THE UNIVERSITY OF CHICAGO

ON VOLUME LEAKAGE BASED ATTACKS AGAINST SECURE OUTSOURCED DATABASES AND THEIR RELATION TO THE TURNPIKE PROBLEM AND TILINGS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
JESSE ALEXANDER STERN

CHICAGO, ILLINOIS
JUNE 2024

Dedicated to my family, for their endless love and support throughout my life, especially during difficult times.

To my loving and brilliant wife, who has filled every day since we met with joy and helped me improve in countless ways.

“. . . theories rest on conditions. Science does not inquire whether those conditions are fulfilled, but only what the result would be on the hypothesis of their fulfillment . . ." . . . What does his lucid explanation amount to but this, that in theory there is no difference between theory and practice, while in practice there is?

-Benjamin Brewster *The Yale Literary Magazine, 1881-1882*

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

There are many people in both my personal and professional life who have supported me both directly and indirectly in this work. While I cannot name everyone here, there are a few people I wish to explicitly acknowledge here.

First, I want to thank my adviser David Cash. David has supported my research here since very early on and has always had exceptional insight and advice when working together on projects. What really stood out to me however has been the immense amount of support David has provided me even when I was working on independent projects. David was always available to offer excellent advice about how to organize and prioritize my research as well as helping me stay on track within the program as a whole. David's care and consideration for his students are always evident, and I greatly appreciate the time and effort he has put into helping me improve as a researcher and achieve my goals within the program.

I want to thank all of my collaborators throughout the years. Not only have these collaborations been the source of numerous joint results, but they have also helped me grow as a researcher. Discussing problems with other people deeply engaged on the same problems has a way of offering insights not only on the problems in question, but also into different ways of thinking about and solving problems in general.

I also want to thank my colleague Neng Huang. Neng was always very selfless with his time as I worked on my tiling research. Working independently on that project presented me with many new challenges. Having someone to speak to periodically about these results and receive feedback from was always extremely helpful and I am very grateful for the many conversations he took the time to have with me on this topic.

Last, but certainly not least, I want to thank my wife Yichen Hou. A Ph.D. can be a long and grueling experience at times. Being able to share that experience and offer mutual support has made it infinitely easier and improved the quality of both my life and my research.

ABSTRACT

When outsourcing data, one of the things the host learns about the data is the number of records returned in response to each user query. We refer to this as volume leakage and study attacks on databases that leak the volumes of range queries over database attributes. We provide two new attacks within this area, one against noisy volume leakage and another against volume leakage of d-dimensional databases for $d > 1$ (where previously work only considered the case of $d = 1$).

To build these attack algorithms, we prove the equivalence of this problem to the turnpike problem. We then take ideas from algorithms previously used to solve the turnpike problem and expand upon them to yield our attacks. This connection also allows us to directly apply numerous theoretical results from the turnpike problem to the problem of database reconstruction from volume leakage. Further, we prove several new theoretical results with respect to the database reconstruction problem using ideas from the turnpike literature.

In order to address the issue of determining uniqueness of some reconstruction of a database from volume leakage, we are forced to take a step back and study the mathematical foundations of the problem. We do this by reducing the problem of database reconstruction to that of tiling finite multisets with a tile of fixed size. Unfortunately, many of the questions we asked with respect to multisets were still open, even with respect to sets of contiguous elements. Thus, we start by proving structural results about the properties of tilings of finite intervals of integers and prove upper and lower bounds on the number of such tilings.

In an initial effort to generalize these tiling results and work towards our desired database reconstruction results, we generalize our tiling results in two major ways. The first is to generalize them to tilings of d-dimensional finite sets. The second is to generalize them to certain families of non-contiguous finite sets. We hope that further generalizations of our latter results will yield an algorithm for efficiently determining uniqueness of a database reconstruction attack's output.

CHAPTER 1

INTRODUCTION

Big data plays a critical role in modern computation, but with big data comes massive storage requirements. Further, it is often the case that many people require access to the same data. Rather than maintaining a personal server room, many have turned to the more convenient and efficient solution of cloud databases. While outsourcing ones data in this way has many advantages, trusting a vendor with potentially sensitive data requires careful analysis of the potential security vulnerabilities of such a scheme. Even if one's data is in encrypted, storing it on a cloud database has risks, as the vendor or other parties able to eavesdrop can learn information about what encrypted information users are accessing.

While different settings result in distinct forms of leakage, one form of leakage that is extremely difficult to entirely remove is volume leakage. At a high level, we say that a query to retrieve some subset of outsourced data leaks query volume if some adversarial party is able to infer the number of records sent back to the user in response to the query. The study of volume leakage-based attacks was initiated by Kellaris et al. [16] where they showed that, after viewing the volume leakage of sufficiently many queries over ranges of record attributes, they could recover how many records were associated with each attribute. This initial work on volume-based attacks left many open questions which we broadly sort into two categories:

- *Practical questions about such attacks in real-world environments.* This includes relaxations of assumptions such as those on the query distribution and noise, as well as potential defenses such as padding the database via the inclusion of dummy records.
- *Theoretical questions about the underlying mathematics of the associated reconstruction problem.* This includes information theoretic questions about the set of solutions consistent with the leakage (such as the number of such solutions and their properties), as well as worst-case runtime analysis of attacks.

A fundamental contribution of our work is our reduction from the problem of reconstructing a database from multiset volume leakage to the turnpike problem, a well-studied computer science problem that is also connected to problems in biology and physics. From this, we are able to port numerous practical and theoretical results from the turnpike literature. This reduction also allows us to develop new practical and theoretical results based on the ideas of a wider range of prior works.

Most of the recent cloud database security literature has focused on more practical questions. In this dissertation, we consider questions of both of the above types. In terms of practical questions, our first contribution is to classify some of the various noise models one can work under in this setting. Different methods for modeling noise correspond to distinct real-world settings, and the diversity of of noise models used in recent works reflects this. Thus, we formalize the noise models previously used in the literature on cloud database attacks here, as well as contribute several new ones. With these noise models in hand, we focus in on the natural noise model where each volume has noise added independently from some range (naturally modeling cases where an eavesdropper's measure of the number of records returned is off by some additive or multiplicative factor). We then give a new noisy volume leakage-based attack that allows for database reconstruction in the presence of such noise. Further, we prove that our attack can, for some databases, uniquely reconstruct the database[1] despite the presence of noise.

As to theoretical contributions, we prove upper and lower bounds on the number of valid reconstructions of a database from volume leakage, utilizing bounds from the turnpike literature in the case of multiset volume leakage and using a new construction in the case of a lower bound given set volume leakage. We prove a reduction from turnpike (and thus, from database reconstruction as well given their equivalence) to a new finite tiling problem. From this, we investigate the mathematical foundations of this problem via tilings. With

1. Up to reflection of the databases attribute, which is an information-theoretic restriction that exists even when there is no noise present.

respect tilings, we are able to give characterizations of all valid tilings of finite contiguous sets of integers by tiles of fixed size. We then work to generalize these results to higher dimensions and to the tiling of non-contiguous sets. To apply these results back to outsourced database attacks, we would like to eventually extend these results to tilings of multisets. While we are unable to complete this program here, we make meaningful progress towards this goal. Achieving this generalization of our tiling results has the potential to yield an efficient algorithm for determining if a database reconstruction from multiset volume leakage is unique, significantly improving attack capabilities on volume leakage.

1.1 Organization

This dissertation is organized into four chapters as well as one appendix.

- Chapter 1 provides a high-level introduction to the overall focus of the dissertation and highlights some of the key prior works that have influence across several other chapters. We also cover some of the most prominent definitions and concepts that appear in later chapters.

- Chapter 2 focuses on new attacks against databases that support range queries and leak query volumes. We establish a connection to prior work by proving the equivalence of this problem in 1-dimension to the turnpike problem. Utilizing this connection, several theoretical results are adapted to apply to database attacks, and a new attack is developed that works on noisy data. We also give the first attack against d-dimensional databases using their multiset volume leakage. Further, our attack using noisy leakage and attack against d-dimensional databases can be combined in a natural way to yield attacks on noisy volume leakage of d-dimensional databases supporting range queries.

- Chapter 3 is concerned with improving our understanding of the underlying mathematical structure of the turnpike problem. We begin by reducing the turnpike problem to the tiling problem of finding tilings of a finite multiset by a tile of fixed size. We then work to establish a foundation for the problem of studying tilings of multisets by studying the related problem of tiling contiguous sets, providing characterizations of and bounds on the number of such tilings.

- Chapter 4 covers generalizations of the results of Chapter 3 to tilings of d-dimensional finite sets. Further, we work to begin removing the restriction from Chapter 3 of the set to be tiled needing to be contiguous.

- The appendix contains full psudocode for all of our algorithms and their sub-routines from chapter 2, as well as psudocode for the main backtracking algorithm for turnpike from prior work [27].

1.2 Preliminaries

We briefly cover some key concepts and definitions that will be used throughout this work.

1.2.1 Databases

We define a 1d-database DB with domain size N and m records as a multiset of values $\mathsf{DB} = \{\{r_1, r_2, \ldots, r_m\}\}$ where $r_i \in [N]$ for all $i \in [m]$. Alternatively, we typically write $\mathsf{DB} = (v_1, \ldots, v_N)$, where $v_i = \sum_{j=1}^{m} \mathbb{1}_{[r_j = i]}$, which corresponds to the number of records taking value i in the database. We say that a database supports **range queries** if and only if a user can make a query of the form $[x, y]$ for $x, y \in [N]$ such that $x \leq y$ and receive all $r_i \in [x, y]$ on response.

We define a d-dimensional database DB with i^{th} dimensional domain size N_j for each $j \in [d]$ and m records as a multiset of tuples $\mathsf{DB} = \{\{r_1, r_2, \ldots, r_m\}\}$ where $r_i \in [N_1] \times \ldots \times [N_d]$ for all $i \in [n]$. Alternatively, we typically write $\mathsf{DB} = (v_{(1,\ldots,1)}, \ldots, v_{(N_1,\ldots,N_d)})$, where $v_{\mathbf{x}} = \sum_{k=1}^{m} \mathbb{1}_{[r_k = \mathbf{x}]}$, which corresponds to the number of records taking each of the d values in the tuple $\mathbf{x}$ in the corresponding dimension of the database. In a natural generalization of range queries to a 1d-database, a range query to a d-dimensional database is a d-dimensional rectangle in $[N_1] \times \ldots \times [N_d]$.

1.2.2 Volume Leakage

We say that a database leaks query volume if and only if some adversarial party learns the number of records returned in response to any user query. Kellaris et al. [16] introduced volume-based reconstruction attacks. Their paper considers a model in which an adversary is observing range queries made to a domain of size N. In their model, they note that an adversary must observe $\Omega(N^4 \log N)$ uniformly random queries in order to estimate v_i, the number of queries that return exactly i records. In our notation, we will refer to this type of leakage as the **multiset model**, and we present attackers whose input is a multiset Vols of volumes, where i is in Vols exactly v_i times. Implicitly these attacks are run only after observing sufficiently many queries to construct the multiset.

Later work, beginning with [12], presents attacks that work in a model making fewer assumptions about the query distribution and the number of queries the adversary observes. Although there are a few different models, we will consider one in which an adversary observes $\Omega(N^2 \log N)$ range queries in order to observe each record at least once. In this model, the adversary cannot estimate the counts accurately (via a lower bound from [16]). Instead, the adversary can determine whether or not at least one range query contains exactly i records (but not their multiplicity). We refer to this model as the **set model** and note that is strictly stronger than the multiset model, in the sense that any attack in the set model can be run in the multiset model and that the model makes fewer assumptions.

Although we formally define the honest-but-curious setting in Figure 1.1, we will generally consider these models in the setting after the adversary has either captured the multiset or set of volume information respectively. Similarly to Kellaris et al., [16], we focus on the volume leakage of databases that support range queries.

```
$\mathrm{G}_{\Pi,\mathcal{L},\mathcal{S},\mathsf{DataGen},\mathsf{QueryGen}}(\mathcal{A})$ :
$b \leftarrow_{\$} \{0,1\}$
$\mathsf{DB} \leftarrow \mathsf{DataGen}$
$C \leftarrow \Pi.\mathsf{Setup}(\mathsf{DB})$
$(\ell, \mathsf{stL}_{\mathcal{L}}) \leftarrow \mathcal{L}_{\mathsf{setup}}(\mathsf{DB}, \perp)$
$(C', \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(\ell, \perp)$
If $b = 0$ run $b' \leftarrow \mathcal{A}^{\mathcal{O}}(C)$
Else run $b' \leftarrow \mathcal{A}^{\mathcal{O}}(C')$
Return $[b = b']$

$\mathcal{O}()$ :
$q \leftarrow \mathsf{QueryGen}$
$c \leftarrow \Pi.\mathsf{Query}(q, C)$
$(\ell, \mathsf{stL}_{\mathcal{L}}) \leftarrow \mathcal{L}_{\mathsf{query}}(q, \mathsf{stL}_{\mathcal{L}})$
$(c', \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(\ell, \mathsf{st}_{\mathcal{S}})$
If $b = 0$ return $c$
Else return $c'$
```

Figure 1.1: Formal game defining an honest but curious adversary against a cryptographic scheme Π.

CHAPTER 2
NOISY D-DIMENSIONAL DATABASE RECONSTRUCTION FROM VOLUME LEAKAGE

2.1 Introduction

In this chapter, we provide a security evaluation of schemes supporting range queries and leaking volume with the following contributions:

- We formally provide an equivalence between volume-based reconstruction and the turnpike problem. From this equivalence, we import theoretical results and new attacks to volume-based reconstruction setting. This includes new guarantees on attacks from prior work [16], new attacks, and bounds on the size of homometric solutions for the set and multiset cases.

- We provide new set and multiset volume-based reconstruction attacks in noisy models based on backtracking algorithms from the turnpike problem literature. Our attacks perform better than attacks from prior work in the regime we care about, and our techniques give an attack on a previously unconsidered noise model.

- We give the first volume-based reconstruction attack against databases with dimension greater than one and give attacks for the multiset volume leakage. Our attacks use a recursive backtracking structure to generalize to any dimension. The form of our algorithms also lends themselves to "partial progress," which boosts performance by running a lower dimensional attack and seeding the higher dimensional attack with the result.

These results are part of an as-of-yet unpublished joint work with David Cash and Alex Hoover.

2.1.1 Modified Sets of Queries

In practice, it may be the case that some range quires a database supports are never actually made by the client and thus, no volume leakage with respect to such queries can be expected. Thus, following up on the initial attacks utilizing volume leakage [12, 16], several lines of work began to study similar attacks in circumstances where only certain types of query volumes are leaked. Gui et al. [13] consider this setting by restricting the window size of the queries that are leaked (i.e. any given query may only be over at most b attributes for some constant b). In a later work, Kornaropoulos et al. [19] consider databases that support range queries as constructed by Demertzis et al. [8]. For these databases, only a subset of range queries are actually supported, and client's queries are answered by returning several smaller range queries or a superset of the client's query utilizing only those range queries that are supported. Unlike the set model, neither of these models is strictly stronger than the multiset model. Given that the best set of allowable queries for such a model is unclear as is their relationship with the multiset and set models, we do not consider them in this work.

Consideration has also been given to models where the attacker has some additional ability that allows them to influence queries or their volumes. One example of this allows the attacker to perform an injection attack and add records to the database [2, 34], which impacts the volumes of queries the adversary sees in a key way. Such capabilities can significantly improve the ability of an attacker to reconstruct a database, even in the presence of noise. While we do not allow the attacker access to such powerful tools in this work, it is possible our attacks could all be improved with access to some additional capability or information based on context.

2.1.2 The Turnpike Problem and Equivalent Problems

The turnpike problem is one of several problems with identical underlying mathematical structures. The first appearance of such a problem was in X-ray crystallography in the case of solving the phase problem [22, 23]. In this setting, the peak positions output from the Patterson function are the inter-atomic distance vectors, with the height of a peak providing information as to the multiplicity of atoms with the corresponding inter-atomic distance. Later, the problem of inferring DNA structure from partial digest data appeared in biology, with the earliest attempts to solve this problem algorithmicly appearing in the late 70s [30]. A polynomial factorization approach to the problem was later discovered by Rosenblatt and Seymour [26]. Not only did this provide methods for solving the problem, but established the properties of homometric sets (i.e. the properties of the set of all solutions to any given instance).

In both the contexts of X-ray crystallography and partial digest, measurement errors in the data are significant issues. Thus, there has been both theoretical and algorithmic progress in studying these problems in such a setting. On the theory side, it was shown that additive or multiplicative error applied to each value in the multiset results in the corresponding reconstruction problem becoming strongly NP-complete [5]. Following up on these results, it was then also proven that missing data leads to the problem being NP-hard, and adding additional data makes the problem hard to approximate (as hard as approximating MAX CLIQUE) [6].

2.1.3 Other Prior Work

While d-dimensional access pattern leakage attacks have been considered [9, 10], there has not yet been any work on d-dimensional volume leakage attacks. On the other hand, there has been work in higher dimensions for the problems of solving the phase problem in X-ray crystallography using the Patterson function and the Turnpike problem. Unfortunately, while we prove the equivalence of solving the phase problem using the Patterson function, the Turnpike problem, and database reconstruction from multiset volume leakage in 1-dimension, all three of these problems are distinct in higher dimensions.

With respect to quantifying how many distinct databases have the same leakage, there have been several efforts, but none particularly focused on the volume leakage setting. Kornaropoulos et al. [19] consider how many databases have the same leakage, but do so in various keyword-based settings and generally with access pattern. As it does not consider volume leakage in either the multiset or set models, their results and our own are disjoint. A follow-up work [18] provides a more general framework for considering what databases are information-theoretically indistinguishable, which they refer to as the reconstruction space of the database. Unfortunately, they do not offer non-trivial bounds on the size of a database's reconstruction space given its multiset or set volume leakage, as their results are not specifically tuned for this purpose.

2.2 Volume-based Reconstruction in 1-Dimension

The study of database reconstruction in 1-dimension from volume leakage was initiated by Kellaris et al. [16]. They give two different types of attacks. The first is a polynomial factorization-based method and the other is a backtracking algorithm. One of our main contributions in this section is to prove that multiset volume-based reconstruction attacks and turnpike reconstruction are reducible to one another in linear time and, from this connection, highlight several important prior results from the turnpike literature and their implications for multiset volume-based reconstruction. Motivated by the work of Zhang on the turnpike problem [35], our second contribution in this section is a proof that the homometric set size (or reconstruction space as it is referred to in [18]) of a database $\mathsf{DB} = (v_1, \ldots, v_N)$ can be exponential in N. Lastly, we conclude this section with a review of the backtracking algorithm of Kellaris et al. [16] (independently discovered in the turnpike literature by Skeina et al. [27]), as this algorithm motivates our own for attacks on multidimensional and noisy leakage.

We begin with the definitions of multiset volume-based reconstruction attacks and set volume-based reconstruction attacks.

Definition 1. *For a 1d-database* $\mathsf{DB} = (v_1, \ldots, v_N)$ *with* m *records and domain size* N*, a* ***multiset volume-based reconstruction attack*** *is one which takes as input*

$$\mathsf{Vols} = \left\{\left\{\sum_{k=i}^{j} v_k : 1 \leq i \leq j \leq N\right\}\right\}$$

and outputs a database $\mathsf{DB}' = (v'_1, \ldots, v'_N)$ *with domain size* N *and* m *records. We say that a multiset volume-based reconstruction attack* ***succeeds*** *if and only if*

$$\mathsf{Vols} = \left\{\left\{\sum_{k=i}^{j} v'_k : 1 \leq i \leq j \leq N\right\}\right\}$$

Definition 2. *For a 1d-database* $\mathsf{DB} = (v_1, \ldots, v_N)$ *with* m *records and domain size* N*, a* ***multiset volume-based reconstruction attack*** *is one which takes as input* (Vols_{set}, N) *where we let*

$$\mathsf{Vols}_{set} = \left\{\sum_{k=i}^{j} v_k : 1 \leq i \leq j \leq N\right\}$$

and outputs a database $\mathsf{DB}' = (v'_1, \ldots, v'_N)$ *with domain size* N *and* m *records. We say that a multiset volume-based reconstruction attack* ***succeeds*** *if and only if*

$$\mathsf{Vols}_{set} = \left\{\sum_{k=i}^{j} v'_k : 1 \leq i \leq j \leq N\right\}$$

It will often be convenient to, with respect to some DB and some leakage model, refer to those databases that are indistinguishable from DB. Thus, if an attack A like the ones defined above succeeds by outputting when DB' when given the leakage of DB, we say that DB and DB' are **information theoretically consistent**. This is always with respect to some leakage model that will, in this work, always be clear from context. We now proceed with the definitions of turnpike reconstruction and the turnpike problem.

Definition 3. *A* ***turnpike reconstruction*** *takes as input a multiset D of $\binom{N}{2} - N$ positive integers and outputs a set of natural numbers L' of size N that includes 0. We say that a turnpike reconstruction* ***succeeds*** *if and only if*

$$D = \{\{x - y \in (L - L) : x > y\}\}.$$

The ***turnpike problem*** *is a decision problem that is true if and only if, its input is a multiset D of $\binom{N}{2} - N$ positive integers and there exists a set L' such that a turnpike reconstruction that outputs L' on input D succeeds.*

We can now prove that multiset volume-based reconstruction attacks and turnpike reconstructions are reducible to one another in linear time.

Corollary 1. *For any multiset volume-based reconstruction attack A, there exists a turnpike reconstruction A' such that*

- *The runtime of A' is the runtime of A plus $O(N)$.*
- *If A succeeds on $\mathsf{DB} = (v_1, \ldots, v_N)$ with probability p, then A' succeeds on*

$$D = \left\{\left\{\sum_{k=i}^{j} v_k : 1 \leq i \leq j \leq N\right\}\right\} \cup \{\{0\}\}$$

  with probability p.

Proof. A' runs A on its input, adds the element 0 to the output DB' of A, then outputs the result as L'. Notice that, if we index the elements $x_i \in L$ by their value (in increasing order starting with 0), v_i equals $x_i - x_{i-1}$ as desired. □

Corollary 2. *For any turnpike reconstruction R, there exists a multiset volume-based reconstruction attack R' such that*

- *The runtime of R' is the runtime of R plus $O\Big(\binom{N}{2}\Big)$.*
- *If, for some set of non-negative integers L of size N that includes 0, R succeeds on $D = \{\{x - y \in (L - L) : x > y\}$ with probability p, then R' succeeds on $\mathsf{DB} = L \setminus \{0\}$ with probability p.*

Proof. R' runs R on its input, removes the element 0 from the output L' of R, then outputs the result as DB'. □

2.2.1 Application of Turnpike Results to Database Reconstruction

With the above connection to turnpike, it follows that the polynomial factorization of Kellaris et al. [16] is equivalent to the polynomial factorization method from X-ray crystallography of Rosenblatt and Seymour [26], while their backtracking algorithm is equivalent to the backtracking algorithm of Skiena, Smith, and Lemke for solving turnpike [27]. We now prove several interesting corollaries about both database reconstruction algorithms, as well as the database reconstruction problem in general.

Corollary 3. *There exists databases $\mathsf{DB} = (v_1, \ldots, v_N)$ such that, given Vols, the number of databases information theoretically consistent with DB is at least $(n+1)^{0.8107144}/2$. Further, given Vols for any DB, the number of databases consistent with DB is at most $(n+1)^{2.4649654}/2$. If DB is not sparse (i.e. has no v_i with no records), then the upper bound can be improved to $(n+1)^{1.2324827}/2$.*

Proof. The polynomial factorization approach of Rosenblatt and Seymour [26] was later analyzed by Lemke and Werman [21], where the bulk of these results were proven. Skiena et al. [27] put these bounds into the context of the turnpike problem and extend the upper bound to the case where a point can appear multiple times on a line (which translates to the sparse case for databases in our setting). Thus, these results follow from theirs and our reduction. As the size of a turnpike instance is 1 greater than that of the corresponding database reconstruction problem (i.e. $N = n + 1$), we have modified the results accordingly. □

Corollary 4. *For any* $\mathsf{DB} = (v_1, \ldots, v_N)$*, given* Vols*, the polynomial factorization attack of Kellaris et al. runs in polynomial time with respect to* $|\mathsf{Vols}|$*.*

Proof. Kellaris et al. are aware of this in the case of unique reconstruction (though they do not highlight this directly, it follows from their use of LLL and the bound on the degree of the polynomial). The polynomial upper bound on homometric set size of Lemke and Werman [21] and Skiena et al. [27] is proven via a polynomial factorization-based argument that shows the number of non-self-reciprocal irreducible factors is upper bounded by a polylog factor. Lemke and Werman [21] then did runtime analysis in the presence of such bounds to prove that their problem could be solved in pseudo-polynomial time (i.e. in polynomial time with respect to the largest distance in the given interpoint distance set). Thus, the last step of the Kellaris et al. method that tries all possible ways to sort irreducible factors into two reciprocal products runs in polynomial time with respect to $|\mathsf{Vols}|$. □

Corollary 5. *For any* $\mathsf{DB} = (v_1, \ldots, v_N)$*, given* Vols*, the backtracking attack of Kellaris et al. has worst case runtime* $\Omega(2^{(N/5)-2})$*.*

Proof. Zhang [35] gives a family of exponential time instances for the backtracking algorithm of Skiena and Sundaram [28] to which the Kellaris et al. algorithm is equivalent. □

Motivated by the family of turnpike instances defined by Zhang [35], we can define a family of databases such that, when given access only to set leakage, they have exponentially many informational theoretically consistent reconstructions from that leakage. To put this another way, the homometric set (or reconstruction space) of DB with set volume-based leakage can be exponential in N.

