} : [q]^k \to [q]^n$ of $(n, k, t)$-DenseBall$_q^L$, the reduction constructs a circuit $\mathcal{C}' : [q]^k \times [V] \to [q]^n$ such that $\mathcal{C}'(\boldsymbol{x}, \boldsymbol{y}) := \mathcal{C}(\boldsymbol{x}) + \mathcal{C}_H^{n,V,q}(\boldsymbol{y})$, where $\mathcal{C}_H^{n,V,q} : [V] \to \mathcal{H}_q^n(\boldsymbol{0}, t)$ is the circuit defined in Lemma A.9. It then calls its $(q^k \cdot V_q^n(t), q^n)$-Pigeon$^L$ oracle on $\mathcal{C}'$, receiving as output $L$ pairs $(\boldsymbol{x}_1, z_1), \dots, (\boldsymbol{x}_N, z_L) \in [q]^k \times [V]$. Finally, it returns the codewords $\mathcal{C}(\boldsymbol{x}_1), \dots, \mathcal{C}(\boldsymbol{x}_L)$ and center $\boldsymbol{y} := \mathcal{C}'(\boldsymbol{x}_1, z_1)$.

Constructing $\mathcal{C}_H^{n,V,q}(\boldsymbol{y})$ requires at most $\mathrm{poly}(\log V, n) = \mathrm{poly}(n)$ time by Lemma A.9, and so the reduction runs in polynomial time. Moreover, $\mathcal{C}'$ is a valid instance of $(q^k \cdot V_q^n(t), q^n)$-Pigeon$^L$. Furthermore, $\mathcal{C}'(\boldsymbol{x}_1, z_1) = \cdots = \mathcal{C}'(\boldsymbol{x}_L, z_L) = \boldsymbol{y}$ by assumption, and so $\Delta(\boldsymbol{y}, \mathcal{C}(\boldsymbol{x}_i)) \le t$ for all $i \in [L]$ by definition of $\mathcal{C}'$ and $\mathcal{C}_H^{n,V,q}(\boldsymbol{y})$. The reduction is therefore correct. □

We now use Theorem 3.10 to show that SDP and DenseBall with parameters corresponding to the (list) Hamming bound are in (one or more of) PPP, PWPP, and PMPP.

**Corollary 3.11** ((List) Hamming bound in PPP, PWPP, and PMPP)**.** *Let $n, k, t, q, L \in \mathbb{Z}^+$ with $2 \le q \le \mathrm{poly}(n)$ and $L \le \mathrm{poly}(n)$. Let*

$$\widetilde{L} := \frac{q^k \cdot V_q^n(t)}{q^n} \ .$$

*Then:*

1. *If $\widetilde{L} > 1$, then $(n, k, 2t)$-SDP$_q$ is in PPP.*
2. *If $\widetilde{L} \ge 1 + 1/\mathrm{poly}(n)$, then $(n, k, 2t)$-SDP$_q$ is in PWPP.*
3. *If $\widetilde{L} \ge L$, then $(n, k, t)$-DenseBall$_q^L$ is in $(q^k \cdot V_q^n(t), q^n)$-PMPP$^L$.*

*Proof.* Items 1 and 2 follow from Theorem 3.10, the respective characterizations of PWPP and PPP in Theorems 2.7 and 2.8, and the equivalence between $(n, k, 2t)$-SDP$_q$ and $(n, k, t)$-DenseBall$_q^2$ in Lemma 2.16. Item 3 follows from Theorem 3.10 and the definition of PMPP. □

### 3.2.2 E pluribus duo

We now give a simple reduction from SDP to DenseBall$^L$ for $L \ge 2$. The idea is that a ball of small radius $t$ containing $L \ge 2$ codewords $\mathcal{C}(\boldsymbol{x}_1), \ldots, \mathcal{C}(\boldsymbol{x}_L)$ must contain a pair of codewords at small distance $d$. Clearly such a pair must exist at distance at most $2t$, but in fact when $L$ is larger it is possible to get a better upper bound. So, the reduction simply computes the distance $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j))$ between each pair of codewords $\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j)$ for $i \ne j$, and outputs a pair $\boldsymbol{x}_i, \boldsymbol{x}_j$ that minimizes $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j))$.

**Theorem 3.12.** *Let $n, k, t, q, L \in \mathbb{Z}^+$ be such that*

$$L \le \left\lceil \frac{q^k \cdot V_q^n(t)}{q^n} \right\rceil .$$

*Then there is a $\mathrm{poly}(n, L, \log q)$-time Karp reduction from $(n, k, d)$-SDP$_q$ to $(n, k, t)$-DenseBall$_q^L$, where $d := d_{[q]^n, \Delta}(t, L)$.*

*Proof.* Let $\mathcal{C} : [q]^k \to [q]^n$ be the input instance of $(n, k, d)$-SDP$_q$. The reduction calls its $(n, k, t)$-DenseBall$_q^L$ oracle on $\mathcal{C}$, receiving as output vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L \in [q]^k$ and $\boldsymbol{y} \in [q]^n$. It then outputs a pair of vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$ that minimizes $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j))$ among all pairs $\boldsymbol{x}_i, \boldsymbol{x}_j$ with $i \ne j$.

It is clear that the reduction runs in $\mathrm{poly}(n, L, \log q)$ time, and it remains to show correctness. We note that $(n, k, t)$-DenseBall$_q^L$ is total and that $\mathcal{C}$ is a valid instance of $(n, k, t)$-DenseBall$_q^L$ by assumption and Theorem 3.9. Therefore, the output vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L \in [q]^k$ and center $\boldsymbol{y}$ must satisfy $\Delta(\boldsymbol{y}, \mathcal{C}(\boldsymbol{x}_i)) \le t$ for all $i \in [L]$. So, it follows by the definition of $d_{[q]^n, \Delta}(t, L)$ (see Definition 2.1) that $\Delta(\mathcal{C}(\boldsymbol{x}_i), \mathcal{C}(\boldsymbol{x}_j)) \le d$ holds for some $i \ne j$, as needed. □

### 3.2.3 The Elias-Bassalygo bound

We start by giving the Johnson bound (see [GRS23, Theorem 7.3.1]).

**Theorem 3.13** (Johnson bound; [Joh62])**.** *Let $n, k, d, q \in \mathbb{Z}^+$ and $q \ge 2$, and let*

$$J_q(\delta) := \left(1 - \frac{1}{q}\right) \cdot \left(1 - \left(1 - \frac{q\delta}{q-1}\right)^{1/2}\right) . \tag{3}$$

*If $\mathcal{C}$ is a $(n, k, d)_q$ code and $t < J_q(d/n) \cdot n$, then $\mathcal{C}$ is a $(t, qdn)$-list decodable code.*

We now state the Elias-Bassalygo bound. See [GRS23, Theorem 8.1.1] and its proof.[12]

**Theorem 3.14** (Elias-Bassalygo bound; [Bas65])**.** *Let $n, k, d \in \mathbb{Z}^+$, let $q$ be a prime power. Then every $(n, k, d)_q$ code satisfies*

$$q^k \leq \frac{q^{n+1} dn}{V_q^n(t)} \ ,$$

*where $t := \lfloor J_q(d/n) \cdot n \rfloor - 1$.*

We are now ready to show that the problem of computing a pair of codewords satisfying the Elias-Bassalygo bound is in PMPP.

**Corollary 3.15** (Elias-Bassalygo bound in PMPP)**.** *Let $n, k, d, t, q \in \mathbb{Z}^+$ with $t < J_q(d/n) \cdot n$, let $L := qnd+1$, and suppose that*

$$L \leq \left\lceil \frac{q^k \cdot V_q^n(t)}{q^n} \right\rceil .$$

*Then there is a $\mathrm{poly}(n, q)$-time Karp reduction from $(n, k, d-1)$-$\mathrm{SDP}_q$ to $(q^k \cdot V_q^n(t), q^n)$-$\mathrm{Pigeon}^L$. In particular, $(n, k, d-1)$-$\mathrm{SDP}_q$ is in $(q^k \cdot V_q^n(t), q^n)$-$\mathsf{PMPP}^L$.*

*Proof.* The reduction first reduces $(n, k, d-1)$-$\mathrm{SDP}_q$ to $(n, k, t)$-$\mathrm{DenseBall}_q^L$ using Theorem 3.12, which applies because $d_{[q]^n,\Delta}(t, L) < d$ by the contrapositive of the Johnson bound (Theorem 3.13). It then reduces $(n, k, t)$-$\mathrm{DenseBall}_q^L$ to $(q^k \cdot V_q^n(t), q^n)$-$\mathrm{Pigeon}^L$ using Theorem 3.10. □

## 3.3 Hardness of SDP and DenseBall

### 3.3.1 PWPP-hardness of SDP

We now turn to proving *hardness results* for SDP and DenseBall. In each of our hardness reductions we first assume that a gadget code, specified by a circuit $\mathcal{C}_A$, is given to the reduction as auxiliary input. We then instantiate the gadget code $\mathcal{C}_A$ with an explicit efficiently computable code $\mathcal{C}_A$ to obtain a true Karp reduction.

**Theorem 3.16.** *Let $k, m, n, d, d' \in \mathbb{Z}^+$ be such that $m < k$ and $d < d'$. Let $q$ be a prime power. Then there is a Karp reduction from $(q^k, q^m)$-$\mathrm{Pigeon}^2$ to $(n, k, d)$-$\mathrm{SDP}_q$ that takes a circuit $\mathcal{C}_A : \mathbb{F}_q^m \to \mathbb{F}_q^n$ defining an $(n, m, d')_q$ code as auxiliary input.*

*Proof.* On input an instance $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^m$ of $(q^k, q^m)$-$\mathrm{Pigeon}^2$, the reduction first computes a circuit $\mathcal{C}'(\boldsymbol{x}) := \mathcal{C}_A(\mathcal{C}(\boldsymbol{x}))$. It then calls its $(n, k, d)$-$\mathrm{SDP}_q$ oracle on $\mathcal{C}'$, receiving as output $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{F}_q^n$, $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ such that $\mathcal{C}'(\boldsymbol{x}_1) = \mathcal{C}'(\boldsymbol{x}_2)$. Finally, it outputs $\boldsymbol{x}_1, \boldsymbol{x}_2$.

It is clear that the reduction runs in polynomial time (recall that $\mathcal{C}_A$ is given as auxiliary input), and it remains to show correctness. Because $\mathcal{C}$ is compressing, there exist $\boldsymbol{x}_1' \neq \boldsymbol{x}_2'$ such that $\mathcal{C}'(\boldsymbol{x}_1') = \mathcal{C}'(\boldsymbol{x}_2')$, i.e., $\mathcal{C}'$ has distance 0. It is therefore a valid instance of $(n, k, d)$-$\mathrm{SDP}_q$. Moreover, by the construction of $\mathcal{C}'$ and the guarantee of the $(n, k, d)$-$\mathrm{SDP}_q$ oracle, we have that $\Delta(\mathcal{C}_A(\mathcal{C}(\boldsymbol{x}_1)), \mathcal{C}_A(\mathcal{C}(\boldsymbol{x}_2))) \leq d$. But, because $\mathcal{C}_A$ has distance $d' > d$, this implies that $\mathcal{C}(\boldsymbol{x}_1) = \mathcal{C}(\boldsymbol{x}_2)$, as needed. □

We will instantiate the reduction in Theorem 3.16 with codes $\mathcal{C}_A$ meeting the Zyablov bound, which can be computed in polynomial time (see [GRS23, Theorem 10.2.1]). In fact, what we state is the special case of the (effective) Zyablov bound with sub-constant rate.[13]

[12] The non-asymptotic version of Elias-Bassalygo bound that we state appears in the *proof* of [GRS23, Theorem 8.1.1].

[13] We note that the premise of [GRS23, Theorem 10.2.1] is stated with the requirement $\delta < 1/2$. While $\delta < 1/2$ is necessary for the $q = 2$ case, for general constant $q$, the result holds for $\delta < 1 - 1/q$, as in Theorem 3.17. It is also evident from the proof of the Zyablov bound that the result extends to superconstant $q$ as well—i.e., that for any prime power $q$ that is an unbounded function of $n$ and any $k = o(n)$, there is an efficiently computable family of codes $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with relative distance $1 - \varepsilon$.

