ON THE ALGORITHMIC ASPECTS OF TURÁN'S THEOREM

By

MENG-TSUNG TSAI

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Martín Farach-Colton

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2017

ABSTRACT OF THE DISSERTATION

On the Algorithmic Aspects of Turán's Theorem

by Meng-Tsung Tsai

Dissertation Director: Martín Farach-Colton

Turán's Theorem gives an upper bound on the number of edges of n-node, K_r-free graphs, or equivalently it can be restated as that every n-node, m-edge graph has an independent set of size $n^2/(n+2m)$. We illustrate how to apply Turán's Theorem to algorithmic problems in several ways.

The complexity of dictionary operations, insertion for example, in external memory is well studied. However, the complexity of a batch of n operations is less known, and is seldom as easy as summing up the complexity of individual operations. We obtain lower bounds for batched predecessors by showing the necessity of fetching a set of information that preserves some "independence", where Turán's Theorem applies. We also prove lower bounds for batched deletions in cross-referenced dictionaries based on the existence of an adversarial input that forbids some patterns, where Turán's Theorem again applies.

In addition, we present an interesting class of problems that are NP-hard to approximate. Turán's theorem is useful in the proof that a large class of problems that can be defined in a simple framework are all NP-hard to approximate.

Acknowledgements

I would like to first thank my advisor Dr. Martín Farach-Colton. He always encourages me to work on what greatly interests me, always provides me with useful discussions whenever I get stuck on a problem, and always finds ample resources for me to move my research forward. My Ph.D. study is thusly full of adventure and fun, and best of all, it goes very smoothly.

I would like to sincerely thank my committee members, Dr. Eric Allender and Dr. William Steiger. I often dropped by their offices and asked more or less a random question. I thank them for always putting aside the work on hand, and giving me useful feedbacks, which greatly accelerated my pace on this dissertation.

I would like to thank everybody who has made helpful comments to my dissertation. They are Dr. Peyman Afshani, Dr. Michael A. Bender, Dr. Jeremy T. Fineman, Dr. Mayank Goswami, Dr. Dzejla Medjedovic, and Dr. Pablo Montes, who are the co-authors of the three papers [21], [5], [10], part of this dissertation, as well as the external committee member, Dr. Rezaul A. Chowdhury. They point out my blind points, and help me greatly to improve the quality of this dissertation.

Last but not the least, I would like to thank my family members: my parents, sister, brother, and my fiancé for their consideration and support so that I can focus myself on my Ph.D. study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Turán's Theorem [94] is one of the most fundamental results in extremal graph theory. It proves an upper bound on the number of edges that any n-node graph G can have if G contains no clique of size r, as follows.

Theorem 1.1 (Turán's Theorem (First Formulation))**.** *Let G be any n-node graph that contains no clique of size r. Then, the number of edges that G has is at most*

$$\left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}.$$

Observe that a clique in G is an independent set in the complement graph $\bar{G}$, and therefore Turán's Theorem can be restated as the following formulation. We will give a formal proof for the equivalence in Section 2.2.

Theorem 1.2 (Turán's Theorem (Second Formulation))**.** *Any n-node, m-edge graph G has an independent set of size at least*

$$\frac{n^2}{n+2m}.$$

In this dissertation, we will illustrate how to apply Turán's Theorem, mainly the second formulation, to algorithmic problems in several ways, part of my previous publications [5, 10, 21]. We focus ourselves on the parts where we use Turán's Theorem in Chapter 2, and defer the full details of these algorithmic problems to Chapters 3, 4, and 5. We summarize the algorithmic problems that we solved as follows.

The batched predecessor problem. The main question we answered in Chapter 3 is a problem in the external memory model, in which each disk page has size B and each I/O can read or write a disk page. Note that the performance is measured in the number of I/Os rather than the number of CPU instructions because the former is the bottleneck of

the entire performance. Given a sequence of N updates (insertions or deletions), one can implement the updates with amortized

$$\mathcal{O}\left(\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}}\right) \text{ I/Os for any constant } \varepsilon > 0, \tag{1.1}$$

while retaining the cost of queries to be optimum. We are interested in whether the amortized cost of predecessor search can be analogously better than $\mathcal{O}(\log_B N)$, the cost in the worst case.

We show, unless a preprocessing with an $\Omega(N^{4/3})$ number of I/Os for is used, the amortized cost of predecessor search cannot be better than that in the worst case in the pointer machine model. We lower bound the amortized cost by analyzing the ***generalized diameter*** of the graph underlying the structure of any feasible data structure. We introduce in Section 2.4 the generalized diameter where we apply Turán's Theorem. We refer readers to Chapter 3 for the full details of the batched predecessor problem.

Cross-referenced dictionaries. The main question we answered in Chapter 4 is a problem in the comparison-based external memory model. It is known that the amortized cost for updates in Equation (1.1) is the best possible (see the survey in [5]), if the cost for queries needs to match the information-theoretical optimum. We show, however, that the cost in Equation (1.1) cannot be attained when the input elements have dimension more than 1. To state the result formally, we are given a sequence of insertions of tuples (a_i, b_i) for $i \in [1, N]$, followed by a sequence of deletions $(d_k, *)$ for $k \in [1, N^{1/2}]$ that deletes all (a_i, b_i)'s whose $a_i = d_k$. To answer range queries on b's in a linear number of I/Os, the amortized cost of insertions and deletions has a lower bound $\Omega(\log_B N/B^{2/3})$.

Not every input sequence, induced by the rank of a_i's and b_i's, yields the hardness result. Thus, we require our proof to rely on an adversary input sequence, the concatenation of a sufficiently large set S of ***independent permutations***; that is, every pair of permutations preserves a certain property. We resort to Turán's Theorem to obtain a lower bound of $|S|$ in Section 2.3. We refer readers to Chapter 4 for the full details of our analysis for the cross-referenced dictionaries.

rSUM-hardness yields APX-hardness. In Chapter 5, we prove the APX-hardness for every problem in a class of maximization problems, each of which is encoded by a polynomial f. This class includes several natural problems, listed in Section 5.1. Let $f \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ be an r-variate polynomial, and let S be ***f-root-free*** if for all distinct $x_1, x_2, \ldots, x_r \in S$, $f(x_1, x_2, \ldots, x_r) \neq 0$. We denote by $\mathcal{P}_f$ the predicate of deciding if a given set S *has* a root for f whose coordinates are distinct, i.e. S is ***not*** f-root-free, and by Max-RF(f) the problem of finding the largest subset of a given $S \subset \mathbb{Z}$ that is f-root-free.

Our main theorems are:

Theorem 1.3. *If $f \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ is a homogeneous linear polynomial consisting of $r \geq 3$ variables, then* Max-RF(f) *is* APX*-hard and cannot be approximated to within $1 - \varepsilon_r$ unless* P = NP *for any*

$$\varepsilon_r < \Delta_r \equiv \frac{1}{744.64 + 601.44(r-3)}. \tag{1.2}$$

Theorem 1.3 restricts the characteristic polynomials to be linear. We show two ways to combine linear polynomials that yield APX-hard maximization problems.

Theorem 1.4. *Let $f = \prod_{i=1}^{k} \ell_i$ where k is a constant and each $\ell_i \in \mathbb{Z}[x_1, x_2, \ldots, x_{r_i}]$ is a homogeneous linear polynomial consisting of r_i variables. If $r = \max_{1 \leq i \leq k} r_i$ is at least 3, then* Max-RF(f) *is* APX*-hard, and cannot be approximated to within $1 - \varepsilon_r$ unless* P = NP *for any $\varepsilon_r < \Delta_r$.*

We can think of the products of polynomials as a disjunctive combination, in that an r-tuple is a root of f if it is a root of any of the constituent linear polynomials. We also have a ***conjunctive*** generalization, as follows. We say an r by k matrix M is ***strongly full rank*** if $k \leq r$ and every $k \times k$ submatrix of M is full rank. Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ be vectors of a same dimensionality, and let $\mathbf{M} = (\mathbf{v_1}|\mathbf{v_2}|\ldots|\mathbf{v_k})$ be the matrix where $M_{ij} = \mathbf{v_j}[i]$. We call $\mathbf{M}$ the ***aggregation*** of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$. We say a vector space is in ***general position*** if it has a set of basis vectors whose aggregation is strongly full rank.