Lemma 1. *For any $\epsilon > 0$, there exists a $\mathsf{DB} = (v_1, \ldots, v_N)$ such that there are $(1-\epsilon)\binom{N/3}{N/6}$ distinct databases on which a set volume-based reconstruction attack succeeds.*

Proof. For any database DB and any i and j such that $1 \leq i < j \leq N$, we call $v_i, \ldots v_j$ a **dense region** if and only if $\forall k \in [i, j](v_k = 1)$. For any database DB and any i and j such that $1 \leq i < j \leq N$, we call $v_i, \ldots v_j$ a **sparse region** if and only if $\sum_{k=i}^{j} v_k = 2(j-i+1)+1$. For any $\ell \in \mathbb{Z}^+$, consider any database $\mathsf{DB}(v_1, \ldots, v_N)$ where $N = (2r+1)(2\ell+1)$ made up of $\ell + 1$ dense regions of size $2r + 1$ (i.e. $j - i + 1 = 2r + 1$) with exactly one sparse region of size r between each dense region. Notice that, regardless of the distribution of records within sparse regions, $\mathsf{Vols}_{\text{set}} = [N]$. Each sparse region has $\binom{2r}{r}$ configurations and thus, if $\ell = 1$, we have $N = 4r + 2$ with a homometric set of size $\binom{2r}{r}$. To get the claim, we not only allow for greater ℓ, but also also allow for each sparse region to have any size, but without changing the number of records (and without changing the sum of the sizes of all of the sparse regions). This results in a homometric set of size $\binom{2r\ell}{r\ell}$ for $N = (2r + 1)(2\ell + 1)$. As ℓ increases, this approaches a homometric set size of $\binom{N/3}{N/6}$. $\square$

2.2.2 Backtracking Algorithm

We now review backtracking-based multiset volume-based reconstruction attack by [16], but written in a format closer to that of the equivalent backtracking algorithm for solving turnpike [27]. For Vols of size $\binom{N}{2} + N$, the algorithm maintains a **backtracking pyramid** (which we denote by B) with N rows and i elements in row i. We use $(i, j)_B$ to indicate the element in row i and column j of the backtracking pyramid. The algorithm then works to fill in $(i, j)_B$ for all i and j in such a way that $(i, j)_B$ is equal to the number of records with an index in the range of v_i to v_{n+i-j}. As the size of B is exactly that of Vols, filling in B with exactly the elements of Vols serves as a multiset volume-based reconstruction attack.

The algorithm proceeds by taking the largest unassigned volume from Vols and assigning it the first empty location in B along either the left or right side of (i.e. $(i,1)_B$ or $(j,j)_B$ for the minimum i or j such that that position of B has not been assigned a volume). After the k^{th} such assignment, notice that we will learn $k-1$ additional volumes. For example, if we just assigned a volume $(j,j)_B$ that implies that, for all i such that $i<j$, $(i,i)_B$ has been assigned a volume. Additionally, as $(i,i)_B$ and $(j,j)_B$ corresponds to the number of records with an index in the range of v_i to v_{n+i-i} and v_j to v_{n+j-j}, it follows that $(i,i)_B-(j,j)_B$ corresponds to the number of records with an index in the range of v_i to v_j. Thus, when we assign a new volume, we can take its difference with $k-1$ other elements along the side of B it was placed and use these values to fill in $k-1$ additional elements of B. While this means that the algorithm proceeds for at most N steps, the guess of assigning a volume to the left or right means the upper bound on its runtime is 2^N. We provide pseudocode for this algorithm in algorithm 1, which runs algorithms 2, 3, and 4 as subroutines.

2.3 Noisy Volume-based Reconstruction in 1 Dimension

We begin by defining various noise models, then give a novel noisy multiset reconstruction attack

Definition 4. *For a 1d-database $\mathsf{DB}=(v_1,\dots,v_N)$ with m records and domain size N, a **noisy multiset reconstruction attack** is one which takes as input the domain size N and*

$$\mathit{Vols}_{\sim}\left\{\left\{z_{i,j}+\sum_{k=i}^{j} v_k : 1\leq i\leq j\leq M\right\}\right\},$$

*where $z_{i,j}\leftarrow^{\$}\chi_{i,j,\mathsf{DB}}$ is a set of volumes drawn from a noise distribution that can depend on the database. Then, the attack outputs a database $\mathsf{DB}'=(v'_1,\dots,v'_N)$ with domain size N. We say that a noisy multiset reconstruction attack **succeeds** if and only if the probability* *that*

$$\mathsf{Vols}_{\sim} = \left\{\left\{ z_{i,j} + \sum_{k=i}^{j} v'_k : 1 \leq i \leq j \leq N \right\}\right\}$$

is greater than 0.

Definition 5. *For a 1d-database* $\mathsf{DB} = (v_1, \ldots, v_N)$ *with* m *records and domain size* N*, a* ***noisy set reconstruction attack*** *is one which takes as input the domain size* N *and*

$$\mathsf{Vols}_{\sim set} = \left\{ z_{i,j} + \sum_{k=i}^{j} v_k : 1 \leq i \leq j \leq N \right\},$$

where $z_{i,j} \leftarrow^{\$} \chi_{i,j,\mathsf{DB}}$ *is a set of volumes drawn from a noise distribution that can depend on the database. Then, the attack outputs a database* DB' *with domain size* N*. We say that a noisy set reconstruction attack* ***succeeds*** *if and only if the probability that*

$$\mathsf{Vols}_{\sim set} = \left\{ z_{i,j} + \sum_{k=i}^{j} v'_k : 1 \leq i \leq j \leq N \right\}$$

is greater than 0.

The noisy multiset reconstruction attack was initially defined in [13]. They also define two other noise models. While we do not give any additional novel algorithms for this problem in this work, we are able to prove results with respect to these models via the connection to the turnpike problem.

For a noisy multiset reconstruction attack, for each element of DB, we may be able to reconstruct the elements of DB up to some error. In such cases, it may make more sense to output a set of possible values for each element of DB as opposed to a single one. This motivates a new attack definition.

Definition 6. *For a 1d-database* $\mathsf{DB} = (v_1, \ldots, v_N)$ *with* m *records and domain size* N*, a* ***noisy multiset full reconstruction attack*** *is one which takes as input the domain size*

N and

$$\left\{\left\{ z_{i,j} + \sum_{k=i}^{j} v_k : 1 \leq i \leq j \leq N \right\}\right\},$$

where $z_{i,j} \leftarrow^{\$} \chi_{i,j,\mathsf{DB}}$ is a set of volumes drawn from a noise distribution that can depend on the database. Then, the attack outputs a N tuple of set approx $-$ $\mathsf{DB} = (V_1, \ldots, V_N)$. We say that a noisy multiset full reconstruction attack ***succeeds*** *if and only if for all $i \in [N]$ we have that $v' \in V_i$ if and only if there exists a DB' with i^{th} element v' such that a noisy multiset reconstruction attack with input DB and output DB' succeeds.*

One type of noise, considered by Gui et al. [13] captures a real-world setting where an adversary is confident they have observed every range query at least once but they may have also recorded volumes that do not correspond to range queries on the data. An example of this would be an adversary that observes both continuous range queries from users along with a few discontinuous range queries (a query on a disjunction of ranges).

Definition 7. *For a 1d-database $\mathsf{DB} = (v_1, \ldots, v_N)$ with m records and domain size N* ***spurious volumes reconstruction attack*** *is one which takes as input the number of records n, the domain size M, and*

$$\mathsf{Vols}_{spur} = \left\{ \sum_{k=i}^{j} v_k : 1 \leq i \leq j \leq N \right\} \cup Z,$$

where $Z \leftarrow^{\$} \chi_{\mathsf{DB}}$ is a set of volumes drawn from a noise distribution that can depend on the database. Then, the attack outputs a database DB' with domain size N.

Another noise model was proposed by [34]. While we do not discuss this noise model at length, we include it here for completeness. At a high level, this “noise" can be introduced as a defense against volume-based reconstruction by padding the database with fake records.

Definition 8. *For a 1d-database $\mathsf{DB} = (v_1, \ldots, v_N)$ with m records and domain size N, a* ***noisy record-padded multiset reconstruction attack*** *is one which takes as input the* *domain size N, a positive integer s and*

$$\left\{\left\{\sum_{k=i}^{j} v_k + z_k : 1 \le i \le j \le N\right\}\right\},$$

where $z_k \in [0, s] \xleftarrow{\$} \chi_{i,j,\mathsf{DB}}$ is a set of additional records drawn from a noise distribution that can depend on the database. Then, the attack outputs a database DB' with domain size N.

The last noise model we define models the case of missing volumes (or at least missing multiplicity of some volumes). This formalization is motivated by a noisy turnpike model proven to be NP-hard [6].

Definition 9. *For a 1d-database $\mathsf{DB} = (v_1, \ldots, v_N)$ with m records and domain size N, a* ***multiset superset reconstruction attack*** *is one which takes as input the domain size N and*

$$\left\{\left\{\sum_{k=i}^{j} v_k : 1 \le i \le j \le N\right\}\right\} \cup X,$$

where X is a multiset set of volumes drawn from a noise distribution that can depend on the database and V. Then, the attack outputs a database DB' with domain size N.

2.3.1 Algorithm for Noisy Multiset Reconstruction Attack

In this section, we first give a high-level overview of a backtracking algorithm for partial digest [28] (recall that partial digest is equivalent to the turnpike problem as well as noisy multiset reconstruction). We then give a new noisy multiset full reconstruction attack algorithm. While our algorithm also takes a backtracking approach, it differs from the prior work in several key ways. For simplicity, we begin by assuming that the database is 1-dimensional and that the noise is such that $\forall a, b \in \mathbb{Z}(k \in [a, b] \Leftrightarrow \Pr(z_{i,j} = k) > 0$. The former restriction can be removed by using ideas from our multidimensional algorithm in the following section, whereas the latter condition may or may not be easy to remove depending on the noise distribution in question.

Our approach is motivated by the work of Skiena and Sundaram [28]. In their work they use a backtracking algorithm to solve the Turnpike problem where a small amount of multiplicative noise has been applied to each interpoint distance. To deal with the noise, they treat elements as intervals (as opposed to integers like in the standard backtracking algorithm for the Turnpike problem [27]). There are two key issues with this. The first is that, as one proceeds in constructing the backtracking pyramid, assignments and implications are no longer unambiguous (beyond the choice of whether the next largest distance will be placed on the left or right side of the backtracking pyramid). For example, if we calculate that some volume is equal to $x - y$ for $x \in [8, 10]$ and $y \in [3, 5]$, then we only know that $x - y \in [\min[x] - \max[y], \max[x] - \min[y]] = [3, 7]$. If our multiset of volume intervals contains the intervals $[2, 4], [4, 6], [5, 7], [6, 8]$, then any of these is a valid interval to place at this point in the backtracking pyramid. Thus, if all elements in the base of the pyramid intersect, one can end up with a runtime on the order of $n!$ as opposed to 2^n. To address this, they adopt a grouping idea of Chang's [4], where they group several intervals based on their mean. While this can lead to a runtime improvement, it can also cause the algorithm to miss valid solutions.

The second issue with this algorithm is that, even once a backtracking pyramid is fully constructed, its elements are intervals and thus, one may still need to do an additional backtracking process of guessing values within those intervals in order to arrive at a solution, a process which could easily take exponential time. This issue is best illustrated by the example in figure 2.1.

In this work, we do not take the mean of any intervals, nor do we add the volume intervals we start with to our backtracking pyramid directly. Instead, we maintain a quadratic number of counters that keeps track of how many copies of various intervals can exist in the

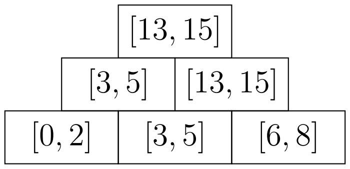

Figure 2.1: Example of a complete backtracking pyramid from the algorithm of Skiena et al. Suppose we are given $\mathsf{Vols} = \{\{1, 4, 4, 7, 14, 14\}\}$ with error range $[-1, 1]$. Then the intervals we run our backtracking procedure on would be $V = \{\{[0, 2], [3, 5], [3, 5], [6, 8], [13, 15], [13, 15]\}\}$. While constructing the backtracking pyramid using the algorithm of Skiena and Sundaram [28] is efficient in this case (avoiding the first issue), it is not clear how to efficiently extract a potential solution.

backtracking pyramid at once. As the backtracking progresses and some of these counters reach 0, we are able to refine the intervals we have put into the backtracking pyramid prior and reduce their length. By doing this, we are able to ensure that our multiset volume-based reconstruction attack always succeeds, that it still takes on the order of 2^n time to produce a backtracking pyramid (as opposed to order $n!$ time), and that it does not need to run an additional backtracking procedure on our final backtracking pyramid in order to produce a solution.

We now describe our algorithm at a high level, beginning with its initialization:

1. Take each noisy volume v and replace it with an interval $[x, y]$, such that a x is the minimum possible value of v prior to adding noise and y is the maximum possible value of v prior to adding noise. We call the resulting multiset of intervals V and we let V' be the set (not multiset) containing exactly those elements in V.

2. We construct an interval pyramid I with m rows and k columns in row k where $m = |V'|$. We associate each element of I with a unique interval $[x, y]$. The interval $[x, y]$ associated with the element of I in row i and column j is such that x is the min element of the j^{th} smallest interval in V' and y is the max of the i^{th} largest interval of V'.

3. Each element of I has a counter that is set to the number of subsets of its associated interval in V (counting multiplicity).

The algorithm then proceeds by interval backtracking, with the key modification that, we add the maximum possible interval from any implication to our pyramid, whether or not it is in our set of volume intervals. Further, when a new interval $[x, y]$ is added to the backtracking pyramid B, it is also added to the bucket in I associated with an interval that is the minimum superset of $[x, y]$ in I.[1] Then the counter of every element of I associated with a superset of $[x, y]$ is reduced by 1. When a counter in I hits 0, we restrict that element of I as well as every other element of I that is above it in I and which share a diagonal with it (so all those up and to the right and up and to the left), then move all elements out of restricted buckets and into the first non-restricted bucket further down in I. When doing this, we are able to take the intersection of the elements being moved and the interval of its new bucket, providing us an opportunity to refine the intervals and improve our attack. When we get an improvement in this way, we can then also look back at other intervals in the backtracking pyramid and look for further opportunities for refinement before continuing.

The full algorithm begins with Algorithm 5, but is split across multiple subroutines. To reduce clutter in the pseudocode of all our algorithms, we omit the process of taking the intersection of each interval with $[\max[\mathsf{Vols}] + a, \max[\mathsf{Vols}] + b]$ (or if $\max[\mathsf{Vols}] + a < 0$) at each step (which we must do to avoid getting volumes that are clearly impossible). Before proceeding with a more detailed description of our algorithm, we must rigorously define how I will be represented, as well as how we will reference elements of both our backtracking pyramid B and interval pyramid I.

Each element of I is a 4-tuple $([x, y], c, r, \{\{\}\})$ with an associated interval, counter, indicator variable for "restriction," and bucket (i.e. multiset) respectively. We use $(i, j)_B$ and $(i, j)_I$ to indicate row i and column j of B and the 4-tuple there in I respectively. We use $[x, y]_{(i,j)_B}$ and $[x, y]_{(i,j)_I}$ to refer to the interval stored in $(i, j)_B$ and the first element of $(i, j)_I$ respectively. We use $(i, j)_c$ to indicate the second element (i.e. the value of the

1. Sometimes the "minimum" interval is ambiguous which we discuss in further detail in the summary of algorithm 7

counter) of $(i,j)_I$. We use $\text{Res}((i,j)_I)$ to indicate the value of the third element of $(i,j)_I$ and say that $(i,j)_I$ is **restricted** if and only if $\text{Res}((i,j)_I) = 1$.

We now summarize NoisyReconstruction and each of its subroutines.

- Algorithm 5 - NoisyReconstruction takes as input the noisy volumes and noise bounds. It calculates how many distinct identifiers (i.e. N) the database has, then runs Setup to generate the multiset intervals and I. It then chooses to branch left or right in placing the next largest volume intervals from Vols into B. Once it makes this choice, it calls the main interval backtracking subroutine IBT to assign the volume and manage any implications. If one of the subroutines ever outputs "Backtrack", it manages the backtrack and tries a new path. This continues until all paths have failed or B is filled correctly.

- Algorithm 6 - Setup is passed the the noisy volumes and noise bounds by NoisyReconstruction. It uses this to generate V, which replaces each element of Vols with an interval, the size of which is based on the noise bound. It then uses V to generate an interval pyramid I, each element of which keeps track of how many intervals of various lengths can exist in B. It then passes V and I back to NoisyReconstruction.

- Algorithm 7 - IBT is passed a volume interval to assign to B, as well as the location in B to assign it to. It is also given B and I. IBT runs in much the same ways as interval backtracking, except it also needs to identify which bucket in I to place the assigned and implied volume intervals. If there is some $[x,y]_{(i,j)_I}$ that is the unique smallest interval that is a superset of the assigned or implied volume interval in question, then it is placed $(i,j)_I$. Each time a volume interval is placed into a bucket, a counter update C is run. It is worth noting that there may be several intervals in the same row of I that are all supersets of the assigned or implied volume interval in question. In such cases, there is no unique smallest interval as relied upon above. Thus, when this is the case, we instead place our current assigned or implied volume interval in the element of I which is the unique minimum superset of all of the supersets on the same level I. For example, suppose the volume interval $[3, 4]$ is implied, and I has buckets for intervals $[2, 4]$ and $[3, 5]$ at its base. As these are on the base of I, there is no bucket for $[3, 4]$ below them, but there ss a bucket for $[2, 5]$ above them. In this case, $[3, 4]$ would be placed in the bucket of I associated with $[2, 5]$.

- Algorithms 8 and 9 - After a volume interval is placed into a bucket in I, we call C to update the counters of that bucket and all buckets that are supersets of it. If the volume interval is being moved between buckets in I as opposed to being added to I for the first time, we run C' instead of C. If some bucket's counter drops below 0, we know we need to backtrack and return to IBT to try a new path. If a buckets counter drops to 0, we mark it as one that should be restricted by adding it to Z. Once we are done updating counters, we run Res to actually apply restrictions to buckets and manage any elements that may be in them.

- Algorithm 10 - Res goes through the list of Z given to it by C or C' and, one at a time, restricts them. Res then restricts all superset of this bucket in I that share a diagonal with it (i.e. they are up and to the right or up and to the left of the initially restricted bucket). For each bucket that is restricted, the elements in it are "pushed" from them to a nearby bucket in I that is not restricted. If our initially restricted bucket is $(i, j)_I$ with associated volume interval $[x, y]_{(i,j)_I}$, then everything up and to the right of it are such that $[x, y']_{(i',j)_I}$ for various y' and i'. Thus, the restriction of $(i, j)_I$ tells us that the lower bound of x is no longer a possible value for the intervals in $(i', j)_I$ to take. Thus, we are able to move these elements out of $(i', j)_I$ down and to the right from it, restrict $(i', j)_I$, and potentially improve the moved elements (by taking the intersection of the interval being moved with the interval associated with its destination bucket). When elements are moved in this way, we then need to call C' to adjust the counter accordingly. We also need to call out interval optimization subroutine Optimize so as to make adjustments that might be implied by improvements made to intervals during the restriction process.

- Algorithm 11 - Optimize goes through the list of Z given to it by C or C' and, one at a time, restricts them. Res then restricts all superset of this bucket in I that share a diagonal with it (i.e. they are up and to the right or up and to the left of the initially restricted bucket). For each bucket that is restricted, the elements in it are "pushed" from them to a nearby bucket in I that is not restricted. If our initially restricted bucket is $(i,j)_I$ with associated volume interval $[x,y]_{(i,j)_I}$, then everything up and to the right of it are such that $[x,y']_{(i',j)_I}$ for various y' and i'. Thus, the restriction of $(i,j)_I$ tells us that the lower bound of x is no longer a possible value for the intervals in $(i',j)_I$ to take. Thus, we are able to move these elements out of $(i',j)_I$ down and to the right from it, restrict $(i',j)_I$, and potentially improve the moved elements (by taking the intersection of the interval being moved with the interval associated with its destination bucket). When elements are moved in this way, we then need to call C' to adjust the counter accordingly. We also need to call out interval optimization subroutine Optimize to make adjustments that might be implied by improvements made to intervals during the restriction process.

To assist in understanding the above description, we give an example execution across several figures, starting with figure 2.2. We use the same example as we used for interval backtracking (i.e. $\text{Vols} = \{\{1,4,4,7,14,14\}\}$ with error range $[-1,1]$). In contrast to the prior interval backtracking algorithm, we are able to achieve exact reconstruction for this instance without the need for an additional backtracking algorithm in post-processing. Our additional overhead in maintaining I is at most polynomial at each step.

While NoisyReconstruction outputs a backtracking pyramid B with intervals as elements, this doesn't immediately translate to it outputting some database DB' as required for a

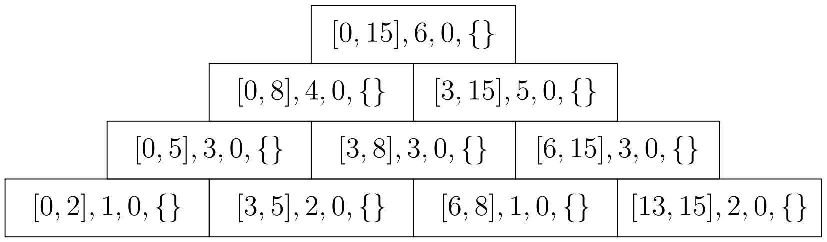

Figure 2.2: Example I after running Setup as part of NoisyReconstruction.

Initialization of I given $\mathsf{Vols} = \{\{1,4,4,7,14,14\}\}$. As the error range is $[-1,1]$, the setup also outputs $V = \{\{[0,2],[3,5],[3,5],[6,8],[13,15],[13,15]\}\}$. For improved legibility, parenthesis around the tuples in I are suppressed and multisets are written with only one pair of brackets.

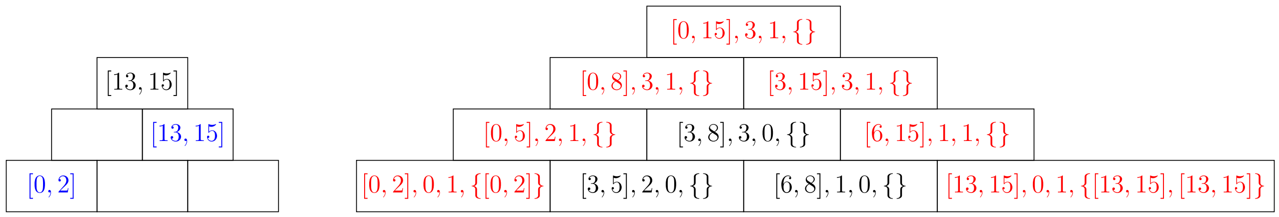

Figure 2.3: Example run of NoisyReconstruction, part 1.

Each element of B (shown left) is placed in a bucket in I (shown right) for which the associated interval is its minimum superset. This drops the counter of two elements of I to 0, restricting most elements of I (highlighted in red).

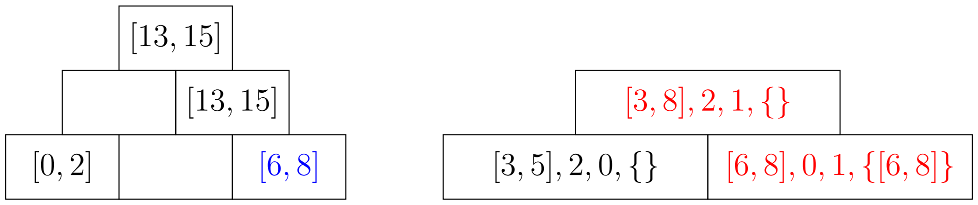

Figure 2.4: Example run of NoisyReconstruction, part 2.

We are left with $V = \{\{[3,5],[3,5],[6,8]\}\}$ and so guess that $[6,8]$ should be assigned to the right. We redact the elements of I restricted in the previous step for the sake of visual clarity.

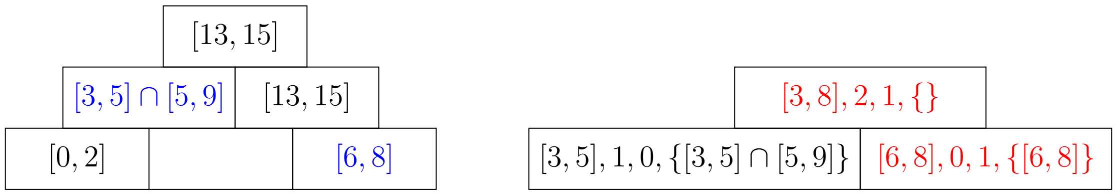

Figure 2.5: Example run of NoisyReconstruction, part 3.

The placement of $[6,8]$ leaves all but one bucket in I restricted. The implied volume interval on the far left is $[5,9]$. The minimum superset of $[5,9]$ in I is $[3,15]$ which is restricted (and no longer shown). This pushes it down to $[3,8]$ which is also restricted, so it ends in the bucket for $[3,5]$ in I. Thus, the volume must be in $[3,5]\cap[5,9]$.

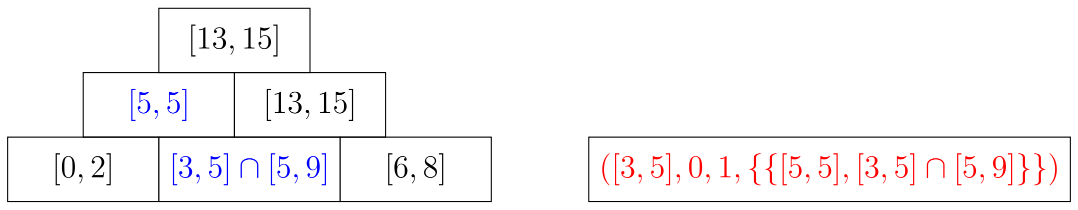

Figure 2.6: Example run of NoisyReconstruction, part 4.

The previous implication was improved by using I resulting in a non-trivial intersection. This triggers an update. Before the update however, we finish calculating the remaining implications from the assignment of $[6, 8]$.

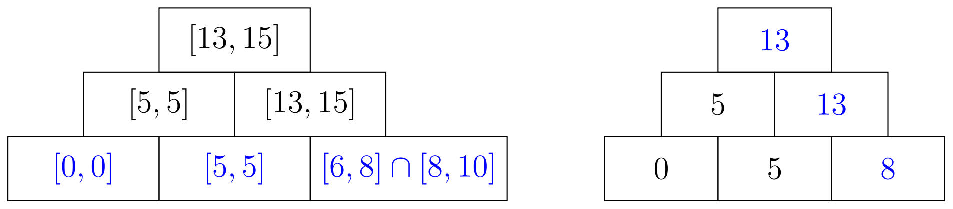

Figure 2.7: Example run of NoisyReconstruction, part 5.