**Theorem 3.17** (Zyablov bound [Zya71])**.** *For any (efficiently computable) $k = k(n) = o(n)$, constant prime power $q$, and constant $\varepsilon > 0$, there is a $\mathrm{poly}(n)$-time algorithm that takes as input $n$ and computes (a circuit representing) a code $\mathcal{C}_q^{\mathsf{Zyb}} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with relative distance $\delta \geq 1 - 1/q - \varepsilon$ for sufficiently large $n$.*

We now show that SDP is PWPP-hard on $q$-ary codes of relative distance $\delta := d/n$ less than $1 - 1/q$ and any constant rate. This hardness corresponds to the red shaded region in the top plot in Figure 1. We remark that $(n, k, \delta n)$-SDP$_q$ is easy for $\delta \geq 1 - 1/q$. In particular, for such any such $\delta$, this problem can be solved by choosing any collection of polynomially many codewords and outputting the closest pair among them. So, Corollary 3.18 shows essentially tight hardness of SDP in terms of $\delta$.

**Corollary 3.18** (PWPP-hardness of SDP)**.** *Let $q$ be a fixed prime power and let $\varepsilon, \varepsilon' > 0$ be constants. Then for all sufficiently large $n, k, d \in \mathbb{Z}^+$ with $k \leq n \leq \mathrm{poly}(k)$ and $d \leq (1 - 1/q - \varepsilon')n$, there is a Karp reduction from $(q^k, q^{k^{1-\varepsilon}})$-Pigeon$^2$ to $(n, k, d)$-SDP$_q$.*

*In particular, $(n, Rn, \delta n)$-SDP$_q$ is PWPP-hard for any constant rate for any constant rate $R \in (0, 1]$ and constant relative distance $\delta \in (0, 1 - 1/q)$.*

*Proof.* Apply Theorem 3.16 with the circuits $\mathcal{C}_A = \mathcal{C}_q^{\mathsf{Zyb}}$ constructed in Theorem 3.17. □

*Remark* 3.19. We remark that the assumption $k \leq n$ (equivalently, $R \leq 1$) in Corollary 3.18 is not necessary, except to ensure that the instance $\mathcal{C}'$ of SDP output by the reduction meets the definition of a code used there. In fact, modifying Corollary 3.18 slightly shows PWPP-hardness of "$(n, k, d)$-SDP$_q$" for any $n \geq \Omega(k^\varepsilon)$ for constant $\varepsilon > 0$ (and any constant relative distance less than). This is because the hard instances $\mathcal{C}'$ of SDP constructed in Corollary 3.18 are such that either $\mathcal{C}'(\boldsymbol{x}) = \mathcal{C}'(\boldsymbol{y})$ or $\Delta(\mathcal{C}'(\boldsymbol{x}), \mathcal{C}'(\boldsymbol{y}))$ is large for $\boldsymbol{x} \neq \boldsymbol{y}$.

We also extend Corollary 3.18 to codes in systematic form (and in particular to codes with distance $d \geq 1$, which are represented by injective circuits; see Section 1.4). This hardness corresponds to the red shaded region in the bottom plot in Figure 1.

**Corollary 3.20** (PWPP-hardness of sysSDP)**.** *Let $q$ be a fixed prime power and let $\varepsilon, \varepsilon' > 0$ be constants. Then for all sufficiently large $n, k, d \in \mathbb{Z}^+$ with $k \leq n - k \leq \mathrm{poly}(k)$ and $d \leq (1 - 1/q - \varepsilon')(n - k)$, there is a Karp reduction from $(q^k, q^{(n-k)^{1-\varepsilon}})$-Pigeon$^2$ to $(n, k, d)$-sysSDP$_q$.*

*In particular, $(n, Rn, \delta n)$-sysSDP$_q$ is PWPP-hard for any constant rate $R \in (0, 1)$ and constant relative distance $\delta \in (0, (1 - 1/q) \cdot (1 - R))$.*

*Proof.* Combine Corollary 3.18 and Lemma 2.13 with block length $n - k$. □

### 3.3.2 PMPP-hardness of DenseBall

We next show a reduction from Pigeon$^L$ for $L \geq 2$ to DenseBall analogous to Theorem 3.16.

**Theorem 3.21.** *Let $k, m, n, t, L, L_A, L' \in \mathbb{Z}^+$ and let $q$ be a prime power. Suppose that $L' \leq \min\{\mathrm{poly}(n), q^{n-k}\}$ and $L \leq \min\{\lceil L'/L_A \rceil, q^{k-m}\}$. Then there is a Karp reduction from $(q^k, q^m)$-Pigeon$^L$ to $(n, k, t)$-DenseBall$_q^{L'}$ that takes a circuit $\mathcal{C}_A : \mathbb{F}_q^m \to \mathbb{F}_q^n$ defining a $(t, L_A)$-list decodable-code with minimum distance at least 1 (i.e., $\mathcal{C}_A$ is injective) as auxiliary input.*

*Proof.* Let $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^m$ be the input instance of $(q^k, q^m)$-Pigeon$^L$. The reduction first computes $\mathcal{C}' : \mathbb{F}_q^k \to \mathbb{F}_q^n$, $\mathcal{C}'(\boldsymbol{x}) := \mathcal{C}_A(\mathcal{C}(\boldsymbol{x}))$. It then calls its $(n, k, t)$-DenseBall$_q^{L'}$ oracle on $\mathcal{C}'$, receiving as output $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L'} \in \mathbb{F}_q^n$ and $\boldsymbol{y} \in \mathbb{F}_q^k$. Next, it computes $\boldsymbol{z}_1 := \mathcal{C}'(\boldsymbol{x}_1), \ldots, \boldsymbol{z}_{L'} := \mathcal{C}'(\boldsymbol{x}_{L'})$, and computes an index $i^*$ that maximizes $|X_i|$, where $X_i := \{\boldsymbol{x}_j : j \in [L'], \mathcal{C}'(\boldsymbol{x}_j) = \boldsymbol{z}_i\}$. Finally, it outputs the vectors in $X_{i^*}$.

The reduction runs in polynomial time (recall that $\mathcal{C}_A$ is given as auxiliary input), and it remains to show correctness. Because $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ and $L' \leq q^{n-k}$, there exist $\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{L'} \in \mathbb{F}_q^n$ such that $\mathcal{C}'(\boldsymbol{x}'_1) = \cdots = \mathcal{C}'(\boldsymbol{x}'_{L'})$ by the construction of $\mathcal{C}'$. It follows that $\mathcal{C}'$ is a valid instance of $(n, k, t)$-DenseBall$_q^{L'}$. Moreover, by the guarantee of the $(n, k, t)$-DenseBall$_q^{L'}$ oracle, it holds that $\Delta(\boldsymbol{y}, \boldsymbol{z}_i) \leq r$ for all $i \in [L']$. On the other hand,

because $\mathcal{C}_A$ is $(t, L_A)$-list decodable $|\{\boldsymbol{z}_i : i \in [L']\}| \leq L_A$. Therefore, $\mathcal{C}'$ must map at least $\lceil L'/L_A \rceil$ vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L'}$ to $\boldsymbol{z}_{i^*}$, i.e.,

$$|X_{i^*}| \geq \lceil L'/L_A \rceil \geq L \ .$$

Furthermore, because $\mathcal{C}_A$ is injective, the fact that $\mathcal{C}'(\boldsymbol{x}) = \mathcal{C}_A(\mathcal{C}(\boldsymbol{x})) = \boldsymbol{z}_{i^*}$ for all $\boldsymbol{x} \in X_{i^*}$ implies that $\mathcal{C}(\boldsymbol{x}) = \mathcal{C}(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in X_{i^*}$, as needed. □

We now state a result on explicit codes that nearly achieve list decoding capacity [GR08] (see also [GRS23, Theorem 17.3.8]). We note in passing that these codes are folded Reed-Solomon codes, but we will only use them in a black-box way as our gadget codes $\mathcal{C}_A$ in Theorem 3.21.

**Theorem 3.22** ([GR08], [GRS23, Theorem 17.3.8])**.** *For any constant rate $R^* \in (0,1)$, any sufficiently small constant $\varepsilon > 0$, and all sufficiently large $n \in \mathbb{Z}^+$, there exist linear $q$-ary $((1 - R^* - \varepsilon)n, L_A)$-list-decodable codes $\mathcal{C}_q^{\mathsf{FRS}}$ of dimension $R^* \cdot n$ and sufficiently large block length $n$, for some*

$$L_A \leq \left(\frac{n}{\varepsilon}\right)^{O(1/\varepsilon)} \ , \quad q \leq \left(\frac{n}{\varepsilon}\right)^{O(1/\varepsilon^2)} \ .$$

*Furthermore, there is a $\mathrm{poly}(n)$-time algorithm for computing (a circuit representing) such codes $\mathcal{C}_q^{\mathsf{FRS}}$.*

Finally, we use Theorem 3.22 to prove PMPP-hardness of DenseBall.

**Corollary 3.23** (PMPP-hardness of DenseBall)**.** *For any constants $R, \rho \in (0,1)$ and positive integers $m := m(n) < o(n)$ and $L := L(n) \leq \mathrm{poly}(n)$, there exists a prime power $q := q(n) \leq \mathrm{poly}(n)$ and a list size $L' := L'(n) \leq \mathrm{poly}(n)$ such that there is a Karp reduction from $(q^k, q^m)$-Pigeon$^L$ to $(n, k, \rho n)$-DenseBall$_q^{L'}$, where $k := k(n) = \lfloor Rn \rfloor$.*

*In particular, for these parameters, $(n, k, \rho n)$-DenseBall$_q^{L'}$ is $(q^k, q^m)$-PMPP$^L$-hard.*

*Proof.* Let $R^* := (1-\rho)/2 > 0$ and $\varepsilon := (1-\rho)/2 > 0$, so that $\rho = 1 - R^* - \varepsilon$. By Theorem 3.22, there exists an efficiently computable $(\rho n, L_A)$-list-decodable code $\mathcal{C}^{\mathsf{FRS}} : \mathbb{F}_q^{R^* n} \to \mathbb{F}_q^n$ with

$$L_A \leq (n/\varepsilon)^{O(1/\varepsilon)} \leq \mathrm{poly}(n) \ ,$$

and

$$q \leq (n/\varepsilon)^{O(1/\varepsilon^2)} \leq \mathrm{poly}(n) \ .$$

Furthermore, since $m = o(n)$ and $R^*$ is a constant, it follows that for sufficiently large $n$, $R^* \cdot n < m$. Therefore, by trivially truncating the input of $\mathcal{C}^{\mathsf{FRS}}$, we obtain a $(\rho n, L_A)$-list-decodable code $\mathcal{C}_A : \mathbb{F}_q^m \to \mathbb{F}_q^n$. The result then follows from Theorem 3.21. □

As with SDP, we also show hardness of DenseBall for codes in systematic form. (See Section 1.4.)

**Corollary 3.24** (PMPP-hardness of sysDenseBall)**.** *For any constants $R \in (0,1)$ and $0 < \rho < 1 - R$ and positive integers $m := m(n) < o(n)$ and $L := L(n) \leq \mathrm{poly}(n)$, there exists a prime power $q = q(n) \leq \mathrm{poly}(n)$ and a list size $L' = L'(n) \leq \mathrm{poly}(n)$ such that there is a Karp reduction from $(q^k, q^m)$-Pigeon$^L$ to $(n, k, \rho n)$-sysDenseBall$_q^{L'}$, where $k := k(n) = \lfloor Rn \rfloor$.*

*In particular, for these parameters, $(n, k, \rho n)$-sysDenseBall$_q^{L'}$ is $(q^k, q^m)$-PMPP$^L$-hard*

*Proof.* Combine Corollary 3.23 and Lemma 2.15 with block length $n - k$. □

# 4 Finding short lattice vectors is in PMPP

In this section, we show that the problem of finding suitably short non-zero lattice vectors (in $\ell_p$ norms) can be placed in $(A, B)$-$\mathsf{PMPP}^L$, with a smooth tradeoff between the length of the vector obtained and $L$ and $B$. In particular, when $L = 2$ and $A \approx B$, we find vectors whose length is at most the bound given by Minkowski's celebrated theorem [Min10] (up to a factor of $1 + 1/n^C$ for an arbitrarily large constant $C > 0$), and when $L = \mathrm{poly}(n)$ and $A \approx LB$, we find shorter vectors, corresponding to a stronger bound due to Blichfeldt [Bli29]. (Blichfeldt proved his bound in the $\ell_2$ norm, but we generalize it. Again, we match Blichfeldt's bound up to a factor of $1 + 1/n^C$ for an arbitrarily large constant $C > 0$.)