Theorem 1.5. *Let $f = \sum_{i=1}^{k} \ell_i^2$ be a r-variate polynomial*[1] *where k is a constant and each $\ell_i \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ is a homogeneous linear polynomial. If the solution set of $f = 0$ is in*

[1] *We note here that $r - 1 \geq d$ and $d \geq 2$ together yield $r \geq 3$.*

general position and has dimension d at least 2, then Max-RF(f) *is* APX*-hard, and cannot be approximated to within* $1 - \varepsilon_r$ *unless* P = NP*, for any*

$$\varepsilon_r < \Gamma_r \equiv \frac{1}{140(6\lceil (r-2)/6 \rceil + 1)}.$$

As promised, Theorem 1.5 can be viewed as the ***conjunctive*** generalization of Theorem 1.3, because the f-root-freeness of some set S implies that S is ℓ_i-root-free for every i.

Our results rely on extending the NP-hardness to APX-hardness for several problems. We show how to achieve the extensions in Section 2.5 by appealing to Turán's Theorem. We refer readers to Chapter 5 for the full proof of Theorems 1.3, 1.4, and 1.5.

Chapter 2

Preliminaries

2.1 Notation

We define some notions that are used throughout this chapter. Let $\alpha(G)$ be the size of maximum independent set of graph G. Let $\kappa(G)$ be the size of maximum clique of graph G. Let $\mathbb{E}[X]$ be the expectation value of random variable X. Let $\mathbb{P}[E]$ be the probability that event E happens.

2.2 Proof of Turán's Theorem

For the sake of self-contained, we first revisit the probabilistic proof of Theorem 1.2 that was introduced in [88]. Then, we prove that Theorem 1.1 and Theorem 1.2 are equivalent. This immediately gives a probabilistic proof for Theorem 1.1, which was proved by a combinatorial argument [94].

Proof of Turán's Theorem (Second Formulation). We prove this by a probabilistic proof. Let $G = (V, E)$ where V denotes the node set of G and E denotes the edge set of G. Thus, $|V| = n$ and $|E| = m$. Let $f : V \to \{1, 2, \ldots, n\}$ be a bijective function, and note that there are $n!$ different such f's. Let $I_f = \{x \in V : f(x) < f(y) \text{ for all } (x, y) \in E\}$ be a subset of V with respect to f. Clearly, I_f is an independent set of G because I_f cannot contain both x and y if $(x, y) \in E$. Our goal is to show that

$$|I_f| \geq \frac{n^2}{n + 2m} \text{ for some } f. \tag{2.1}$$

Let π be a random function sampled uniformly at random from the $n!$ possible f's. Let $\deg(x)$ be the degree of node x and $\bar{d} = \sum_{x \in V} \deg(x)/n$ be the average degree. For every

node $x \in V$, we have the probability

$$\mathbb{P}\left[x \in I_\pi\right] = \mathbb{P}\left[\pi(x) < \pi(y)\ \forall (x,y) \in E\right] = \frac{1}{1+\deg(x)}. \tag{2.2}$$

Therefore, the expected size of I_π is

$$\begin{aligned}
\mathbb{E}\left[|I_\pi|\right] &= \sum_{x \in V} \mathbb{P}\left[x \in I_\pi\right] && \text{(linearity of expectation)} \\
&= \sum_{x \in V} \frac{1}{1+\deg(x)} && \text{(By (2.2))} \\
&\geq \sum_{x \in V} \frac{1}{1+\bar{d}} && \left((1+x)^{-1} \text{ is convex for } x \geq 0\right) \\
&= \sum_{x \in V} \frac{1}{1+2m/n} && \text{(handshaking lemma)} \\
&= \frac{n^2}{n+2m}
\end{aligned}$$

Since there exists some f whose $|I_f| \geq \mathbb{E}\left[|I_\pi|\right]$, we are done. □

Theorem 2.1. *The two formulation of Turán's Theorem in Chapter 1 are equivalent, i.e. Theorem 1.1 ⇔ Theorem 1.2.*

Proof. (Theorem 1.1 ⇒ Theorem 1.2) Let G be any n-node, m-edge graph. Since $\alpha(G) = \kappa(\bar{G})$, we know that $\bar{G} = (\bar{V}, \bar{E})$ contains no clique of size $r = \alpha(G) + 1$. Thus,

$$\begin{aligned}
& \left(1 - \frac{1}{\alpha(G)}\right)\frac{n^2}{2} \geq |\bar{E}| && \text{(Theorem 1.1)} \\
\Rightarrow\ & \left(\alpha(G) - 1\right)\frac{n^2}{2} \geq \alpha(G)|\bar{E}| && (2.3) \\
\Rightarrow\ & \left(\frac{n^2}{2} - |\bar{E}|\right)\alpha(G) \geq \frac{n^2}{2} && (2.4) \\
\Rightarrow\ & \left(\frac{n^2}{2} - \left(\binom{n}{2} - m\right)\right)\alpha(G) \geq \frac{n^2}{2} && (2.5) \\
\Rightarrow\ & \alpha(G) \geq \frac{n^2}{n+2m} && (2.6)
\end{aligned}$$

(Theorem 1.1 ⇐ Theorem 1.2) Let G be any n-node graph that contains no clique of

size r. Since $\kappa(G) = \alpha(\bar{G})$, we have that $r - 1 \geq \kappa(G) = \alpha(\bar{G})$. Thus,

$$\frac{n^2}{n + 2|\bar{E}|} \leq r - 1 \qquad \text{(Theorem 1.2)}$$

$$\Rightarrow \quad \frac{n^2}{n + 2\left(\binom{n}{2} - |E|\right)} \leq r - 1 \qquad (2.7)$$

$$\Rightarrow \quad n^2 \leq (r-1)(n^2 - 2|E|) \qquad (2.8)$$

$$\Rightarrow \quad 2(r-1)|E| \leq (r-2)n^2 \qquad (2.9)$$

$$\Rightarrow \quad |E| \leq \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2} \qquad (2.10)$$

□

Combining Theorems 1.2 and 2.1 yields a proof of Theorem 1.1.

2.3 On a Certain Set of Independent Permutations

We begin with the definition of independent permutations. Then, we discuss how large a set of independent permutations can be.

Definition 2.2. *Let π_A, π_B be two bijective functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$, a.k.a. permutation functions. Let z be a factor of n, where $qz = n$ for some $q \in \mathbb{N}$. We define that π_1, π_2 are **z-chunk independent** iff for every $k, \ell \in \{1, 2, \ldots, q\}$, the intersection of A_k and B_ℓ has a constant size, independent of n, where*

$$A_k = \{\pi_A(x) : (k-1)z + 1 \leq x \leq kz\} \text{ and } B_\ell = \{\pi_B(x) : (\ell - 1)z + 1 \leq x \leq \ell z\}.$$

Definition 2.3. *We say a set S of permutation functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$ are **z-chunk independent** iff for every two permutation functions in S are z-chunk independent.*

Our main theorem is:

Theorem 2.4. *For every integer $n \geq 1$, every constant $\delta \in (0, 1/2)$ whereas n^δ divides n, there exists an n^δ-chunk independent set S of permutation functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n\}$ whose size is at least n^C for any constant $C > 0$.*

Proof. We construct a graph $G = (V, E)$ of $n!$ nodes where each node represents a unique permutation function, and connect an edge between two nodes iff the represented permutation functions are not z-chunk independent where $z = n^\delta$ for some constant δ. Our goal is thus to prove that G has an independent set of size n^C.

By Theorem 1.2, we have that

$$\alpha(G) \geq \frac{|V|^2}{|V| + 2|E|} \tag{2.11}$$

$$= \frac{|V|^2}{|V| + \sum_{x \in V} \deg(x)} \tag{2.12}$$

Because of symmetry, G is d-regular. Let f be the permutation function where $f(x) = x$, and π be a permutation function sampled uniformly at random from the $n!$ possible ones. Let v_f and v_π be the nodes in G that represent f and π, respectively. We have that:

$$d = |V| \mathbb{P}\left[(v_f, v_\pi) \in E\right] \tag{2.13}$$

$$\leq |V| n^{1-\delta} \binom{n^\delta}{T} \left(\frac{1}{n^{1-\delta}}\right)^T n^{1-\delta} \qquad \text{(note that } \delta > 0\text{)}$$

$$\leq |V| n^{2-2\delta-T(1-2\delta)} \qquad \text{(note that } \delta < 1/2\text{, and pick a sufficiently large } T\text{)}$$

$$\leq |V|/n^C \text{ for any constant } C > 0 \tag{2.14}$$

Combining Equations (2.12) and (2.14), we get

$$\alpha(G) \geq \frac{|V|^2}{(1+d)|V|} \geq \frac{|V|}{1 + |V|/n^C} = n^{C - o(1)} \text{ for any constant } C > 0. \tag{2.15}$$

□

Remark 2.5. *If $\delta = 0$ or $\delta > 1/2$, it is clear that Theorem 2.4 does not hold. For $\delta = 1/2$, we do not know whether Theorem 2.4 hold or not, which is no easier than the almost orthogonal vector problem for $\{0,1\}^n$. The corresponding almost orthogonal vector problem for $\mathbb{R}^n$ needs a number theoretical construction due to Tao [87].*

2.4 Generalized Diameter

We generalize the definition of graph diameter to k nodes for any $k \geq 2$, denoted by ***k-diameter*** $d_k(G)$, and prove a lower bound of k-diameter by Turán's Theorem. 2-diameter

is exactly the graph diameter; i.e. the greatest ***distance*** between any pair of nodes, where the distance between two nodes is the number of edges in a shortest path that connects them. For general k, we define the k-diameter as follows:

Definition 2.6. *Let k-diameter be the maximum **hyper-distance** among any group of k nodes, where the hyper-distance between k nodes is the number of edges in a minimum node-cardinality spanning tree that connects them.*

We recall the definition of ***r-independent set*** before proceeding to the helper lemma (Lemma 2.7) where the r-independent set is a set of nodes whose pairwise distance is more than r. Clearly, 1-independent set is exactly the independent set. Let $\alpha_r(G)$ be the cardinality of the maximum r-independent set.