We can now proceed with the update. As this leads to the improvement of the bottom left and bottom right elements (shown left), another update is required which manages to find the exact value of all volumes (shown right). Thus, for $\mathsf{Vols} = \{\{1, 4, 4, 7, 14, 14\}\}$ and error range $[-1, 1]$, the only possible database is $DB = (0, 5, 8)$. As I is not meaningfully used in these updates, we redact I for the sake of visual clarity.

noisy multiset reconstruction attack to succeed. Thus, we define an additional procedure in DBguess (i.e. algorithm 12) that takes B as input and outputs an actual DB' and allows us to reason formally about the success of NoisyReconstruction. At a high-level, DBguess takes an interval in B, assigns it an arbitrary value within that interval, then runs Algorithm 11 update B based on the assignment. We now prove the correctness of NoisyReconstruction(Vols, $[a, b]$).

Lemma 2. *Let Vols be noisy multiset leakage from* DB *with the additive noise applied to each volume being in $[a, b]$ such that, for every range query, the probability that any given element in $[a, b]$ is the amount of noise applied to that range queries volume leakage is non-zero (i.e. all amount of noise in the range are possible for any particular range). DBguess(NoisyReconstruction(Vols, $[a, b]$)) is a noisy multiset reconstruction attack that always succeeds.*

Proof. We walk through a single path of NoisyReconstruction that leads to a complete backtracking pyramid B and prove that any outcome of running DBguess on the output of NoisyReconstruction results in a DB' that is information-theoretically consistent with the leakage. To do this, we focus on proving that, for every element in B, replacing any element $[x, y]$ in B with an interval $[z, z]$ for $z \in [x, y]$ and running Optimize does not lead to any information theoretically impossible state of B. There are essentially three checks done throughout the algorithm that influence the intervals in B. The first is a backtracking-based calculation for implications, the correctness of which follows from the correctness of interval backtracking [28]. The second is that, as mentioned prior, we always make sure that intervals obey some global upper and lower bound on their ranges, which are 0 and $\min[\text{Vols}] + a$ in the lower bound and $\max[\text{Vols}] + b$ in the upper bound, the correctness of which is straightforward. The third influence on intervals comes from I which enforces the rule that no interval can appear in B more than it appears in V. The fact that it achieves this however is non-trivial and thus, we pay special attention to steps involving I when we reach the discussion of them in this proof.

During setup, the elements of Vols are converted to intervals that capture all possible elements they could correspond to. What we mean by this is, if we only knew that $x \in \mathsf{Vols}$ and the error range is $[a, b]$, then x could have been anywhere in that range $[a + x, b + x]$ prior to the inclusion of error. That being said, we know that no volume is negative, so we must take the min of each element with 0. Thus, without considering any interplay between the volumes, each volume interval is as tight as possible.

Proceeding to backtracking, we begin by establishing that the "largest" interval to be placed next is unambiguous. All intervals in V are of equal length (exactly length $1 + b - a$ to be precise) and thus, we can simply take the element of V with the largest maximum element and place it left or right in the backtracking pyramid.[2] As to implication, the location of implications is correct for the same reason as in standard backtracking, as are the initial intervals computed before utilizing I.

We must now justify how I operates. After setup, each element of I has a counter correctly tracking how many intervals within any range could possibly exist in B. When an interval $[x, y]$ is placed into a bucket in I, the placement follows three major criteria.

1. The bucket it is placed into cannot be restricted.
2. The bucket it is placed into is such that there is no other bucket on the same level (i.e. at the same height in the interval pyramid I) that is a superset of $[x, y]$.
3. The bucket it is placed into is such that its associated interval is the minimum superset of $[x, y]$ (i.e. $[x, y] \subset [x', y']_{(i',j')_I}$) among all elements of I for which criteria 1 and 2 hold.

We now consider each of these criterion in order and justify them. As to the first, a bucket can be restricted for one of two reasons: its counter is 0 or the counter of a bucket

2. Under a different noise model where this may not hold we can break ties in the maximum element of an interval by taking the interval with the largest minimum element.

that shares a diagonal with it and is below it in I (i.e. closer to the base of the interval pyramid I) has a counter set to 0. Let $(i,j)_C = 0$ for some i and j. As we only decrement a counter when a subset of the associated interval (i.e. a subset of $[x,y]_{(i,j)_I}$) is added to B, if $(i,j)_C = 0$ we know there can be no more intervals in this range. Thus, the attempt to add an additional interval to this bucket signals a contradiction and causes a backtrack as desired. Alternatively, let us assume that $(i,j)_C > 0$, but that there is an element of I below it on the same diagonal with a counter set to 0. Without loss of generality, let us assume that this element appears below and to the left of $(i,j)_I$ (i.e. There exists an $i' > i$ such that $(i',j)_C = 0$). By construction, we have that $[x,y]_{(i,j)_I}$ and that $[x,y']_{(i',j)_I}$. To put this more plainly, the diagonal they share is one that contains all of the intervals with a minimum of x. This means we can refine the interval we are placing by increasing its lower bound to some x' greater than x, where x' is equal to the minimum element of all of the intervals on the first unrestricted diagonal that is down and to the right of and perpendicular to the diagonal shared by $(i,j)_I$ and $(i,j)_I$. We refer the read to the full interval backtracking pyramid displayed in figure 2.2 along with the interval volume assignment in figure 2.4 to see an example of exactly this type of occurrence.

As to the second criterion, we reuse an example from earlier to exemplify its application. Suppose the volume interval $[3,4]$ is implied and I has buckets for intervals $[2,4]$ and $[3,5]$ at its base. As these are on the base of I, by construction, there is no bucket for $[3,4]$ below them, but there is a bucket for $[2,5]$ above them. In this case, $[3,4]$ would be placed in the bucket of I associated with $[2,5]$. The reason for this is that $[3,4] \cap [2,4]$ and $[3,4] \cap [3,5]$ are both non-empty. If we attempted to place $[3,4]$ into either bucket prior to the restriction of the bucket associated with the interval $[2,5]$, we would be making an assumption that may be premature. This is best displayed with an example and thus, without loss of generality, assume we placed $[3,4]$ in the bucket associated with $[2,4]$. But suppose the bucket for $[2,4]$ should actually be filled entirely with intervals that are eventually refined down to the value of 2. Then placing $[3, 4]$ in this bucket as well would cause the counter to fall below 0 and backtrack, even though the counter for the bucket associated with $[3, 5]$ may not yet be 0. Thus, this backtrack is incorrect. On the other hand, if we leave the interval $[3, 4]$ in the bucket for $[2, 5]$, then if the same scenarios occurs again and the bucket associated with $[2, 4]$ is filled with 2, this causes a restriction to the bucket for $[2, 5]$, pushing the intervals $[3, 4]$ out of that bucket and into the bucket associated with the interval $[3, 5]$ as required.

The third and final criterion is fairly straightforward compared to the other 2. As the goal of the counters in I is to track how many more of each interval can be added to B, if $[x, y]$ is being added to B by assignment or implication, we wish to decrement the counter for each interval in I that is a superset of it (excepting those rules of by criteria 1 and 2 for the reasons expressed prior). Thus, by the rules of C and C' which manage the counters in I, the interval $[x, y]$ should be placed in the bucket associated with the interval that is the minimum superset of $[x, y]$ among all elements of I for which criteria 1 and 2 hold.

While the above considerations with respect to I are primarily focused on the subroutine IBT, the same reasoning also justifies the actions of the subroutine Res, which manages the restriction of elements of I and the shifting of those elements to new buckets as appropriate. The correctness of Optimize follows from the correctness of interval backtracking as, if we update any value in B using I, then it follows that we must recalculate all of the standard backtracking implications to make sure we have not missed any cascading improvement that would affect global consistency. Lastly, DBguess simply guesses values in the range of base elements of B and updates the rest of B accordingly using Optimize. We know that any solution found in this way is correct by the correctness of the intervals calculated using some global bounds on intervals, standard interval backtracking arithmetic, and refinements utilizing I. Further, we cannot have missed any solution as, if we replace any element of B with some element not in the associated interval, then this would change which counter is being decremented in I and cause at least one counter to drop below 0. This would imply that some value appears more often in B than we know is possible given V and thus, the assumption that this is an information-theoretically valid solution is false. $\square$

Note that, one enumerates all paths of NoisyReconstruction that result in a complete backtracking pyramid, then enumerates all paths of DBguess on each of these backtracking pyramids, then this algorithm becomes a full noisy multiset reconstruction attack that always succeeds. The proof of this is essentially that of Lemma 2.

2.4 Volume-based Reconstruction in Multiple Dimensions

In this section, we give a backtracking algorithm for database reconstruction from multiset volume leakage in 2-dimensions. While we focus on two 2-dimensional databases, the algorithm naturally generalizes to d-dimensional databases for arbitrarily large d. We discuss how to do this at a high level at the end of this section. We now formally define the reconstruction problem in question for higher dimensions (both in terms of multiset and set leakage).

Definition 10. *For a d-dimensional-database* $\mathsf{DB} = (v_{(1,\ldots,1))}, \ldots, v_{(M_1,\ldots,M_d)})$ *with m records and* i^{th} *dimensional domain size* M_j *d-**dimensional multiset volume-based reconstruction attack** is one which takes as input the tuple of domain sizes for each dimensions* $(M_1, \ldots, M_d)$ *and*

$$\mathsf{Vols} = \left\{\left\{ \sum_{v_{\boldsymbol{x}} \in R_{(\boldsymbol{y},\boldsymbol{z})}} v_{\boldsymbol{x}} : (1, \ldots, 1) \leq \boldsymbol{y} \leq \boldsymbol{z} \leq M_1 \times \ldots \times M_d \right\}\right\}$$

where $R_{(\boldsymbol{y},\boldsymbol{z})} = \{v_{\boldsymbol{x}} : \boldsymbol{y} \leq \boldsymbol{x} \leq \boldsymbol{z}\}$ *and outputs a database* $\mathsf{DB}' = (v'_{(1,\ldots,1))}, \ldots, v'_{(M_1,\ldots,M_d)})$ *with* i^{th} *dimensional domain size* M_i *and m records. We say that a d-dimensional set volume-*

based reconstruction attack ***succeeds*** *if and only if*

$$\mathsf{Vols} = \left\{\left\{ \sum_{v_{\boldsymbol{x}} \in R_{(\boldsymbol{y},\boldsymbol{z})}} v_{\boldsymbol{x}} : (1,\ldots,1) \leq \boldsymbol{y} \leq \boldsymbol{z} \leq N_1 \times \ldots \times N_d \right\}\right\}$$

where $R'_{(\boldsymbol{y},\boldsymbol{z})} = \{v'_{\boldsymbol{x}} : \boldsymbol{y} \leq \boldsymbol{x} \leq \boldsymbol{z}\}$

Definition 11. *For a d-dimensional-database* $\mathsf{DB} = (v_{(1,\ldots,1))}, \ldots, v_{(M_1,\ldots,M_d)})$ *with* m *records and* i^{th} *dimensional domain size* M_j ***d-dimensional set volume-based reconstruction attack*** *is one which takes as input the tuple of domain sizes for each dimensions* $(N_1, \ldots, N_d)$ *and*

$$\mathsf{Vols}_{set} = \left\{ \sum_{v_{\boldsymbol{x}} \in R_{(\boldsymbol{y},\boldsymbol{z})}} v_{\boldsymbol{x}} : (1,\ldots,1) \leq \boldsymbol{y} \leq \boldsymbol{z} \leq N_1 \times \ldots \times N_d \right\}$$

where $R_{(\boldsymbol{y},\boldsymbol{z})} = \{v_{\boldsymbol{x}} : \boldsymbol{y} \leq \boldsymbol{x} \leq \boldsymbol{z}\}$ *and outputs a database* $\mathsf{DB}' = (v'_{(1,\ldots,1))}, \ldots, v'_{(N_1,\ldots,N_d)})$ *with* i^{th} *dimensional domain size* M_i *and* m *records. We say that a d-dimensional set volume-based reconstruction attack* ***succeeds*** *if and only if*

$$\mathsf{Vols}_{set} = \left\{ \sum_{v_{\boldsymbol{x}} \in R_{(\boldsymbol{y},\boldsymbol{z})}} v_{\boldsymbol{x}} : (1,\ldots,1) \leq \boldsymbol{y} \leq \boldsymbol{z} \leq N_1 \times \ldots \times N_d \right\}$$

where $R'_{(\boldsymbol{y},\boldsymbol{z})} = \{v'_{\boldsymbol{x}} : \boldsymbol{y} \leq \boldsymbol{x} \leq \boldsymbol{z}\}$

As the volume leakage one has access to in such attacks is strictly weaker than access pattern leakage, we know that the number of possible reconstructions that are information theoretically consistent with the leakage is at least as large when given volume leakage as access pattern leakage. Thus, unlike the in the 1-dimensional case, we can use the result of [9] Falzone et al. to conclude that the worst case number of valid reconstructions is exponential (as there this lower bound is proven, but with access pattern leakage).

2.4.1 Attack Overview

We now give a high level description of our 2-dimensional multiset volume-based reconstruction attack, which succeeds on any valid input DB. The first important thing to note is that, similar to the backtracking algorithms for volume-based database reconstruction [16] and the turnpike problem [27], one can use small number of guesses to infer a polynomial number of additional assignments of volumes to range queries. In 1-dimension, this allows one to solve the problem after n volume assignment guesses, despite the input including volumes from $\binom{n}{2}$ distinct range queries. At a high level, the principles that dominate the behavior for the 1-dimensional backtracking algorithm are as follows:

1. At each step, the largest volume is assigned to one of the two ranges that starts from either end point and for which all of its supersets have already been assigned.

2. One takes the difference between the volumes of the newly assigned range with each range that shares and end point with it (each of which is the volume of some range that would not yet have a volume associated with it).

These ideas can be naturally generalized to d-dimensions as follows:

1. At each step, the largest volume is assigned to some range that starts from either end point and for which all of its supersets have already been assigned.

2. One takes the difference between the volume of the newly assigned range with each range that shares all but one boundary with it (each of which is the volume of some range that would not yet have a volume associated with it).

In both cases, if any volume is assigned to more ranges then the number of time it is observed in the leakage, then one backtracks and try a new path. Both items present greater difficulty in higher dimensions. The first of these two generalizations is straightforward, though it vastly complicates tracking what options are available at each step. The basic idea is that, as you are assigning the largest volume to some range, if some superset of it does not yet have a volume associated with it, you know that range has at least as large a volume as the range you just assigned a volume. If the database is dense (has no range queries with no associated records), then the range that is a superset of the one assigned will be strictly larger and this will always result in backtracking. If they have equal volume, it can at best give the same result as having assigned the largest volume to the range that is a superset of the one just assigned a volume. An example of how to do this with nested backtracking pyramids can be found in Algorithm 13.

As to the second generalization, we prove that this is reasonable for an algorithm such as Algorithm 13 with Lemma 3.

Lemma 3. *Let* $q = ([a_1, b_1], \cdots, [a_n, b_n])$ *be the most recent range query assigned a volume. Let* q' *be a superset of* q *that shares all but one boundary with it, and assume without loss of generality that* $q' = ([a_1, b_1], \cdots, [a_n, b'_n])$ *for* $b'_n > b_n$*. Then the volume of* $q'' = ([a_1, b_1], \cdots, [b_n, b'_n])$ *equals the volume of* q' *minus that of* q *and the volume of* q'' *cannot have been implied in this way in a previous step of the backtracking algorithm generalized to d-dimensions.*

Proof. The fact that the volume of q'' equals the volume of q' minus that of q follows immediately. Suppose q'' already had an associated volume from a prior step of the algorithm. If q'' had a volume prior prior, then we would have been able to take the differences of the volumes of q' and q'' to give a volume to q. As q is assumed to have just been assigned a volume, this is a contradiction. □

Computing implied values in this way and, with backtracking, trying every potential guess at each step also ensures correctness of the algorithm, as no implied query would be computed in two different ways.

CHAPTER 3
TILINGS OF CONTIGUOUS FINITE SUBSETS OF $\mathbb{Z}$ WITH TILES OF FIXED SIZE

3.1 Introduction

In this chapter, we study the number of finite tiles $A \subset \mathbb{Z}$ of size α that can translationally tile any C that is a finite contiguous subset of $\mathbb{Z}$, results that previously appeared in [31]. Under these restrictions the tile A can be translated any number of times to cover exactly C, but cannot be rotated or reflected. Further we consider two tiles A and A' to be congruent (i.e. not distinct) if and only if one can be transformed into the other via some translation. The study of this problem is motivated by the fact that we can reduce the turnpike problem to the problem of finding a tiling for a specific multiset by a tiling of size square root the size of the the multiset, a result we also prove in this chapter.

For any $\alpha \in \mathbb{Z}^+$ and $C = [x_1] \times [x_2] \times \ldots [x_d]$ where $x_1, \ldots, x_d \in \mathbb{Z}^+$ (which we refer to as a finite contiguous C), we classify exactly which A of size α can tile C. More specifically, we give an efficient[1] method for enumerating all elements of $\mathcal{T}(\alpha, C)$, where $(A, B) \in \mathcal{T}(\alpha, C)$ if and only if

1. $A, B \subset \mathbb{Z}$
2. $A + B = C$
3. $|A| = \alpha$
4. $|C| = \alpha|B|$,

where we use $A + B$ to mean the Minkowsji sum of A and B. Further, we assume (A, B) is some canonical representative (to be formally defined later) of the class of all (A', B') such that A is congruent to A'. This classification of the elements of $\mathcal{T}(\alpha, C)$ also allows us to prove a partial order on $|\mathcal{T}(\alpha, C)|$ with respect to α for any finite contiguous C.

1. By efficient, we mean polynomial time with respect to $|\mathcal{T}(\alpha, C)|$.

We then study the extremal question as to the the growth rate of $\max_{\alpha,C}[|\mathcal{T}(\alpha, C)|]$ with respect to $|C|$. The trivial bounds for this value are very poor, with a lower bound of roughly $\log n$ and an upper bound of $\binom{n}{n/2}$. We improve these bounds for finite contiguous C to the upper bound of

$$n^{\frac{(1+\epsilon)\log n}{\log\log n}}$$

and an infinitely often super linear lower bound. More specifically, the lower bound states that there exists some infinite $N \subset \mathbb{Z}^+$,

$$\forall n \in N \exists \alpha \in \mathbb{Z}^+(|\mathcal{T}(\alpha, C)| > \omega(n)$$

where $n = |C|$ and C is both finite and contiguous.

Translational tilings of $\mathbb{Z}$ [7, 20, 33], and of $\mathbb{Z}^d$ [1, 11, 17, 32] more generally, are natural problems that have been studied in many prior works. While the vast majority of work in this area seeks to tile the infinite set $\mathbb{Z}^d$, Pederson and Wang [24] initiated the study of tiling finite intervals of $\mathbb{Z}$. This was later followed by the independent work of Bodini and Rivals [3], with the combined works outlining necessary and sufficient conditions for such tilings. Additionally, Bodini and Rival [3], as well as Rivals [25], initiated the study of counting the number of such tilings. We expand upon the prior works in the following ways:

1. Introducing new characterizations of tilings of $[n]$ that allows for the specification of a fixed tile size.

2. Proving a partial order on $|\mathcal{T}(\alpha, C)|$ with respect to α for any finite contiguous C.

3. Proving strong upper and lower bounds as to the number of such tilings.

We now discuss how each of these contributions and the prior works that relate to them.

3.1.1 Characterizations of Tilings

With respect to the tilings of finite intervals of $\mathbb{Z}$, a number of our results align with results of Bodini and Rivals [3] and Pederson and Wang [24], but allow for specification of the additional parameter α (i.e. a specific tile size). For us to specify properties of tilings of finite intervals of $\mathbb{Z}$ by tiles of size α, it must be the case that any properties we highlight must also be true of tilings of finite intervals of $\mathbb{Z}$ without a specified tile size. Thus, one can recover many of the structural results of these prior works from our own by considering them for all α. Unfortunately, there does not appear to be a straightforward way to derive our results from those prior without writing new proofs that take tile size into consideration at each step. Despite this, there are several properties of tilings utilized in our proofs that were proved in these prior works and which we summarize in Lemma 5. For completeness, we provide a proof Lemma 5 in the terminology of this paper.

3.1.2 Counts on the Number of Distinct Tilings

As to the counting of tilings of finite subsets of $\mathbb{Z}^d$, Bodini and Rivals [3] and Rivals [25] began working towards this by counting the tilings of finite intervals of the discrete line (i.e. the case of $d = 1$). While the recent work of Benjamini, Kozma, and Tzalik counts the number of tiles from some finite contiguous subset of $\mathbb{Z}^d$, the tiles counted are those that tile the infinite set $\mathbb{Z}^d$ as a whole, which is a fundamentally different question then tiling a finite subset of $\mathbb{Z}^d$. Given this, we focus on the results of Bodini and Rivals [3] and Rivals [25], who begin by proving that the number of tilings of a finite interval of $\mathbb{Z}$ is equal to the elements of the integer sequences A067824 and A107067 as indexed by The On-Line Encyclopedia of Integer Sequences [29], though we note that the equivalence of sequences A067824 and A107067 was independently established by Karhumaki, Lifshits, and Rytter [15]. Unfortunately, neither sequence has clear upper or lower bounds. Further, our goals differ somewhat from those of Rivals as we wish to study fixed tile sizes and extend beyond finite intervals to some finite non-contiguous C. Thus, even if one did derive a satisfactory upper or lower bound from these integer sequences, this would not provide an upper of lower bound on the number of tilings for any particular fixed tiles size. This means that we require new formulas for counting tilings, even in the case of finite intervals of $\mathbb{Z}$.

3.1.3 Upper and Lower Bounds on the Number of Distinct Tilings

While we are unaware of any previous literature counting the number of tilings of $[n]$ with tiles of a fixed tile size α, one can derive some trivial upper and lower bounds on the number of such tilings. For an upper bound on $\mathcal{T}(\alpha, [n])$ and without relying on prior work, one can simply note that there are at most $\binom{n-1}{\alpha-1}$ ways to make a tile of size α using elements in $[n]$ and such that the tile always includes 1. While many of these tiles would fail to tile $[n]$, by setting α to $n/2$ we get a very crude upper bound of $\binom{n}{n/2}$. Another more in-depth approach would be to use our arguments from Lemma 12 and Corollary 7 along with Theorem 5 of Bodini and Rivals [3], but this would result in the upper bound we obtain raised to the $\log n$.[2] To shave of this additional $\log n$ multiplicative factor from the exponent, we are thus motivated to give Definition 19 and prove Lemma 6.

As to lower bounds for $\mathcal{T}(\alpha, [n])$, we note that a trivial lower bound of $\log n$ can be obtained without prior work by considering tilings of $[2^k]$ for tiles of size $2^{k/2}$. To achieve an improved lower bound, we require a new formula for enumerating the elements of $\mathcal{T}(\alpha, [n])$. The way we chose to enumerate the elements of $\mathcal{T}(\alpha, [n])$ in Lemma 6 does not appear conducive to good lower bound arguments. Thus, in Lemma 10, we establish a second formula for enumerating the elements of $\mathcal{T}(\alpha, [n])$ utilizing the inclusion-exclusion principle. From this, we begin by proving a nearly linear lower bound on $|\mathcal{T}(\alpha, [n])|$. We are then able to apply a more fine-grained analysis to achieve a super-linear lower bound for infinitely

2. This is due to not knowing how the distinct tilings are distributed with respect to tile size, necessitating a multiplication by the number of distinct tile sizes (which itself equals the number of divisors of the size of the set to be tiled).

many n and certain α chosen based upon n. We note that such a lower bound cannot be achieved for all n nor for all α, as if n is prime or $\alpha = 1$ there is at most a single tiling of $[n]$.

3.2 Definitions

For $n \subset \mathbb{Z}^+$ and $x, y \subset \mathbb{Z}$, we let $[n] \triangleq \{1, \ldots, n\}$ and $[x, y] \triangleq \{x, x+1, x+2, \ldots, y-1, y\}$ where $x \leq y$. We take p_i to be the i^{th} prime and we use $x|y$ to mean x divides y. Let the divisor function $\sigma_0(n)$ for $n \in \mathbb{Z}$ equal the number of positive divisors of n. We write $\omega^*(n)$ and $\Omega^*(n)$ to the prime omega functions, where $\omega^*(n)$ equals the number of distinct prime factors of n (ignoring multiplicity of these prime factors) and $\Omega^*(n)$ equals the total number of prime factors of n (i.e. the sum of the exponents across all prime factors of n).[3] Throughout this work we use log to mean $\log_2$. All sets discussed in this work are assumed to be finite. For sets A and B, let $A + B = \{a + b : a \in A, b \in B\}$ be the Minkowski sum of A and B. For a finite set $A \subset \mathbb{Z}$, we use $\min[A]$ and $\max[A]$ to indicate the minimum and maximum element of A. For a function f, we write $\min_x[f(x)]$ and $\max_x[f(x)]$ to indicate the value of $f(x)$ for any choice of x the minimizes or maximizes $f(x)$ respectively.[4] When we take the max or min of a set of sets, we take the max or min respectively from the union over all elements of the set (e.g. $\max\big[\{\{1,2\}\{5,6\}\}\big] = 6$). Similarly, if we use a set of sets S in a set difference operation, we treat S as the union of its elements. Let $\text{proj}_i(x)$ be the projection of $x \in \mathbb{Z}^d$ onto the i^{th} dimension (i.e. $\text{proj}_i(x_1, \ldots, x_i, \ldots, x_d) = x_i$). Further, we let $\text{proj}_i(A)$ for a set A equal the set $\{\text{proj}_i(x) | x \in A\}$.

Definition 12. *We call a set C **contiguous** if, for some $d \in \mathbb{Z}^+$, we have $C = [x_1] \times [x_2] \times \ldots [x_d]$ where $x_1, \ldots, x_d \in \mathbb{Z}^+$.*[5]

3. The prime omega functions are usually denoted by $\omega(n)$ and $\Omega(n)$ for ignoring and counting multiplicity of the prime factors respectively, however we use $\omega^*(n)$ and $\Omega^*(n)$ respectively so as to avoid confusion with asymptotic notation.