To that end, we first prove the following technical proposition. We the derive the above results as corollaries.

**Proposition 4.1.** *There is an algorithm that takes as input an integer $p \geq 1$, radius $r \geq 1$, an integer $q \geq 1$, integers $A > B \geq 1$, and a basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$ for a lattice $\mathcal{L}(\mathbf{B})$, makes a single query to a $(A, B)$-Pigeon$^L$, and outputs a lattice vector $\boldsymbol{y} \in \mathcal{L}(\mathbf{B})$ with the following behavior. If, $L \leq \lceil A/B \rceil$, $q \geq 100pn/r$ and*

$$q^n \det(\mathcal{L}) \leq B < A \leq \left(1 - \frac{20n}{rq}\right) \cdot q^n \cdot \mathrm{vol}(\mathcal{B}_p^n(r)) \ ,$$

*then $0 < \|\boldsymbol{y}\|_p \leq d_{\ell_p^n}(r, L)$. Furthermore, the algorithm runs in time $\mathrm{poly}(n, L, \log A, \log r, \log \|\mathbf{B}\|)$.*

*Proof.* On input $p \geq 1$, $r \geq 1$, $q \geq 1$, and a basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$ for a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, the algorithm behaves as follows. The algorithm first uses the procedure from Corollary A.7 to construct the circuit $\mathcal{C}_p : [A] \to (\mathbb{Z}^n/q \cap \mathcal{B}_p^n)$, which gives an injective mapping from $[A]$ to $(\mathbb{Z}^n/q \cap \mathcal{B}_p^n)$. (Notice in particular that the upper bound on $A$ is sufficient to apply Corollary A.7.) The algorithm also constructs the injective circuit $\mathcal{C}_{\mathcal{L}} : (\mathcal{P}(\mathcal{L}) \cap \mathbb{Z}^n/q) \to [q^n \det(\mathcal{L})]$ where $\mathcal{P}(\mathbf{B}) := \{\mathbf{B}\boldsymbol{z} \ : \ \boldsymbol{z} \in [0, 1)^n\}$, as described in [BJP+19]. Finally, let $\mathcal{C} : [A] \to [B]$ be the circuit defined by $\mathcal{C}(x) := \mathcal{C}_{\mathcal{L}}(\mathcal{C}_p(x) \bmod \mathcal{L})$. (Since $B \geq q^n \det(\mathcal{L})$, this circuit is well defined.) The algorithm then calls its $(A, B)$-Pigeon$^L$ oracle on input $\mathcal{C}$, receiving as output distinct $x_1, \ldots, x_L \in [A]$ such that $\mathcal{C}(x_i) = \mathcal{C}(x_j)$ for all $i, j$. It then outputs $\mathcal{C}_p(x_i) - \mathcal{C}_p(x_j)$ where $i \neq j$ is chosen to minimize $\|\mathcal{C}_p(x_i) - \mathcal{C}_p(x_j)\|_p$.

Clearly the reduction runs in the claimed time. We first observe that the reduction does in fact output a non-zero lattice vector. Indeed, since $\mathcal{C}(x_i) = \mathcal{C}(x_j)$, we see that $\mathcal{C}_{\mathcal{L}}(\mathcal{C}_p(x_i) \bmod \mathcal{L}) = \mathcal{C}_{\mathcal{L}}(\mathcal{C}_p(x_j) \bmod \mathcal{L})$. Since $\mathcal{C}_{\mathcal{L}}$ is injective, $\mathcal{C}_p(x_i) \bmod \mathcal{L} = \mathcal{C}_p(x_j) \bmod \mathcal{L}$, i.e., $\mathcal{C}_p(x_i) - \mathcal{C}_p(x_j)$ is a lattice vector. Furthermore, since $x_i \neq x_j$ and $\mathcal{C}_p$ is injective, $\mathcal{C}_p(x_i) \neq \mathcal{C}_p(x_j)$. Therefore $\mathcal{C}_p(x_i) - \mathcal{C}_p(x_j)$ is a non-zero lattice vector, as claimed.

It remains to show that there exists an $i \neq j$ such that $\|\mathcal{C}_p(x_i) - \mathcal{C}_p(x_j)\|_p \leq d_{\ell_p^n}(r, L)$. To see this, notice that by definition $\mathcal{C}_p(x_1), \ldots, \mathcal{C}_p(x_L) \in \mathcal{B}_p^n(r)$. Then, by the definition of $d_{\ell_p^n}(r, L)$, there must exist $i \neq j$ such that

$$\|\mathcal{C}(x_i) - \mathcal{C}(x_j)\|_p \leq d_{\ell_p^n}(r, L) \ ,$$

as needed. □

Recall that we are interested in $\gamma$-HSVP$_p$ for $\gamma \approx \mathrm{vol}(\mathcal{B}_p^n(1))^{-1/n}$.

**Corollary 4.2.** *For any sufficiently large polynomial $s(n)$, efficiently computable non-decreasing functions $\alpha := \alpha(B) > 1$ and $L := L(B) \leq \lceil \alpha \rceil$ with $L \leq \mathrm{polylog}(B)$, $\gamma$-HSVP$_p \in (\alpha B, B)$-$\mathsf{PMPP}^L$ where*

$$\gamma \leq (1 + o(1)) \cdot \mathrm{vol}(\mathcal{B}_p^n(1))^{-1/n} \cdot (\alpha^*)^{1/n} \cdot d_{\ell_p^n}(1, L^*) \ ,$$

*where $\alpha^* := \alpha(2^{s(n)})$ and $L^* := L(2^{s(n)})$.*

*In particular, $\gamma_M := 2\,\mathrm{vol}(\mathcal{B}_p^n(1))^{-1/n}$ corresponds to Minkowski's bound, and we have $(1 + o(1))\gamma_M$-HSVP$_p \in \mathsf{PWPP}$. Blichfeldt's bound for the $\ell_2$ norm corresponds to $\gamma_B := (1 + o(1)) \cdot \sqrt{2}\,\mathrm{vol}(\mathcal{B}_2^n(1))^{-1/n}$, and we see that, e.g., $(1 + o(1))\gamma_B$-HSVP $\in (2LB, B)$-$\mathsf{PMPP}^L$ for any $L = \mathrm{poly}(\log B)$.*

*Proof.* By Lemma 2.21, it suffices to solve $(\gamma, s(n)/(10n))$-HSVP$_p$, where recall that $(\gamma,\ell)$-HSVP$_p$ is the special case of HSVP$_p$ in which the input lattice has determinant at most $2^\ell$. On input a basis $\mathbf{B} \in \mathbb{Z}^{n\times n}$ for a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, the reduction does the following. Let $q := 2^n$ and $B := q^n \det(\mathcal{L}) \leq 2^{s(n)}$. The reduction computes an $r \geq 100pn/q$ such that

$$\left(1 - \frac{20n}{rq}\right) \cdot q^n \cdot \mathrm{vol}(\mathcal{B}_p^n(r)) = A(B) \ .$$

(Notice that such an $r$ exists because the function on the left is continuous, unbounded, and can be less than $A(B) > B \geq q^n$ for $r \geq 100pn/q$.) The reduction then runs the procedure from Proposition 4.1 on input $p$, $r$, $q$, $A(B)$, $B$, and $\mathbf{B}$, and outputs the resulting non-zero lattice vector $\boldsymbol{y} \in \mathcal{L}$.

The reduction is clearly efficient. Notice that the conditions necessary to apply Proposition 4.1 are satisfied, so $\boldsymbol{y}$ is a non-zero lattice vector with

$$\|\boldsymbol{y}\|_p \leq d_{\ell_p^n}(r, L(B)) = r d_{\ell_p^n}(1, L(B)) \leq r d_{\ell_p^n}(1, L^*) \ .$$

And, notice that $r$ satisfies

$$r^n \leq (1 + o(1)) \cdot q^n A(B)/\mathrm{vol}(\mathcal{B}_p^n(1)) = \alpha(B) \cdot \det(\mathcal{L})/\mathrm{vol}(\mathcal{B}_p^n(1)) \ .$$

Since $\alpha(B)$ is non-decreasing and $B \leq 2^{s(n)}$,

$$r^n \leq (1 + o(1)) \cdot \alpha^* \cdot \det(\mathcal{L})/\mathrm{vol}(\mathcal{B}_p^n(1)) \ ,$$

and the result follows. □

# 5 Inclusions

We have defined a hierarchy of complexity classes $(A(n), B(n))^{L(n)}$-PMPP and we now consider inclusions between these classes. Recall that due to the Merkle–Damgård construction, $(2^n, 2^{n-1})$-PMPP$^2$ is equal to $(2^{p(n)}, 2^n)$-PMPP$^2$ for any polynomial $p(n)$. We now consider what inclusions the Merkle–Damgård construction and the Merkle tree construction imply when $L \neq 2$.

## 5.1 Merkle–Damgård construction for multicollisions

We now examine the trade-off between the size of a multicollision and the compression of a function for the Merkle–Damgård construction. (We note that if one is slightly more careful with floors and ceilings, then one can get a slightly better dependence of $L'$ on $L$, and in particular, one can achieve a strict generalization of the case $L' = L = 2$ mentioned above. However, this makes the notation quite unwieldy, so we resist the urge to do this.)

**Theorem 5.1.** *For $L' := L^{\lceil (m-b)/(a-b) \rceil}$, there is a Karp reduction from $(q^a, q^b)$-Pigeon$^L$ to $(q^m, q^b)$-Pigeon$^{L'}$ that runs in $\mathrm{poly}(|\mathcal{C}|, L')$ time where $|\mathcal{C}|$ is the size of the input.*

*Proof.* We assume without loss of generality that $\ell := (m-a)/(a-b) = (m-b)/(a-b) - 1$ is an integer. On input a circuit $\mathcal{C} : [q]^a \to [q]^b$, the reduction behaves as follows. Define $\mathcal{C}_i : [q]^{b+i(a-b)} \to [q]^b$ as follows. $\mathcal{C}_1 := \mathcal{C}$, and for $i > 1$, $\mathcal{C}_i(x, y_1, \ldots, y_i) := \mathcal{C}(\mathcal{C}_{i-1}(x, y_1, \ldots, y_{i-1}), y_i)$ for $x \in [q]^b$ and $y_j \in [q]^{(a-b)}$.

The reduction calls its $(q^m, q^b)$-Pigeon$^{L'}$ oracle on $\mathcal{C}_{\ell+1} : [q]^m \to [q]^b$, receiving as output distinct $z_1, \ldots, z_{L'} \in [q]^m$ such that $\mathcal{C}_{\ell+1}(z_1) = \cdots = \mathcal{C}_{\ell+1}(z_{L'})$. Let $z_j := (x_{1,j}, y_{1,j}, \ldots, y_{\ell,j})$ where $x_{1,j} \in [q]^a$ and $y_{i,j} \in [q]^{a-b}$. Finally, the reduction computes for all $1 \leq j \leq L'$ and $1 \leq i \leq \ell$, $x_{i+1,j} := \mathcal{C}(x_{i,j}, y_{i,j})$, and it outputs any collection of distinct strings $(x_{i,j_1}, y_{i,j_1}), \ldots, (x_{i,j_L}, y_{i,j_L}) \in [q]^a$ such that $\mathcal{C}(x_{i,j_1}, y_{i,j_1}) = \cdots = \mathcal{C}(x_{i,j_L}, y_{i,j_L})$.

This reduction clearly runs in time $\mathrm{poly}(|\mathcal{C}|, L')$. To show correctness, we must show that a collection of distinct colliding stings $(x_{i,j_1}, y_{i,j_1}), \ldots, (x_{i,j_L}, y_{i,j_L})$ actually exists.

We prove by induction on $i$ that if there exist distinct

$$(x_{1,j_1}, y_{1,j_1}, \ldots, y_{i,j_1}), , \ldots, (x_{1,j_{L^i}}, y_{1,j_1}, \ldots, y_{i,j_{L^i}}) \in [q]^{a+i(a-b)}$$

such that

$$\mathcal{C}_i(x_{1,j_1}, y_{1,j_1}, \ldots, y_{i,j_1}) = \cdots = \mathcal{C}_i(x_{1,j_{L^i}}, y_{1,j_{L^i}}, \ldots, y_{i,j_{L^i}}) \ ,$$

then there exist distinct

$$(x_{i,j_1}, y_{i,j_1}), \ldots, (x_{i,j_L}, y_{i,j_L}) \in [q]^a$$

such that

$$\mathcal{C}(x_{i,j_1} \circ y_{i,j_1}) = \cdots = \mathcal{C}(x_{i,j_L} \circ y_{i,j_L}) \ .$$

I.e., if there exists an $L^i$-wise collision in $\mathcal{C}_i$, then there exists an $L$-wise collision in $\mathcal{C}$.