Our helper lemma is:

Lemma 2.7.

$$d_k(G) \geq k(r+1)/2 \text{ if } k \leq \alpha_r(G)$$

for any connected graph G.

Proof. Since $k \leq \alpha_r(G)$, there exist k nodes that form an r-independent set S. It suffices to prove that any tree spanning S has at least $k(r+1)/2$ edges.

Let T be any such tree of ℓ edges. Let $v_1, v_2, \ldots, v_k$ be the order that $v_i \in S$ first appears in the Eulerian tour of T that starts from an arbitrary node. Since the Eulerian tour of T traverses each edge in T twice, and for each i the number of edges in the Eulerian tour between the first appearance of v_i and v_{i+1} cannot be less than their distance, we have:

$$2\ell \geq d(v_k, v_1) + \sum_{1 \leq i \leq n-1} d(v_i, v_{i+1}) \geq k(r+1).$$

Because T is arbitrarily picked, we are done. □

Based on Lemma 2.7, we are able to obtain a lower bound of $d_k(G)$ if the $\alpha_r(G)$ can be bounded. A particular example is for regular graphs:

Theorem 2.8. *For any connected t-regular n-node graph G where $t \geq 2$, a lower bound for the k-diameter of G is*

$$d_k(G) \geq k\left(2 + \log_{t-1} \frac{n}{4tk}\right).$$

Proof. Since a connected 2-regular graph is a cycle graph, the statement immediately holds. We assume that $t > 2$ for the rest of the proof.

Let G^i be the i-th graph product of G, where an edge $(u, v) \in G^i$ iff there is a path of length i that connects u and v. Let $G^+ = G \cup G^2 \cup \cdots \cup G^r$. Clearly, an independent set of G^+ is an r-independent set of G and the degree of each node in G^+ is at most

$$\sum_{0 \le i \le r-1} t(t-1)^i \le 2t(t-1)^{r-1} - 1 \tag{2.16}$$

By Turán's Theorem,

$$\alpha_r(G) \ge \frac{n}{2t(t-1)^{r-1}}. \tag{2.17}$$

To apply Lemma 2.7, we require $k \le \alpha_r(G)$. This can be asserted by letting

$$\frac{n}{2t(t-1)^{r-1}} \ge k. \tag{2.18}$$

$$\Leftrightarrow \; 1 + \log_{t-1} \frac{n}{2tk} \ge r \tag{2.19}$$

We are done by picking r as the greatest possible. □

2.5 Extending the MIS-based NP-hardness Results

The NP-hardness of many problems is based on the NP-hardness of Maximum Independent Set Problem (MIS). Usually, the NP-hardness reduction suffices to be an APX-hardness reduction with resorting to Turán's Theorem. We illustrate how to achieve this for several example problems.

2.5.1 Hardness of Max-3SUM

We define Max-3SUM to be the maximization version of the 3SUM problem: Given a set S of integers, find the largest $T \subseteq S$ so that T has no 3 (distinct) elements that sum to 0, that is, the largest $T \subseteq S$ that ***fails*** the 3SUM test. We say such T a 3SUM-***free set***. For the ease to explain the use of Turán's Theorem, we restrict ourselves to the case that the three elements are distinct. For the case of allowing repetition, we refer readers to Chapter 5.

In Chapter 5, we show that:

Lemma 2.9. *For any n-node m-edge graph $G = (V, E)$, there exists an injective function $f : V \cup E \to \mathbb{Z}$ so that G has an independent set of size k iff the set*

$$S = \{f(x) : x \in V \cup E\}$$

has a 3SUM*-free subset of size $m + k$, where the function f can be found and evaluated in time polynomial in n.*

Since MIS is NP-hard, by Lemma 2.9 we know that Max-3SUM is NP-hard. By Turán's Theorem, Lemma 2.9 indeed implies something further:

Theorem 2.10. Max-3SUM *is* APX*-hard.*

Proof. Suppose for contradiction that there exists a polynomial-time algorithm $\mathcal{A}$ that can compute a $(1-\varepsilon)$-approximation for the Max-3SUM problem where ε is some constant > 0. For every n-node m-edge graph $G = (V, E)$ where $m \leq \delta n$ for some constant δ, we use the function f mentioned in Lemma 2.9 to obtain $S = \{f(x) : x \in V \cup E\}$ and then run $\mathcal{A}$ with the input S. The output of $\mathcal{A}$ will give a quantity

$$r \in [(1-\varepsilon)(\alpha(G) + m), (\alpha(G) + m)] \tag{2.20}$$

in a bounded range by Lemma 2.9. On the other hand, by Turán's Theorem we have that

$$\alpha(G) \geq \frac{n^2}{n + 2m} \geq \frac{n}{1 + 2\delta} \geq \frac{m}{\delta(1 + 2\delta)}. \tag{2.21}$$

Combining Equations (2.20) and (2.21), we get

$$\alpha(G) \geq r - m \geq (1-\varepsilon)\alpha(G) - \varepsilon m \geq \left[1 - (1 + \delta + 2\delta^2)\varepsilon\right] \alpha(G). \tag{2.22}$$

Equation (2.22) suggests that if algorithm $\mathcal{A}$ can pick ε as an arbitrarily small positive constant, then $r - m$ gives an accurate estimation of $\alpha(G)$, up to a constant multiplicative error arbitrarily close to 1. This leads to a contradiction because MIS has no PTAS for 3-regular graphs that have $1.5n$ edges. As a result, for some constant $\varepsilon > 0$, such algorithm $\mathcal{A}$ does not exist, meaning that Max-3SUM is APX-hard. □

2.5.2 Hardness of MIS for Bounded-Girth Graphs

Let us recall that ***girth*** of graph G is the length of a shortest cycle in G. Let C_g be the collection of all graphs that have girth $> g$. Clearly, C_3 is the collection of all triangle-free graphs. It is known in the literature that MIS for C_3 is APX-hard [34] and MIS for C_g for any constant $g \geq 3$ is NP-hard [69]. Here, we gives a complementary result:

Lemma 2.11. *For any n-node m-edge graph G, there exists a mapping $f : G \to H$ so that*

1. *H has $n + 2\ell m$ nodes, $(2\ell + 1)m$ edges, and girth $\geq 6\ell + 3$ for some constant ℓ,*
2. *H has an independent set of size $k + \ell m$ iff G has an independent set of size k,*

where $f(G)$ can be computed in time polynomial in n.

Proof. Initially, let H be a copy of G. Then, for each edge $(u, v) \in H$, we replace the edge with a path $u - x_{uv}^1 - x_{uv}^2 - \cdots - x_{uv}^{2\ell} - v$ of length $2\ell + 1$, where x_{uv}^i are newly added nodes. It is clear that the resulting graph H satisfies the first condition.

(2. $\Leftarrow$) If G has an independent set S, we construct an independent set I of H by setting $I = S$ initially. Then, for each edge $(u, v) \in G$, if $u \in S$ add $x_{uv}^2, x_{uv}^4, \ldots, x_{uv}^{2\ell}$ to I, or otherwise add $x_{uv}^1, x_{uv}^3, \ldots, x_{uv}^{2\ell-1}$ to I. I thus has size $|S| + \ell m$ and is an independent set of H because for every edge $(u, v) \in G$, u, v cannot be both contained in S.