4. If such an x does not exist we treat this as undefined though this circumstance does not occur in this work.

5. One could alternatively define contiguous C to be C such that $C = [x_1, y_1] \times [x_2, y_2] \times \ldots [x_d, y_d]$ for

When $d = 1$, a contiguous C is simply a finite interval of the discrete line, though we fix its minimum point to be at 1 for convenience.

Definition 13. *For finite sets $A, B, C \subset \mathbb{Z}^d$ such that $|A| = \alpha$ and $|B| = \beta$, the pair (A, B) is a **valid translational tiling** of C if and only if $A + B = C$ and $|C| = \alpha\beta$. We refer to A as the **tile** and B as the **translations**.*

We will typically drop the term "translational" from the above definition and simply refer to (A, B) as a **valid tiling** of C. Definition 13 allows for infinitely many tilings of all finite sets as, to tile C, one can set $A = C + m$ and $B = C - m$ for all $m \in \mathbb{Z}^d$. So that we may count the number of tilings up to such translations, we give the following definition.

Definition 14. *The **congruence class** of a tiling (A, B) is the set of all tilings (A', B') such that $A + m = A'$ and $B - m = B'$ for some $m \in \mathbb{Z}^d$ (we would also then refer to A and A' as themselves congruent). Let the **canonical representative** of each congruence class of tilings be (A, B) such that $(0, \ldots, 0) \in B$ and $\forall i(min[proj_i(B)] \geq 0)$.*

For the remainder of the paper, we presume all tilings are the canonical representative of the congruence class of tilings to which they belong. With this, we can now define the set of fixed tile size tilings of a finite set C.

Definition 15. *For finite sets $A, B, C \subset \mathbb{Z}^d$ such that $|A| = \alpha$ and $|B| = \beta$, we define $\mathcal{T}(\alpha, C)$ to be the set of all (A, B) that are the canonical representative of their congruence class of tilings, are valid tilings of C, and are such that $|A| = \alpha$. Further, we define $\mathcal{T}((\alpha_1, \alpha_2, \ldots, \alpha_d), C)$ to be the set of all (A, B) that are the canonical representative of their congruence class of tilings, are valid tilings of C, and are such that $\forall i \in [d](|proj_i(A)| = \alpha_i)$.*

The remaining definitions in this section are purely with respect to finite intervals of $\mathbb{Z}$ (i.e C such that $d = 1$). So that we may refer to a specific member of $\mathcal{T}(\alpha, C)$ when necessary, we require the following.

$x_1, \ldots, x_d, y_1, \ldots, y_d \in \mathbb{Z}$ and $\forall i(x_i \leq y_i)$, however this complicates some statements and proofs without increasing the generality of the results.

Definition 16. *Let C be a finite subset of $\mathbb{Z}$. For $T = (A, B)$ and $T' = (A', B')$ such that $T, T' \in \mathcal{T}(\alpha, C)$, we say that $T < T'$ if and only if $min[A \setminus A'] < min[A' \setminus A]$. Otherwise, $T = T'$. We use $T_i = (A_i, B_i)$ to represent the i^{th} valid tiling of C according to this total order.*

To see that Definition 16 is valid, we require that it indeed defines a total order on $\mathcal{T}(\alpha, C)$ when $d = 1$. To see this notice that, if $i \neq j$, then $A_i \neq A_j$, as there is a unique B for tiling C with translates of any fixed A. For the rest of the properties of a total order, we can see A_i as corresponding to an integer that is the sum of 2^k for all $k \in A_i$. Thus, this being a total order follows from any subset of the integers being totally ordered by their value. We define $a_{(i,j)}$ to denote the i^{th} smallest element of A_j. We will usually drop the subscript j from this and other notation when it is clear from context or irrelevant (e.g. writing a_i as opposed to $a_{(i,j)}$). We define $b_{(i,j)}$ similarly to $a_{(i,j)}$, but with reference to B instead of A.

Definition 17. *Let $(A_j, B_j) \in \mathcal{T}(\alpha, [n])$ be the j^{th} valid tiling of $[n]$. We define the **first segment** and **first rift** of the j^{th} tiling (i.e. $s_{(1,j)}$ and $r_{(1,j)}$ respectively) to be:*

- $s_{(1,j)} \triangleq \{x \in A : x < \min[C \setminus A]\}$
- $r_{(1,j)} \triangleq \{x \in C \setminus A : x < \min[A \setminus s_1]\}$.

*As we frequently require the size of the first segment and first rift of a tiling, we let $k_{(s,j)} \triangleq |s_{(1,j)}|$ and $k_{(r,j)} \triangleq |r_{(1,j)}|$. For $i > 1$, we define the i^{th} **segment** and i^{th} **rift** of the j^{th} **tiling** (i.e. $s_{(i,j)}$ and $r_{(i,j)}$ respectively) recursively as follows:*

- $s_{(i,j)} \triangleq \left\{x \in A : \max[s_{i-1}] < x < \min[(C \setminus A) \setminus \{y \in C : y \leq \max[r_{i-1}]\}]\right\}$
- $r_{(i,j)} \triangleq \left\{x \in C \setminus A : \max[r_{i-1}] < x < \min\left[A \setminus \left(\bigcup_{k=1}^{i-1} s_k\right)\right]\right\}$.

To put the above more intuitively, the i^{th} segment of a valid tiling T_j is the i^{th} set of consecutive (relative to C) elements of A_j where as the the i^{th} rift of a tiling T_j is the elements of C between s_i and s_{i+1}. We define S_j and R_j to be the set of all non-empty $s_{(i,j)}$ and $r_{(i,j)}$ respectively. For example, let $A = \{1,2,5,6,9,10\}$, $B = \{0,2\}$ and $C = [12]$. Then $s_1 = \{1,2\}$, $s_2 = \{5,6\}$, and $s_3 = \{9,10\}$ while $r_1 = \{3,4\}$ and $r_2 = \{7,8\}$. The aforementioned segments and rifts would then be exactly the elements of S and R respectively, as all other segments and rifts are empty in this example. We define $\mathcal{T}(\alpha, C, (k_s, \cdot))$ to be

$$\mathcal{T}(\alpha, C, (k_s, \cdot)) \triangleq \{(A_i, B_i) \in \mathcal{T}(\alpha, C) : |s_{(1,i)}| = k_s\}$$

and define $\mathcal{T}(\alpha, C, (k_s, \cdot))$ to be

$$\mathcal{T}(\alpha, C, (k_s, k_r)) \triangleq \{(A_i, B_i) \in \mathcal{T}(\alpha, C) : (|s_{(1,i)}| = k_s) \wedge (|r_{(1,i)}| = k_r)\}.$$

3.3 Turnpike Reduces to Tiling Multisets

While our results in this chapter focus primarily on translational tilings of sets, we do this because similar results about multisets are currently out of reach. Thus, we focus on establishing a mathematical foundation of results related to sets that we hope to generalize to multisets in future work. To put the overall program in context and motivate these results, we briefly discuss tiling multisets and reduce the turnpike problem to this problem.

Definition 18. *For multisets $A, B, C \subset \mathbb{Z}$, let $A + B$ equal the multiset $\{a + b | a \in A, b_B\}$ sucht that distinct pairs of elements in A and B summing tot he same value increases that sums multiplicity in $A + B$. We say that A tiles a multiset C if and only if there exists a B such that $A + B = C$.*

For the definition of the turnpike problem, we refer the reader to Definition 3.

Lemma 4. *There exists a polynomial time reduction from the turnpike problem to the problem of, given a multiset $C \subset \mathbb{Z}$, finding a tile $A \subset \mathbb{Z}$ such that A tiles $C/$*

Proof. Given the distance multiset D of $\binom{n}{2}$ integers (counting multiplicity), and we build a new multiset D' by including in it the elements of D, the elements of $-1 \cdot D$, and the element 0 with multiplicity n. We can then ask if there exists a tile of size n that tiles D'. Due to our use of B in this chapter for the set of translations of a tile, we use P in this proof for th backtracking pyramids described in Chapter 2. Let P be the backtracking pyramid containing the elements of D. From P, we can build a new construct P' where we take the n copies of 0 from D' and add them to P as a part of a new bottom row, then reflect the elements of D across this new row of 0, but make the reflected copies of the elements of D below the row of all 0 negative. Notice that any A of size n that tiles D', is exactly a diagonal of such a P', and that each is just an additive shift (or translation) of each other diagonal in P'. As the tilings of D' are in one-to-one correspondence with valid backtracking pyramid P (or equivalently, with sets of n points on a line whose set $\binom{n}{2}$ interpoint distances equal D), the reduction is complete. □

3.4 Formulas for Enumerating $\mathcal{T}(\alpha, [n])$

In this section, we define two formulas for counting the exact number of distinct tilings for any α and $C = [n]$. By summing over all tile sizes, To prove the first of these formulas is correct, we do two things:

- Define necessary and sufficient conditions as to the size of segments and rifts in valid tilings.
- Group points into *meta-points* such that each element of the meta-point is in the same segment or rift as each other element of the meta-point.

Taken together, we are able to calculate $|\mathcal{T}(\alpha, [n])|$ by taking the sum of a small number of $|\mathcal{T}(\alpha', [n'])|$ for $n' < n$ where α' and n' are straightforward to calculate from α and n. This first formula is useful for proving both our partial order on $|\mathcal{T}(\alpha, [n])|$ with respect to tile size for fixed n, as well as for proving our upper bound on $|\mathcal{T}(\alpha, [n])|$. Unfortunately, it is less useful for deriving a strong lower bound. Thus, we define our second formula for counting $|\mathcal{T}(\alpha, [n])|$, the correctness of which we prove from the first formula using a combinatorial argument.

Before proceeding further with our results, we present a result characterizing the tilings of discrete intervals that was independently proven by Pederson and Wang [24] as well as by Bodini and Rivals [3]. For completeness, we present a proof of this as Lemma 5, but with the statement of the lemma and proof in the notation of this work.

Lemma 5 ([3, 24]). *For all $\alpha, n \subset \mathbb{Z}$ and any valid tiling $(A_i, B_i) \in \mathcal{T}(\alpha, [n])$, all segments of the i^{th} tiling have size k_s and the length of any rift of the i^{th} tiling is divisible by k_s.*

Proof. Once a k_s and k_r have been selected and knowing that $r_1 \neq \emptyset$, the only way to tile r_1 with translates of elements of A is with translates of elements of s_1. As elements of s_1 are consecutive as are those of r_1, the only way to do this is as in Lemma 7 (i.e. by defining B^* to be $\{x \cdot k_s | x \in [0, k_r/k_s]\}$ and let B^* be a subset of B). This also justifies the restriction of $\mathcal{R}$ that $k_s | k_r$, as otherwise tiling r_1 with translates of s_1 would not be possible (example: for $s_1 = \{1, 2\}$ and $s_2 = \{x, x+1\}$, the number of elements in $r_1 = [3, x-1]$ must be divisible by 2).

Now we prove that $\forall i[(|s_i| \neq 0) \implies (|s_i| = k_s)]$. By definition, $|s_1| = k_s$. Suppose this is the case for all s_j such that $j < i$. Consider s_i. If $|s_i| = 0$ the statement holds. Suppose then that $|s_i| \neq 0$ and that $|s_i| > k_s$. Then, as k_s is an element of B^*, we have that $s_i \cap (s_i + k_s) \neq \emptyset$ which is a contradiction. Suppose $|s_i| < k_s$. Let $I = \big[\max[s_i] + 1, \min[s_i + k_s] - 1\big]$ and observe that $1 \leq |I| < k_s$. The lower bound on $|I|$ follows directly from $|s_i| < k_s$ and the translation of s_i by k_s, while the upper bound on $|I|$ follows from the fact that $|s_i| > 0$ and the fact that we are taking the max from s_i for lower bound I, but we are taking the min from $s_i + k_s$ the upper bound I. Taken together, with the fact that $|s_i| > 0$, we have that $|I| < k_s$. Notice that introducing any element to B smaller then any element of B^* would result in a collision between translates of s_1. Given this, I must be tiled by some translate of s_j for $j < i$, but $|s_j| = k_s > |r^*|$ and thus, we have a contradiction. Together, these prove that $\forall i[(|s_i| \neq 0) \implies (|s_i| = k_s)]$ as desired. $\square$

With this, we proceed with defining the first counting formula and proving its correctness. We do this in two main steps in which we:

- Prove that the restrictions to the size of the first segment and rift (i.e. k_s and k_r respectively) and the sub-cases we sum over based upon these these values are **sufficient** to yield a valid tiling. This proves that our formula acts as a lower bound to $|\mathcal{T}(\alpha, [n])|$.
- Prove that the restrictions to the size of the first segment and rift (i.e. k_s and k_r respectively) and the sub-cases we sum over based upon these these values are **necessary** to yield a valid tiling. This proves that our formula acts as an upper bound to $|\mathcal{T}(\alpha, [n])|$.

As the value produced by our formula is both an upper and lower bound on $|\mathcal{T}(\alpha, [n])|$, it follows that it calculates the exact value of $|\mathcal{T}(\alpha, [n])|$. We can now define the first formula in full detail.

Definition 19. *For $\mathcal{S} = \{k_s \in \mathbb{Z}^+ : k_s|\alpha\}$ and*

$$\mathcal{R}_{k_s} = \{k_r \in \mathbb{Z}^+ : (k_s|k_r) \wedge (k_s + k_r|k_s\beta) \wedge \big((k_s = \alpha) \iff (k_r = 0)\big)\}$$

we define the set $\Psi_{(\alpha,[n],(k_s,k_r))}$ to be

$$\Psi_{(\alpha,[n],(k_s,k_r))} \triangleq \begin{cases} 0, & \alpha \nmid n \\ 1, & k_r = 0 \\ \left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s+k_r}\right]\right)\right| - \left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s+k_r}\right], (1, \cdot)\right)\right|, & \text{otherwise} \end{cases}$$

Using this definition, we prove that the following method can be used to count the tilings of $[n]$ by sets of size α.

Lemma 6.

$$|\mathcal{T}(\alpha,[n])| = \sum_{k_s\in\mathcal{S}} \sum_{k_r\in\mathcal{R}_{k_s}} \Psi_{(\alpha,[n],(k_s,k_r))}.$$

To prove Lemma 6, we first prove that the right hand side is an upper bound for the left hand side. We then use this to prove equality in all cases of $\Psi_{(\alpha,[n],(k_s,k_r))}$ as well as justifying the definitions of $\mathcal{S}$ and $\mathcal{R}_{k_s}$. After these lemmas, we proceed with the formal proof of Lemma 6.

We note that Lemma 7 and Lemma 8 require a structural result similar to Theorem 4 of Bodini and Rivals [3] and Corollary 2.2 of Pedersen and Wang [24] in their proofs. Unfortunately, the structural result we require is distinct from these, as it requires the additional allowance for the case of fixed tile size. This necessitates a new proof which is implicitly contained in the proofs of Lemma 7 and Lemma 8.

Lemma 7.

$$|\mathcal{T}(\alpha,[n])| \geq \sum_{k_s\in\mathcal{S}} \sum_{k_r\in\mathcal{R}_{k_s}} \Psi_{(\alpha,[n],(k_s,k_r))}.$$

Proof. The definition of $\Psi_{(\alpha,[n],(k_s,k_r))}$ gives us three cases, which are where its value equals either 0, 1, or in which its value is based on $|\mathcal{T}(\alpha',[n'])|$ (where the negative term can be seen as subtracting away the case where $k_s=1$). The first case, where $|\mathcal{T}(\alpha,C)|=0$, need not be handled for the lower bound, as such cases only reduce the value of the sum. For $|\mathcal{T}(\alpha,C)|=1$, as $k_r=0$ and $\alpha|n$, we can always set $A=[\alpha]$ and $B=\{x\cdot\alpha : x\in[0,\beta-1]\}$, resulting in $A+B=C$. Lastly, we handle the case where the value of $\Psi_{(\alpha,[n],(k_s,k_r))}$ is based upon $|\mathcal{T}(\alpha',[n'])|$. To address this case, we define an injective mapping

$$f_{k_{(s,i)},k_{(r,i)}} : \mathcal{T}(\alpha/k_{(s,i)},[n/(k_{(s,i)}+k_{(r,i)})]) \to \mathcal{T}(\alpha,[n],(k_{(s,i)},k_{(r,i)}))$$

where we let $T_i = (A_i, B_i) \in \mathcal{T}(\alpha, [n], (k_{(s,i)}, k_{(r,i)}))$ and $T_j = (A_j, B_j) \in \mathcal{T}(\alpha/k_{(s,i)}, [n/(k_{(s,i)} + k_{(r,i)})])$ and assume $k_{(s,i)}, k_{(r,i)} \geq 2$. For notational convenience, we suppress the subscripts of f for the remainder of this proof. We abuse notation slightly and let $f(A_j) = A_i$ (or $f(B_j) = B_i$) if and only if $f(T_j) = T_i$. To construct such an f, we map T_j to T_i such that $m \in A_j + B_j$ if and only if $[(m-1)(k_{(s,i)} + k_{(r,i)}) + 1, m(k_{(s,i)} + k_{(r,i)})] \in A_i + B_i$. We call this the ***key property*** of f. The main idea behind the key property is that, as T_i tiles a larger set than T_j, we can expand each point of A_j to be multiple consecutive points in A_i.[6]

To complete the proof, we define f and show it is injective. Let T_j be an arbitrary valid tiling in $\mathcal{T}(\alpha/k_{(s,i)}, [n/(k_{(s,i)} + k_{(r,i)})])$ such that $k_{(s,j)} \geq 2$. As T_j is arbitrary, maintaining the key property forces us to make sure that $[2(k_{(s,i)}+k_{(r,i)})] \subset A_i+B_i$ for any A_i and B_i such that $f_{T_j} = (A_i, B_i)$ for some j. Thus, let $[k_{(s,i)}]$ and $[k_{(s,i)}+k_{(r,i)}+1, 2k_{(s,i)}+k_{(r,i)}]$ both be subsets of A_i. Given these facts about A_i, it follows that $B^* = \{x \cdot k_{(s,i)} : x \in [0, k_{(r,i)}/k_{(s,i)}]\}$ is a subset of B_i. This gives us that $[2(k_{(s,i)}+k_{(r,i)})] \subset A_i+B_i$ as desired. For other $m \in A_j$, we let $[(m-1)(k_{(s,i)}+k_{(r,i)})+1, (m-1)(k_{(s,i)}+k_{(r,i)})+k_{(s,i)}]$ be in $f(A_j) = A_i$ to ensure that key property is maintained. To see this, notice that this implies that $B^* + [(m-1)(k_{(s,i)} + k_{(r,i)})+1, (m-1)(k_{(s,i)}+k_{(r,i)})+k_{(s,i)}] = [(m-1)(k_{(s,i)}+k_{(r,i)})+1, m(k_{(s,i)}+k_{(r,i)})] \subset A_i+B_i$ as desired. For all $b_{(\ell,j)} \in B_j$, we have $A_j + b_{(\ell,j)} \subset A_j + B_j$ by definition. Let m be in $A_j + b_{(\ell,j)}$ where $b_{(\ell,j)} \neq 0$. Notice, that by adding

$$B^* + \left\{ \frac{b_{\ell,j}(k_{(s,i)} + k_{(r,i)})}{k_{(s,i)}} \right\}$$

to $f(B_j) = B_i$, we get that $[(m-1)(k_{(s,i)} + k_{(r,i)}) + 1, m(k_{(s,i)} + k_{(r,i)})] \subset A_i + B_i$ as desired. □

To give an example of the above using $A_i + B_i = [24]$ and $A_j + B_j = [48]$, the tiling

6. The opposite direction also holds in that on can compress consecutive points in A_i down to single points of A_j, but this follows from the upper bound, not the lower bound

$T_{(k',j)} = (\{1,3,9,11\}, \{0,1,12,13\})$ would map to

$$T_{(k,i)} = (\{1,2,5,6,17,18,21,22\}, \{0,2,24,26\})$$

via f. In order to prove that the sum from Lemma 6 (along with the given definitions for $\mathcal{S}$ and $\mathcal{R}_{k_s}$) act as an upper bound to the number distinct tilings of $C = [n]$, we require the following definition.

Definition 20. *For $(A, B) \in \mathcal{T}(\alpha, C)$, let the x^{th} meta-point of C with respect to A (denoted by x^*) be the set $[(x-1)(k_s + k_r) + 1, x(k_s + k_r)]$. We use the terminology of segments to refer to consecutive sets of meta-points in C and refer to these as* ***meta-segments*** *(i.e. s_i^*). We extend the idea of rifts to* ***meta-rifts*** *(i.e. r_i^*) similarly. More formally we have that*

- $s_1^* \triangleq \{x^* \subset A : \max[x^*] < \min[C \setminus A]\}$
- $r_1^* \triangleq \Big\{x^* \subset C \setminus A : \max[x^*] < \min\big[A \setminus s_1^*\big]\Big\}.$
- $s_i^* \triangleq \Big\{x^* \subset A : \max[s_{i-1}^*] < x < \min\Big[(C \setminus A) \setminus \{y \in C : y \leq \max[r_{i-1}^*]\}\Big]\Big\}$
- $r_i^* \triangleq \left\{x^* \subset C \setminus A : x^* \subset \left(\max[r_{i-1}^*], \min\left[A \setminus \Big(\bigcup_{k=1}^{i-1} s_k^*\Big)\right]\right)\right\}.$

Lemma 8. *Suppose that $\alpha|n$, $k_s \in \mathcal{S}$, and $k_r \in \mathcal{R}_{k_s} \setminus \{0\}$. Then it follows that*

$$\Psi_{(\alpha,[n],(k_s,k_r))} = \left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s + k_r}\right]\right)\right| - \left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s + k_r}\right], (1, \cdot)\right)\right| = \big|\mathcal{T}(\alpha, [n], (k_s, k_r))\big|.$$

In addition, no selection of k_s and k_r such that $k_s \notin \mathcal{S}$ and $k_r \notin \mathcal{R}_{k_s}$ has any valid tilings associated with them.

Proof. Lemma 5 justifies the first restriction of $\mathcal{S}$, as A is made up of segments, so if each segment has cardinality k_s, then it must be the case that $k_s|\alpha$. Further, from the proof of Lemma 5, we can conclude that $(s_1 \cup s_2) + B^*$ must tile exactly $[2(k_s + k_r)]$ in any valid tiling as done in Lemma 7. For example, if $s_1 = \{1, 2\}$ and $s_2 = \{7, 8\}$, we know that $\{0, 2, 4\} \subset B$ as these are necessary to tile $r_1 = [3, 6]$ with translates of s_1. These elements of B then also sum with the elements s_2 so that $(s_1 \cup s_2) + \{0, 2, 4\} = [12]$. Unless $C = [12]$, there are two possible ways the next elements of C (i.e. $\{13, 14\}$) can be tiled. Either these elements are in s_3 or they are tiled by further translates of s_1. As we will see below, this decision for $C = [n]$ in this example ends up being akin to the choice of whether or not to include 3 in A for $C = [n/(k_s + k_r)] = [n/6]$ (and with the size of A reduced by a factor of $k_s = 2$).

If $n = 2(k_s + k_r)$, we have found the unique valid tiling for this k_s and k_r. In terms of meta-points, this case corresponds to tiling the set $\{1^*, 2^*\}$, where $1^* = [k_s + k_r]$ and $2^* = [k_s + k_r + 1, 2(k_s + k_r]$. Consider the case of $n = \ell(k_s + k_r)$ for $\ell > 2$. There are two ways to tile $2(k_s + k_r) + 1$ in $A + B$. Either $2(k_s + k_r) + 1 \in s_3$ or $2(k_s + k_r) + 1 \in s_1 + b_i$ for some $b_i \in B$. It cannot be the case that $2(k_s + k_r) + 1 \in s_2 + b_i$ for some $b_i \in B$, as this would imply that $(s_1 + b_i) \cap [2(k_s + k_r)] \neq \emptyset$ which would not yield a valid tiling. Suppose $2(k_s + k_r) + 1 \in s_3$. It follows that $(s_1 \cup s_2 \cup s_3) + B^* = [3(k_s + k_r)]$. The number of times we repeat this process determines $|s_i^*|$, as each such decision to add s_i for a new i as soon as possible essentially adds one new point to the first meta-segment. Suppose $2(k_s + k_r) + 1 \in s_1 + b_i$. It follows that $\min[s_3] > 4(k_s + k_r)$. This is because the number of elements between $s_1 + b_i$ and $s_2 + b_i$ is $2(k_s + k_r) - k_s$, but $s_3 + B^*$ is a set of $2(k_s + k_r)$ consecutive elements. Thus, s_3 (and by extension, $s_3 + B^*$) cannot appear until at least $4(k_s + k_r) + 1$. Thus, this decision of how to tile $2(k_s + k_r) + 1$ leads to a meta-point being added to a meta-rift.

The choice between the two options outlined above as to how to tile $2(k_s + k_r) + 1$ is repeated once every $k_s + k_r$ elements and are the only ones, as repeating the case analysis from above leads to similar contradictions. As these potential tilings align exactly with those in Lemma 7, we know these tilings are valid. Thus, the solutions as to how to tile $[n]$ are exactly the ways to tile $[n/(k_s + k_r)]$ with a valid tiling for which $k_{(s,j)} \geq 2$ (which is accounted for by the negative term with $k_{(s,j)}$ fixed to 1). To justify the second restriction (i.e. $k_s + k_r | k_s\beta$) of $\mathcal{R}_{k_s}$ notice that, due to the fact that segments are of length k_s and the definition of B^*, we have that $k_s + k_r$ elements are grouped into meta-points and are either tiled or not tiled as a group. Once r_1 is tiled by translates of s_1, the first $a \cdot (k_s + k_r)/k_s$ elements of C will be tiled. Thus, it must be that $(\alpha \cdot \mathcal{T}(k_s + k_r)/k_s) | \alpha\beta$, as otherwise translates of these $a \cdot (k_s + k_r)/k_s$ elements could not tile C. This divisibility requirement simplifies to the restriction $(k_s + k_r) | k_s\beta$ as required. For the last restriction to elements of $\mathcal{R}_{k_s}$, its necessity follows from the definition of segments and rifts. □

We now handle the other two cases for $\Psi_{(\alpha,[n],(k_s,k_r))}$ relevant to the upper bound.