The base case when $i = 0$ is trivial. So, we assume that the result is true for $i-1$, and suppose that there exist distinct strings

$$(x_{1,j_1}, y_{1,j_1}, \ldots, y_{i,j_1}), , \ldots, (x_{1,j_{L^i}}, y_{1,j_1}, \ldots, y_{i,j_{L^i}}) \in [q]^{a+i(a-b)}$$

such that

$$\mathcal{C}_i(x_{1,j_1}, y_{1,j_1}, \ldots, y_{i,j_1}) = \cdots = \mathcal{C}_i(x_{1,j_1}, y_{1,j_{L^i}}, \ldots, y_{i,j_{L^i}}) \ .$$

Let $g$ be the number of distinct values taken by the

$$(x_{i,j_k}, y_{i,j_k}) := (\mathcal{C}_{i-1}(x_{1,j_k}, y_{1,j_k}, \ldots, y_{i-1,j_k}), y_{i,j_k})$$

for different choices of $k$. If $g \geq L$, then we are done, since $\mathcal{C}(x_{i,j'_1}, y_{i,j'_1}) = \cdots = \mathcal{C}(x_{i,j'_g}, y_{i,j'_g})$ forms a $g$-wise collision for $L \geq g$ of distinct elements under $\mathcal{C}$. If $g < L$, then by the pigeonhole principle there must exist some subset of the $(x_1, j_k, y_{1,j_k}, \ldots, y_{i-1,j_k}, y_{i,j_k})$ that form an $L^{i-1}$-wise collision under $\mathcal{C}_{i-1}$. Since by assumption all of these have the same value of $y_{i,j_k}$, these must all have distinct values for $(x_1, j_k, y_{1,j_k}, \ldots, y_{i-1,j_k})$. So, we get a collision of $L^{i-1}$ distinct elements under $\mathcal{C}_{i-1}$, and the result follows from the induction hypothesis. Either way, the result follows.

In particular, we see that the reduction succeeds because $L' = L^{(m-b)/(a-b)} = L^{\ell+1}$. □

## 5.2 A reduction using Merkle trees and list-recoverable codes

We now show how to show a non-trivial relationship between PMPP with different parameters, using Merkle trees and list-recoverable codes.

**Definition 5.2.** *For a circuit $\mathcal{C} : [N]^r \to [N]$, we define the $r$-ary, depth-$d$ Merkle tree built from $\mathcal{C}$ as the circuit $\mathcal{C}_d : [N]^{r^d} \to [N]$ defined recursively as follows. $\mathcal{C}_1 = \mathcal{C}$, and for all $i \geq 2$ we define $\mathcal{C}_i : [N]^{r^i} \to [N]$ as*

$$\mathcal{C}_i\big(x_1, \ldots, x_{r^i}\big) = \mathcal{C}_{i-1}\big(\mathcal{C}(x_1, x_2, \ldots, x_r), \mathcal{C}(x_{r+1}, \ldots, x_{2r}), \ldots, \mathcal{C}(x_{r^i-r+1}, \ldots, x_{r^i})\big) \ .$$

**Lemma 5.3.** *There is an efficient algorithm that takes as input a circuit $\mathcal{C} : [N]^r \to [N]$ and $x_1 = (x_{1,1}, \ldots, x_{1,r^d}), \ldots, x_L = (x_{L,1}, \ldots, x_{L,r^d}) \in [N]^{r^d}$ such that $\mathcal{C}_d(x_i) = \mathcal{C}_d(x_j)$ for all $i, j$ and outputs distinct $w_1, \ldots, w_{L'} \in [N]^r$ such that $\mathcal{C}(w_i) = \mathcal{C}(w_j)$ for all $i$, $j$. Furthermore, if there exists some $1 \leq j \leq r^{d-1}$ such that $\{(x_{i,2j-1}, x_{i,2j})\}$ contains at least $\ell$ distinct elements, then $L' \geq \lceil \ell^{1/d} \rceil$*

*Proof.* The algorithm behaves as follows. Let $x^0_{i,j} := x_{i,j}$. For $1 \leq k \leq d$ and $1 \leq j \leq r^{d-k}$, the algorithm computes $x^k_{i,j} := \mathcal{C}(x^{k-1}_{i,r(j-1)+1}, \ldots, x^{k-1}_{i,rj})$. The algorithm simply outputs the set $S_{i^*,j^*,k^*} := \{(x^k_{i,(r-1)j+1}, \ldots, x^k_{i,rj}) \ : \ \mathcal{C}(x^k_{i,(r-1)j+1}, \ldots, x^k_{i,rj}) = x^{k^*}_{i^*,j^*}\}$ with largest size.

Clearly the algorithm is efficient. Furthermore, by definition the output $w_1, \ldots, w_{L'}$ of the algorithm satisfies $\mathcal{C}(w_i) = \mathcal{C}(w_j)$ for all $i, j$,.

Notice that $x^d_{i,1} = \mathcal{C}(x_i)$. For each $0 \leq k \leq d-1$, let $\ell_k := \max_j |\{(x^k_{i,(r-1)j+1}, x^k_{i,rj})\}|$. We also define $\ell_d = 1$. Notice that $\ell_0 \geq \ell$ by assumption. Therefore, there exists some $1 \leq k \leq d$ such that $\ell_{k-1}/\ell_k \geq \ell^{1/d}$. After parsing definitions, we see that for this choice of $k$, there must exist a set $S_{i,j,k}$ with size at least $\ell^{1/d}$. And, since this set has integer size, the size must actually be at least $\lceil \ell^{1/d} \rceil$, as needed.) □

We now define list-recoverable codes. Our definition is actually a special case of a more general definition. In the more general definition, each codeword only needs to be close (in Hamming distance) to a codeword whose characters $\sigma_i$ come from the sets $T_i$.

**Definition 5.4.** *A circuit $\mathcal{C} : [A] \to [N]^n$ defines a $(\ell, L)$-list-recoverable code if $\mathcal{C}$ is injective and for every collection of sets $T_1, \ldots, T_n \subseteq [N]$ with $|T_i| \leq \ell$,*

$$|\mathrm{Image}(\mathcal{C}) \cap (T_1 \times \cdots \times T_n)| \leq L$$

We use the following theorem from [GLS+22]. (We write the special case that applies to our special case definition of list-recoverable codes. In [GLS+22], this corresponds to the case when $\varepsilon = 1$.)

**Theorem 5.5** (Special case of [GLS+22, Theorem 5.1])**.** *There exists a $(\ell, L)$-list-recoverable code $\mathcal{C} : [q^k] \to [q]^n$ provided that $q \geq 2^{C(L + n \log L)}$ is a prime power and*

$$n/k \geq C\sqrt{\ell} \cdot \frac{L}{L - \ell} \cdot (\log(L/(L - \ell)) + 1) \, ,$$

*where $C > 0$ is some absolute constant. Furthermore, a circuit representing this code can be constructed in randomized time* $\mathrm{poly}(\log q, n)$

We now show the main result of this section, which closely follows [BKP18].

**Theorem 5.6.** *Suppose that there exists an efficiently computable $(\ell - 1, L - 1)$-list-recoverable code $\mathcal{C}_{\mathsf{LR}} : [A] \to [N^r]^{r^{d-1}}$. Then, there is an efficient Karp reduction from $(N^r, N)$-Pigeon$^{L'}$ to $(A, N)$-Pigeon$^{L}$ for $L' := \lceil \ell^{1/d} \rceil$.*

*Proof.* On input a circuit $\mathcal{C} : [N^r] \to [N]$, the reduction constructs $\mathcal{C}^* : [A] \to [N]$ defined by $\mathcal{C}^*(w) := \mathcal{C}_d(\mathcal{C}_{\mathsf{LR}}(w))$, where $\mathcal{C}_d$ is the depth-$d$ Merkle tree defined above. The reduction then uses its Pigeon oracle to compute distinct $z_1, \ldots, z_L \in [A]$ such that $\mathcal{C}^*(z_i) = \mathcal{C}^*(z_j)$ for all $i, j$. Let $x_i := \mathcal{C}_{\mathsf{LR}}(z_i)$. Finally, the reduction uses the procedure from Lemma 5.3 on input $\mathcal{C}$ and $x_1, \ldots, x_L$ to output distinct $w_1, \ldots, w_{L'}$ such that $\mathcal{C}(w_i) = \mathcal{C}(w_j)$ for all $i$, $j$.

Clearly the reduction is efficient. By Lemma 5.3, in order to prove correctness, it suffices to prove that there exists some $j$ such that $\{(x_{i,(r-1)j+1}, \ldots, x_{i,rj})\}$ contains at least $\ell$ distinct elements. Indeed, since $\mathcal{C}_{\mathsf{LR}}$ is injective and the $z_i$ are distinct, the $x_1, \ldots, x_L$ must all be distinct as well. The result then follows from the list recoverability of $\mathcal{C}_{\mathsf{LR}}$. □

**Corollary 5.7.** *For any integers $r := r(v) \geq 2$, $k := k(v) \geq 1$, and $L := L(v) \geq 2$, $(2^{vr}, 2^v)$-PMPP$^{L'}$ reduces to $(2^{vrk}, 2^v)$-PMPP$^{L}$ under randomized reductions, where*

$$L' \geq \Omega\big(L^{\log r/(2 \log r + \log k + \log(L)/2)}\big) \, .$$

*Proof.* Take $q := 2^{vr}$, $A := q^k$, $\ell := L/2$, and

$$n := \left\lceil C\sqrt{\ell} \cdot 2 \cdot (\log(2) + 1) \cdot k \right\rceil = O(\sqrt{L}k) \, .$$

Let $d := \lceil \log_r n \rceil + 1 \leq \log_r n + 2$ so that $r^{d-1} \geq n$. By Theorem 5.5, there is a $(\ell - 1, L - 1)$-list-recoverable code $\mathcal{C} : [q^k] \to [q]^{r^{d-1}}$ that can be constructed in randomized polynomial time. Combining this with Theorem 5.6 gives a reduction from $(2^{vr}, 2^v)$-Pigeon$^{L'}$ to $(2^{vrk}, 2^v)$-Pigeon$^{L}$ for $L' \geq \ell^{1/d}$. The result follows by noting that

$$\log(L') \geq \frac{\log \ell}{d} \geq \frac{\log r \log \ell}{\log n + 2 \log r} \geq \frac{\log r \log L - \log r}{\log(L)/2 + \log k + 2 \log r} - O(1) \geq \frac{\log L \log r}{\log k + \log L/2 + 2 \log r} - O(1) \, ,$$

as needed. □

### 5.3 Polynomial Long Choice

Finally, we show that PMPP is contained in the recently defined complexity class PLC [PPY23] for certain parameters. Essentially the same result was recently proven independently by [JLRX24]. ([JLRX24] also proved that PMPP *contains* PLC, though for different parameters.)

**Theorem 5.8.** *For any $L < n$, there exists a Karp reduction from $(2^n, 2^{n-L})$-Pigeon$^{L+1}$ to* UnaryLongChoice *that runs in polynomial time.*

*Proof.* This proof can be seen as a generalization of the proof that PWPP reduces to UnaryLongChoice. Let $\mathcal{C} : \{0,1\}^n \to \{0,1\}^{n-L}$ be the circuit in which we wish to find a $L+1$-wise collision. We let $P_i(a_0, \ldots, a_i, x)$ be the $(i+1)$th bit of $\mathcal{C}(x)$ if $i+1 \leq n-L$. Otherwise, we let $P_i(a_0, \ldots, a_i, x) = 0$. We then feed $P_0, P_1, \ldots, P_{n-2}$ to the PLC oracle to get back distinct $a_0, a_1, \ldots, a_n$. The reduction then outputs $a_{n-L}, a_{n-L+1}, \ldots, a_n$. The reduction clearly runs in polynomial time. Correctness follows from the fact that $a_{n-L}, a_{n-L+1}, \ldots, a_n$ are distinct, and by construction, $\mathcal{C}(a_i) = \mathcal{C}(a_j)$ if $i, j \geq n-L$. Therefore, $\mathcal{C}(a_{n-L}) = \mathcal{C}(a_{n-L+1}) = \cdots = \mathcal{C}(a_n)$. □

**Corollary 5.9.** *For any constant $k$, $(2^{2n}, 2^n)$-PMPP$^k \subseteq$ PLC.*

By Corollary 5.9, for all parameters $A, B, L$ for which $(A,B)$-PMPP$^L$ has been studied to date, $(A,B)$-PMPP$^L \subseteq$ PLC. However, it remains unclear how exactly PMPP is related to PLC for the full range of choices for $A, B, L$.