(2. $\Rightarrow$) Let $I_0 = B_0 \cup W_0$ be an independent set of H where B_0 is a subset of the nodes from G and W_0 is a subset of the newly added nodes. For each edge $(u, v) \in G$, if both u and v are in B_0, then $|W_0 \cap \{x_{uv}^i : i \in [1, 2\ell]\}| = t < \ell$. Thus, we can perform such an operation

$$I_1 \leftarrow \left(I_0 \setminus \{v\} \setminus \{x_{uv}^i : i \in [1, 2\ell]\}\right) \cup \{x_{uv}^2, x_{uv}^4, \ldots, x_{uv}^{2(t+1)}\}$$

while retaining $|I_1| = |I_0|$. Let $I_1 = B_1 \cup W_1$, which are defined analogously. We iterate the above procedure until we reach I_c where $I_c = B_c \cup W_c$ and for every edge $(u, v) \in G$ at least one of u, v is not in B_c. Thus, B_c is an independent set of G. Since W_c has size at most ℓm and $|I_c| = |I|$, we have $|B_c| = |I_c| - |W_c| \geq k$. □

Theorem 2.12. *Maximum Independent Set problem is* APX*-hard for graphs in C_g for any constant $g \geq 3$.*

Proof. The proof is similar to that of Theorem 2.10. Suppose for contradiction that MIS for graphs in C_g can be approximated to within a factor of $(1-\varepsilon)$ in polynomial time by an algorithm $\mathcal{A}$. Then, for any n-node m-edge graph G where $m \leq \delta n$ for some constant δ, we use the function f indicated in Lemma 2.11 with such a constant ℓ that $6\ell + 3 > g$ and run $\mathcal{A}$ with input $f(G)$. The output of $\mathcal{A}$ is

$$r \in [(1-\varepsilon)(\ell m + \alpha(G)), (\ell m + \alpha(G))] \tag{2.23}$$

by Lemma 2.11. On the other hand, by Turán's Theorem we have

$$\alpha(G) \geq \frac{n^2}{n+2m} \geq \frac{n}{(1+2\delta)} \geq \frac{m}{\delta(1+2\delta)}. \tag{2.24}$$

Combining Equations (2.23) and (2.24), we get

$$\alpha(G) \geq r - \ell m \geq (1-\varepsilon)\alpha(G) - \varepsilon \ell m \geq \left[1 - (1 + \delta\ell + 2\delta^2\ell)\varepsilon\right] \alpha(G). \tag{2.25}$$

Again, if ε can be any arbitrarily small positive constant, then $\alpha(G)$ can be approximated to within any constant factor arbitrarily close to 1, a contradiction. □

Corollary 2.13. *Maximum Independent Set problem is* APX*-hard for graphs that have* $\leq \delta n$ *edges and have girth* $\geq g$*, where* n *is the number of nodes in a graph and* g, δ *are any constants that* $g \geq 3$*,* $\delta \geq 3/2$*.*

Proof. In the proof of Theorem 2.12, we have a PTAS reduction from MIS for any n-node m-edge graph that has $m = 1.5n$ edges to MIS for any graph that has $n + 2\ell m$ nodes and $(2\ell+1)m$ edges, where $6\ell + 3 \geq g$. Thus, we require to set

$$\delta \geq \frac{(2\ell+1)m}{n+2\ell m} = \frac{2\ell+1}{2\ell + \frac{1}{1.5}}. \tag{2.26}$$

Hence, any $\delta \geq 1.5$ suffices. □

2.5.3 Hardness of Finding the Largest Subgraph Without Small Cycles

Based on Corollary 2.13 and Turán's Theorem, we are able to generalize the classic NP-hardness result for largest (in terms of the number of nodes) node-induced subgraph that satisfies some ***hereditary property*** [64]. We say $\mathcal{P}$ a hereditary property if for any subgraph $H \subseteq G$, G satisfies $\mathcal{P}$ implies that H satisfies $\mathcal{P}$. It is clear that containing no

cycles of length ℓ is a hereditary property. Therefore, finding the largest node-induced subgraph that contains no cycles of length ℓ is NP-hard. We extend the NP-hardness result to APX-hardness result as follows.

Theorem 2.14. *Finding the largest node-induced subgraph that contains no cycle of length ℓ is* APX*-hard.*

Proof. Let G be any graph of girth $> \ell$ whose number of edges is at most 1.5 times of the number nodes, which in the graph class described in Corollary 2.13. By the definition of girth, G has no cycle of length ℓ. Let H be a copy of G. For each edge $(u, v) \in H$, replace (u, v) with a cycle $u - x_{uv}^1 - x_{uv}^2 - \cdots - x_{uv}^{\ell-2} - v - u$ of length ℓ where x_{uv}^i are newly added nodes. Clearly, every cycle of length ℓ in H has the form $u - x_{uv}^1 - x_{uv}^2 - \cdots - x_{uv}^{\ell-2} - v - u$ for some u, v. Thus, to remove at least one node from each cycle of length ℓ, one can simply consider the removal of u or v, or both, because the removal of a newly added node eliminates the only one cycle of length ℓ where it locates. Hence, H has a node-induced subgraph of size $k + m(\ell - 2)$ iff G has an independent set of size k. If there exists an algorithm can approximate the largest node-induced subgraph to within a factor of $(1 - \varepsilon)$, then one can get the quantity

$$r \in [(1 - \varepsilon)(\alpha(G) + m(\ell - 2)), \alpha(G) + m(\ell - 2)]. \tag{2.27}$$

On the other hand, by Turán's Theorem we have

$$\alpha(G) \geq \frac{n^2}{n + 2m} \geq \frac{n}{4} \geq \frac{m}{6}. \tag{2.28}$$

Combining Equations (2.27) and (2.28),

$$\alpha(G) \geq r - (\ell - 2)m \geq (1 - \varepsilon)\alpha(G) - \varepsilon(\ell - 2)m \geq [1 - (6\ell - 11)\varepsilon]\, \alpha(G). \tag{2.29}$$

Thus, the ε cannot be arbitrarily small. We are done. □

Chapter 3

Batched Predecessor Problem

3.1 Problem Definition and Background

A ***static dictionary*** is a data structure that represents a set $S = \{s_1, s_2, \ldots, s_n\}$ subject to the following operations:

PREPROCESS(S):	Prepare a data structure to answer queries.
SEARCH(q, S):	Return TRUE if $q \in S$ and FALSE otherwise.
PREDECESSOR(q, S):	Return $\max_{s_i \in S}\{s_i < q\}$.

The traditional static dictionary can be extended to support batched operations. Let $Q = \{q_1, \ldots, q_x\}$. Then, the ***batched predecessor*** problem can be defined as follows:

BATCHEDPRED(Q, S):	Return $A = \{a_1, \ldots, a_x\}$, where $a_i =$ PREDECESSOR(q_i, S).

In this chapter we prove lower bounds on the batched predecessor problem in ***external memory*** [6], that is, when the dictionary is too large to fit into main memory. We study tradeoffs between the searching cost and the cost to preprocess the underlying set S. We present our results in the pointer-machine I/O model [86].

We focus on query size $x \leq n^c$, for constant $c < 1$. Thus, the query Q can be large, but is still much smaller than the underlying set S. This query size is interesting because, although there is abundant parallelism in the batched query, common approaches such as linear merges and buffering [13, 28, 30] are suboptimal.

Our results show that the batched predecessor problem in external memory cannot be solved asymptotically faster than $\Omega(\log_B n)$ I/Os per query element if the preprocessing

is bounded by a polynomial; on the other hand, the problem ***can*** be solved asymptotically faster, in $\Theta((\log_2 n)/B)$ I/Os, if we impose no constraints on preprocessing [21]. These bounds stand in marked contrast to single-predecessor queries, where one search costs $\Omega(\log_B n)$ even if preprocessing is unlimited.

We assume that S and Q are sorted. Without loss of generality, Q is sorted because Q's sort time is subsumed by the query time. Without loss of generality, S is sorted, as long as the preprocessing time is slightly superlinear. We consider sorted S throughout the chapter. For notational convenience, we let $s_1 < s_2 < \cdots < s_n$ and $q_1 < q_2 < \cdots < q_x$, and therefore $a_1 \leq a_2 \leq \cdots \leq a_x$.

Given that S and Q are sorted, an alternative interpretation of this problem is as follows: ***how can we optimally merge two sorted lists in external memory?*** Specifically, what is the optimal algorithm for merging two sorted lists in external memory when one list is some polynomial factor smaller than the other?

Observe that the naïve linear-scan merging is suboptimal because it takes $\Theta(n/B)$ I/Os, which is greater than the $\mathcal{O}(n^c \log_B n)$ I/Os of a B-tree-based solution. Buffer trees [13, 28, 30] also take $\Theta(n/B)$ I/Os during a terminal flush phase. This chapter shows that with polynomial preprocessing, performing independent searches for each element in Q is optimal, but it is possible to do better for higher preprocessing.