Lemma 9. *For some $\mathcal{T}(\alpha, [n], (k_s, k_r))$, if $\alpha \nmid n$, then $|\mathcal{T}(\alpha, [n], (k_s, k_r))| = 0$. Otherwise, if $k_r = 0$, then $|\mathcal{T}(\alpha, [n], (k_s, k_r))| = 1$.*

Proof. If $a \nmid n$, then $|C| = \alpha\beta$ is impossible and the lemma holds. If $k_r = 0$, then $r_1 = \emptyset$ and $A = [\alpha]$. Consider trying to change B from $B = \{x \cdot \alpha : x \in [0, b-1]\}$ as defined in Lemma 7. We attempt to do this via induction on the elements of B. The base case would be to change 0, but this is not allowed by the definition of $\mathcal{T}$. Suppose that b_i is the first element that should be adjusted and assume without loss of generality that we cannot reduce it below $b_{i-1} + 1$ or increase it to be greater then $b_{i+1} - 1$. If we increase b_i, then $b_i + 1$ is no longer in $A + B$ which is a contradiction. If we decrease b_i, then $A + b_{i-1} \cap A + b_i \neq \emptyset$. Thus, b_i cannot be changed while still yielding a valid tiling. □

With the prior results of this section in hand, Lemma 6 immediatly follows.

Proof of Lemma 6. Lemma 8 bounds shows equality in the recursive case of $\Psi_{(\alpha,[n],(k_s,k_r))}$ and justifies the restrictions to $\mathcal{S}$ and $\mathcal{R}_{k_s}$. Lemma 9 shows equality in the other two cases of $\Psi_{(\alpha,[n],(k_s,k_r))}$. Summing over all valid k_s and k_r yields the lemma. □

While the formula from Lemma 6 is the one we use to prove our upper bound, it is not in a convenient form for the purpose of lower bound analysis. Thus, we give an alternative formula that we prove to be equivalent and which we use to prove our lower bound.

Lemma 10. *Let $\mathcal{P}_{(n,k)}$ be the set of products of k distinct prime divisors of n.*

$$|\mathcal{T}(\alpha,[n])| = \sum_{k\in[n]} \sum_{v\in\mathcal{P}_{(n,k)}} (-1)^{k+1}\Big(|\mathcal{T}(\alpha,[n/v])| + |\mathcal{T}(\alpha/v,[n/v])|\Big).$$

Proof. Let $k_{(s,i)}$ and $k_{(r,i)}$ be the size of the first segment and rift of the sumset tile (A_i, B_i). We prove this by showing this formula's equivalence to the formula from Lemma 6. We break the tilings from Lemma 6 into two case: tilings of $\mathcal{T}(\alpha,[n])$ such that $k_s > 1$ and tilings such that $k_s = 1$. We define a function f_1 that takes as input $(\alpha,[n])$ and an element of $\mathcal{T}(\alpha,[n/v])$, and outputs an element of $\mathcal{T}(\alpha,[n])$ such that $v|(k_s+k_r)$ and $k_s = 1$. Further we prove that, for any $(A_i,B_i)\in\mathcal{T}(\alpha,[n])$ such that $v|(k_{(s,i)}+k_{(r,i)})$ and $k_{(s,i)} = 1$, there exists a unique $(A_j,B_j)\in\mathcal{T}(\alpha,[n/v])$ such that $f_1((\alpha,[n]),(A_j,B_j)) = (A_i,B_i)$. Similarly, we define a function f_2 that takes as input $(\alpha,[n])$ and an element of $\mathcal{T}(\alpha/v,[n/v])$, and outputs an element of $\mathcal{T}(\alpha,[n])$ such that $v|k_s$. Further we prove that, for any $(A_i,B_i)\in\mathcal{T}(\alpha,[n])$ such that $v|k_{(s,i)}$, there exists a unique $(A_j,B_j)\in\mathcal{T}(\alpha/v,[n/v])$ such that $f_2((\alpha,[n]),(A_j,B_j)) = (A_i,B_i)$.

We begin with f_1. For any $(A_j,B_j)\in\mathcal{T}(\alpha,[n/v])$ such that $f_1((\alpha,[n]),(A_j,B_j)) = (A_i,B_i)$, we let $A_i = \{x : x/v \in A_j\}$ and $B_i = \{y : y/v \in B_j\} + [0,v]$. $k_{(s,i)} = 1$ follows from the fact that $v \geq 2$. As for $d|(k_{(s,i)} + k_{(r,i)})$, one can see from the definition of B_i that the first rift must end at some multiple of v which implies that $v|(k_{(s,i)} + k_{(r,i)})$. For any $(A_i,B_i)\in\mathcal{T}(\alpha,[n])$ such that $v|(k_{(s,i)} + k_{(r,i)})$ and $k_{(s,i)} = 1$, it follows from Lemma 6 that the single point segments of (A_i,B_i) occur only at positions t such that $p_i|(t-1)$ (this follows from the division of n by $k_s + k_r$ in the last case of the definition of $\Psi_{(\alpha,[n],(k_s,k_r))}$). Thus, f_1 is invertable and the claim follows. As for f_2, for any $(A_j,B_j)\in\mathcal{T}(\alpha/v,[n/v])$ such that

$f_1((\alpha,[n]),(A_j,B_j)) = (A_i,B_i)$, we let $A_i = \{x : \lfloor x/v \rfloor \in A_j\}$ and $B_i = \{y : y/v \in B_j\}$. $v|k_{(s,i)}$ follows from the fact that $\lfloor x/v \rfloor$ has the same value for every v consecutive values of x. For any $(A_i,B_i) \in \mathcal{T}(\alpha,[n])$ such that $v|k_{(s,i)}$, it follows from Lemma 6 that every segment and rift is divisible by v. Thus, f_2 is invertable and the claim follows.

Let $\{p_1,\ldots,p_m\}$ be the prime divisors of n. Notice that f_1 and f_2 map to disjoint subsets of $\mathcal{T}(\alpha,[n])$, but that the union of their codomains on inputs from $\mathcal{T}(\alpha,[n/p_i])$ and $\mathcal{T}(\alpha/p_i,[n/p_i])$ respectively for all $i \in [m]$ is exactly $\mathcal{T}(\alpha,[n])$. The issue then in simply summing the size of these codomains is that an element of $\mathcal{T}(\alpha,[n/p_i])$ and an element of $\mathcal{T}(\alpha,[n/p_j])$ for $i \neq j$ may map to the same element $(A_z,B_z) \in \mathcal{T}(\alpha,[n])$ by f_1. By the definition of f_1, this would imply that $p_ip_j|(k_{(s,z)} + k_{(r,z)})$ and $k_{(s,z)} = 1$, which means we can remove the over counting by subtracting cases for which v is composed of 2 distinct prime factors of n (though now we may be under counting). More generally, we can apply the inclusion-exclusion principle with respect to the number of prime factors of v to arrive at an exact count as desired. □

3.5 Upper and Lower Bound Calculations

In this section, we have three primary results:

1. A partial order on $|\mathcal{T}(\alpha,[n])|$ with respect to α.

2. A upper bound on $|\mathcal{T}(\alpha,[n])|$ for sufficiently large n utilizing $\Psi_{(\alpha,[n],(k_s,k_r))}$.

3. A lower bound on $|\mathcal{T}(\alpha,[n])|$ for specific α and infinitely many n that is super-linear in n, proved by analyzing the formula from Lemma 10.

Prior to beginning the proof (or series of proofs) necessary to prove each of these results, we provide a high level outline as to our approach. Further, before we dive into any of these, we wish to establish the following useful corollary.

Corollary 6. $\mathcal{T}(\alpha, [n]) = \mathcal{T}(\beta, [n])$.

Proof. Let $P_j = (A, B)$ be a tiling of $[n]$ such that $|A| = \alpha$ and $|B| = \beta$. Define P_j' to be (A', B'), where $A' = B + \{1\}$ and $B' = A - \{\min[A]\}$. Notice that P_j' is a tiling of $[n]$ such that $|A'| = \beta$ and $|B'| = \alpha$. □

As our results allow for fixed tile sizes, it is natural to ask, for a fixed $[n]$, are there more tiles of size α or of size α' that tile C? While we are unable to prove a total order on $|\mathcal{T}(\alpha, [n])|$ with respect to α and fully resolve this question, we are able to prove a partial order by leveraging the intuition that, the closer α is in its prime factorization to being a square root of n, the more distinct tiles there will be of size α that tile $[n]$. The difficulty in proving this statement is in formalizing this notion of "closeness" to being a square root of n. Our proof simplifies this question by proving an order with respect to α and α' in cases where one has strictly higher multiplicity in each of its prime factors (and such that neither has too high of a multiplicity in any prime factor).

With this restriction as to the α we compare, we can utilize the recursive case of $\Psi_{(\alpha,[n],(k_s,k_r))}$ to give an inductive proof and derive our desired partial order. In Lemma 11, we first prove a base case with respect to α by comparing tiles of size 1 to tiles of size p_i. We then make our inductive hypothesis with respect to both α and n, as to apply our inductive step, we will be relying on the recursion from Definition 19, which reduces the size of both α and n. Lastly, we pick apart the elements from the sets $\mathcal{S}$ and $\mathcal{R}_{k_s}$ for the cases in question, separating these pairs into groups of cases that can be more easily compared via an equality or inequality. Once all cases have been accounted for in this manner and with all inequalities being in the same direction, the proof will be complete.

Lemma 11. *Let $\alpha = p_1^{\mu_1} \cdot p_2^{\mu_2} \cdot \ldots \cdot p_k^{\mu_k}$, let $n = p_1^{\psi_1} \cdot p_2^{\psi_2} \cdot \ldots \cdot p_k^{\psi_k}$, and assume $\forall i (2\mu_i \leq \psi_i)$. It follows that, for all j such that $1 \leq \mu_j$, we have $|\mathcal{T}(\alpha/p_j, [n])| < |\mathcal{T}(\alpha, [n])|$.*

Proof. We proceed by induction on

$$\Omega^*(\alpha) = \sum_{i=1}^{k} \mu_i$$

and n, where $\Omega^*(\alpha)$ is the prime omega function. For $\Omega^*(\alpha) = 1$ and all n, we have that $|\mathcal{T}(\alpha/p_j, [n])| = |\mathcal{T}(1, [n])| = 1$. By assumption we have $p_j^2|n$ and thus, in conjunction with Lemma 6, we have that $|\mathcal{T}(p_j, [n])| \geq |\mathcal{T}(1, [n/p_j])| + |\mathcal{T}(p_j, [n/p_j]) \geq 2$. Thus, the base case with respect to $\Omega^*(\alpha)$ holds for all n. Assuming that $|\mathcal{T}(\alpha/p_j, [n])| < |\mathcal{T}(\alpha, [n])|$ holds for all α such that $\Omega^*(\alpha) \leq m$ and for contiguous C such that $|C| < n$, we prove that this implies it holds for $\Omega^*(\alpha) = m + 1$ and n. Without loss of generality, let this case be such that

$$\alpha = p_1^{\mu_1+1} \cdot p_2^{\mu_2} \cdot \ldots \cdot p_k^{\mu_k}$$

and assume that $2\mu_1 \leq \psi_1$. By the assumption that $\forall i(2\mu_i \leq \psi_i)$, it follows that $\mathcal{T}(\alpha, [n], (k_s, k_r))$ has strictly more potential values for k_s (i.e. has larger $|\mathcal{S}|$) when compared to $\mathcal{T}(\alpha/p_1, [n], (k_s, k_r))$, but that each value of $k_s \in \mathcal{S} \setminus \{\alpha, \alpha/p_1\}$ has exactly one more option for k_r (i.e. $|\mathcal{R}_{k_s}|$ is one larger in the case of α/p_1). This latter fact follows from the requirement that $k_s + k_r | k_s\beta$ and that the case with α/p_1 results in an associated β' such that $\beta' = p_1\beta$, as we have $(\alpha/p_1)(\beta') = n$ by the definition of a valid tiling. Thus, $k_s + k_r = p_1 k_s \beta$ is possible in the case of $\mathcal{T}(\alpha/p_i, [n], (k_s, k_r))$, but not $\mathcal{T}(\alpha, [n], (k_s, k_r))$. The only exception to this is when we have $\mathcal{T}(\alpha, [n], (\alpha, k_r))$ and $\mathcal{T}(\alpha/p_1, [n], (\alpha/p_1, k_r))$ as in both such cases the only possible k_r is 0.

Based on the above, we begin by separating out the easiest cases to compare for both $\mathcal{T}(\alpha, [n])$ and $\mathcal{T}(\alpha/p_1, [n])$, then proceed with handling the outliers. First, we have that

$$|\mathcal{T}(\alpha, [n], (\alpha, 0))| = |\mathcal{T}(\alpha/p_1, [n], (\alpha/p_1, 0))| = 1.$$

Next, for $k_s \in \mathcal{S} \setminus \{\alpha, \alpha/p_1\}$ and k_r such that $k_s + k_r | k_s\beta$, we have that

$$|\mathcal{T}(\alpha, [n], (k_s, k_r))| = \left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s + k_r}\right]\right)\right| - \left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s + k_r}\right], (1, \cdot)\right)\right|$$

as well as

$$|\mathcal{T}(\alpha/p_1, [n], (k_s, k_r))| = \left|\mathcal{T}\left(\alpha/p_1k_s, \left[\frac{n}{k_s + k_r}\right]\right)\right| - \left|\mathcal{T}\left(\alpha/p_1k_s, \left[\frac{n}{k_s + k_r}\right], (1, \cdot)\right)\right|.$$

By our inductive hypothesis, we have that

$$\left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s + k_r}\right]\right)\right| > \left|\mathcal{T}\left(\alpha/p_1k_s, \left[\frac{n}{k_s + k_r}\right]\right)\right|.$$

By Lemma 6 and our prior observation about the relative number of k_r in such cases, we have that

$$\left|\mathcal{T}\left(\alpha/k_s, \left[\frac{n}{k_s + k_r}\right], (1, \cdot)\right)\right| = \left|\mathcal{T}\left(\alpha/p_1k_s, \left[\frac{n}{k_s + k_r}\right], (1, \cdot)\right)\right| + 1.$$

Thus, it follows that $|\mathcal{T}(\alpha, [n], (k_s, k_r))| \geq |\mathcal{T}(\alpha/p_1, [n], (k_s, k_r))|$ for all such k_s and k_r.

We now classify all remaining tilings for tiles of size α/p_1. In this case we have $\mathcal{T}(\alpha/p_1, [n], (k_s, k_r))$ for $k_s \in \mathcal{S} \setminus \{\alpha/p_1\}$ and k_r such that $k_s + k_r = p_1k_s\beta$. As $n = \alpha\beta$ by definition, we also have that

$$\frac{n}{k_s + k_r} = \frac{\alpha\beta}{p_1k_s\beta} = \frac{\alpha}{p_1k_s}$$

from which it immediately follows that

$$|\mathcal{T}(\alpha/p_1, [n], (k_s, (p_1\beta - 1)k_s))| = |\mathcal{T}(\alpha/p_1k_s, [\alpha/p_1k_s])| - |\mathcal{T}(\alpha/p_1k_s, [\alpha/p_1k_s], (1, (\cdot)| = 1.$$

Further, this implies that $\sum_{k_s} |\mathcal{T}(\alpha/p_1, [n], (k_s, (p_1\beta - 1)k_s))| = |\mathcal{S} \setminus \{\alpha, \alpha/p_1\}|$ for $k_s \in \mathcal{S} \setminus \{\alpha, \alpha/p_1\}$.

Lastly, we show that there are a strictly greater then $|\mathcal{S} \setminus \{\alpha, \alpha/p_1\}|$ tilings in $\mathcal{T}(\alpha, [n])$ we have not yet counted, thus proving the lemma. Consider the tilings in $\mathcal{T}(\alpha, [n], (\alpha/p_1, k_r))$. Notice that, by our assumption that $\forall i(2\mu_i \leq \psi_i)$ and with $k_s = \alpha/p_i$, it follows that

$$|\mathcal{R}_{\alpha/p_i}| > |\mathcal{S} \setminus \{\alpha, \alpha/p_1\}|.$$

As each $k_r \in \mathcal{R}_{\alpha/p_i}$ has at least one valid tiling associated with it, this proves the lemma. $\square$

We now move on to proving our upper bound on $|\mathcal{T}(\alpha, [n])|$. Lemma 12 leverages the fact that the formula from Lemma 6 has a recursive structure, with a branching factor based upon $|\mathcal{S}|$ and the size of $|\mathcal{R}_{k_s}|$ for each k_s and at most logarithmic depth. We then use the fact that the definitions of $\mathcal{S}$ and $\mathcal{R}_{k_s}$ are based around divisibility to estimate the maximum number of branches based on the number of divisors of α and β (and thus, the maximum number of valid tilings). We conclude the upper bound proof with Corollary 7, which combines the result of Lemma 12, an estimate on the number of divisors of an integer, and an ideal setting of α to yield our upper bound. A similar proof to the one just outlined likely works to prove an upper bound on the number of tilings of an interval of the discrete line via the structural results Bodini and Rivals [3]. However, it would introduce an additional $\sigma_0(n)$ multiplicative factor to Lemma 12, resulting in a final upper bound that would be larger by approximately a $\log n$ multiplicative factor in the exponent.

Lemma 12. $|\mathcal{T}(\alpha, [n])| \leq (\sigma_0(\alpha) \cdot \sigma_0(\beta) - \sigma_0(\alpha) - \sigma_0(\beta) + 2)^{\log n}$.

Proof. By Corollary 6, we can assume without loss of generality that $\alpha \geq \beta$. We prove that the number of tilings that do not violate the restrictions of Lemma 7 on $\mathcal{S}$ and $\mathcal{R}_{k_s}$ is exactly $\sigma_0(\alpha) \cdot \sigma_0(\beta) - \sigma_0(\alpha) - \sigma_0(\beta) + 2$. The number of valid choices of k_s is exactly $\sigma_0(\alpha)$. As to $\mathcal{R}_{k_s}$, we first handle the case of $k_s = \alpha$. As the third restriction forces $k_r = 0$, this results in a single valid k_r. Next, consider when $k_s = 1$. In this case, $k_s | k_r$ for any choice of k_r, so the first restriction is satisfied. Lastly, the second restriction simplifies to $k_r + 1 | \beta$. The number of k_r that satisfy this is exactly $\sigma_0(\beta) - 1$, as the only divisor of β we cannot form with the sum $k_r + 1$ is 1.

We can now handle any remaining cases. Let $div(\beta)$ be the set of divisors of β. Notice that $div(k_s\beta)$ is exactly $\big(k_s \cdot div(\beta)\big) \cup div(k_s) \cup div(\beta)$. Due to the second restriction of $\mathcal{R}_{k_s}$ (i.e. $k_s + k_r | k_s\beta$) and that fact that $k_s + k_r > k_s$, we can narrow down options for k_r from $div(k_s\beta)$ to $(k_s \cdot div(\beta)) \cup div(\beta)$. By the first restriction of $\mathcal{R}_{k_s}$, we have that $k_s | k_r$, so we can further simplify valid options for k_r from $k_s \cdot div(\beta) \cup div(\beta)$ to $k_s \cdot div(\beta)$. Lastly, $k_r \neq k_s$ and thus, the number of valid options for k_r equals

$$|k_s \cdot div(\beta)| - 1 = \sigma_0(\beta) - 1.$$

Thus, for all k_s such that $k_s | a$ and $k_s \neq a$, we have $\sigma_0(\beta) - 1$ options for β. When $k_s = \alpha$, we have exactly 1 option for k_s. Each of these $\sigma_0(\alpha) \cdot \sigma_0(\beta) - \sigma_0(\alpha) - \sigma_0(\beta) + 2$ sub-cases then divides n by at least 2. Thus, if we very conservatively assume that each sub-case decreases n by a factor of 2, we get that

$$|\mathcal{T}(\alpha, [n])| \leq (\sigma_0(\alpha) \cdot \sigma_0(\beta) - \sigma_0(\alpha) - \sigma_0(\beta) + 2)^{\log n}$$

as desired. $\square$

By Theorem 317 of Hardy and Wright [14] (which is attributed to Wigert (1907)) we have that, for any $\epsilon > 0$ and infinitely many sufficiently large n,[7]

$$2^{(1-\epsilon)\log(n)/\log\log(n)} < \sigma_0(n) < 2^{(1+\epsilon)\log(n)/\log\log(n)}.$$

Thus, for infinitely many n, we have that $\sigma_0(n) \sim 2^{\log(n)/\log\log(n)}$.

7. Only the lower bound is infinitely often. The upper bound holds for all sufficiently large n.

Corollary 7. *For $C = [n]$, all n, and any $\epsilon > 0$ we have that*

$$\max_{\alpha}\left[|\mathcal{T}(\alpha, C)|\right] \leq (2^{\frac{(1+\epsilon)\log n}{\log\log\sqrt{n}}} - 2^{\frac{\log\sqrt{n}}{\log\log\sqrt{n}}+1} + 2)^{\log n} < n^{\frac{(1+\epsilon')\log n}{\log\log n}}.$$

Proof. Notice that the inner $\sigma_0(\alpha)\cdot\sigma_0(\beta) - \sigma_0(\alpha) - \sigma_0(\beta) + 2$ is maximized when $\alpha = \beta = \sqrt{n}$. By Lemma 12, we have at most $(\sigma_0(\sqrt{n})^2 - 2\sigma_0(\sqrt{n}) + 2)^{\log n}$ valid tilings. Using the bounds of Wigert [14] yields the statement. □

This is a significant improvement over the trivial upper bound of $\binom{n}{2}$. Next, we prove our super-linear lower bound on $|\mathcal{T}(\alpha, [n])|$. The primary lemmas that achieve this are Lemma 14 and Lemma 15.

We begin our series of lower bound proofs with Lemma 13. All of the lemmas related to our lower bound rely on our formula for counting $|\mathcal{T}(\alpha, [n])|$ from Lemma 10. Thus, the primary purpose of Lemma 13 is to highlight this by applying Lemma 10 to calculate $|\mathcal{T}(2^{\lfloor k/2\rfloor}, [2^k])|$. The choice of n and α greatly simplify this lemma, as the choice of $n = 2^k$ means that their are no negative terms, and our choice of $\alpha = 2^{\lfloor k/2\rfloor}$ leads to an opportunity to apply Corollary 6.

Lemma 13. *For $k \in \mathbb{Z}^+$, we have that $|\mathcal{T}(2^{\lfloor k/2\rfloor}, [2^k])| > \frac{2^k}{\sqrt{2k}}$.*

Proof. In this case, Lemma 10 simplifies to

$$|\mathcal{T}(2^{\lfloor k/2\rfloor}, [2^k])| = |\mathcal{T}(2^{\lfloor k/2\rfloor}, [2^{k-1}])| + |\mathcal{T}(2^{\lfloor k/2\rfloor - 1}, [2^{k-1}])|.$$

Notice that, if we continue to reapply Lemma 10 in this way, we eventually end up with only the term $|\mathcal{T}(1, [1])|$ with multiplicity $\binom{k}{k/2}$. This is because there it take k divisions by 2 before the set to be tiled resolves to [1], but exactly $k/2$ of these must divide the tile size by

2 to reach a tile size of 1. Thus, we have that

$$\begin{aligned}\mathcal{T}(2^{\lfloor k/2 \rfloor}, [2^k])| &= \binom{k}{k/2} |\mathcal{T}(1, [1])| \\ &= \binom{k}{k/2} \\ &> \frac{2^k}{\sqrt{2k}}.\end{aligned}$$

□

The next two preliminary lower bound lemmas are Lemma 14 and Lemma 15. Lemma 15 extends Lemma 13 by multiplying n by 3^2. While this appears to be a minor change, it serves to illustrate the idea that increasing the number of prime factors of n can have the effect of increasing the number of valid tilings by some constant factor in the exponent. However, now that we have more then one prime factor of n and unlike in Lemma 15, applying Lemma 10 results in some positive and some negative terms. Thus, we require an inequality that allows us to remove all of the negative terms at the cost of positive terms of relativity little importance. We provide such an inequality and prove its correctness in Lemma 14, then prove Lemma 15 utilizing it.