## 6 Black-box separations

We now present two black-box separation results between Pigeon with different parameters. (We note that the recent independent work of Jain, Li, Robere, and Xun [JLRX24] contains exciting black-box separations. Their results are formally incomparable to ours, but we feel that the results in [JLRX24] are more interesting than our own black-box separations.)

The first result shows that there is no *fine-grained* black-box reduction in certain parameter regimes. The second proof result rules out deterministic Karp black-box reductions (e.g., reductions in the style of Merkle-Damgård that make a single oracle call to a collision-finding oracle) in certain parameter regimes.

**Definition 6.1.** *A black box reduction from $(2^b, 2^a)$-Pigeon$^L$ to $(2^s, 2^r)$-Pigeon$^{L'}$ is an oracle algorithm $\mathcal{A}^{f, \text{Colfinder}^f}$ such that for any oracles $f : \{0,1\}^b \to \{0,1\}^a$ and Colfinder$^f$ that finds $L'$-wise collisions in oracle circuits $\mathcal{C}^f : \{0,1\}^s \to \{0,1\}^r$, $\mathcal{A}$ succeeds in finding an $L$-wise collision in $f$.*

*We call such a reduction a black-box Karp reduction if it makes at most one query to the Colfinder$^f$ oracle.*

We assume without loss of generality that $\mathcal{A}$ never queries the same value to $f$ twice, that $x_1, x_2, \ldots, x_L$ that form the $L$-wise collision in $f$ were all queried to $f$ by $\mathcal{A}$ at some point, when Colfinder returns $w_1, w_2, \ldots, w_{L'}$ on input $C^f$, the reduction computes $C^f(w_1), C^f(w_2), \ldots, C^f(w_{L'})$ (and makes the necessary queries to $f$ to do so).

### 6.1 Simulation approach

**Lemma 6.2** ([STKT06])**.** *Let $2 \leq L \leq q$, $\alpha < 1$, and $q = \alpha N^{(L-1)/L}$. If the random variables $X_1, X_2, \ldots, X_q$ are uniformly distributed on $[N]$, then,*

$$\Pr[\exists i_1 < \cdots < i_L,\ X_{i_1} = \cdots = X_{i_L}] \leq 1/L! \,.$$

**Lemma 6.3** ([STKT06])**.** *Let $e_N = (1 - \frac{1}{N})^{-N}$, $2 \leq L \leq q$, and $q = (L!)^{1/L} N^{(L-1)/L} + L < N$. If the random variables $X_1, X_2, \ldots, X_q$ are uniformly distributed on $[N]$, then*

$$\Pr[\exists i_1 < \cdots < i_L,\ X_{i_1} = \cdots = X_{i_L}] \geq \frac{1}{2} - (\frac{L!}{N})^{1/L} \ln(e_N) \,.$$

**Lemma 6.4** ([Ber80]). *Let $X_1, X_2, \ldots, X_q$ be i.i.d random variables with support on at most $N$ values and $Y_1, Y_2, \ldots, Y_q$ be uniformly distributed on $[N]$, then for any $L$*

$$\Pr[\exists i_1 < \cdots < i_L,\ X_{i_1} = \cdots = X_{i_L}] \geq \Pr[\exists i_1 < \cdots < i_L,\ Y_{i_1} = \cdots = Y_{i_L}]$$

**Theorem 6.5.** *For any integers $L, L' \geq 2$ and positive integers $b > a$ and $s > r$, there is no black-box reduction $\mathcal{A}^{f,\mathsf{Colfinder}^f}$ from $(2^b, 2^a)$-Pigeon$^L$ to $(2^s, 2^r)$-Pigeon$^{L'}$ that makes $q_f$ queries to $f$ and $q_{\mathsf{CF}}$ to $\mathsf{Colfinder}^f$ if*

$$q_{\mathsf{CF}} \left( \frac{T^2}{2^{s+1}} + \frac{1}{2} + 2eL'2^{-r/L'} \right)^{10\log(q_{\mathsf{CF}})} \leq \frac{1}{2}\,,$$

*$T < 2^r$, and*

$$10 q_{\mathsf{f}} q_{\mathsf{CF}} \log(q_{\mathsf{CF}}) T + q_{\mathsf{f}} < 2^{a\frac{L-1}{L}-1}\,,$$

*where $T := L'2^{r\frac{L'-1}{L'}} + L'$.*

**Corollary 6.6.** *Let $L' \geq 2$ be a constant and $L > L'$. For every function $p(n) = \mathrm{poly}(n)$, there exists $n_0$ such that for all $n > n_0$, there is no black-box reduction from $(2^{2n}, 2^n)$-Pigeon$^L$ to $(2^b, 2^n)$-Pigeon$^{L'}$ which runs in time $p(n)$ for any $b > n$.*

*Proof of Theorem 6.5.* Say for the sake of contradiction that there exists a reduction $\mathcal{A}^{f,\mathsf{Colfinder}}$ that makes at most $q_f$ queries to $f$ and at most $q_{\mathsf{CF}}$ queries to $\mathsf{Colfinder}$ and succeeds in finding a $L$-wise collision in $f$ for every $f, \mathsf{Colfinder}$ pair. Let $f : \{0,1\}^b \to \{0,1\}^a$ be a random function. We will use $\mathcal{A}$ to create an algorithm $\mathcal{M}$ which finds a $L$-wise collision in a random function with probability close to 1 but makes relatively few queries to the random function $f$, which contradicts Lemma 6.2.

We first create an algorithm $\mathcal{M}^f$ that will do exactly what $\mathcal{A}$ does, except it will simulate $\mathsf{Colfinder}$ using the birthday paradox. Let $T' = 10\log(q_{\mathsf{CF}})((L'!)^{1/L'}2^{r\frac{L'-1}{L'}} + L')$. $\mathcal{M}$ will then simulate the $i$th call to $\mathsf{Colfinder}$ as follows: $\mathcal{M}^f$ will sample $T'$ values $x_1^i, x_2^i, \ldots, x_{T'}^i$ uniformly at random from $\{0,1\}^s$ and look for $j_1 < j_2 < \cdots < j_{L'}$ such that the $x_{j_i}$ are distinct and $C_i^f(x_{j_1}) = C_i^f(x_{j_2}) = \cdots = C_i^f(x_{j_{L'}})$. The simulation then outputs $x_{j_1}, x_{j_2}, \ldots, x_{j_{L'}}$.

Let us now analyze the probability that $\mathcal{M}$ succeeds in finding a $L$-wise collision for any fixed $f$ with high probability. For the analysis let us group all $x_i^j$ into $10\log(q_{\mathsf{CF}})$ groups of size $T = (L'!)^{1/L'}2^{r\frac{L'-1}{L'}} + L'$. We say the event $\mathsf{RepeatInG}_k$ occurs if not all elements in $G_k$ are distinct. We say the event $\mathsf{NoCol}_k$ occurs if $G_k$ does not contain $a_1 < a_2 < \cdots < a_{L'}$ such that $C_i^f(a_1) = C^f(a_2) = \cdots = C^f(a_{L'})$. We say that the event $\mathsf{SimFail}_i$ occurs if simulation $i$ fails to find a $L'$-wise collision in $C_i^f$. We say the event $\mathcal{M}\mathsf{Fail}$ occurs if any of

the $q_{\mathsf{CF}}$ simulation steps fails.

$$\begin{aligned}
\Pr_{x_j^i}[\mathcal{M}\mathsf{Fail}] &= \Pr_{x_j^i}\left[\bigcup_{i\in[q_{\mathsf{CF}}]} \mathsf{SimFail}_{\mathsf{i}}\right] \\
&\leq \sum_{i\in[q_{\mathsf{CF}}]} \Pr_{x_j^i}[\mathsf{SimFail}_{\mathsf{i}}] \\
&\leq \sum_{i\in[q_{\mathsf{CF}}]} \Pr_{x_j^i}\left[\bigcap_{k\in 10\log(q)} (\mathsf{RepeatInG}_k \cup \mathsf{NoCol}_k)\right] \\
&\leq q_{\mathsf{CF}}\left(\Pr_{x_j^1}[(\mathsf{RepeatInG}_1 \cup \mathsf{NoCol}_1)]\right)^{10\log(q_{\mathsf{CF}})} \\
&\leq q_{\mathsf{CF}}\left(\Pr_{x_j^1}[\mathsf{RepeatInG}_1] + \Pr_{x_j^1}[\mathsf{NoCol}_1]\right)^{10\log(q_{\mathsf{CF}})} \\
&\leq q_{\mathsf{CF}}\left(\Pr_{x_j^1}[\mathsf{RepeatInG}_1] + \frac{1}{2} + (\frac{L'!}{2^r})^{1/L'}\ln(e_{2^r})\right)^{10\log(q_{\mathsf{CF}})}
\end{aligned}$$

The final inequality follows from treating each $C^f(x_j^1)$ as a random variable on $\{0,1\}^r$ and combining Lemma 6.4 with Lemma 6.3. We also know that the probability we sample the same $x_j^1$ twice is $\leq \frac{T(T-1)}{2\cdot 2^s} \leq \frac{T^2}{2\cdot 2^s}$ by a well known bound on the birthday paradox.

$$\leq q_{\mathsf{CF}}\left(\frac{T^2}{2^{s+1}} + \frac{1}{2} + (\frac{L'!}{2^r})^{1/L'}\ln(e_{2^r})\right)^{10\log(q_{\mathsf{CF}})} \leq \left(\frac{T^2}{2^{s+1}} + \frac{1}{2} + 2eL'2^{-r/L'}\right)^{10\log(q_{\mathsf{CF}})} \leq \frac{1}{2}$$

So with probability greater $\frac{1}{2}$, $\mathcal{M}$ succeeds in finding a collision in a random $f$. Since each evaluation of $C^f$ requires at most $q_{\mathsf{f}}$ queries, $\mathcal{M}$ makes at most $q_{\mathsf{f}}q_{\mathsf{CF}}T + q_{\mathsf{f}}$ queries to $f$. This is a contradiction since by Lemma 6.2 $2^{a\frac{L-1}{L}-1}$ queries are required to have a $\frac{1}{L!}$ chance of finding a collision in a random function with an $a$ bit output.

□

## 6.2 KNY approach

**Theorem 6.7.** *Let $L = L(n), L' = L'(n), a = a(n), s = s(n), r = r(n)$ and ensure $s - r = \Omega(n), L(n) = \mathrm{poly}(n), L'(n) = \mathrm{poly}(n)$ and*

$$s \leq \frac{a(L-1) - (L-1)\cdot\omega(\log(n))}{L'-1}\ .$$

*For every $p(n) = \mathrm{poly}(n)$, there exists $n_0$ such that for every $n > n_0$, there is no deterministic black-box Karp reduction from $(2^{2n}, 2^{a(n)})$-Pigeon$^{L(n)}$ to $(2^{s(n)}, 2^{r(n)})$-Pigeon$^{L'(n)}$ that runs in time $p(n)$.*

The oracle $\Gamma$ we will use to show that no black box construction exists is as follows (it is the natural generalization of the oracle given in [KNY18]).

**Construction 6.8.** *The oracle $\Gamma$ consists of a tuple $(f, \mathsf{Colfinder}^f)$*

1. *The function $f = \{f\}_{n\in\mathbb{N}}$: For every $n$, the function $f$ is a uniformly chosen function from $2n$ bits to $a(n)$ bits.*
2. *The function $\mathsf{Colfinder}^f$: This function consists of an infinite set of permutations, where for every $n \in \mathbb{N}$, and every circuit $C^f : \{0,1\}^{s(n)} \to \{0,1\}^{r(n)}$, there are $L'$ uniformly and independently*

*chosen permutations* $\pi^1_{C_f}, \pi^2_{C_f}, \ldots, \pi^{L'}_{C_f}$ *over* $\{0,1\}^{s(n)}$. *When given an input* $C^f$, $\mathsf{Colfinder}^f$ *sets* $x_1 = \pi^1_{C^f}(0^{s(n)})$, *and* $x_i = \pi^i_{C^f}(t_i)$ *where* $t_i$ *is the lexicographically smallest* $t_i$ *such that* $C^f(x_1) = C^f(x_i)$ *(for* $1 < i \leq q$*). It then outputs* $(x_1, x_2, \ldots, x_{L'(n)})$.