Dittrich et al. [39] consider multisearch problems where queries are simultaneously processed and satisfied by navigating through large data structures on parallel computers. They give a lower bound of $\Omega(x \log_B(n/x) + x/B)$ under stronger assumptions: no duplicates of nodes are allowed, the ith query has to finish before the $(i+1)$st query starts, and $x < n^{1/(2+\varepsilon)}$, for a constant $\varepsilon > 0$.

Buffering is a standard technique for improving the performance of external-memory algorithms [13, 28, 30]. By buffering, partial work on a set of operations can share an I/O, thus reducing the per-operation I/O cost. Queries can similarly be buffered. In this chapter, the number of queries, x, is much smaller than the size, n, of the data structure being queried. As a result, as the partial work on the queries progresses, the query paths can diverge within the larger search structure, eliminating the benefit of buffering.

In order to show results in the I/O pointer-machine model, we define a graph whose nodes are the blocks on disk of the data structure and whose edges are the pointers between blocks. Since a block has size B, it can contain at most B pointers, and thus the graph is fairly sparse. We show that any such sparse graph has a large set of nodes that are far apart. If the algorithm must visit those well-separated nodes, then it must perform many I/Os. The crux of the proof is that, as the preprocessing increases, the redundancy of the data structure increases, thus making it hard to pin down specific locations of the data structure that must be visited. We show that if the data structure is reasonable in size—in our case $\mathcal{O}(n^{4/3-\varepsilon})$—then we can still find a large, well dispersed set of nodes that must be visited, thus establishing the following lower bound:

Theorem 3.1 (Lower bound, I/O pointer-machine model)**.** *Let S be a set of size n. In the I/O pointer-machine model, if* Preprocessing(S) *uses $\mathcal{O}(n^{4/3-\varepsilon})$ blocks of space and I/Os, for any constant $\varepsilon > 0$, then there exists a constant c and a set Q of size n^c such that computing* BatchedPred(Q, S) *requires $\Omega(x \log_B(n/x) + x/B)$ I/Os.*

3.2 The Proof

Here we analyze the batched predecessor problem in the I/O pointer-machine model. We show that if the preprocessing time is $\mathcal{O}(n^{4/3-\varepsilon})$ for any constant $\varepsilon > 0$, then there exists a query set Q of size x such that reporting BatchedPred(Q, S) requires $\Omega(x/B+x \log_B n/x)$ I/Os. Before proving our theorem, we briefly describe the model.

I/O pointer machine model. The I/O pointer machine model [86] is a generalization of the pointer machine model introduced by Tarjan [89]. Many results in range reporting have been obtained in this model [3, 4].

To answer BatchedPred(Q, S), an algorithm preprocesses S and builds a data structure comprised of n^k blocks, where k is a constant to be determined later. We use a directed graph $\mathcal{G} = (V, E)$ to represent the n^k blocks and their associated directed pointers. Every algorithm that answers BatchedPred(Q, S) begins at the start node v_0 in V and at each step picks a directed edge to follow from those seen so far. Thus, the nodes in a computation are all reachable from v_0. Furthermore, each fetched node contains elements from S,

and the computation cannot terminate until the visited set of elements is a superset of the answer set A. A node in V contains at most B elements from S and at most B pointers to other nodes.

Let $\mathcal{L}(W)$ be the union of the elements contained in a node set W, and let $\mathcal{N}(a)$ be the set of nodes containing element a. We say that a node set W ***covers*** a set of elements A if $A \subseteq \mathcal{L}(W)$. An algorithm for computing A can be modeled as the union of a set of paths from v_0 to each node in a node set W that covers A.

To prove a lower bound on $\textsc{BatchedPred}(Q, S)$, we show that there is a query set Q whose answer set A requires many I/Os. In other words, for every node set W that covers A, a connected subgraph spanning W contains many nodes. We achieve this result by showing that there is a set A such that, for every pair of nodes $a_1, a_2 \in A$, the distance between $\mathcal{N}(a_1)$ and $\mathcal{N}(a_2)$ is large, that is, all the nodes in $\mathcal{N}(a_1)$ are far from all the nodes in $\mathcal{N}(a_2)$. Since the elements of A can appear in more than one node, we need to guarantee that the node set V of $\mathcal{G}$ is not too large; otherwise the distance between $\mathcal{N}(a_1)$ and $\mathcal{N}(a_2)$ can be very small. For example, if $|V| \geq \binom{n}{2}$, every pair of elements can share a node, and a data structure exists whose minimum pairwise distance between any $\mathcal{N}(a_1)$ and $\mathcal{N}(a_2)$ is 0.

First, we introduce two measures of distance between nodes in any (undirected or directed) graph $G = (V, E)$. Let $d_G(u, v)$ be the length of the shortest (di-)path from node u to node v in G. Furthermore, let $Ł_G(u, v) = \min_{w \in V} (d_G(w, u) + d_G(w, v))$. Thus, $Ł_G(u, v) = d_G(u, v)$ for undirected graphs, but not necessarily for directed graphs.

For each $W \subseteq V$, define $f_G(W)$ to be the minimum number of nodes in any connected subgraph H such that (1) the node set of H contains $W \cup \{v_0\}$ and (2) H contains a path from v_0 to each $v \in W$. Observe that $f_G(\{u, v\}) \geq Ł_G(u, v)$. The following lemma gives a more general lower bound for $f_G(W)$. In other words, the size of the graph containing nodes of W is linear in the minimum pairwise distance within W.

Lemma 3.2. *For any directed graph $G = (V, E)$ and any $W \subseteq V$ of size $|W| \geq 2$, $f_G(W) \geq r_W|W|/2$, where $r_W = \min_{u,v \in W, u \neq v} Ł_G(u, v)$.*

Proof Sketch. Consider the undirected version of G, and consider a TSP of the nodes in

W. It must have length $r_W|W|$. Any tree that spans W must therefore have size at least $r_W|W|/2$. Finally, $f_G(W)$ contains a tree that spans W. □

Our next goal is to find a query set Q such that every node set W that covers the corresponded answer set A has a large r_W. The answer set A will be an independent set of a certain kind, that we define next. For a directed graph $G = (V, E)$ and an integer $r > 0$, we say that a set of nodes $I \subseteq V$ is ***r-independent*** if $Ł_G(u, v) > r$ for all $u, v \in I$ where $u \neq v$. The next lemma guarantees a substantial r-independent set.

Lemma 3.3. *Given a directed graph $G = (V, E)$, where each node has out-degree at most $B \geq 2$, there exists an r-independent set I of size at least $\frac{|V|^2}{|V|+4r|V|B^r}$.*

Proof. Construct an undirected graph $H = (U, F)$ such that $U = V$ and $(u, v) \in F$ iff $Ł_G(u, v) \in [1, r]$. Then, H has at most $2r|V|B^r$ edges. By Turán's Theorem [88], there exists an independent set of the desired size in H, which corresponds to an r-independent set in G, completing the proof. □

In addition to r-independence, we want the elements in A to occur in few blocks, in order to control the possible choices of the node set W that covers A. We define the ***redundancy*** of an element a to be $|\mathcal{N}(a)|$. Because there are n^k blocks and each block has at most B elements, the average redundancy is $\mathcal{O}(n^{k-1}B)$. We say that an element has ***low redundancy*** if its redundancy is at most twice the average. We show that there exists an r-independent set I of size n^ε (here ε depends on r) such that no two blocks share the same low-redundancy element. We will then construct our query set Q using this set of low-redundancy elements in this r-independent set.[1]

Finally, we add enough edges to place all occurrences of every low-redundancy element within $\rho < r/2$ of all other occurrences of that element. We show that we can do this by adding few edges to each node, therefore maintaining the sparsity of G. Since this augmented graph also contains a large r-independent set, all the nodes of this set cannot share any low-redundancy elements.

[1] Our construction does not work if the query set contains high redundancy elements, because high redundancy elements might be placed in every block.

The following lemma shows that nodes sharing low-redundancy elements can be connected with low diameter and small degrees.

Lemma 3.4. *For any $k > 0$ and $m > k$ there exists an undirected k-regular graph H of order m having diameter $\log_{k-1} m + o(\log_{k-1} m)$.*

Proof. In [24], Bollobás shows that a random k-regular graph has the desired diameter with probability close to 1. Thus there exists some graph satisfying the constraints. □

Consider two blocks B_1 and B_2 in the r-independent set I above, and let a and b be two low-redundancy elements such that $a \in B_1, b \notin B_1$ and $a \notin B_2, b \in B_2$. Any other pair of blocks $B_1^{'}$ and $B_2^{'}$ that contain a and b respectively must be at least $(r - 2\rho)$ apart, since $B_i^{'}$ is at most ρ apart from B_i. By this argument, every node set W that covers A has $r_W \geq (r - 2\rho)$. Now, by 3.2, we get a lower bound of $\Omega((r - 2\rho)|W|)$ on the query complexity of Q. We choose $r = c_1 \log_B(n/x)$ and get $\rho = c_2 \log_B(n/x)$ for appropriate constants $c_1 > 2c_2$. This is the part where we require the assumption that $k < 4/3$ as shown in Theorem 3.1, where n^k was the size of the entire data structure. We then apply 3.3 to obtain that $|W| = \Omega(x)$.