Lemma 14. *For $k \geq 1$ and α such that $3 | \alpha$ and $3^2 \nmid \alpha$,*

$$|\mathcal{T}(\alpha, [2^k 3^2])| \geq |\mathcal{T}(\alpha, [2^{k-1} 3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1} 3^2])| + |\mathcal{T}(\alpha/3, [2^k])|.$$

Proof. Our approach to this proof will be to expand $|\mathcal{T}(\alpha, [2^k 3^2])|$ using Lemma 10, isolate the terms in the sum $|\mathcal{T}(\alpha, [2^{k-1} 3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1} 3^2])| + |\mathcal{T}(\alpha, [2^k])|$ and prove that the sum of the remaining terms from the inclusion-exclusion formula is non-negative. By Lemma

10, we have that

$$|\mathcal{T}(\alpha, [2^k 3^2])| = |\mathcal{T}(\alpha, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3^2])| + |\mathcal{T}(\alpha, [2^k 3])| + \\ |\mathcal{T}(\alpha/3, [2^k 3])| - |\mathcal{T}(\alpha, [2^{k-1}3])| - |\mathcal{T}(\alpha/6, [2^k 3])|.$$

We can now apply Lemma 10 again, this time to the term $|\mathcal{T}(\alpha, [2^k 3])|$ to get

$$\begin{aligned}
|\mathcal{T}(\alpha, [2^k 3^2])| =& |\mathcal{T}(\alpha, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3^2])| + |\mathcal{T}(\alpha, [2^{k-1}3])| + |\mathcal{T}(\alpha/2, [2^{k-1}3])| + \\
& |\mathcal{T}(\alpha, [2^k])| + |\mathcal{T}(\alpha/3, [2^k])| - |\mathcal{T}(\alpha, [2^{k-1}])| - |\mathcal{T}(\alpha/6, [2^{k-1}])| \qquad (3.1) \\
& + |\mathcal{T}(\alpha/3, [2^k 3])| - |\mathcal{T}(\alpha, [2^{k-1}3])| - |\mathcal{T}(\alpha/6, [2^k 3])| \\
=& |\mathcal{T}(\alpha, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3^2])| + |\mathcal{T}(\alpha, [2^{k-1}3])| + |\mathcal{T}(\alpha/2, [2^{k-1}3])| + \\
& |\mathcal{T}(\alpha/3, [2^k])| - |\mathcal{T}(\alpha/6, [2^{k-1}])| + |\mathcal{T}(\alpha/3, [2^k 3])| - \qquad (3.2) \\
& |\mathcal{T}(\alpha, [2^{k-1}3])| - |\mathcal{T}(\alpha/6, [2^k 3])| \\
=& |\mathcal{T}(\alpha, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3])| + |\mathcal{T}(\alpha/3, [2^k])| - \\
& \qquad (3.3) \\
& |\mathcal{T}(\alpha/6, [2^{k-1}])| + |\mathcal{T}(\alpha/3, [2^k 3])| - |\mathcal{T}(\alpha/6, [2^k 3])|
\end{aligned}$$

where equation 5 follows from equation 4 due to $3|a$ and $3 \nmid |C|$. Thus, as $a \nmid |C|$, no valid tilings were dropped between these two lines. We now set aside the terms in $|\mathcal{T}(\alpha, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3^2])| + |\mathcal{T}(\alpha, [2^k])|$, leaving us with

$$|\mathcal{T}(\alpha/2, [2^{k-1}3])| - |\mathcal{T}(\alpha/6, [2^{k-1}])| + |\mathcal{T}(\alpha/3, [2^k 3])| - |\mathcal{T}(\alpha/6, [2^k 3])|.$$

Notice that

$$|\mathcal{T}(\alpha/2, [2^{k-1}3])| \geq |\mathcal{T}(\alpha/6, [2^{k-1}])|$$

and

$$|\mathcal{T}(\alpha/3, [2^k 3])| \geq |\mathcal{T}(\alpha/6, [2^k 3])|.$$

Thus,

$$|\mathcal{T}(\alpha/2, [2^{k-1}3])| - |\mathcal{T}(\alpha/6, [2^{k-1}])| + |\mathcal{T}(\alpha/3, [2^k 3])| - |\mathcal{T}(\alpha/6, [2^k 3])| \geq 0$$

as required. □

We now wish to leverage Lemma 14 to prove an improved lower bound in the case of $|\mathcal{T}(2^{\lfloor k/2 \rfloor} \cdot 3, [2^k 3^2])|$. To do this, the main insight that is required is that, if one repeatedly applies Lemma 14, first to some given $\mathcal{T}(\alpha, [n])$ and then again to some of the terms that result from applying Lemma 10 to $\mathcal{T}(\alpha, [n])$, there will begin to be “collisions" between some of the resulting terms. In Lemma 15, we are able to create many such collisions and utilize high multiplicity terms to improve our lower bound on $|\mathcal{T}(\alpha, [n])|$ with respect to n.

Lemma 15. *For $k \in \mathbb{Z}^+$, for $\alpha = 2^{\lfloor k/2 \rfloor} \cdot 3$, we have that $|\mathcal{T}(\alpha, [2^k 3^2])| = \omega(n)$.*

Proof. By Lemma 14, we know that

$$|\mathcal{T}(\alpha, [2^k 3^2])| \geq |\mathcal{T}(\alpha, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/2, [2^{k-1}3^2])| + |\mathcal{T}(\alpha/3, [2^k])|.$$

Consider the terms on the right side of the inequality. Notice that we can also apply Lemma 14 to the first two terms while leaving the third unchanged. If we continue to reapply Lemma 14 to the first two of the three terms that result (keeping the third term each time), we will be left with a sum of terms in the form of Pascal's triangle.[8] More formally, all of the terms

8. Alternatively known as Pingala's triangle, Yang Hui's triangle, and several other names due to its repeated independent discovery.

in the sum would be of the form

$$\binom{j}{i}|\mathcal{T}(\alpha/2^i3,[2^{k-j}])|,$$

where the coefficient of $\binom{j}{i}$ follows from the observation that, when visualized as Pascal's triangle, the term $|\mathcal{T}(\alpha/2^i3,[2^{k-j}])|$ appears in the j^{th} row and i^{th} column.

To clarify that this is indeed the case, notice that the first term on the right hand side of Lemma 14 equals the term from the left hand side with $|\mathcal{C}|$ divided by two. By contrast, the second term on the right hand side of Lemma 14 equals the term from the left hand side with both α and $|\mathcal{C}|$ divided. Thus, when we consider the number of ways to output the aforementioned third term of

$$\binom{j}{i}|\mathcal{T}(\alpha/2^i3,[2^{k-j}])|,$$

it follows that Lemma 14 was applied $j+1$ times (in j cases, $|\mathcal{C}|$ was divided by 2, while the final application removed the 3^2 term). Further, the tile size α was divided by 2 only i times. Thus, there are $\binom{j}{i}$ copies of this term when utilizing Lemma 14 to fully expand $|\mathcal{T}(\alpha,[2^k3^2])|$ (except for the third right hand side terms according to Lemma 14 which never expanded as mentioned prior).

We now wish to focus on a subset of terms from this expansion that balances reasonably high multiplicity with a fairly large set of tilings, as this will maximize the value of their product. This desire for a large number of distinct tilings means we wish to have fairly large $|\mathcal{C}|$, with $\alpha = \sqrt{|\mathcal{C}|}$, an objective that is achieved by consedering all of the terms in the middle vertical slice of cases when visualized as Pascal's triangle. For all cases in this slice, if we only focus only on the term with the 3 terms removed, we have that there are

$$\binom{(1-w)k}{(1-w)(k/2)}|\mathcal{T}(2^{w(k/2)},[2^{wk}])| = \frac{2^{(1-w)k}}{\sqrt{(1-w)(2k)}} \cdot \frac{2^{wk}}{\sqrt{w(2k)}}$$
$$= \frac{2^k}{\sqrt{(1-w)(2k)}\sqrt{w(2k)}}$$

distinct tilings for $0 < w \leq 1$. There are k such middle terms (or pairs of middle terms) for various settings of w. The denominator is minimized in the case explored in Lemma 13 where in it is equal to $\sqrt{2k}$ and it is maximized in the case of $w = 0.5$ in which case the denominator is equal to k (though this is the only case with such a high denominator). As there are k such cases and we can take their sum, we have a super-linear number of distinct tilings. $\square$

Lemma 15 is essentially a finite example of a more general method for improving upon Lemma 13. As mentioned prior, increasing the number of distinct prime factors in n in the correct manner has the effect of increasing the number of valid tilings. The method for doing this is to take the argument from Lemma 15 and continue to apply it, removing each prime one at a time, steadily multiplying the number of cases of $|\mathcal{T}(2^{k(w/2)},[2^{wk}])|$ for various $0 \leq w \leq 1$. The issue in doing this is that Lemma 14 cannot be applied when their are prime factors other than 2 and 3 and thus, we will require we will require a generalized version of Lemma 14. We prove such a generalization in Lemma 16, the proof of has much more in common with the proof of Lemma 10 then Lemma 15. We now proceed with the proof Lemma 16.

Lemma 16. *Let p_i be the i^{th} prime, and let t, t_{p_m}, t_1, and t_2 be defined as*

$$t \triangleq \mathcal{T}\Big(\alpha, \Big[2^k \textstyle\prod_{i=2}^{m} p_i^2\Big]\Big) \qquad t_1 \triangleq \mathcal{T}\Big(\alpha, \Big[2^{k-1} \textstyle\prod_{i=2}^{m} p_i^2\Big]\Big)$$
$$t_{p_m} \triangleq \mathcal{T}\Big(\tfrac{\alpha}{p_m}, \Big[2^k \textstyle\prod_{i=2}^{m-1} p_i^2\Big]\Big) \qquad t_2 \triangleq \mathcal{T}\Big(\tfrac{\alpha}{2}, \Big[2^{k-1} \textstyle\prod_{i=2}^{m} p_i^2\Big]\Big)$$

where we define the products to be equal to 1 *if* $m = 1$ *and* 0 *if* $m = 0$. *It follows that*

$$|t| \geq |t_{p_m}| + |t_1| + |t_2|.$$

Proof. Throughout this proof, we will use (A_t, B_t) to refer to A and B such that $(A, B) \in t$. We prove the lemma via a combinatorial argument in which we prove that t_{p_m}, t_1 and t_2 have size equal to that of three disjoint subsets of t. More specifically, we define mappings f_{p_m}, f'_1, and f'_2 from t_{p_m}, t_1 and t_2 respectively to t, such that their codomains are disjoint. All three mappings are similar to mappings f_1 or f_2 from the proof of Lemma 10. This is especially true of f'_1 and f'_2, whereas f_{p_m} requires a more complex piece-wise definition. After defining a mapping, we briefly analyze several properties of elements of that mappings codomain. By the fact that the elements of the codomain of each mapping have mutually-exclusive properties, we are able to conclude that the codomains of the mappings are disjoint sets as desired.

For convenience, we briefly review the definitions of f_1 and f_2 from Lemma 10 at a high level. For these functions formal definitions, we refer the reader back to Lemma 10. $f_1(x, y)$ takes a pair as input, where x is a target set of tilings $\mathcal{T}(\alpha, [n])$ and where y is some $(A, B) \in \mathcal{T}(\alpha, [n/v])$ for fixed $v \geq 2$. The output of f_1 is some $(A', B') \in \mathcal{T}(\alpha, [n])$ for which $k_s = 1$. To achieve this, when imagined as a process or algorithm, f_1 essentially "pushes apart" the elements of $[n/v]$ by adding $v-1$ points between each (following a similar process for A), then builds a larger B' from B so as to cover $[n]$ with $A' + B'$. $f_2(x, y)$ takes a pair as input, where x is a target set of tilings $\mathcal{T}(\alpha, [n])$ and where y is some $(A, B) \in \mathcal{T}(\alpha/v, [n/v])$ for fixed $v \geq 2$. The output of f_2 is some $(A', B') \in \mathcal{T}(\alpha, [n])$ for which $k_s > 1$. f_2 is defined in much the same as f_1, except that one adds v contiguous points to A' for each element of A.

We begin by defining f'_1 using the definition of f_1. For $(A_{t_1}, B_{t_1}) \in t_1$, let $f'_1(A_{t_1}, B_{t_1}) \triangleq f_1(t, (A_{t_1}, B_{t_1}))$. By Lemma 10 all elements in the codomain of f_1 are such that $k_{(s,t)} = 1$.

Further, notice that $k_{(r,t)}$ is always odd. To see this, if $k_{(s,t_1)} > 1$, then $k_{(r,t)} = 1$ by a single point being inserted between the first and second points of the first segment. Thus, suppose $k_{(s,t_1)} = 1$. If $k_{(r,t_1)}$ is even, then an odd number of points are added to the first rift to get the first rift of the tiling of t and thus, $k_{(r,t)}$ is odd. Conversely, if $k_{(r,t_1)}$ is odd, then an even number of points is added to the first rift to get the first rift of the tiling of t and thus, $k_{(r,t)}$ is odd. Thus, in all cases, an element of the codomain of f_1 has an odd valued $k_{(r,t)}$.

Next we define f_2' using the definition of f_2 from Lemma 10. Let $f_2'(A_{t_2}, B_{t_2}) = f_2(t, (A_{t_2}, B_{t_2}))$. This gives us a tiling of t with even $k_{(s,t)}$ and $k_{(r,t)}$. The first of these facts follows from the fact that each segment from the tiling from t_2 has had its length exactly doubled. The latter observation then follows from $k_s | (k_s + k_r)$ and the fact that $k_{(s,t)}$ is even. Just as in Lemma 10, as elements of the codomain of f_1 are tilings such that $k_{(s,t)} = 1$ elements of the codomain of f_2 have even $k_{(s,t)}$, we know that the codomains of f_1 and f_2 are disjoint.

We now give a piece-wise definition of f_{p_m}, with one definition when $k_{(s,t_{pm})}$ of the input is even and another when $k_{(s,t_{pm})}$ of the input is odd. For some, $(A_{t_{pm}}, B_{t_{pm}}) \in t_{pm}$, if $k_{(s,t_{pm})}$ is even, we begin by applying $f_1(t, (A_{t_{pm}}, B_{t_{pm}})) = (A', B')$. This increases the size of $A_{t_{pm}} + B_{t_{pm}}$ by a multiplicative factor of p_m and does not change the size of $A_{t_{pm}}$. This would tile exactly the first p_m fraction of points of $A_t + B_t \in t$ (i.e. $A' + B' = [|A_t + B_t|/p_m]$). Thus, we build our final A'' from A' by letting $A'' = A' + \{x \cdot p_m | x \in [0, p_m]\}$. This then gives us our desired output of f_{p_m} on such inputs, which is $(A'', B') = (A_t, B_t)$. As $k_{(s,t_{pm})}$ is even, so is $k_{(r,t_{pm})}$. As the number of points added to the first rift is $p_m - 1$ (which is even), we have that $k_{(r,t)}$ is even. This makes the codomain of f_{p_m} disjoint from the codomain of f_1' in this case. Further, it follows from the portion of the definition based on f_1 that $k_{(s,t)} = 1$. This makes the codomain of f_{p_m} disjoint from the codomain of f_2' in this case.

Now for the case when $k_{(s,t_{pm})}$ of the input to f_{p_m} is odd. For some, $(A_{t_{pm}}, B_{t_{pm}}) \in t_{pm}$, we begin by applying

$$f_2(t, (A_{t_{pm}}, B_{t_{pm}})) = (A^*, B^*).$$

This would tile exactly the first p_m fraction of points of $A_t + B_t$ (i.e. $A^* + B^* = [|A_t + B_t|/p_m]$). Thus, we build our final B^{**} from B^* by letting $B^{**} = (B^* + \{x \cdot p_m | x \in [0, p_m]\}) \cup \{x \cdot p_m | x \in [0, p_m]\}$ to get us our desired output of f_{p_m} on such inputs of $(A^*, B^{**}) = (A_t, B_t)$. As $k_{(s,t_{pm})}$ is odd and we are multiplying the size of the first segment by p_m (which is also odd), we are left with an odd value for $k_{(s,t)}$. This makes the codomain of f_{p_m} disjoint from the codomain of f_2' in this case. Further, it follows from the first part of the mapping that $k_{(s,t)} > 1$. This makes the codomain of f_{p_m} disjoint from the codomain of f_1' in this case. As we have shown the codomains of each function to be disjoint, the lemma follows. □

It is unclear exactly what super-linear lower bound Lemma 15 achieves nor what improvement utilizing Lemma 16 to generalize Lemma 15 enables. We leave this analysis to future work, but conjecture that a lower bound that is linear times some super-polylogarithmic factor is achievable. While one could expand upon the infinite families of values of n for which such lower bounds hold, one cannot prove any such bound for all n. This follows immediately from the fact that, if n is prime, the only tile sizes that tile n are when $\alpha \in \{1, n\}$ and in each case, there is only a single distinct tiling of $[n]$.

CHAPTER 4
NON-CONTIGUOUS TILINGS IN HIGHER DIMENSIONS

4.1 Introduction

In this chapter, we generalize the results of the previous chapter to tiling of $C \subset \mathbb{Z}^d$ and to some families of non-contiguous C. As with the previous chapter, these results originally appeared in [31]. While there is prior work in on tiling the infinite set $\mathbb{Z}^d$ [1, 11, 17, 32], none of these works consider tilings of a finite subset of $\mathbb{Z}^d$. Bodini and Rivals [3] posit that their results generilize to d-dimensions, a claim that our results can be used to verify (while also allowing for the handleing of fixed tile sizes).

We conjecture that the number of tilings of any finite contiguous C by tiles of size α is an upper bound on the number of tilings of any finite $C' \subset \mathbb{Z}^d$ by tiles of size α. In an effort to begin working towards this and other results for non-contiguous C, we prove that any A of size α that tiles some finite contiguous C itself has at most as many tilings by tiles of size α' (for any $\alpha' \in \mathbb{Z}^+$) as there are tilings of $[\alpha]$ by tiles of size α'.

To relate our results back to the turnpike problem and database reconstruction, one would not only need to characterize the tilings of finite non-contiguous $C \subset \mathbb{Z}$, but of tilings of any finite multiset of integers. Despite this, we believe that the true difficulty is a full characterization of tilings of finite non-contiguous $C \subset \mathbb{Z}$, and that the multiset case would follow readily from this in a similar fashion to the generalization of results on tilings of $C \subset \mathbb{Z}$ to tilings of $C \subset \mathbb{Z}^d$.

In this section, we rely upon the definitions of the previous sections, as well as the folliwng definiton of a projection. Let $\text{proj}_i(x)$ be the projection of $x \in \mathbb{Z}^d$ onto the i^{th} dimension (i.e. $\text{proj}_i(x_1, \ldots, x_i, \ldots, x_d) = x_i$). Further, we let $\text{proj}_i(A)$ for a set A equal the set $\{\text{proj}_i(x) | x \in A\}$. We now discuss some of the prior work related to these topics in greater detail.

4.1.1 Generalizations to Higher Dimension and Tiling Non-Contiguous Sets

While our aforementioned results consider more restricted sets of tilings (by allowing for an additional restriction in the form of a fixed tile size), we also work to reduce the number of restrictions on the underlying set to be tiled. We do this by considering tilings in higher dimensions as well as tilings of sets with non-contiguous elements. In their conclusion, Bodini and Rivals [3] claim that many of their results can be extended to tilings of a d-dimensional rectangle and give a brief high-level description as to the technique for doing so. We prove that our results with respect to fixed tile size can be extended to d-dimensions. For the structural results of this form, these proceed in essentially the manner Bodini and Rivals [3] describe (though again, with the additional need to take into account fixed tile size). As one can remove the fixed tile size restriction in these results by summing over all α, these results also confirm the claim of Bodini and Rivals [3]. In addition we note that our upper and lower bounds from the 1-dimensional case still apply in d-dimensions. As to our counting results, their extension to d-dimensions follows readily from the generalization of the structural results to d-dimensions.

As to the tiling of finite non-contiguous C we prove an upper bound on the number of such tilings with a tile of size α for a particular family of such C. More specifically, we prove that any A of size α that tiles $C = [x_1] \times [x_2] \times \dots [x_d]$ has at most as many tilings of size α' (for any $\alpha' \in \mathbb{Z}^+$) as there are tilings of $[\alpha]$ by tiles of size alpha (i.e. $|\mathcal{T}(\alpha', A)| \leq |\mathcal{T}(\alpha', [\alpha])|$).

4.1.2 Prior Work On Tilings of Some Finite Non-Contiguous Multisets

While there does not appear to be prior work on the number of tilings of non-contiguous finite subsets of integers, given the reduction from turnpike to tilings from lemma 4, some research on the Turnpike Problem [27] implicitly studies the number of “tilings" of a restricted family of finite multisets of integers. By the tiling of a multiset, we mean that the elements of the multiset are covered by translates of a given tile a number of times exactly equal to their multiplicity in the multiset. Thus, a count on the number of valid turnpike reconstructions is also a bound on the number of distinct tiles that can tile some highly specific related multiset. As metnioned in chapter 2, Skiena, and Smith, and Lemke [27] give a polynomial[1] upper bound on the number of solutions to the turnpike problem. They do this by transforming the problem into a question about the factorization of related polynomials, then counting the number of irreducible factors that are not self-reciprocal. This is interesting, as this polynomial formulation of the problem has a similar flavor to characterizations of tilings of finite intervals studied in the algebraic approach of Bodini and Rivals [3] and Theorem 2.5 of Pederson and Wang [24]. While we did not use this approach in our method of counting, we leave as an open question whether or not some of the techniques from either body of work can be applied to the other to yield improved results.

4.2 Extending to $\mathbb{Z}^d$

We now extend the above results for finite subsets of $\mathbb{Z}$ to results for $\mathbb{Z}^d$ for arbitrary $d \in \mathbb{Z}^+$. In their conclusion, Bodini and Rivals [3] correctly claim that their results about tiling finite intervals of the discrete line can be extended to $\mathbb{Z}^d$ and briefly mention what the characterization of the tilings in this setting would be. As we work with the additional restriction of fixed tile size we adjust our claim appropriately, but we note that our core idea appears to align witht heir own. Further, our proof verifies their claim, as one can sum over all tile sizes to recover their claim. Once we have extended our structural results to $\mathbb{Z}^d$, we use this to show that our upper and lower bounds from the previous section still hold in $\mathbb{Z}^d$.[2] The extension of our structural results appears as Lemma 18 and proceeds by induction however, the base case of the induction is somewhat non-trivial in that it requires

1. The exact degree of the polynomial bound depends on if the underlying point set is permitted to have points with multiplicity greater than 1, though they handle both cases.

2. The fact that the lower bound still holds follows immediately from the fact that one can set $d = 1$ and use the same case and argument as in Lemma 16.

a fact about the distribution of the elements of A with respect to B for any (A, B) that is valid tiling of $[n]$. For the sake of clarity, we separate out this base case structural result and present it as Lemma 17 before proving Lemma 18.

Lemma 17. *Let $C = [n]$. For $m \in \mathbb{Z}$, either $|A \cap (B+m)| = 1$ or $|C \cap (B+m)| < |B+m|$.*

Proof. We proceed by induction on the $\Omega^*(n)$. For the base case of $\Omega^*(n) = 1$, we have that exactly one of $|A|$ and $|B|$ is n. Assume without loss of generality that $|A| = n$, then $A = C$ and $B + m$ is either in C and has a unique intersection with A or is not in C. Thus, the base case holds. Suppose the lemma holds for $\Omega^*(n) = k - 1$ and we prove that it holds for $\Omega^*(n) = k$. Notice that, as $k > 1$, either A or B has segments of size greater than 1. Suppose without loss of generality that A has segments of size greater than 1. Thus, we can group C into meta-points such that, for each meta-point, the consecutive elements it encompasses are either all in A or all not in A. Further, by the definition of B, its segment size is 1 and its minimum rift size is the segment size of A minus 1. Thus, it follows that each meta-point of C is such that either its first element is in B or it has no elements in B. Let $w > 1$ be the number of points per meta-point and suppose m is such that $|C \cap (B + m)| = |(B + m)|$ (if not, we are done). As meta-points have size greater then 1, we can reconsider the intersection of $|A \cap (B+m)| = 1$ with respect n/w points and $\lfloor m/w \rfloor$ and apply our inductive hypothesis to say that exactly one meta-point of C that is entirely elements of A intersects a meta-point of $B + \lfloor m/w \rfloor$. This gives us an intersection of $A \cap (B + \lfloor m/w \rfloor)$ at the first point of a meta point. If we replace $\lfloor m/w \rfloor$ by m as required, this increases the shift by at most $w - 1$. As the intersection between A and $B + \lfloor m/w \rfloor$ is in the first point of a meta-point and the meta-point in question contains w consecutive elements, $|A \cap (B + m)| = 1$ still holds and the lemma follows. $\square$

We can now prove our main lemma of the section.

Lemma 18. *Let $C = C_1 \times C_2 \times \ldots \times C_d$ for C_i such that $C_i = [n_i]$ for some $n_i \in \mathbb{Z}^+$. We have that*

$$|\mathcal{T}((\alpha_1, \alpha_2, \ldots, \alpha_d), C)| = |\mathcal{T}(\alpha_1, C_1)| \cdot |\mathcal{T}(\alpha_2, C_2)| \cdot \ldots \cdot |\mathcal{T}(\alpha_d, C_d)|.$$

More specifically, (A, B) is an element of valid tilings of $\mathcal{T}((\alpha_1, \alpha_2, \ldots, \alpha_d), C)$ if and only if
$A = \{(x_1, x_2, \ldots, x_d) : x_i \in A_i\}$ and $B = \{(y_1, y_2, \ldots, y_d) : y_i \in B_i\}$ where $(A_i, B_i) \in \mathcal{T}(\alpha_i, C_i)$. Further, for $m \in \mathbb{Z}^d$, either $|A \cap (B + m)| = 1$ or $|C \cap (B + m)| < |B + m|$.

Proof. To tile the elements of $C_1 \times \min[C_2] \times \ldots \times \min[C_d]$, the only elements of A that can be utilized are those in $C_1 \times \min[C_2] \times \ldots \times \min[C_d]$. Thus, the elements of A in $C_1 \times \min[C_2] \times \ldots \times \min[C_d]$ and the elements of B in $\mathbb{Z}^+ \times \{0\} \times \ldots \times \{0\}$ exactly correspond to the one-dimensional tiling of C_1 with $|A| = \alpha_1$.

We prove the lemma via two nested inductive arguments. We begin with induction on the dimension d. For $d = 1$, this follows by definition and Lemma 17. Thus, suppose this is the case for some d and we wish to prove that the lemma still holds for $d + 1$. To do this, we prove by induction on z that, for all $z \in [n_{d+1}]$ there exists a $k \in [z]$ such that, $C_1 \times C_2 \times \ldots \times C_d \times z$ is tiled by

$$(A \cap (C_1 \times C_2 \times \ldots \times C_d \times k)) + \{b \in B : \text{proj}_{d+1} = z - k\}.$$

For the case of $z = 1$ it follows from our inductive hypothesis relative to d that $C_1 \times \ldots \times C_d \times 1$ can only be tilings as described in the lemma. More specifically, one takes the d dimensional tiling, then includes 1 and 0 as the $(d + 1)^{\text{th}}$ element in the tuples of each element of A and B respectively. Now, for our inductive hypothesis relative to z, suppose the claim holds for the subsets of A and B that tile exactly the elements of $C_1 \times C_2 \times \ldots \times C_d \times [z - 1]$ for $z \leq n_{d+1}$. We wish to prove this holds for $C_1 \times C_2 \times \ldots \times C_d \times z$.