**Lemma 6.9.** *There exists a polynomial time, oracle-aided algorithm* $\mathcal{A}$ *such that for any function* $f : \{0,1\}^{2n} \to \{0,1\}^{a(n)}$, *and any oracle aided circuit* $C^f : \{0,1\}^{s(n)} \to \{0,1\}^{r(n)}$, *it holds that*

$$\Pr_{\mathsf{Colfinder}}\left[ \begin{array}{c} x_1, x_2, \ldots, x_{L'} \text{ are distinct} \\ C^f(x_1) = C^f(x_2) = \cdots = C^f(x_{L'}) \end{array} \;:\; (x_1, x_2, \ldots, x_{L'}) \leftarrow \mathcal{A}^{(f,\mathsf{Colfinder}^f)}(1^n, C^f) \right] \geq 1 - 1/\operatorname{poly}(n)$$

*Proof.* Fix $n$ and $f$. The algorithm on input $C^f$, sends $C^f$ to $\mathsf{Colfinder}^f$ and outputs the result $(x_1, x_2, \ldots, x_{L'})$. It holds by definition that $C^f(x_1) = C^f(x_2) = \cdots = C^f(x_{L'})$. All that is left to show is that $x_1, x_2, \ldots, x_{L'}$ are all distinct.

Notice there for any $u(n)$, there are at most $u(n) \cdot 2^{r(n)}$ values for $x$ such that $|(C^f)^{-1}(C^f(x))| \leq u(n)$. The proof for this is fairly simple. Group each input according to its output, there are $2^{s(n)}$ inputs, and at most $2^{r(n)}$ groups. If we count the number of elements in the groups with less than $u(n)$ elements, we will clearly end up with $\leq 2^{r(n)} \cdot u(n)$ elements. Thus, there at at least $2^{s(n)} - 2^{r(n)}u(n)$ $x$ for which there are $|d| > u(n)$. Let $u(n) = n^2 L'(n)^2$. We thus have that

$$\Pr_{x_i}\left[ \; |(C^f)^{-1}(C^f(x))| > L'(n)^2 n^2 \; \right] \geq \frac{2^{s(n)} - L'(n)^2 n^2 2^{r(n)}}{2^{s(n)}} \geq 1 - \frac{L'(n)^2 n^2}{2^{s(n)-r(n)}} = 1 - \frac{L'(n)^2 n^2}{2^{-\Omega(n)}} = 1 - \operatorname{negl}(n) \; .$$

Let us now consider the probability that our proposed algorithm does not output distinct $x_i$. Assume we pick $x_1$ such that $|(C^f)^{-1}(C^f(x))| > n^2 L'(n)^2$, then if we pick $L' - 1$ elements independently from $(C^f)^{-1}(C^f(x))$ (which is what $\mathsf{Colfinder}$ does), then by the birthday paradox, we will have probability $\geq 1 - \frac{L'^2}{n^2 L'^2} = 1 - 1/n^2$ that $x_1, x_2, \ldots, x_{L'}$ are distinct. Therefore, the probability that we succeed is $(1 - \operatorname{negl}(n))(1 - 1/n^2) = 1 - 1/\operatorname{poly}(n)$, as claimed. □

Theorem 6.7 then follows from the following lemma.

**Lemma 6.10.** *For any collision-finding oracle* $\mathsf{Colfinder}^f$ *and any polynomial-time deterministic black-box Karp reduction* $\mathcal{A}^{(f,\mathsf{Colfinder}^f)}$ ,

$$\Pr_{f}\left[ \begin{array}{c} x_1, x_2, \ldots, x_L \text{ are distinct} \\ f(x_1) = f(x_2) = \cdots = f(x_L) \end{array} \;:\; (x_1, x_2, \ldots, x_L) \leftarrow \mathcal{A}^{(f,\mathsf{Colfinder}^f)}(1^n) \right] < o(1) \; .$$

*Proof.* We may assume without loss of generality that (1) $\mathcal{A}^{(f,\mathsf{Colfinder})}$ never queries $f$ on the same input twice; (2) $\mathcal{A}^{(f,\mathsf{Colfinder})}$ always queries all of the elements $x_1, \ldots, x_L$ that it outputs; $\mathcal{A}^{(f,\mathsf{Colfinder})}$ always makes a query to $\mathsf{Colfinder}$; and (3) when $\mathsf{Colfinder}$ outputs some collision $w_1, \ldots, w_{L'}$ in some circuit $\mathcal{C}^f$, $\mathcal{A}^{(f,\mathsf{Colfinder})}$ proceeds to compute $\mathcal{C}^f(w_1), \ldots, \mathcal{C}^f(w_{L'})$, making all necessary new queries to $f$ along the way to do so. Notice also that because $\mathsf{Colfinder}^f$ is fixed, the first element $w_1$ output by $\mathsf{Colfinder}^f$ on input some circuit $\mathcal{C}^f$ is fixed and independent of $f$ for fixed $\mathcal{C}^f$.

Let $\mathcal{F} \subseteq \{f : \{0,1\}^{2n} \to \{0,1\}^{a(n)}\}$ be the set of all functions $f$ for which $\mathcal{A}^{(f,\mathsf{Colfinder})}$ succeeds, i.e., for which $(x_1, \ldots, x_L) \leftarrow \mathcal{A}^{(f,\mathsf{Colfinder}^f}(1^n)$ is in fact a valid $L$-wise collision in $f$. We will show that for every $f \in \mathcal{F}$, there is an *encoding* bit string $E_f$ such that $f$ can be completely recovered from $E_f$ and $|E_f| < a(n)2^{2n} - \omega(1)$. This implies that $|\mathcal{F}| < o(2^{a(n)2^{2n}})$, which is equivalent to the result.

To that end, fix some $f$ for which $\mathcal{A}^{(f,\mathsf{Colfinder})}$ finds distinct elements $x_1, \ldots, x_L \in \{0,1\}^{2n}$ such that $f(x_1) = \cdots = f(x_L)$. Let $z_1, \ldots, z_q$ be all queries made by $\mathcal{A}^{(f,\mathsf{Colfinder})}$ to $f$. By assumption, there exist indices $i_1, \ldots, i_L \in [q]$ such that $z_{i_j} = x_j$ for all $j$. Furthermore, the algorithm $\mathcal{A}^{(f,\mathsf{Colfinder})}$ makes precisely one query to $\mathsf{Colfinder}$, receiving as output $w_1, \ldots, w_{L'} \in \{0,1\}^{s(n)}$.

Our encoding $E_f$ will be the following. Let $Y$ be the list of all $f(z_i)$ (in order) *except* for when $i = i_j$ for some $j$. Let $Y^*$ be the list of all $f(z)$ where $z \neq z_i$ for any $i$ (in some canonical order—say the lexicographic

order on such $z$). Notice that $Y$ and $Y^*$ together contain all values of $f$ except for its value on the points $z_{i_1}, \ldots, z_{i_L}$. Therefore $|Y| + |Y^*| = a(n)(2^{2n} - L)$ Then,

$$E_f = i_1 \circ \cdots \circ i_L \circ f(x_1) \circ w_2 \cdots \circ w_{L'} \circ Y^* \circ Y \ ,$$

where the $i_j$ are written in binary. Notice that

$$|E_f| = \lceil \log q \rceil \cdot L + a(n) + (L'-1) \cdot s(n) + a(n)(2^{2n} - L) = a(n)2^{2^n} + O(\log n) \cdot L + (L'-1) \cdot s(n) - a(n)(L-1) \ .$$

By our assumption on the parameters $a, s, L, L'$, we have $a(n)(L-1) > \omega(\log n)L + (L'-1)s(n)$. So, $|E_f| < a(n)2^{2n} - \omega(1)$ as needed.

It remains to show that $E_f$ can be used to recover $f$. To see this, first notice that $E_f$ contains enough information to list all responses to all $f$ queries made by $\mathcal{A}^{(f,\mathsf{Colfinder})}$. In particular, for all queries to $f$ except the $i_j$th query for any $j$, we can simply use $Y$ to respond to the queries. For the $i_j$th query, we can respond with $f(x_1)$, which is included in $E_f$. Using these responses, the fixed value of $w_1$, and the $w_2, \ldots, w_{L'}$ included in $E_f$, we can all queries and responses made by $\mathcal{A}^{(f,\mathsf{Colfinder})}$, therefore recovering the values of $f(z_1), \ldots, f(z_q)$. The remaining values of $f(z)$ can be read off of $Y^*$. The result follows. □

# 7 A non-black-box non-separation

In this section, we use non-black-box techniques to prove non-trivial relationships between versions of Pigeon with different parameters $L$. To do so, we use beautiful ideas due to Rothblum and Vasudevan [RV22], who showed an analogous result for collision-resistant hash functions.

**Theorem 7.1.** *Let $k := k(n) \geq 2$, $m := m(n) \geq 2$, $L_1 := L_1(n) \geq 2$, and $L_2 := L_2(n) \geq 2$ be efficiently computable polynomially bounded integers that satisfy $L_1 \leq 2^{(k-1)n-m}$, $L_2 \leq 2^{n-m}$, $k - 1 \leq L_2/2$, and*

$$L_1 > (L_2 + 1) \cdot \sqrt{2(k-1)/L_2} - 2(k-1) \ .$$

*Suppose that there exist efficient deterministic algorithm that solves $(2^{kn}, 2^{m+n})$-Pigeon$^{L_1}$ and an efficiently deterministic algorithm that solves $(2^n, 2^m)$-Pigeon$^{L_2}$. Then, there is also an efficient deterministic algorithm algorithm that solves $(2^{kn}, 2^m)$-Pigeon$^{L'}$, where*

$$L' := \left\lceil L_1 \cdot \sqrt{L_2/(2(k-1))} - L_2 + \sqrt{2(k-1)L_2} \right\rceil \ .$$

*Proof.* Suppose that $\mathcal{A}_{L_1}$ is an efficient deterministic algorithm that solves $(2^{kn}, 2^{m+n})$-Pigeon$^{L_1}$ and $\mathcal{A}_{L_2}$ is an efficient algorithm that solves $(2^n, m)$-Pigeon$^{L_2}$. We construct an algorithm $\mathcal{A}_{L'}$ that solves $(2^{kn}, 2^m)$-Pigeon$^{L'}$.

Before doing so, we will need a number of definitions. For a circuit $\mathcal{C} : \{0,1\}^{kn} \to \{0,1\}^m$ and $\alpha \in \{0,1\}^n$, we define the circuit $\mathcal{C}_\alpha : \{0,1\}^{kn} \to \{0,1\}^{m+n}$ as $\mathcal{C}_\alpha(x) := (\mathcal{C}(x), g_\alpha(x))$, where $g_\alpha : \{0,1\}^{kn} \to \{0,1\}^n$ is defined as follows. To compute $g_\alpha$, we first interpret $\alpha \in \{0,1\}^n$ as an element in the finite field $\mathbb{F}_{2^n}$. We similarly interpret $x \in \{0,1\}^{kn}$ as a list of $k$ field elements, $x_0, \ldots, x_{k-1} \in \mathbb{F}_{2^n}$. Then, $g_\alpha(x) = x_0 + x_1\alpha + \cdots + x_{k-1}\alpha^{k-1}$. In other words, $g_\alpha(x)$ interprets $x$ as the coefficients of a polynomial with degree at most $k-1$ and evaluates that polynomial at $\alpha$.

Then, we define the function $f_\mathcal{C} : \{0,1\}^n \to \{0,1\}^m$ as follows. To compute $f_\mathcal{C}(\alpha)$, we first run $\mathcal{A}_{L_1}$ on input $\mathcal{C}_\alpha$, receiving as output (distinct) $x_1, \ldots, x_{L_1} \in \{0,1\}^{kn}$ such that $\mathcal{C}_\alpha(x_1) = \cdots = \mathcal{C}_\alpha(x_{L_1})$. Then, $f_\mathcal{C}(\alpha) := \mathcal{C}(x_1)$ (or, equivalently $f_\mathcal{C}(\alpha) = \mathcal{C}(x_i)$ for any $i$, since $\mathcal{C}_\alpha(x_i) = \mathcal{C}_\alpha(x_1)$ and $\mathcal{C}(x_i)$ is a substring of $\mathcal{C}_\alpha(x_i)$). In other words, $f_\mathcal{C}(\alpha)$ is "the image under $\mathcal{C}$ corresponding to the $L_1$-wise collision found by $\mathcal{A}_{L_1}$ in $\mathcal{C}_\alpha$." (Notice that $f_\mathcal{C}$ is defined in terms of $\mathcal{A}_{L_1}$. This is what makes our proof non-black-box.)