Proof of Theorem 3.1. We partition S into S_ℓ and S_h by the redundancy of elements in these n^k blocks and claim that there exists $A \subseteq S_\ell$ such that query time for the corresponded Q matches the lower bound.

Let S_ℓ be the set of elements of redundancy no more than $2Bn^k/n$ (i.e., twice of the average redundancy). The rest of elements belong to S_h. By the Markov inequality, we have $|S_\ell| = \Theta(n)$ and $|S_h| \leq n/2$. Let $\mathcal{G} = (V, E)$ represent the connections between the n^k blocks as the above stated. We partition V into V_1 and V_2 such that V_1 is the set of blocks containing some elements in S_ℓ and $V_2 = V \setminus V_1$. Since each block can at most contain B elements in S_ℓ, $|V_1| = \Omega(n/B)$.

Then, we add some additional pointers to $\mathcal{G}$ and obtain a new graph $\mathcal{G}'$ such that, for each $e \in S_\ell$, every pair $u, v \in \mathcal{N}(e)$ has small $\mathrm{L}_{\mathcal{G}'}(u, v)$. We achieve this by, for each $e \in S_\ell$, introducing graph H_e to connect all the n^k blocks containing element e such that the diameter in H_e is small and the degree for each node in H_e is $\mathcal{O}(B^\delta)$ for some constant δ. By 3.4, the diameter of H_e can be as small as

$$\rho \leq \frac{1}{\delta} \log_B |H_e| + o(\log_B |H_e|) \leq \frac{k-1}{\delta} \log_B n + o(\log_B n).$$

We claim that the graph $\mathcal{G}'$ has a $(2\rho+\varepsilon)$-independent set of size n^c, for some constants $\varepsilon, c > 0$. For the purpose, we construct an undirected graph $H(V_1, F)$ such that $(u, v) \in F$ iff $Ł_{\mathcal{G}'}(u, v) \leq r$. Since the degree of each node in $\mathcal{G}'$ is bounded by $\mathcal{O}(B^{\delta+1})$, by 3.3, there exists an r-independent set I of size

$$|I| \geq \frac{|V_1|^2}{|V_1| + 4r|V|\mathcal{O}(B^{r(\delta+1)})} \geq \frac{n^{2-k}}{4r\mathcal{O}(B^{r(\delta+1)+2})} = n^c.$$

Then, $r = ((2-k-c)\log_B n)/(\delta+1) + o(\log_B n)$. To satisfy the condition made in the claim, let $r > 2\rho$. Hence, $(2-k-c)/(\delta+1) > 2(k-1)/\delta$. Then, $k \to 4/3$ for sufficiently large δ. Observe that, for each $e \in S_\ell$, e is contained in at most one node in I; in addition, for every pair $e_1, e_2 \in S_\ell$ where e_1, e_2 are contained in separated nodes in I, then $Ł_{\mathcal{G}'}(u, v) \geq \varepsilon$ for any $u \ni e_1, v \ni e_2$. By 3.2, we are done. □

Chapter 4

Cross-referenced Dictionaries

4.1 Problem Definition and Background

Dictionaries remain the most well studied class of data structures. A dictionary supports insertions, deletions, membership queries, and usually successor, predecessor, and extract-min operations. But surprisingly basic questions about dictionaries remain unanswered.

Some of these basic questions arose as far back as the pre-computer era, whenever people indexed large collections of data. The library of Alexandria is thought to have contained over 600,000 volumes, partitioned first into seven broad topics and then shelved alphabetically by author [52,100]. Each volume is thought to have had tags called ***pinakes***,[1] which contained metadata. Pinakes were also compiled into a separate volume, which is thought to have been the first library catalog. Thus, people could search for books using the pinakes, but only by scanning through the pinakes in $\langle subject, author \rangle$ order. It was many centuries before there were any libraries of size comparable to that of Alexandria after that library was destroyed, and during that time, libraries were indexed using content-addressable-memory systems—that is, "curators, slaves or freedmen" [55,58].

Circa 1295, the library at the Collège de Sorbonne at the Université de Paris introduced indexes [78], in the sense that there were volumes compiled for the purpose of locating books according to a variety of criteria. For the first time it became possible to search the content of a library according to distinct orders (by author, subject, or collection[2]), while the books themselves were stored on the shelves in any convenient order. In 1791, Enlightenment thinkers of the French Revolution introduced card catalogs as indexes, making it easier for

[1] "Pinakes" is ancient Greek for "tables", and is thus consistent with modern database nomenclature.

[2] No title indexes were compiled, since titles were not fixed at that time [77].

the indexes to track the changing collection [54].

Today books are stored on the shelves according to a subject-classification scheme (usually the Dewey Decimal System [38] or the Library of Congress Classification System (LC) [71]) to allow for browsing, but they are also indexed in other orders (author, title, subject, keyword). Each particular index on the books is a dictionary ordered by a different key.

Specifying Operations of a Dictionary.

The actual data structure at work organizing a library is not merely a set of dictionaries, but a ***system*** of ***cross-referenced dictionaries***, which we call a ***compound dictionary***. We call a compound dictionary an L***-dictionary*** if it consists of L cross-referenced dictionaries. A compound dictionary maintains a ***cross-reference invariant***, where each dictionary—which we sometimes call an ***index***—stores the same set of items but orders them according to a different comparison function. Thus, every time that a book is inserted into or deleted from the library, each index needs to be updated.

An abstraction of a compound dictionary is as follows. The L-dictionary maintains a set $\mathcal{S} \subseteq \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_L$. Each (potentially infinite) key space $\mathcal{U}_i$ is totally ordered. Items can be inserted, deleted, and queried:

- INSERT($\boldsymbol{x}$): $\mathcal{S} \longleftarrow \mathcal{S} \cup \{\boldsymbol{x}\}$. That is, add $\boldsymbol{x}$ to $\mathcal{S}$.
- DELETE(i, x): $\mathcal{S} \longleftarrow \mathcal{S} - \mathcal{U}_1 \times \cdots \times \mathcal{U}_{i-1} \times \{x\} \times \mathcal{U}_{i+1} \times \cdots \times \mathcal{U}_L$.
  That is, remove all tuples $\langle *, \ldots, *, x, *, \ldots, * \rangle$ whose ith component is x.
- LOOKUP(i, x): return $\mathcal{S} \cap \mathcal{U}_1 \times \cdots \times \mathcal{U}_{i-1} \times \{x\} \times \mathcal{U}_{i+1} \times \cdots \times \mathcal{U}_L$.
  That is, return all tuples $\langle *, \ldots, *, x, *, \ldots, * \rangle$ whose ith component is x.
- RANGE(i, r_1, r_2): return $\mathcal{S} \cap \mathcal{U}_1 \times \cdots \times \mathcal{U}_{i-1} \times [r_1, r_2] \times \mathcal{U}_{i+1} \times \cdots \times \mathcal{U}_L$, where $[r_1, r_2] = \{x \mid r_1 \leq x \leq r_2\}$.
  That is, return all tuples $\langle *, \ldots, *, x, *, \ldots, * \rangle$ in $\mathcal{S}$ for which $r_1 < x < r_2$.

We refer to the index on $\mathcal{U}_i$ as $\mathbf{I}_i$.

Observe that compound dictionaries are distinct from multi-dimensional indexes because delete and query operations on compound dictionaries specify only a single coordinate, whereas delete and query operations on a multi-dimensional dictionary might allow all or some of the coordinates of the deleted item or queried rectangle to be specified.

Compound Dictionaries in Databases.

The compound dictionary is one of the most (if not the most) widely used data-structural abstraction, because it appears in essentially every relational database management system (RDBMS). In database terminology, indexes are sometimes also called ***tables***, and the elements that are inserted and deleted are typically called ***rows***.

The actual specification of a database is slightly different: indexes can be defined on tuples of fields; deletions can only be specified on so-called ***primary keys***; and in some databases, only U_1 can be primary; some fields may not have any index associated with them; etc. Our version of the problem is similar enough to capture the essential algorithmic challenge of compound dictionaries.

The Complexity of Deletes in a Compound Dictionary.

Considering that compound dictionaries have been around for 720 years and are the basic data structure of databases, it may seem surprising that the algorithmic literature is largely silent on this data structure.