Let A' be the elements of A in $C_1 \times C_2 \times \ldots \times C_d \times 1$. As $(1, \ldots, 1, z)$ must be tiled, this must either be because $(1, \ldots, 1, z) \in A$ or $(1, \ldots, 1, z)$ is covered by a translation of an element of A in $C_1 \times C_2 \times \ldots \times C_d \times [z-1]$. First, suppose $(1, \ldots, 1, z) \in A$. Let $B^* \subset B$ be such that

$$A' + B^* = C_1 \times C_2 \times \ldots \times C_d \times 1$$

and let $b' = (y_1, \ldots, y_{d+1})$ for $y_{d+1} \in [z-1]$ be an element of B that translates an element of $A \cap (C_1 \times C_2 \times \ldots \times C_d \times [z-1])$ to $C_1 \times C_2 \times \ldots \times C_d \times z$. For now, let us assume that $y_{d+1} = z - 1$. By our inductive hypothesis on d, we have that either $|A \cap (B+m)| = 1$ or $|C \cap (B+m)| < |B+m|$ for $m \in \mathbb{Z}^d$. Thus, it follows that $|((1, \ldots, 1, z) + B^*) \cap (A' + b')| = 1$ or $|C \cap A' + b'| < |A' + b'|$ which implies that $C_1 \times C_2 \times \ldots \times C_d \times z$ cannot be tiled by elements of A' if it $(1, \ldots, 1, z) \in A$. Notice then that this together with our inductive hypothesis on z implies that, for $k \in [z-1]$, $A \cap (C_1 \times C_2 \times \ldots \times C_d \times k) = \emptyset$, or that $A \cap (C_1 \times C_2 \times \ldots \times C_d \times k) = A'$. This observation then allows us to circle back and inductively remove the restriction that $y_{d+1} = z - 1$ from the prior argument. Taken together, these restrictions turn the case of tiling $C_1 \times C_2 \times \ldots \times C_d \times z$ when $(1, \ldots, 1, z) \in A$ into tiling a d-dimensional space with fixed B^*, at which point we can apply the inductive hypothesis d to conclude that the claim holds.

The argument for the case of $(1, \ldots, 1, z) \notin A$ proceeds similarly, so we merely outline the differences at a high level. To cover $(1, \ldots, 1, z)$, it follows that there exists a b' such that $(A \cap (C_1 \times C_2 \times \ldots \times C_d \times k)) + b' = (A' + (0, \ldots, 0, k-1)) + b'$ covers $(1, \ldots, 1, z)$. Adding any element $a' \in C_1 \times C_2 \times \ldots \times C_d \times z$ to A then creates an issue, as $a' = (1, \ldots, 1, z) + m$ for some m such that $\text{proj}_{d+1}(m) = 0$. At this point we treat $(A \cap (C_1 \times C_2 \times \ldots \times C_d \times k)) + b'$ and $a' + B^*$ as A and B respectively from the statement that, for $m \in \mathbb{Z}^d$, either $|A \cap (B+m)| = 1$ or $|C \cap (B+m)| < |B+m|$. □

Lemma 18 can then be leveraged to prove that $\max\limits_{\alpha, C}[|\mathcal{T}(\alpha, C)|]$ is maximal when $d = 1$

Lemma 19. *Let $C = [x_1] \times [x_2] \times \dots [x_d]$ for $x_1, \dots, x_d \in \mathbb{Z}^+$ and $C' = [n]$ for $n = |C|$. It follows that*

$$\max_{\alpha, C}[|\mathcal{T}(\alpha, C)|] \le \max_{\alpha', C'}[|\mathcal{T}(\alpha', C')|].$$

Proof. Recall that we use $[a]$ for the set of integers 1 through a and $[a, b]$ for the set of integers from a to b (inclusive of a and b). We give a mapping from any $(A, B) \in \mathcal{T}(\alpha, C)$ to a distinct $(A', B') \in \mathcal{T}(\alpha', C')$. We know by Lemma 18 that $\prod_{i \in [d]} |\mathrm{proj}_i(A)| = |A|$. We define A' in terms of A by specifying the elements of

$$A' \cap \Big[\prod_{i \in [k]} x_i\Big]$$

inductively with respect to k with $k \le d$. For $k = 1$, we let $A' \cap [x_1] = \mathrm{proj}_i(A)$. Suppose that the elements of

$$A' \cap \Big[\prod_{i \in [k']} x_i\Big]$$

are defined for all $k' < k$ and that $1 < k$. Then we define the elements of

$$A' \cap \Big[\prod_{i \in [k-1]} x_i, \prod_{i \in [k]} x_i\Big]$$

by

$$\bigcup_{z \in \mathrm{proj}_k(A)} \left(\left(A' \cap \Big[\prod_{i \in k-1} x_i\Big]\right) + (z - 1)\right).$$

We can prove by induction on d that there exists a B' such (A', B'). For $d = 1$, we have that $C = C'$ and $(A, B) = (A', B')$. Suppose the lemma holds up to $d - 1$. Then in the case of d, our definition of A' from A as well as the fact that

$$\prod_{i \in [d]} |\mathrm{proj}_i(A)| = |A|,$$

we have that $|A| = |A'|$. Consider C' condensed into x_d meta-points of size

$$\prod_{i \in [d-1]} x_i$$

each. We know by the inductive hypothesis that there exists a $B^* \subset B'$ such that $A' + B^*$ tiles exactly each meta-point with at least one element of A'. This tiles exactly the meta-points indexed by elements of $\text{proj}_d(A)$. By Lemma 18, $\text{proj}_d(A)$ is such that $(\text{proj}_d(A), B') \in \mathcal{T}(|\text{proj}_d(A)|, [x_i])$ for some B' and thus, the lemma follows. □

Note that one cannot hope to achieve an improvement of Lemma 19 that yields equality for all α, n and d, as evidenced by the following counterexample.

Corollary 8. *Let $C = [x_1] \times [x_2] \times \ldots [x_d]$ for $x_1, \ldots, x_d \in \mathbb{Z}^+$ and $C' = [n]$ for $n = |C|$. $\exists \alpha, n, d \in \mathbb{Z}^+$ such that*

$$\max_{\alpha, C}[|\mathcal{T}(\alpha, C)|] < \max_{\alpha', C'}[|\mathcal{T}(\alpha', C')|].$$

Proof. Let $\alpha = 2$ and $C = [3] \times [2]$. The only valid tiling of C' is

$$(\{(1,1),(2,1)\}, \{(0,0),(0,1),(0,2)\})$$

where as $C' = [6]$ has the tilings $T_1 = (\{1,2\}, \{0,2,4\})$ and $T_2 = (\{1,4\}, \{0,1,2\})$. □

4.3 Tilings for Non-Contiguous C

For all α and n, we hypothesis that $C = [n]$ has at least as many tilings as any α' and C' such that $|C'| = n$. More formally, we claim the following

Conjecture 1.

$$\forall \alpha, n, d \in \mathbb{Z}^+, \forall C \subset \mathbb{Z}^d \Big[(|C| = n) \implies |\mathcal{T}(\alpha, C)| \leq |\mathcal{T}(\alpha, [n])| \Big].$$

While we are unable to resolve this conjecture, we make some progress towards this by proving that Conjecture 1 holds when C can tile $[n]$ for some n. We begin by proving this only for the case of $d = 1$ in Lemma 20, but we briefly show how to extend this to any d in Corollary 11. Before we proceed with Lemma 20, we make two useful structural claims. Corollary 9 clarifies that the first rift in any tiling is the smallest. Corollary 10 is a structural result about valid tilings of $[n]$ that shows the segments only ever appear as the first k_s elements of a meta-point (where meta-points are as defined in Definition 20).

Corollary 9. *For any $(A, B) \in \mathcal{T}(\alpha, [n])$ with j rifts we have that $\forall i \in [1, j](|r_1| \leq r_i)$.*

Proof. Suppose not, then consider $B^* \subset B$ such that B^* is the minimum size subset set of B for which $[1, k_s + k_r] \subset A + B^*$ holds. We know from Lemma 8 that elements in $[1, k_s + k_r]$ must be covered by shifts of elements in s_1 by elements in B^*. Let r_i be such that $|r_i| < |r_1|$, then $s_i + B^* \cap s_{i+1} \neq \emptyset$, which contradicts our assumption that $(A, B) \in \mathcal{T}(\alpha, [n])$. $\square$

Corollary 10. *For all $(A, B) \in \mathcal{T}(\alpha, [n])$, we have that*

$$\exists! M \subset \left[0, \frac{n}{(k_r/k_s) + 1} - 1\right] \left(A = \bigcup_{m \in M} ([1, k_s] + \{m(k_s + k_r)\})\right).$$

Put another way, there is a unique subset of the meta-points of C with respect to A such that the elements of A are the first k_s elements of these meta-points.

Proof. Notice that, by the definition of k_s, it is certainly the case that $[1, k_s] \subset A$. As is shown in the arguments of Lemma 7, the remaining segments of A outside of s_1 must begin at intervals of $k_s + k_r$. The upper bound on the interval of which M is a subset then comes from the fact that an element of A beyond this point would be translated by an element of B to a value greater than n, making the tiling invalid. $\square$

With this, we can now proceed with proving the main lemma of the section.

Lemma 20. *For any $\alpha, \alpha', n \in \mathbb{Z}^+$ and any $(A, B) \in \mathcal{T}(\alpha, [n])$, we have that $|\mathcal{T}(\alpha', A)| \leq |\mathcal{T}(\alpha', [\alpha])|$.*

Proof. We prove the lemma via the following steps:

1. We define a compression function f for taking certain well structured sets of integers and outputting a new set of integers with larger groups of contiguous elements (i.e. larger segment size).

2. We define an injective function g_1 that maps any $(A, B) \in \mathcal{T}(\alpha, [n])$ such that A has at least one rift, to some $(A', B') \in \mathcal{T}(\alpha, [n/(k_r/k_s) + 1)])$.

3. We define an injective function g_2 that maps any $(\mathcal{A}, \mathcal{B}) \in \mathcal{T}(\alpha', A)$ to some $(\mathcal{A}', \mathcal{B}') \in \mathcal{T}(\alpha', [A'])$.

4. We an injective function g_3 thats maps any $(\mathcal{A}, \mathcal{B}) \in (\mathcal{T}(\alpha', A)$ to an element of $\mathcal{T}(\alpha', [\alpha])$, where g_3 simply applies g_2 to $(\mathcal{A}, \mathcal{B})$ a number of times based upon $g(A, B)$ for $(A, B) \in \mathcal{T}(\alpha, [n])$.

Define the function $f(A, C)$, where there exists a B such that $(A, B) \in \mathcal{T}(\alpha, C)$, to be such that

$$f(A, C) = \{a - k_s \lfloor a/(k_s + k_r) \rfloor : a \in A)|\}.$$

Note that, while C is not explicitly mentioned on the right hand side of the equality, it is implicit by the use k_s and k_r and thus, C is in fact required in the input for the function to be well defined. Let C be a finite contiguous set of integers. By corollary 9, the intuition as to the effect of f is that it takes the first k_s elements of A from each meta-point of C with respect to A and makes them contiguous by removing elements besides the first k_s of each meta-point, followed by "shifting" those remaining appropriately (which formalize in a moment).

Before defining g_1 and g_2, we define $\tilde{B}$ to be such that

$$\tilde{B} = \Big\{ b_i \in B : i = 1 \bmod \frac{k_r}{k_s} + 1 \Big\}.$$

By corollary 10, B is defined in such a way that the elements of $A + \tilde{B}$ cover exactly the first k_s elements of each meta-point of $[n]$ with respect to A. More formally, we have that

$$(A, \tilde{B}) \in \mathcal{T}\Bigg(\alpha, \bigcup_{m \in [0, \frac{n}{(k_r/k_s)+1} - 1]} ([1, k_s] + \{m(k_s + k_r)\}) \Bigg).$$

We can now define both g_1 and g_2. Let $g_1(A, B) = (f(A, [n]), f(\tilde{B}, [0, n-1]) = (A', B')$. By Corollary 10, this transformation is such that $|A'| = |A|$ and $|B'| = |\tilde{B}| = k_s|B|/k_r$. This compression of the element of both A and $\tilde{B}$ leads to A' and B' such that $(A', B') \in \mathcal{T}(\alpha, [n/(k_r/k_s) + 1)])$. Similarly we define $g'(\mathcal{A}, \mathcal{B}) = (f(\mathcal{A}, [n]), f(\mathcal{B}, [0, n-1]) = (\mathcal{A}', \mathcal{B}')$. In the case of g_2, we now have that $|\mathcal{A}'| = |\mathcal{A}|$ and $|\mathcal{B}'| = |\mathcal{B}|$. Further, We have that $(\mathcal{A}', \mathcal{B}') \in \mathcal{T}(\alpha', A')$ where $\alpha' = |\mathcal{A}|$.

We now define g_3 from g_2, using g_1 to show its correctness. Suppose A has ℓ distinct lengths of rift. Certainly, ℓ is finite and less than or equal to the total number of rifts in A. Notice though that applying g_1 to (A, B) results in an (A', B') with one less distinct rift length. Thus, it follows that $g_1^{\ell}(A, B) = ([\alpha], \{0\})$. Knowing this, observe that we can then apply g_2 the same number of times (i.e. ℓ times) to $(\mathcal{A}, \mathcal{B}) \in \mathcal{T}(\alpha', A)$, giving us $g_2^{\ell}(\mathcal{A}, \mathcal{B}) = (\mathcal{A}', \mathcal{B}')$ such that $\mathcal{A}' + \mathcal{B}' = [\alpha]$. Thus, we define g_3 to be $g_2^{\ell}(\mathcal{A}, \mathcal{B})$ where ℓ is the number of rifts in $\mathcal{A} + \mathcal{B}$. As g_2 is injective, it follows that distinct tilings $(\mathcal{A}, \mathcal{B}) \in \mathcal{T}(\alpha', A)$ map to distinct elements of $\mathcal{T}(\alpha', [\alpha])$ by g_3. Thus, the lemma follows. □

We now generalize Lemma 20 to arbitrary dimension.

Corollary 11. *Let $C = C_1 \times C_2 \times \ldots \times C_d$ for C_i such that $C_i = [n_i]$ for some $n_i \in \mathbb{Z}^+$.*

For all $(A, B) \in |\mathcal{T}((\alpha_1, \alpha_2, \ldots, \alpha_d), C)|$ *and any set of* $\alpha'_i \in \mathbb{Z}^+$ *for all* i*, we have that*

$$|\mathcal{T}((\alpha'_1, \alpha'_2, \ldots, \alpha'_d), A)| \leq |\mathcal{T}((\alpha'_1, \alpha'_2, \ldots, \alpha'_d), [\alpha_1] \times [\alpha_2] \times \ldots \times [\alpha_d])|.$$

Proof. Applying the function g_3 as in Lemma 20 to each dimension one at a time injectivley maps any element of $\mathcal{T}((\alpha'_1, \alpha'_2, \ldots, \alpha'_d), A)$ to an element of $\mathcal{T}((\alpha'_1, \alpha'_2, \ldots, \alpha'_d), [\alpha_1] \times [\alpha_2] \times \ldots \times [\alpha_d])$. $\square$

REFERENCES

[1] I. Benjamini, G. Kozma, and E. Tzalik. The number of tiles of $\mathbb{Z}^d$. *arXiv preprint arXiv:2303.07956*, 2023.

[2] L. Blackstone, S. Kamara, and T. Moataz. Revisiting leakage abuse attacks. In *ISOC Network and Distributed System Security Symposium – NDSS 2020*, San Diego, CA, USA, Feb. 23–26, 2020. The Internet Society.

[3] O. Bodini and E. Rivals. Tiling an interval of the discrete line. In *Annual Symposium on Combinatorial Pattern Matching*, pages 117–128. Springer, 2006.

[4] W. Chang. personal communication with steven skiena and gopalakrishnan sundaram.

[5] M. Cieliebak and S. Eidenbenz. Measurement errors make the partial digest problem np-hard. In *Latin American Symposium on Theoretical Informatics*, pages 379–390. Springer, 2004.

[6] M. Cieliebak, S. Eidenbenz, and P. Penna. Partial digest is hard to solve for erroneous input data. *Theoretical computer science*, 349(3):361–381, 2005.

[7] E. M. Coven and A. Meyerowitz. Tiling the integers with translates of one finite set. *Journal of Algebra*, 212(1):161–174, 1999.

[8] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis. Practical private range search revisited. In *Proceedings of the 2016 International Conference on Management of Data*, pages 185–198, 2016.

[9] F. Falzon, E. A. Markatou, Akshima, D. Cash, A. Rivkin, J. Stern, and R. Tamassia. Full database reconstruction in two dimensions. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 443–460, Virtual Event, USA, Nov. 9–13, 2020. ACM Press.

[10] F. Falzon, E. A. Markatou, Z. Espiritu, and R. Tamassia. Attacks on encrypted range search schemes in multiple dimensions. Cryptology ePrint Archive, Report 2022/090, 2022. `https://eprint.iacr.org/2022/090`.

[11] R. Greenfeld and T. Tao. The structure of translational tilings in $\mathbb{Z}^d$. *Discrete Analysis*, 2020.

[12] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 315–331, Toronto, ON, Canada, Oct. 15–19, 2018. ACM Press.

[13] Z. Gui, O. Johnson, and B. Warinschi. Encrypted databases: New volume attacks against range queries. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 361–378, London, UK, Nov. 11–15, 2019. ACM Press.

[14] G. H. Hardy, E. M. Wright, et al. *An introduction to the theory of numbers.* Oxford University Press, 1979.

[15] J. Karhumaki, Y. Lifshits, and W. Rytter. Tiling periodicity. *Discrete Mathematics & Theoretical Computer Science*, 12, 2010.

[16] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic attacks on secure outsourced databases. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1329–1340, Vienna, Austria, Oct. 24–28, 2016. ACM Press.

[17] M. N. Kolountzakis and M. Matolcsi. Algorithms for translational tiling. *Journal of Mathematics and Music*, 3(2):85–97, 2009.

[18] E. M. Kornaropoulos, N. Moyer, C. Papamanthou, and A. Psomas. Leakage inversion: Towards quantifying privacy in searchable encryption. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 1829–1842, Los Angeles, CA, USA, Nov. 7–11, 2022. ACM Press.

[19] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia. Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks. In *2021 IEEE Symposium on Security and Privacy*, pages 1502–1519, San Francisco, CA, USA, May 24–27, 2021. IEEE Computer Society Press.

[20] J. C. Lagarias and Y. Wang. Tiling the line with translates of one tile. *Inventiones Mathematicae*, 124(1):341–365, 1996.

[21] P. Lemke and M. Werman. On the complexity of inverting the autocorrelation function of a finite integer sequence, and the problem of locating n points on a line, given the (n atop 2t) unlabelled distances between them. 1988.

[22] A. L. Patterson. A fourier series method for the determination of the components of interatomic distances in crystals. *Physical Review*, 46(5):372, 1934.

[23] A. L. Patterson. A direct method for the determination of the components of interatomic distances in crystals. *Zeitschrift für Kristallographie-Crystalline Materials*, 90(1-6):517–542, 1935.

[24] S. Pedersen and Y. Wang. Universal spectra, universal tiling sets and the spectral set conjecture. *Mathematica Scandinavica*, pages 246–256, 2001.

[25] E. Rivals. Counting the tiles of an interval of the discrete line. 2007.

[26] J. Rosenblatt and P. D. Seymour. The structure of homometric sets. *SIAM Journal on Algebraic Discrete Methods*, 3(3):343–350, 1982.

[27] S. S. Skiena, W. D. Smith, and P. Lemke. Reconstructing sets from interpoint distances. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 332–339, 1990.

[28] S. S. Skiena and G. Sundaram. A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology*, 56:275–294, 1994.

[29] N. Sloane. The on-line encyclopedia of integer sequences, 1964.

[30] M. Stefik. Inferring dna structures from segmentation data. *Artificial Intelligence*, 11(1-2):85–114, 1978.

[31] J. Stern. On the number of distinct tilings of finite subsets of $\mathbb{Z}^d$. *arXiv preprint arXiv:2303.06717*, 2023.

[32] M. Szegedy. Algorithms to tile the infinite grid with finite clusters. In *Proceedings 39th Annual Symposium on Foundations of Computer Science*, pages 137–145. IEEE, 1998.

[33] R. Tijdeman. *Decomposition of the integers as a direct sum of two subsets*, page 261–276. London Mathematical Society Lecture Note Series. Cambridge University Press, 1995.

[34] S. Wang, R. Poddar, J. Lu, and R. A. Popa. Practical volume-based attacks on encrypted databases. Cryptology ePrint Archive, Report 2019/1224, 2019. `https://eprint.iacr.org/2019/1224`.

[35] Z. Zhang. An exponential example for a partial digest mapping algorithm. *Journal of Computational Biology*, 1(3):235–239, 1994.

APPENDIX

Algorithm 1 Turnpike via Backtracking

```
TurnpikeBT(Vols):
Input: The multiset Vols.
Output: A database $DB'$ such that, if Vols is the multiset volume leakage of some database $DB$, then TurnpikeBT is a successful multiset volume-based reconstruction attack. Otherwise, reject.
 1: Run Intialize(Vols)
 2: if Intialize(Vols) rejects then
 3:     Halt and reject
 4: path := ()
 5: $d := 3$
 6: while $d \leq N$ do
 7:     Run Right($B, d$, Vols)
 8:     if Right($B, d$, Vols) outputs "backtrack" then
 9:         while Last element of path is $L$ do
10:             pop(path)
11:             Return elements from $V_d$ to Vols
12:             Set corresponding elements of $B$ to $-1$
13:             $d--$
14:             if $d = 2$ then
15:                 reject
16:             while Last element of path is $R$ do
17:                 Run Left($B, d$, Vols)
18:                 if Left($B, d$, Vols) outputs "backtrack" then
19:                     pop(path)
20:                     Return elements from $V_d$ to Vols
21:                     Set corresponding elements of $B$ to $-1$
22:                     $d--$
23:                     if $d = 2$ then
24:                         reject
25:         Append $L$ to path
26:         Update $B$, $V_d$, and Vols according to Left($B, d$, Vols)
27:     else
28:         Append $R$ to path
29:         Update $B$, $V_d$, and Vols according to Right($B, d$, Vols)
```

Figure A.1: Standard Backtracking Algorithm for the Turnpike problem [27].

Algorithm 2 Backtracking Initialization

Intialize(Vols):

Input: The multiset Vols.

Output: The multiset Vols and initializations for B and N. If Vols is of the wrong size or initial volume placements fail **reject**.

```
1: $N := \lceil \sqrt{2|\mathsf{Vols}|} \rceil$
2: **if** $|\mathsf{Vols}| \neq \binom{N}{2} + N$ **then**
3:     **reject**
4: $i := 2$
5: $j := 1$
6: **while** $i \leq N$ **do**
7:     **while** $j \leq i$ **do**
8:         $(i, j)_B := -1$
9:         $j++$
10:    $j := 1$
11:    $i++$
12: $(1,1)_B = \max[\mathsf{Vols}]$
13: $\mathsf{Vols} := \mathsf{Vols} \setminus \{\max[\mathsf{Vols}]\}$
14: $(2,2)_B = \max(\mathsf{Vols})$
15: $\mathsf{Vols} := \mathsf{Vols} \setminus \{\max[\mathsf{Vols}]\}$
16: **if** $(1,1)_B - (2,2)_B \in \mathsf{Vols}$ **then**
17:     $(n,1)_B := (1,1)_B - (2,2)_B$
18:     $\mathsf{Vols} := \mathsf{Vols} \setminus \{(1,1)_B - (2,2)_B\}$
19: **else**
20:     Halt and **reject**
```

Figure A.2: Turnpike Backtracking Algorithm Initialization Procedure

Algorithm 3 Adding Volume to the Right

```
Right($B, d, \mathsf{Vols}$):
Input: The backtracking pyramid $B$, depth $d$, and set of remaining volumes $\mathsf{Vols}$.
Output: $(B, V_d, \mathsf{Vols})$ or backtrack if this path was incorrect.
1: $i \coloneqq \min[\{k : (k,k)_B = -1\}]$
2: $(i,i)_B \coloneqq \max[\mathsf{Vols}]$
3: $V_d \coloneqq \max[\mathsf{Vols}]$
4: $j \coloneqq i$
5: $i - -$
6: while $2 \geq i$ do
7:     if $(i,i)_B - (j,j)_B \in \mathsf{Vols}$ then
8:         $(n+i+1-j, i)_B \coloneqq (i,i)_B - (j,j)_B$
9:         $V_d \coloneqq V_d \cup \{(i,i)_B - (j,j)_B\}$
10:        $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{(i,i)_B - (j,j)_B\}$
11:        $i - -$
12:    else
13:        backtrack
```

Figure A.3: Right Branch Procedure of Turnpike Backtracking Algorithm

Algorithm 4 Adding Volume to the Right

```
Left($B, d,$ Vols):
Input: The backtracking pyramid $B$, depth $d$, and set of remaining volumes Vols.
Output: ($B, V_d,$ Vols) or backtrack if this path was incorrect.
1: $i \coloneqq \min[\{k : (k, 1)_B = -1\}]$
2: $(i, 1)_B \coloneqq \max[\mathsf{Vols}]$
3: $V_d \coloneqq \max[\mathsf{Vols}]$
4: $j \coloneqq i$
5: $i - -$
6: while $2 \geq i$ do
7:     if $(i, 1)_B - (j, 1)_B \in \mathsf{Vols}$ then
8:         $(n + i + 1 - j, n + i + 1 - j)_B \coloneqq (i, 1)_B - (j, 1)_B$
9:         $V_d \coloneqq V_d \cup \{(i, 1)_B - (j, 1)_B\}$
10:        $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{(i, 1)_B - (j, 1)_B\}$
11:        $i - -$
12:    else
13:        backtrack
```

Figure A.4: Left Branch Procedure of Turnpike Backtracking Algorithm

Algorithm 5 Noisy Reconstruction from Volume Multiset

```
NoisyReconstruction(Vols, $[a, b]$):
Input: The multiset Vols.
Output: A backtracking pyramid $B$ if Vols is constant with the noisy multiset volume leakage of some database $DB$. Otherwise, reject.
 1: $N \coloneqq \lceil \sqrt{2|\mathsf{Vols}|} \rceil$
 2: if $|\mathsf{Vols}| \neq \binom{N}{2} + N$ then
 3:     reject
 4: $i \coloneqq 2$
 5: $j \coloneqq 1$
 6: while $i \leq N$ do
 7:     while $j \leq i$ do
 8:         $(i, j)_B \coloneqq -1$
 9:         $j++$
10:     $j \coloneqq 1$
11:     $i++$
12: Run Setup(Vols, $[a, b]$)
13: $(1, 1)_B = \max[V]$
14: $V \coloneqq V \setminus \{\max[V]\}$
15: path $\coloneqq ()$
16: $d \coloneqq 2$
```

Figure A.5: Algorithm for database reconstruction from noisy volume multiset leakage. Part 1 of 2.

```
17: while $d \leq N$ do
18:     $a \coloneqq \min[\{k : (k,k)_B = -1\}]$
19:     Set $i$ to be the bottom row of $I$
20:     Set $j$ to be the max integer such that $(i,j)_I$ is not restricted
21:     Run $IBT([x,y]_{(i,j)}, [a,a], B, I)$
22:     if $IBT([x,y]_{(i,j)}, [a,a], , B, I)$ outputs “Backtrack" when running $R$ then
23:         while Last element of path is $L$ do
24:             pop(path)
25:             Undo one execution of $IBT$
26:             $d--$
27:             if $d = 2$ then
28:                 reject
29:             while Last element of path is $R$ do
30:                 $a \coloneqq \min[\{k : (k,1)_B = -1\}]$
31:                 Set $i$ to be the bottom row of $I$
32:                 Set $j$ to be the max integer such that $(i,j)_I$ is not restricted
33:                 Run $IBT([x,y]_{(i,j)}, [a,1], B, I)$
34:                 if $IBT(\max[V], [a,1], B, I)$ outputs “backtrack" when running $R$ then
35:                     pop(path)
36:                     Undo one execution of $IBT$
37:                     $d--$
38:                     if $d = 2$ then
39:                         reject
40:         Append $L$ to path
41:     else
42:         Append $R$ to path
43: Output $(B, I)$
```

Figure A.6: Algorithm for database reconstruction from noisy volume multiset leakage. Part 2 of 2.