We are now ready to describe $\mathcal{A}_{L'}$. The algorithm takes as input a description of a circuit $\mathcal{C} : \{0,1\}^{kn} \to \{0,1\}^m$ and behaves as follows. It first constructs a circuit $\mathcal{C}^* : \{0,1\}^n \to \{0,1\}^m$ such that $\mathcal{C}^*(\alpha) = f_\mathcal{C}(\alpha)$ with $f_\mathcal{C}$ as described above. The algorithm then runs $\mathcal{A}_{L_2}$ on input $\mathcal{C}^*$, receiving as output distinct

$\alpha_1, \ldots, \alpha_{L_2} \in \{0,1\}^n$ such that $f_{\mathcal{C}}(\alpha_1) = \cdots = f_{\mathcal{C}}(\alpha_{L_2})$. Then for $i = 1, \ldots, L_2$, the algorithm computes $(x_{i,1}, \ldots, x_{i,L_2}) \leftarrow \mathcal{A}_{L_1}(\mathcal{C}_{\alpha_i})$, where $\mathcal{C}_{\alpha_i}$ is defined as above. Finally, the algorithm outputs any subset of $L'$ distinct elements among the $x_{i,j}$. (We will argue below that such a subset must exist.)

First, notice that this is actually an efficient algorithm. In particular, the fact that $\mathcal{A}_{L_1}$ is an efficient deterministic algorithm means that we can efficiently compute the circuit $\mathcal{C}^*$ (which uses $\mathcal{A}_{L_1}$ as a subroutine) and in particular that $\mathcal{C}^*$ has description length that is polynomially bounded in the description length of $\mathcal{C}$.

Next, we claim that the algorithm is correct. In particular, we claim that $\mathcal{C}(x_{i,j}) = \mathcal{C}(x_{i',j'})$ for all $i, j, i', j'$ *and* that there exist at least $L'$ distinct elements among the $x_{i,j}$.

Showing that $\mathcal{C}(x_{i,j}) = \mathcal{C}(x_{i',j'})$ amounts to carefully parsing the (admittedly rather complicated) definitions of $\mathcal{C}_\alpha$ and $f_{\mathcal{C}}$. We actually show something slightly stronger, namely that $C_{\alpha_i}(x_{i,j}) = C_{\alpha_{i'}}(x_{i',j'})$. (This is in fact a stronger statement, since $C(x_{i,j})$ is a substring of $C_{\alpha_i}(x_{i,j})$. By the fact that $\mathcal{A}_{L_1}$ is deterministic and the definition of $f_{\mathcal{C}}$, we must have that $\mathcal{C}(x_{i,1}) = f_{\mathcal{C}}(\alpha_i)$ for all $i$. By the correctness of $\mathcal{A}_{L_2}$, we must have that $f_{\mathcal{C}}(\alpha_i) = f_{\mathcal{C}}(\alpha_1)$ for all $i$. And, by the correctness of $\mathcal{A}_{L_1}$, we must have that $\mathcal{C}_{\alpha_i}(x_{i,j}) = \mathcal{C}_{\alpha_i}(x_{i,1})$ for all $i, j$, as needed.

It remains to show that there must be at least $L'$ distinct values of $x_{i,j}$. This is where we use our careful choice of $g_\alpha$. Indeed, since $g_{\alpha_i}(x_{i,j})$ is a substring of $\mathcal{C}_{\alpha_i}(x_{i,j})$, it follows from the above that $g_{\alpha_i}(x_{i,j}) = g_{\alpha_{i'}}(x_{i',j'})$ for any $i, j, i', j'$.

Recall that $g_{\alpha_i}(x_{i,j})$ is a polynomial $p_{i,j}$ with degree at most $k-1$ in $\alpha_i$, where the polynomial $p_{i,j}$ itself depends only on $x_{i,j}$. These polynomials have the property that $p_{i,j}(\alpha_i) = p_{i',j'}(\alpha_{i'})$ for all $i, j, i', j'$. Therefore, applying Corollary 2.23, we see that there are at least $L'$ distinct polynomials $p_{i,j}$. In other words, there are at least $L'$ distinct values $x_{i,j}$, as needed. □

# References


[ALS21] Divesh Aggarwal, Zeyong Li, and Noah Stephens-Davidowitz. A $2^{n/2}$-time algorithm for $\sqrt{n}$-SVP and $\sqrt{n}$-Hermite SVP, and an improved time-approximation tradeoff for (H)SVP. In *Eurocrypt*, 2021. 4, 7

[Bas65] L. A. Bassalygo. New upper bounds for error-correcting codes. *Problems of Information Transmission*, pages 32–35, 1965. 2, 18

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Multi-collision resistant hash functions and their applications. In *Eurocrypt*, 2018. 1, 9

[Ber80] Geoffrey C. Berresford. The uniformity assumption in the birthday problem. *Mathematics Magazine*, 53(5):286–288, 1980. 26

[BFH+23] Romain Bourneuf, Lukáš Folwarczný, Pavel Hubáček, Alon Rosen, and Nikolaj I. Schwartzbach. PPP-completeness and extremal combinatorics. In *ITCS*, 2023. 1, 7

[BJP+19] Frank Ban, Kamal Jain, Christos H. Papadimitriou, Christos-Alexandros Psomas, and Aviad Rubinstein. Reductions in PPP. *Information Processing Letters*, 145:48–52, 2019. 1, 2, 4, 7, 9, 21

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: A paradigm for keyless hash functions. In *STOC*, 2018. 1, 5, 6, 9, 24

[Bli29] H. F. Blichfeldt. The minimum value of quadratic forms, and the closest packing of spheres. *Mathematische Annalen*, 101(1):605–608, 1929. 4, 8, 13, 21

[Dam89] Ivan Bjerre Damgård. A design principle for hash functions. In *CRYPTO*, 1989. 4, 11

[DDv22] Thomas Debris-Alazard, Léo Ducas, and Wessel P. J. van Woerden. An algorithmic reduction theory for binary codes: LLL and more. *IEEE Transactions on Information Theory*, 68(5):3426–3444, 2022. 7

[Del73] Philippe Delsarte. *An algebraic approach to the association schemes of coding theory*. Thesis, Universite Catholique de Louvain, 1973. 8

[DMS03] I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003. 7

[GLS+22] Zeyu Guo, Ray Li, Chong Shangguan, Itzhak Tamo, and Mary Wootters. Improved list-decodability and list-recoverability of Reed-Solomon codes via tree packings. In *FOCS*, 2022. 24

[GN08] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC*, 2008. 4, 7

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inf. Theory*, 54(1):135–150, 2008. 20

[Gri60] J. H. Griesmer. A bound for error-correcting codes. *IBM Journal of Research and Development*, 4(5):532–542, 1960. 7

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM Journal on Discrete Mathematics*, 13(4):535–570, 2000. 13

[GRS23] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. 2023. October 3rd, 2023 book version. 14, 15, 16, 17, 18, 20

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. 2, 16

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016. 1, 7

[JLRX24] Siddhartha Jain, Jiawei Li, Robert Robere, and Zhiyang Xun. On pigeonhole principles and Ramsey in TFNP. http://arxiv.org/abs/2401.12604, 2024. 5, 6, 9, 25

[Joh62] Selmer M. Johnson. A new upper bound for error-correcting codes. *IRE Trans. Inf. Theory*, 8:203–207, 1962. 17

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *CRYPTO*, 2004. 1, 4, 9

[KL78] Grigorii A. Kabatjanskiĭ and Vladimir I. Levenšteĭn. Bounds for packings on the sphere and in space. *Problemy Peredači Informacii*, 14(1):3–25, 1978. 8

[Knu05] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions*. Addison-Wesley Professional, 2005. 35

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In *Eurocrypt*, 2018. 1, 6, 9, 27

[KY23] Ilan Komargodski and Eylon Yogev. Personal communication, 2023. 5, 7

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982. 4, 7

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, 1989. 4, 11

[Min10] Hermann Minkowski. *Geometrie der Zahlen*. B.G. Teubner, 1910. 2, 13, 21

[MRRW77] R. McEliece, E. Rodemich, H. Rumsey, and L. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Transactions on Information Theory*, 23(2):157–166, 1977. 8

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In *Eurocrypt*, 2016. 4, 7

[NS07] Mridul Nandi and Douglas R. Stinson. Multicollision attacks on some generalized sequential hash functions. *IEEE Transactions on Information Theory*, 53(2):759–767, 2007. 1, 9

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. 1, 7, 10

[Plo60] M. Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960. 2, 15

[PPY23] Amol Pasarkar, Christos Papadimitriou, and Mihalis Yannakakis. Extremal combinatorics, iterated pigeonhole arguments and generalizations of PPP. In *ITCS*, 2023. 5, 7, 11, 25

[RV22] Ron D. Rothblum and Prashant Nalini Vasudevan. Collision-resistance from multi-collision-resistance. In *CRYPTO*, 2022. 1, 5, 6, 9, 29

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(23):201–224, 1987. 4, 7

[Sin64] R. Singleton. Maximum distance $q$-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964. 2, 15

[Sot20] Aikaterini Sotiraki. *New Hardness Results for Total Search Problems and Non-Interactive Lattice-Based Protocols*. Thesis, Massachusetts Institute of Technology, 2020. https://dspace.mit.edu/handle/1721.1/129310. 1, 6, 7, 9

[ST20] Chong Shangguan and Itzhak Tamo. Combinatorial list-decoding of reed-solomon codes beyond the johnson radius. In *STOC*, 2020. 15

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In *ICISC*, 2006. 25

[Sud97] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. 13

[SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-completeness with connections to cryptography. In *FOCS*, 2018. 1, 7

[Var97] Alexander Vardy. Algorithmic complexity in coding theory and the Minimum Distance Problem. In *STOC*, 1997. 7

[YW07] Hongbo Yu and Xiaoyun Wang. Multi-collision attack on the compression functions of MD4 and 3-pass HAVAL. In *ICISC*, 2007. 1, 9

[Zya71] Victor Zyablov. An estimate of the complexity of constructing binary linear cascade codes. *Probl. Peredachi Inf.*, 1971. 19

# A On efficient injections from $[N]$ to sets of size roughly $N$

**Definition A.1.** *For a family of sets $\mathcal{S} := \{S_{n,k}\}_{n\geq 1,k\geq 0}$ with $S_{n,k} \subseteq [-k,k]^n$, $j : \mathbb{N}^2 \times \mathbb{Z} \to \mathbb{N}$ is an* index *of $\mathcal{S}$ if for all $n$ and $k$,*

$$\bigcup_{i=-k}^{k} \{i\} \times S_{n-1,j(n,k,i)} = S_{n,k}$$

*with $j(n,k,i) \leq k$. We say that $j$ is efficiently computable if there is a deterministic algorithm that computes $j(n,k,i)$ in time* $\mathrm{poly}(n, \log k, \log i)$.

**Lemma A.2.** *If a family of sets $\mathcal{S} := \{S_{n,k}\}_{n\geq 1,k\geq 0}$ has an efficiently computable index and if membership in $S_{1,k}$ can be computed in deterministic* $\mathrm{poly}(k)$ *time, then there is a deterministic algorithm that takes as input $n$, $k$, and $a \leq |S_{n,k}|$ and outputs the $a$th element in $S_{n,k}$ in the lexicographic order in time* $\mathrm{poly}(n, k, \log a)$.

The above lemma is useful when $k$ is small. But, we will be interested in cases when $k$ is quite large. In that case, we might not be able to quite find an algorithm that yields an efficient injection from $[|S_{n,k}|]$ to $S_{n,k}$, but the ideas below let us come quite close in some important special cases.