On the other hand, at first glance, there's not that much to say. Insertions, for example, into an L-dictionary are simply L times slower than an insertion into a single dictionary, on both a RAM and in external memory.

Now consider deletes. On a RAM, deletions take $\mathcal{O}(\log N)$ operations on a dictionary and $\mathcal{O}(L \log N)$ on an L-dictionary. As with insertions, a deletion from an L-dictionary can be decomposed into L deletions from regular dictionaries.

Even in external memory, the problem seemed trivial until recently. The B-tree [18] achieves optimal $\mathcal{O}(\log_B N + K/B)$ I/Os for range queries on K elements and $\mathcal{O}(\log_B N)$

I/Os for insertions and deletions. On an L-dictionary implemented using B-trees, the deletion cost is $\mathcal{O}(L \log_B N)$ I/Os. Once again, a deletion to the compound dictionary is a deletion on each dictionary.

But a little bit more is actually going on, because a deletion seems to require a search. Consider a 2-dictionary on $U_1 \times U_2$. An insertion of $\langle u, v \rangle$ consists of adding $\langle u, v \rangle$ into $\mathbf{I}_1$ ordered by u and into $\mathbf{I}_2$ ordered by v. A deletion $\textsc{delete}(1, u)$ seems to require $\textsc{lookup}(1, u)$ to fetch the pair $\langle u, v \rangle$, followed by removing $\langle u, v \rangle$ from both $\mathbf{I}_1$ and $\mathbf{I}_2$. In short, an actual delete from a constituent index requires knowing the key to be deleted. But this seems to require a query to find all the necessary keys.

For a B-tree, this query is not a problem. We get the desired bounds by noting that deletions take the same amount of time as searches. One query to get all keys does not slow down the L deletions from the individual dictionaries.

Compound Dictionaries and Write Optimization.

In recent years, both the theory and practice of external-memory dictionaries have been revolutionized by ***write-optimization*** techniques. Write-optimization is a somewhat informal concept in the file-system and database communities, but it boils down to this: a dictionary is write-optimized if insertions and deletions are substantially better than those of a B-tree, while point queries are as good or nearly as good.

The best (write-optimized) dictionaries maintain (optimal) $\mathcal{O}(\log_{1+B^\varepsilon} N + K/B)$ I/Os for range queries while achieving a substantially improved insertion and deletion cost of $\mathcal{O}(\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}})$ amortized I/Os, for $0 \leq \varepsilon \leq 1$, while [20, 27, 29].

Write optimization techniques are now widespread in the database world [12, 33, 51, 62, 91, 92] and are starting to have an impact on file systems [46, 56, 57, 75, 102].

Deletes and Write Optimization.

Write-optimized databases and file systems show a marked asymmetry between insertions/deletions and queries in that for $\varepsilon < 1$, the cost of inserting and deleting is much lower than the cost of a query. In such data structures, a delete is typically implemented

as the insertion of a tombstone message, which changes the state of the data structure so that subsequent queries no longer see the deleted item. Given the gap in the I/O budgets of insertion/deletions vs queries, it is not possible to determine if an insertion is overwriting a previous insertion, if a delete is deleting an item that actually belongs to the set, etc. This asymmetry introduces an algorithmic issue with compound dictionaries.

An L-dictionary composed of write-optimized dictionaries (WODs) takes time

$$\mathcal{O}\left(L\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}}\right) \tag{4.1}$$

to insert into all indexes. However, consider the deletion algorithm, which includes a search. Searches are much slower than insertions, and so the time to delete is

$$\mathcal{O}\left(\log_{1+B^\varepsilon} N + L\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}}\right). \tag{4.2}$$

Write optimization does help, because the L multiplies a low-order term, but deletions do not enjoy the full benefits of write optimization.

The alternative is to push the slowdown to the query: one could keep a data structure of all the deletions. Suppose that there is a set $D = \{d_1, d_2, ..., d_\ell\}$ of deletion $\textsc{delete}(1, d_i)$. A query $\textsc{range}(2, x, y)$ considers a sequence $\langle a_i, b_i\rangle$, where $x \leq b_i \leq y$. Some of these a_i might belong to D, and any such pair would need to be filtered out of the answer. These lookups in a data structure on D would slow down the queries, thus yielding deletions that match the write-optimization bound for deletions but with suboptimal queries.

In either case, the crux of the difficulty seems to be the jump from a single dictionary to a 2-dictionary. In the remainder of the chapter, we therefore restrict our attention to 2-dictionaries when talking about compound dictionaries.

Deletes and Databases.

So far, we have described the problem of deletes in write-optimized indexes. This problem is of algorithmic interest, certainly, because the run-time of deletes is a big gap in our understanding of indexing. However, we did not come to this problem originally from a consideration of algorithmic issues. Instead, while building TokuDB [93], we had to deal with the issue of deletions. Deletions are a big problem in the design of write-optimized

storage systems. What is particularly interesting to us in this problem is that the pragmatics of building a database so exactly line up with the algorithmics of compound dictionaries.

Warming Up.

Before we consider the problem of deletes in 2-dictionaries, we examine the simpler ***count*** problem on single dictionaries. In its simplest version, the count of a dictionary returns the cardinality of the set $\mathcal{S}$ being indexed.

In many instantiations of a dictionary, such as in a database, dictionaries support overwrite insertions, in which a new insertion with the same key replaces the old key. (Actually, the value associated with the key replaces the old value). In RAM, such operations takes $\mathcal{O}(\log N)$ time, and counts can be computed in $\mathcal{O}(\log N)$ time. In a B-tree, such operations take $\mathcal{O}(\log_B N)$ I/Os, and counts can be computed in $\mathcal{O}(\log_B N)$ I/Os.

In a WOD, however, insertions take very few I/Os compared to queries. There are not enough I/Os in an insertion to resolve whether a particular insertion is a new insertion or an overwrite. It seems that we need a query to resolve this issue, either at the time of insertion or at query time, in order to achieve an accurate count. Once again, the asymmetry between the cost of insertions and the cost of queries in a WOD seems to cause some algorithmic problems for some operations.

Our Results.

In this chapter, we warm up by showing that the count operation is slow if insertions are write optimized. Specifically, we show that it is impossible to achieve $\mathcal{O}(\log_B N)$ I/Os for count in the external-memory comparison model unless insertions take $\Omega(\log_B N)$. That is, no write optimization is possible at all for this problem. This result serves as both a first proof on the limits of write optimization and as a simplified proof that shows some of the techniques of the main result. The details can be found in 4.2.

In 4.3, we establish limits on write optimization for deletions on 2-dictionaries. We prove that one may either achieve fast deletes or optimal range queries but not both. Our result is not as general as the count result, because our lower bound establishes that some parts of the write-optimization tradeoff curve are not achievable, whereas in the count lower bound,

we show that no write optimization is achievable at all. We conjecture that if range queries are optimal, then deletes takes $\Omega(\log_B N)$ (in the I/O comparison model), that is, that no write optimization is possible for this problem either. We leave this conjecture for future work.

Related Lower Bounds.

Brodal and Fagerberg [29] derived lower bounds on the update/query tradeoff for external memory one-dimensional dictionaries. For the predecessor problem, they showed that to achieve query time $\mathcal{O}(\log_B N)$, updates must take $\Omega(\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}})$ I/Os amortized. They also constructed buffered-B trees that achieve this tradeoff, thus essentially solving the 1D predecessor problem for most choices of parameters.

Verbin and Zhang [95] considered problems like 1D range counting, membership, predecessor, etc., in dictionaries. They show that: if the update take is less than 1 I/O, queries must be roughly logarithmic in N; and if the update take $1 + o(1)$ I/Os, then hashing gives $\mathcal{O}(1)$ query time.

Ke Yi [101] considered the range query problem in dictionaries, and showed that essentially all known versions of dynamic B-trees are optimal for this problem, as long as $\log_B(N/M)$ is a constant.

4.2 Counts and Dictionaries

We define ***1-D count problem*** as follows:

- Static Insertion Phase: Preprocess set $\mathcal{S} = \{a_1, a_2, \ldots, a_N\}$.
- Dynamic Insertion Phase: Insert a sequence of $\sqrt{N}$ elements $\mathcal{D} = \{d_1, d_2, \ldots, d_{\sqrt{N}}\}$.
- Counting Phase: Output the count, $|\mathcal{S} \cup \mathcal{D}|$.