Algorithm 6 Setup

```
Setup(Vols, $[a, b]$):
Input: Noisy multiset volume leakage Vols and an interval $[a, b]$ from which the additive noise is drawn.
Output: A multiset of volume intervals $V$ and an interval pyramid $I$ of 4-tuples.
1: $V := \emptyset$
2: Initialize $I$ as n empty pyramid with $M$ rows
3: for all $\{x \in [|\mathsf{Vols}|]\}$ do
4:     $V := V \cup [\min[0, x + a], \min[0, x + b]]$
5: Let $u$ equal the number of unique elements in $V$
6: for all $j \leq u$ do
7:     for all $i \leq j$ do
8:         Set $x$ equal to $j^{\text{th}}$ smallest min element of an interval in $V$
9:         Set $y$ equal to $i^{\text{th}}$ largest max element of an interval in $V$
10:        Set $z$ equal to number of intervals in $V$ that are sub-intervals of $[x, y]$
11:        Set $(i, j)_I$ (i.e. row $i$ column $j$ of $I$) to $([x, y], z, 0, \{\{\}\})$
12: Output $I$
```

Figure A.7: Setup procedure for noisy database reconstruction algorithm.

Algorithm 7 Backtracking

<u>$IBT([x,y],[a,b],B,I)$:</u>

Input: Assigned volume interval $[x,y]$, the position $[a,b]_B$ to assign it to, the backtracking pyramid B, and the interval pyramid I.

Output: Updated B and I based on assignment and associated implications.

```
1:  $(i,j) \coloneqq \min\{i,j \in [N] | [x,y] \subset [x',y']_{(i,j)} \wedge \text{Restricted}((i,j)_I) = 0\}$    ▷ Find correct bucket
2:  $(a,b)_B \coloneqq [x,y]$
3:  $(i,j)_I \coloneqq (i,j)_I \cup [x,y]$    ▷ Add to bucket
4:  Run $C((i,j))$    ▷ Update counters
5:  if $b = 1$ then    ▷ Implication if placed on left
6:      $(N+2-a, N+2-a)_B \coloneqq \text{IntAbs}((1,1)_B, (a,1)_B)$
7:      if There is a unique $(i,j)$ such that $(N+2-a, N+2-a)_B \subset [x',y']_{(i,j)} \wedge \text{Restricted}((i,j)_I) = 0$ and $i$ is maximized then
8:          Set $(i,j)$ to this value
9:      else
10:         Set $h$ to height of $I$
11:         $i \coloneqq \min\{i \in [N-1] | (N+2-a, N+2-a)_B \subset [x',y']_{(h,i)} \wedge \text{Restricted}((h,i)_I) = 0\}$
12:         $j \coloneqq \max\{i \in [N-1] | (N+2-a, N+2-a)_B \subset [x',y']_{(h,j)} \wedge \text{Restricted}((h,j)_I) = 0\}$
13:     $(i,j)_I \coloneqq (i,j)_I \cup (N+2-a, N+2-a)_B$
14:     Run $C((i,j), B, I)$
15:     for all $(a',1) \in \{(a',1) | (a' \neq a) \wedge ((a',1)_B \neq \emptyset)\}$ do    ▷ Compute implications
16:         $(N+1-|a-a'|, N+2-\max\{a,a'\})_B \coloneqq \text{IntAbs}((a,1)_B, (a',1)_B) \cap \text{IntAbs}((N+2-\max\{a,a'\}, N+2-\max\{a,a'\})_B, (N+2-\min\{a,a'\}, N+2-\min\{a,a'\})_B)$
```

Figure A.8: Core interval backtracking procedure for noisy database reconstruction algorithm. Part 1 of 2.

```
17: if $b = a$ then                                  $\triangleright$ Implication if place on right. Works just as left
18:     $(N+2-a,1)_B \coloneqq \text{IntAbs}((1,1)_B,(a,a)_B)$
19:     if There is a unique $(i,j)$ such that $(N+2-a,1)_B \subset [x',y']_{(i,j)} \wedge \text{Restricted}((i,j)_I) = 0$ and $i$ is maximized then
20:         Set $(i,j)$ to this value
21:     else
22:         Set $h$ to height of $I$
23:         $i \coloneqq \min\{i \in [N-1] | (N+2-a,1)_B \subset [x',y']_{(h,i)} \wedge \text{Restricted}((h,i)_I) = 0\}$
24:         $j \coloneqq \max\{i \in [N-1] | (N+2-a,1)_B \subset [x',y']_{(h,j)} \wedge \text{Restricted}((h,j)_I) = 0\}$
25:     $(i,j)_I \coloneqq (i,j)_I \cup (N+2-a,1)_B$
26:     Run $C((i,j),B,I)$
27:     for all $(a',a') \in \{(a',a') | (a' \neq a) \wedge ((a',a')_B \neq \emptyset)\}$ do
28:         $(N+1-|a-a'|,\max\{a,a'\})_B \coloneqq \text{IntAbs}((a,a)_B,(a',a')_B) \cap \text{IntAbs}((N+2-\max\{a,a'\},1)_B,(N+2-\min\{a,a'\},1)_B)$
```

Figure A.9: Core interval backtracking procedure for noisy database reconstruction algorithm. Part 2 of 2.

Algorithm 8 Counter Update

$C((i,j),B,I)$:

Input: The index of I such that a volume interval was just added to the bucket $(i,j)_I$. The backtracking pyramid B and the interval pyramid I.

Output: I (with updated counters and restrictions of its elements).

```
1: $Z := \emptyset$
2: for all $\{i' | 0 \leq i' \leq i\}$ do                ▷ Reduce each superset's counter by 1
3:     for all $\{j' | \max\{1, j + i' - i\} \leq j' \leq \min\{i',j\}\}$ do
4:         if $(i',j')_c > 0$ then
5:             $(i',j')_c - -$
6:             if $(i',j')_c = 0$ then
7:                 $Z := Z \cup (i',j')$
8:         else
9:             Backtrack
10: if $Z \neq \emptyset$ then
11:     Run $R(I,Z)$
12:     Update $I$ based on execution of $R(I,Z)$
```

Figure A.10: Main counter update sub-routine for noisy database reconstruction algorithm. Used when placing intervals into the backtracking pyramid.

Algorithm 9 Counter Update Bucket Move

$C'((i,j),(i',j'),B,I)$:

Input: The index (i,j) such that a volume interval was just moved to $(i,j)_I$ during a restriction update. The index (i',j') of the such that this volume was just removed from $(i',j')_I$. The backtracking pyramid B and the interval pyramid I.

Output: An update to counters in I.

```
1: $Z \coloneqq \emptyset$
2: **if** $j = j'$ **then**                                        ▷ If new bucket is left of original
3:     **for all** $\{z | i' + 2 - j' \leq z < i\}$ **do**          ▷ Reduce each superset of $(i,j)$ counter by 1
4:         **for all** $\{w | i' - j' < z - w \leq i' - j'\}$ **do**           ▷ Ignore supersets of $(i',j')$
5:             **if** $(z,w)_c > 0$ **then**
6:                 $(z,w)_c - -$
7:                 **if** $(z,w)_c > 0$ **then**
8:                     $Z \coloneqq Z \cup (z,w)$
9:             **else**
10:                Backtrack
11: **else**                                                    ▷ If new bucket is right of original
12:     **for all** $\{z | j' + 1 \leq z < i\}$ **do**              ▷ Reduce each superset of $(i,j)$ counter by 1
13:         **for all** $\{w | j' < w \leq j'\}$ **do**                          ▷ Ignore supersets of $(i',j')$
14:             **if** $(z,w)_c > 0$ **then**
15:                 $(z,w)_c - -$
16:                 **if** $(z,w)_c > 0$ **then**
17:                     $Z \coloneqq Z \cup (z,w)$
18:             **else**
19:                 Backtrack
20: **if** $Z \neq \emptyset$ **then**
21:     $R(Z,B,I)$
```

Figure A.11: Counter update sub-routine for noisy database reconstruction algorithm. Used when updating intervals.

Algorithm 10 Restriction Update

$\underline{R(Z, B, I)}$:

Input: Set of buckets in I to restrict Z, the backtracking pyramid B, and the interval pyramid I.

Output: An update to I with restrictions to buckets, changes to their elements and associated updates to counters, as well as the backtracking pyramid B, and the interval pyramid I.

```
1: $\ell := \min[\bigcup_{j\in[|V'|]}\{(|V'|, j) | (|V'|, j)_c \neq 0\}]$
2: $u := \max[\bigcup_{j\in[|V'|]}\{(|V'|, j) | (|V'|, j)_c \neq 0\}]$
3: for all $(i, j) \in Z$ do
4:     $k := 0$
5:     while $k < j$ do
6:         $\text{Res}\big((|V'| - k, j - k)_I\big) := 1$                    ▷ Restrict all up and left
7:         if $(|V'| - k, j - k)_I \neq \emptyset$ then
8:             $z := 1$
9:             while $z \leq k$ do
10:                if $\text{Res}\big((|V'| - k + z, j - k)_I\big) := 0$ then    ▷ Find valid new bucket
11:                    for all $[x, y] \in (|V'| - k, j - k)_I$ do
12:                        $(|V'| - k + z, j - k)_I := (|V'| - k + z, j - k)_I \cup \{[x, y] \cap [x', y']_{(|V'|-k+z,j-k)}\}$    ▷ Move elements down left
13:                        $C'((|V'| - k, j - k), (|V'| - k + z, j - k), B, I)$    ▷ Adjust new bucket's counter
14:                        Run Optimize$(B, I, (a, b))$ (where $(a, b) \in B$ pointed to by $[x, y]$)
15:                    $(|V'| - k, j - k)_I := \emptyset$
16:                    Break While
17:                else if $z = k$ then
18:                    Backtrack
19:                else
20:                    $z{+}{+}$
21:        $k{+}{+}$
22:    $k := 0$
```

Figure A.12: Sub-routine for managing restriction of elements in the interval pyramid of noisy database reconstruction algorithm. Part 1 of 2.

```
23:    while $k < |V'| - j$ do
24:        $\text{Res}\big((|V'| - k, j)_I\big) \coloneqq 1$                    ▷ Restrict all up and right
25:        if $(|V'| - k, j)_I \neq \emptyset$ then
26:            $z \coloneqq 1$
27:            while $z \leq k$ do
28:                if $\text{Res}\big(|V'| - k + z, j + z\big)_I) \coloneqq 0$ then        ▷ Find valid new bucket
29:                    for all $[x, y] \in (|V'| - k, j)_I$ do
30:                        $(|V'| - k + z, j + z)_I \coloneqq (|V'| - k + z, j + z)_I \cup \{[x, y] \cap [x', y']_{(|V'|-k+z,j+z)}\}$    ▷ Move elements down right
31:                        $C'((|V'| - k, j), (|V'| - k + z, j + z), B, I)$    ▷ Adjust new bucket's counter
32:                        Run Optimize$(a, b)$ (where $(a, b) \in B$ pointed to by $[x, y]$)
33:                    $(|V'| - k, j + z)_I \coloneqq \emptyset$
34:                    Break While
35:                else if $z = k$ then                ▷ No valid bucket for element move
36:                    Backtrack
37:                else
38:                    $z + +$
39:        $k + +$
```

Figure A.13: Sub-routine for managing restriction of elements in the interval pyramid of noisy database reconstruction algorithm. Part 2 of 2.

Algorithm 11 Interval Optimization

```
Optimize($B, I, (a, b)$):
Input: The backtracking pyramid $B$, interval pyramid $I$, and position $(a, b)$ in backtracking pyramid $B$ to be updated.
Output: Updated $B$ and $I$ based on assignment and associated implications.
1: for all $(a', b') \in \{(a+k, b+k) | k \in [1-b, N-a] \setminus \{0\}\}$ do ▷ Updates along one diagonal
2:     $(N+1-|a-a'|, \max\{a, a'\})_B := (N+1-|a-a'|, \max\{a, a'\})_B \cap \mathrm{IntAbs}((a, b)_B, (a', b')_B)$
3:     if $(N+1-|a-a'|, \max\{a, a'\})_B \neq \mathrm{IntAbs}((a, b)_B, (a', b')_B)$ then
4:         Let $(i', j')_I$ be the bucket of $(N+1-|a-a'|, \max\{a, a'\})_B$
5:         if There is a unique $(i, j)$ s.t. $(N+1-|a-a'|, \max\{a, a'\})_B \subset [x, y]_{i,j}$ and $\mathrm{Restricted}((i, j)_I) = 0$ that maximizes $i$ then
6:             Set $(i, j)$ to this value
7:         else
8:             Set $h$ to height of $I$
9:             $i := \min\{i \in [N] | ((N, k)_B \subset [x', y']_{(h,i)}) \wedge \mathrm{Restricted}((h, i)_I) = 0\}$
10:            $j := \max\{j \in [N] | ((N, k)_B \subset [x', y']_{(h,j)}) \wedge \mathrm{Restricted}((h, j)_I) = 0\}$
11:            Run $C'((i, j), (i', j'))$
12:        Run Optimize$(B, I, (N+1-|a-a'|, \max\{a, a'\}))$
13: for all $(a', b) \in \{(a+k, b) | k \in [b-a, N-a] \setminus \{0\}\}$ do ▷ Updates along other diagonal
14:     $(N+1-|a-a'|, b+|\max\{a, a'\}-(N+1-|a-a'|)|)_B := (N+1-|a-a'|, b+|\max\{a, a'\}-(N+1-|a-a'|)|)_B \cap \mathrm{IntAbs}((a, b)_B, (a', b)_B)$
15:     if $(N+1-|a-a'|, b+|\max\{a, a'\}-(N+1-|a-a'|)|)_B \neq \mathrm{IntAbs}((a, b)_B, (a', b)_B)$ then
16:         Let $(i', j')_I$ be the bucket of $(N+1-|a-a'|, b+|\max\{a, a'\}-(N+1-|a-a'|)|)_B$
17:         if There is a unique $(i, j)$ s.t. $(N+1-|a-a'|, b+|\max\{a, a'\}-(N+1-|a-a'|)|)_B \subset [x, y]_{i,j}$ and $\mathrm{Restricted}((i, j)_I) = 0$ that maximizes $i$ then
18:             Set $(i, j)$ to this value
19:         else
20:             Set $h$ to height of $I$
21:             $i := \min\{i \in [N] | ((N, k)_B \subset [x', y']_{(h,i)}) \wedge \mathrm{Restricted}((h, i)_I) = 0\}$
22:             $j := \max\{j \in [N] | ((N, k)_B \subset [x', y']_{(h,j)}) \wedge \mathrm{Restricted}((h, j)_I) = 0\}$
23:             Run $C'((i, j), (i', j'))$
24:         Run Optimize$(B, I, (N+1-|a-a'|, b+|\max\{a, a'\}-(N+1-|a-a'|)|))$
25: Output $B$ and $I$
```

Figure A.14: Intervals optimization Sub-routine of noisy database reconstruction algorithm.

Algorithm 12 Noisy DB' Guess

```
DBguess($B, I$):
Input: A backtracking pyramid $B$ and interval pyramid $I$ output by NoisyReconstruction(Vols, $[a, b]$).
Output: A database $DB' = (v'_1, \dots, v'_N)$.
1:  $j := 1$
2:  while $k \leq N$ do
3:      if $|[x, y]_{(N,k)_B}| > 1$ then
4:          Set $(N, k)_B$ to be a uniformly random element in $[x, y]_{(N,k)_B}$
5:          if There is a unique $(i, j)$ s.t. $(N, k)_B \subset [x', y']_{(i,j)}$ and $(i, j)_c > 0$ that maximizes $i$ then
6:              Set $(i, j)$ to this value
7:          else
8:              Set $h$ to height of $I$
9:              $i := \min\{i \in [N] | ((N, k)_B \subset [x', y']_{(h,i)}) \wedge \text{Restricted}((h, i)_I) = 0\}$
10:             $j := \max\{j \in [N] | ((N, k)_B \subset [x', y']_{(h,j)}) \wedge \text{Restricted}((h, j)_I) = 0\}$
11:         Set $(i', j')$ such that $(N, j)_B$ is in $(i', j')_I$
12:         Move $(N, k)_B$ out of $(i', j')_I$ and into $(i, j)_I$
13:         Run $C'((i, j), (i', j'))$
14:         Run Optimize($B, I, (N, k)$)
15:         Update $B$ and $I$ based on output
16:     $k + +$
17: $DB' := ()$
18: $k := 1$
19: while $k \leq N$ do
20:     $v_j := (N, k)_B$
21:     Append $v_k$ to $DB'$
22:     $j + +$
23: Output $DB'$
```

Figure A.15: Algorithm for extracting a single solution from the output of the noisy database reconstruction algorithm.

Algorithm 13 2d-database Reconstruction

2-DimTurnpikeBT(Vols):

Input: The multiset Vols and $(N_1, \ldots, N_d)$.

Output: A 2d-database DB' such that, if Vols is the multiset volume leakage of some database DB, then 2-DimTurnpikeBT is a successful multiset volume-based reconstruction attack. Otherwise, **reject**.

```
1: if If $|\mathsf{Vols}| \neq |N_1 \times N_2|$ then
2:     Halt and reject
3: Build $B_x$ with entries $B_{(y,(i,j))}$ for all $i \in [N_1]$ and $j \in [i]$
4: for all $B_{(y,(i,j))} \in B_x$
5:     $B_{(y,(i,j))}$ let $B_{(y,(i,j))}(a,b)$ be the element in row $a$ column $b$ and initialize each to $-1$
6: $B_{(y,(1,1))}(1,1) := \max[\mathsf{Vols}]$
7: for all $i \in [2, N_1]$ do
8:     Set $C^L_{(y,(i,1))}$, $C^R_{(y,(i,1))}$, $C^L_{(y,(i,i))}$, and $C^R_{(y,(i,i))}$ to 0
9: Choose $x$ or $y$ branch
10: Choose max or min branch
11: if Branch is $x$ and min then
12:     Set $i$ to min element of $[N_1]$ s.t. $B_{(y,(i,1))}(1,1) = -1$
13:     $B_{(y,(i,1))}(1,1) := \max[\mathsf{Vols}]$
14:     for all $i' \in [N_1]$ s.t. $B_{(y,(i',1))}(1,1) \neq -1$ do do
15:         $B_{(y,(N_1+1-|i-i'|,N_1+2-|i-i'|-\min(i,i'))}(1,1) := |B_{(y,(i,1))}(1,1) - B_{(y,(i',1))}(1,1)|$
16:         if $|B_{(y,(i,1))}(1,1) - B_{(y,(i',1))}(1,1)| \in \mathsf{Vols}$ then
17:             $\mathsf{Vols} := \mathsf{Vols} \setminus \{|B_{(y,(i,1))}(1,1) - B_{(y,(i',1))}(1,1)|\}$
18:         else
19:             Backtrack
20:     if $\mathsf{Vols} \neq \emptyset$ then
21:         Return to branch choice on line 9
```

Figure A.16: 2d-database reconstruction algorithm from multiset volume leakage. Part 1 of 3.

```
22: if Branch is $x$ and $\mathsf{max}$ then
23:     Set $i$ to min element of $[N_1]$ s.t. $B_{(y,(i,i))}(1,1) = -1$
24:     $B_{(y,(i,i))}(1,1) \coloneqq \max[\mathsf{Vols}]$
25:     for all $i' \in [N_1]$ s.t. $B_{(y,(i',i'))}(1,1) \neq -1$ do
26:         $B_{(y,(N_1+1-|i-i'|,\min(i,i'))}(1,1) \coloneqq |B_{(y,(i,i))}(1,1) - B_{(y,(i',i'))}(1,1)|$
27:         if $|B_{(y,(i,i))}(1,1) - B_{(y,(i',i'))}(1,1)| \in \mathsf{Vols}$ then
28:             $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{|B_{(y,(i,i))}(1,1) - B_{(y,(i',i'))}(1,1)|\}$
29:         else
30:             Backtrack
31:     if $\mathsf{Vols} \neq \emptyset$ then
32:         Return to branch choice on line 9
33: if Branch is $y$ and $\mathsf{min}$ then
34:     Choose $(i,j)$ s.t. $i \in [N_1]$ and $j \in \{1,i\}$ s.t. either $i = j = 1$ or s.t. $B_{(y,(i,j))}(1,1) \neq$
    $-1$ and $B^L_{(y,(i,j))} > 0$
35:     Substitute in 1 for all instances of $b$ and $b'$ in the following and substitute in $L$ for $D$
36: if Branch is $y$ and $\mathsf{max}$ then
37:     Choose $(i,j)$ s.t. $i \in [N_1]$ and $j \in \{1,i\}$ s.t. either $i = j = 1$ or s.t. $B_{(y,(i,j))}(1,1) \neq$
    $-1$ and $B^R_{(y,(i,j))} > 0$
38:     In the following, substitute in $a$ and $a'$ for all instances of $b$ and $b'$ respectively
39:     In the following, substitute in $R$ for all instances of $D$
40: if Branch is $y$ then
41:     Set $a$ to min element of $[N_2]$ s.t. $B_{(y,(i,j))}(a,b) = -1$
42:     $B_{(y,(i,j))}(a,b) \coloneqq \max[\mathsf{Vols}]$
43:     $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{\max[\mathsf{Vols}]\}$
44:     if $j = 1$ then
45:         Substitute in 1 for all instances of $z$ and $z'$ in the following and substitute in
    $N_1 + 2 - |i - i'| - 2\min\{i, i'\}$ for $t$
46:     else
47:         In the following, substitute in $i$ and $i'$ for all instances of $z$ and $z'$ respectively
48:         $t \coloneqq 0$
49:     for all $i' \in [N_1] \setminus \{i\}$ s.t $B_{(y,(i',z'))}(a,b) \neq -1$ do
50:         if $|B_{(y,(i,z))}(a,b) - B_{(y,(i',z'))}(a,b)| \in \mathsf{Vols}$ then
51:             $B_{(y,(N_1+1-|i-i'|,\min(i,i')+t)}(a,b) \coloneqq |B_{(y,(i,z))}(a,b) - B_{(y,(i',z'))}(a,b)|$
52:             $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{|B_{(y,(i,z))}(a,b) - B_{(y,(i',z'))}(a,b)|\}$
53:         else
54:             Backtrack
```

Figure A.17: 2d-database reconstruction algorithm from multiset volume leakage. Part 2 of 3.

```
55:     for all $a' \in [N_2] \setminus \{a\}$ s.t. $\mathbf{B}_{(y,(i,j))}(a', b') \neq -1$ do
56:         if $|\mathbf{BTP}_{y,(i,j)}(a, 1) - \mathbf{BTP}_{y,(i,j)}(a', b')| \in \mathsf{Vols}$ then
57:             $\mathbf{BTP}_{y,(i,j)}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\}) \coloneqq |\mathbf{BTP}_{y,(i,j)}(a, b) - \mathbf{BTP}_{y,(i,j)}(a', b')|$
58:             $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{|\mathbf{BTP}_{y,(i,j)}(a, b) - \mathbf{BTP}_{y,(i,j)}(a', b')|\}$
59:         else
60:             Backtrack
61:     for all $i' \in [N_1] \setminus \{i\}$ s.t $B_{(y,(i',z'))}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\}) \neq -1$ do
62:         if $B_{(y,(N_1+1-|i-i'|,\min\{i,i'\}+t))}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\}) \in \mathsf{Vols}$ then
63:             $B_{(y,(N_1+1-|i-i'|,\min\{i,i'\}+t))}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\}) \coloneqq |B_{(y,(i,z))}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\}) - B_{(y,(i',z'))}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\})|$
64:             $\mathsf{Vols} \coloneqq \mathsf{Vols} \setminus \{B_{(y,(N_1+1-|i-i'|,\min\{i,i'\}+t))}(N_2+1-|a-a'|, N_2+2-|a-a'|-\min\{a, a'\})\}$
65:         else
66:             Backtrack
67:     $C^D_{(y,(i,j))} \coloneqq C^D_{(y,(i+1,j+1))} - 1$
68:     $i++$
69:     $C^D_{(y,(i,z))} \coloneqq C^D_{(y,(i,z))} - 1$
70:     if $\mathsf{Vols} \neq \emptyset$ then
71:         Return to branch choice on line 9
```

Figure A.18: 2d-database reconstruction algorithm from multiset volume leakage. Part 3 of 3.