**Definition A.3.** *For a family of sets $\mathcal{S} := \{S_{n,k}\}_{n\geq 1,k\geq 0}$ and a function $L(n,k)$, we say that there is* an efficient $L$-sized injection to $\mathcal{S}$ *if there is a deterministic* $\mathrm{poly}(n, \log k, \log a)$*-time algorithm $\mathcal{A}$ that takes as input integers $n \geq 1$, $k \geq 0$, and $a \geq 1$ and outputs $s \in S_{n,k}$ such that if $a_1, a_2 \leq L(n,k)$ and $a_1 \neq a_2$, then $\mathcal{A}(n,k,a_1) \neq \mathcal{A}(n,k,2)$.*

**Definition A.4.** *For a family of sets $\mathcal{S} := \{S_{n,k}\}_{n\geq 1,k\geq 0}$ with index $j$, we say that a function $L : \mathbb{N}^2 \times \mathbb{Z} \to \mathbb{N}$ is* a lower bound for $\mathcal{S}$ consistent with $j$ *if for all $n \geq 1$, $k \geq 0$, and $-k \leq t \leq k$,*

1. *$L(1,k,t) = |S_{1,k} \cap [-k,t]|$;*
2. *$L(n,k,-k-1) = 0$;*
3. *$L(n,k,t) - L(n,k,t-1) \geq 0$ (i.e., $L$ is a non-decreasing function of $t$); and*
4. *$L(n+1,k,t) - L(n+1,k,t-1) \leq L(n, j(n+1,k,t), j(n+1,k,t))$.*

*We say that such $L$ and $j$ are* efficiently computable *if $L(n,k,t)$ and $j(n,k,t)$ can be computed deterministically in time* $\mathrm{poly}(n, \log k, \log t)$. *And, we write $\overline{L}(n,k) := L(n,k,k)$*

This definition might seem rather strange. But, notice, for example, that these conditions imply that if $L$ is a lower bound that is consistent with some index $j$, then $\overline{L}(n,k) \leq |S_{n,k}|$, and more generally that

$$L(n,k,t) \leq \sum_{i=-k}^{t} |S_{n,j(n,k,i)}| \,,$$

so we at least have some justification for the terminology "lower bound."

A very nice example is to take $S_{n,k} := \{\boldsymbol{z} \in \mathbb{Z}^n \ : \ \|\boldsymbol{z}\|_1 = k\}$. Then, a nice index function is simply $j(n,k,i) = k - |i|$. And, the sets $\{i\} \times S_{n-1,j(n,k,i)}$ simply correspond to the slices of the $\ell - 1$ sphere. In this example, one can then take $L(n,k,t)$ to be precisely equal $|\{\boldsymbol{z} \in \mathbb{Z}^n \ : \ \|\boldsymbol{z}\| = k, z_1 \leq t\}|$ since this happens to be efficiently computable. But, one can also use a volume-based estimate of this, estimating $\sum_{i=-k}^{t} S_{n-1,i}$ by $L(n,k,t) := \mathrm{vol}\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_1 = k - \delta \text{ and } x_1 \leq t\}$. For an appropriately chosen slack factor $\delta = \delta(n,k)$, this will satisfy the conditions outlined above.

The following shows that the existence of such an $L$ and $j$ is actually enough to imply an efficient $\overline{L}$-sized injective mapping to $\mathcal{S}$ for $\overline{L}$ that is related to $L$.

**Lemma A.5.** *Suppose that $\mathcal{S} := \{S_{n,k}\}$ with $S_{n,k} \subseteq [-k,k]$ is a doubly indexed family of sets with an efficiently computable index $j$ and efficiently computable lower bound $L$ for $\mathcal{S}$ consistent with $j$. Then, there is an efficient $\overline{L}$-sized injective mapping to $\mathcal{S}$.*

*Proof.* We describe our algorithm $\mathcal{A}(n,k,a)$ recursively as follows. The algorithm first uses binary search to find the minimal value of $t \in [-k,k]$ such that $L(n,k,t) \geq a$. If $n = 1$, it simply outputs $t$. Otherwise, it outputs $(t, \mathcal{A}(n-1, j(n,k,t), a - L(n,k,t-1))$.

First, notice that when $a \leq \overline{L}(n,k)$, there does always exist a minimal value of $t \in [-k,k]$ such that $L(n,k,t) \geq a$, and this will in fact be found by binary search because of the fact that $L(n,k,t)$ is non-decreasing. Notice as well that the running time of the algorithm is in fact $\mathrm{poly}(n, \log k, \log a)$. In particular, the running time of the algorithm satisfies the recurrence $T_{\mathcal{A}}(n,k,a) \leq T_{\mathcal{A}}(n-1, k-t, a - L(n,k,t)) + \mathrm{polylog}(n,k,a)$ with base case $T_{\mathcal{A}}(0,k,a) \leq \mathrm{poly}(\log k)$, and a simple argument shows that this is polynomially bounded.

Furthermore, notice that the fact that $j$ is an index implies that the output of the algorithm is in fact an element in $S_{n,k}$. So, the algorithm does at least output an element of $S_{n,k}$ and does have the claimed running time.

It remains to prove that the algorithm is an $\overline{L}$-sized injection, which we do by induction on $n$. In the base case when $n = 1$, a simple argument using Item 1 above shows that $\mathcal{A}(1,k,a)$ outputs precisely the $a$th element in the set $S_{1,k} \subseteq [-k,k]$. So, in particular, it is injective as a function of $a$ for $n = 1$.

Now, let $n \geq 2$. We assume for induction that the algorithm is injective on input $(n-1,k',a')$ for any $k'$ and any $a' \leq \overline{L}(n-1,k')$. Now, for $a_1 \neq a_2$ with $a_1, a_2 \leq \overline{L}(n,k)$, let $t_i$ be minimal such that $a_i \leq L(n,k,t_i)$. Notice that if $t_1 \neq t_2$, then clearly $\mathcal{A}(n,k,a_1) \neq \mathcal{A}(n,k,a_2)$ (since the first coordinates in the two outputs are $t_1$ and $t_2$ respectively). So, we may assume that $t_1 = t_2 = t$.

Let $k' := j(n,k,t)$ and $a_i' := a_i - L(n,k,t-1)$. It suffices to show that $\mathcal{A}(n-1,k',a_1') \neq \mathcal{A}(n-1,k',a_2')$. By the induction hypothesis, it suffices to show that $0 < a_i' \leq \overline{L}(n-1,k')$. Indeed, by the definition of $t$, we have that $L(n,k,t-1) < a_i \leq L(n,k,t)$, so that $0 < a_i' \leq L(n,k,t) - L(n,k,t-1)$. And, by Item 4, this implies that $a_i' \leq \overline{L}(n-1,k')$ as needed. □

## A.1 Injections into $\ell_p$ balls

**Lemma A.6.** *For any integer $p \geq 1$, let $\mathcal{S} := \{S_{n,k}\}$ where $S_{n,k} := \{\boldsymbol{z} \in \mathbb{Z}^n \ : \ \|\boldsymbol{z}\|_p^p \leq k\}$. Then, $j(n,k,i) := k - |i|^p$ is an efficiently computable index of $\mathcal{S}$ and*

$$L(n,k,t) := \mathrm{vol}_n\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - pk^{1-1/p}(n-1),\ x_1 \leq t\}\right) + 1$$

*is an efficiently computable lower bound for $\mathcal{S}$ that is consistent with $j$, where we interpret this expression as 0 if the set is empty. Furthermore, for $k \geq (10pn)^p$,*

$$\frac{\mathrm{vol}(\mathcal{B}_p^n(k^{1/p}))}{\overline{L}(n,k)} \leq 1 + 10n/k^{1/p}\,.$$

*Proof.* It is immediate that $j$ is an efficiently computable index of $\mathcal{S}$. And, it is immediate that $L$ is efficiently computable (ignoring issues of precision) and that $L$ satisfies all properties needed to be a lower bound for $\mathcal{S}$ *except* for Item 4. So, we only need to prove that $L$ satisfies Item 4.

In other words, we must prove that

$$\mathrm{vol}_{n+1}\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - pk^{1-1/p}n,\ t-1 \leq x_1 \leq t\}\right) \leq \mathrm{vol}_n\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - |t|^p - pk^{1-1/p}(n-1)\}\right)\,.$$

Notice that the left-hand side is zero if, e.g., $|t| \geq k^{1/p}$, so we may assume that $|t| \leq k^{1/p}$. Furthermore, if $t = 0$, a quick check shows that the inequality holds. So, we may assume that $1 \leq |t| \leq k^{1/p}$. Then,

$$\begin{aligned}
&\mathrm{vol}\left(\{\boldsymbol{x} \in \mathbb{R}^{n+1} \ : \ \|\boldsymbol{x}\|_p^p \leq k - pk^{1-1/p}n,\ t-1 \leq x_1 \leq t\}\right) \\
&\quad = \int_{t-1}^{t} \mathrm{vol}_n\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - |r|^p - pk^{1-1/p}(n+1)\}\right) \mathrm{d}r \\
&\quad \leq \mathrm{vol}_n\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - (|t|-1)^p - pk^{1-1/p}(n+1)\}\right) \\
&\quad \leq \mathrm{vol}_n\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - |t|^p + p|t|^{p-1} - pk^{1-1/p}(n+1)\}\right) \\
&\quad \leq \mathrm{vol}_n\left(\{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\|_p^p \leq k - |t|^p - k^{1-1/p}n\}\right)\,,
\end{aligned}$$

as needed.

The "furthermore" follows by recalling that $\mathrm{vol}_n(\mathcal{B}_p^n(r)) = r^n \cdot \mathrm{vol}_n(\mathcal{B}_p^n(1))$. □

**Corollary A.7.** *For any constant integer $p \geq 1$ and $q := q(n)$, let $\mathcal{S} := \{S_{n,k}\}$ where $S_{n,k} := \{\boldsymbol{z} \in \mathbb{Z}^n/q(n) : \|\boldsymbol{z}\|_p^p \leq k\}$. Then, there is a $\mathrm{poly}(n, \log q)$-time computable $L$-sized injection into $\mathcal{S}$ with*

$$L(n,k) \geq (1 - 20n/(qk^{1/p})) \cdot q^n \cdot \mathrm{vol}(\mathcal{B}_p^n(k^{1/p}))$$

*for all $k \geq (100pn/q)^p$.*

*Proof.* Notice that it suffices to consider the case $q = 1$ after rescaling appropriately. Then, combine Lemma A.6 with Lemma A.5. □

## A.2 An injection into the Hamming ball

We will use the following result from Knuth [Knu05].

**Lemma A.8** ([Knu05])**.** *The combinatorial number system defines a bijective function $f_{m,k} : [\binom{m}{k}] \to \binom{[m]}{k}$, which can be computed in time $\mathrm{poly}(m)$.*

**Algorithm 1** An injective map from $[A]$ into the Hamming ball

**Require:** $x \in [A]$
- $t = \max\{t' : V_q^n(t') \leq x\}$ // $x$ encodes a point in hamming sphere of radius $t$
- $u \leftarrow x - V_q^n(t-1) - 1$ // For convenience, we let $V_q^n(-1)$ be 0
- $v \leftarrow \lfloor u/(q-1)^t \rfloor$ // Note that $u \in [0, 1, \ldots, V_q^n(t-1) - 1]$
- $(i_1, i_2, \ldots, i_t) \leftarrow f_{n,t}(v+1)$
- Let $a_1, a_2, \ldots, a_t \in \{0, \ldots, q-2\}$ be the digits of $(u \bmod (q-1)^t)$ when written in base $q-1$.
- $(y_1, y_2, \ldots, y_n) \leftarrow (0, 0, \ldots, 0)$
- **for** $j \in \{i_1, i_2, \ldots, i_t\}$ **do**
  - $y_j = a_j + 1$
- **end for**
- **return** $(y_1, y_2, \ldots, y_n)$

**Lemma A.9.** *There is an algorithm that runs in time $\mathrm{poly}(\log(A), n)$, which on inputs $A, n$ outputs a circuit $\mathcal{C}_H^{n,A,q} : [A] \to \mathbb{F}_q^n$ that is an injective map from $[A]$ into $V_q^n(r)$, where $r$ is the smallest integer such that $V(\mathcal{B}_q^n(0,r)) \geq A$. In particular if $A = V_q^n(r)$ for some $r$, then $\mathcal{C}_H^{n,A,q}$ is bijective.*

*Proof.* It suffices to show an algorithm with running time $\mathrm{poly}(\log(A), n)$ that maps an integer $x \in [A]$ to $\mathcal{B}_q^n(0,r)$. Key to our translation will be the combinatorial number system from Lemma A.8. See Algorithm 1 for details. □



ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il