Theorem 4.1. *In the comparison-based external-memory model, for any algorithm that solves the 1-D count problem using $\mathcal{O}(N \log_B N)$ I/Os for the static insertion phase, there is a constant $c < 1$ so that if it performs at most $c\sqrt{N} \log_B N$ I/Os for the dynamic insertion phase, it must perform $\Omega(\sqrt{N} \log_B N)$ I/Os in the worst case to output the count $|\mathcal{S} \cup \mathcal{D}|$.*

Proof. Let the sorted order of $\mathcal{S}$ be $a_1 < a_2 < \cdots < a_N$. Suppose the adversary reveals that each d_k is in a disjoint subrange of $\mathcal{S}$ as follows: $a_1 \leq d_1 \leq a_{\sqrt{N}}, a_{\sqrt{N}+1} \leq d_2 \leq a_{2\sqrt{N}}, \ldots, a_{(\sqrt{N}-1)\sqrt{N}+1} \leq d_{\sqrt{N}} \leq a_N$. In the comparison-based model, the only information that the algorithm can learn about d_k is the set of possible a_i that might match d_k, i.e. that $d_k = a_i$, for some i, or that there it lies in some open interval (a_i, a_j), but the relative order of a_{i+1} and d_k is unknown, (and symmetrically with a_{j-1}). We say that d_k is ***resolved*** if we know that $d_k = a_i$ or $d_k \in (a_i, a_{i+1})$, for some i. Otherwise it is ***unresolved*** on some interval (a_i, a_j), $j > i + 1$. We note here that in this setting, the algorithm knows that $d_1 < d_2 < \cdots < d_{\sqrt{N}}$, so no extra information can be inferred by comparing pairs of elements in $\mathcal{D}$.

Suppose that the adversary reveals to the algorithm the additional information $|\mathcal{S} \cup \mathcal{D}|$ is either $N + \sqrt{N}$ or $N + \sqrt{N} - 1$. That means at most one member of $\mathcal{D}$ matches some member of $\mathcal{S}$.

To distinguish between the two cases, the algorithm can be forced to identify the predecessors and successors for the each d_k. The adversary never declares that an element of $\mathcal{D}$ is equal to an element of $\mathcal{S}$ and this forces the algorithm to resolve all the intervals. To see this, suppose at the end of Counting Phase some d_k is unresolved in interval (a_i, a_j), $j > i + 1$. For all the other members of $\mathcal{D}$, the adversary chooses to have those elements be distinct from the elements of $\mathcal{S}$; this is possible since the adversary has never declared the existence of an equality between elements of $\mathcal{D}$ and $\mathcal{S}$. Now, the adversary can choose to set $d_k = a_i$ or $a_i < d_k < a_{i+1}$ that results in an incorrect count.

Therefore, all the elements of $\mathcal{D}$ must be resolved, however, if d_k is resolved, then the algorithm knows the successor and predecessor of d_k. It is known that finding the predecessors for each d_k requires $\Omega(\sqrt{N} \log_B N)$ (say, at least $c^+\sqrt{N} \log_B N$) I/Os [22, Theorem 7]. We set $c = c^+/2$, and since the I/Os spent on the second phase is $c\sqrt{N} \log_B N$ (choose $c < c^+/2$), the third phase must pay off the difference $c^+\sqrt{N} \log_B N - c\sqrt{N} \log_B N \in \Omega(\sqrt{N} \log_B N)$, hence proving the theorem. □

4.3 Deletes and 2-Dictionaries

In this section we show that a 2-dictionary supporting optimal range queries cannot achieve the write-optimization bound ($\mathcal{O}(\frac{\log_{1+B^\varepsilon} N}{B^{1-\varepsilon}})$) for any $\varepsilon \in (0, 1/3)$. Brodal and Fagerberg [29] proved a lower bound on the update/query tradeoff for the predecessor problem in one-dimensional external memory dictionaries, and showed that the buffered-B^ε tree achieves the optimal write-versus-query tradeoff (same as the write-optimization bound mentioned earlier). Here we show that such a tradeoff is not possible in cross-referenced dictionaries.

Specifically, we show a lower bound for ***2-Dictionary Deletion Problem*** (2DD), which we define as the problem of performing the following compound-dictionary commands, during three different phases.

- Phase 1: Let $\mathcal{A} = \{a_i\}$ and $\mathcal{B} = \{b_i\}$ be sorted sets of N elements each. This phase consists of inserting a set $\mathcal{S} = \{\langle a_i, b_j\rangle\}$, by performing N insertions $\textsc{insert}(a_i, b_j)$. Define π by $a_i = \pi(b_j)$.
- Phase 2: This phase consists of a set $\mathcal{D} = \{d_i\} \subseteq \mathcal{A}$ of $\sqrt{N}$ deletions $\textsc{delete}(1, d_i)$ on the first coordinate.
- Phase 3: This phase consists of one range query $\textsc{range}(2, b_\ell, b_r)$ on the second coordinate.

In the general setting, inserts, deletes and range queries can be provided in any order. This general problem is obviously harder than the "three phase" problem defined above (the defined problem is an instance), so a lower bound on our problem is a lower bound on the general cross-referenced 2-dictionary maintenance problem.

Our main result is the following theorem:

Theorem 4.2. *For any data structure that solves the 2DD problem, if an insertion takes amortized $\mathcal{O}(\log_B N)$ I/Os and a deletion takes amortized $\mathcal{O}((\log_B N)/B^\alpha)$ I/Os for any constant $\alpha > 2/3$, then some range query $\textsc{range}(2, b_\ell, b_r)$ requires $\omega(\log_B N + K/B)$ I/Os, where $K = |\mathcal{B} \cap [b_\ell, b_r]|$ is the number of b's inserted (but not deleted) in the range $[b_\ell, b_r]$.*

4.3.1 Proof Outline.

As in the proof of Theorem 4.1, we specify that the members of $\mathcal{D}$ come from disjoint ranges of $\mathcal{A}$, each of size $\sqrt{N}$. We perform the allowed I/Os and find the uncertainty ranges for all the deletions. In this proof, we more carefully quantify the uncertainty that remains in all the deletions, because we need this uncertainty to be large enough to lower bound the number of I/Os of a range query.

In other words, counts take $\mathcal{O}(\log_B N)$ I/Os, whereas range queries take $\mathcal{O}(\log_B N + K/B)$. If K is large, then this term dominates the I/O complexity of a range query. Specifically, it is not enough simply to figure out that there are unresolved deletions, since the unresolved deletions could potentially be resolved while answering the range query. Thus, we need to *make sure that the I/Os required to resolve the deletion completely cannot be amortized against those used to answer the range query.*

To begin, we need to refine Theorem 7 from [22] (which states that not all searches can fully resolve in less than $c^+ \log_B N$ I/Os per query, for some constant c^+) with a stronger lower bound on the total size of the unresolved intervals. (See Lemma 4.4.)

Because the remaining uncertainty is large, there are many tuples in $\mathbf{I}_2$ that might be deleted. We want to find a range query that has many such potential deletions. In Lemma 4.6 we show that such ***hot*** regions exist they involves a sufficient number of different d_i's. We need to show that not too many of these d_i's can be filtered out as having been deleted (it is important to know that it is enough for the range query to simply not output any of the d_i's rather than fully resolve them). This is done by showing that not too many of these deletions can be fetched into memory as a byproduct when the algorithm fetches other d_i's. To guarantee this, we require π to satisfy two conditions, CA and CB, that roughly speaking require that π sufficiently "shuffles" the elements of $\mathcal{A}$ and $\mathcal{B}$. Lemma 4.3 guarantees the existence using the probabilistic method and Turán's Theorem [88].

4.3.2 Preliminaries.

We assume that every I/O can read/write a disk page capable of storing $B = \log^\tau N$ tuples for some sufficiently large constant τ, the physical memory can store $M = \mathcal{O}(N^\mu)$ tuples,

and the range query has size $K = N^\delta$ and $\mu < \delta < 1/4$ are constants. The constants τ and δ are found during the course of the proof.

Specifying the insertions and deletions.

Let $a_1, \ldots, a_N$ be the sorted order of $\mathcal{A}$ and $b_1, \ldots, b_N$ the sorted order of $\mathcal{B}$. We assume that $a_i \neq a_{i+1}$ and $b_i \neq b_{i+1}$, for $1 \leq i < N$, that is, $\mathcal{A}$ and $\mathcal{B}$ each have N distinct values. Recall that the N inserted tuples be $(\pi(b_i), b_i)$ for $i \in [N]$, where π is a mapping from $\{b_1, \ldots, b_N\}$ to $\{a_1, \ldots, a_N\}$.

We will break $\mathcal{A}$ into chunks of size $\sqrt{N}$ as we did in the proof of Theorem 4.1. We say that a_i has ***color*** k, abbreviated as $c(a_i) = k$, if $\lceil i/\sqrt{N} \rceil = k$. We say b_i has color k if $\pi(b_i)$ has color k and overload the color function so that $c(T) = \{c(b_i) : b_i \in T\}$. For every fixed constant $r \in (0, 1)$ and $1 \leq t \leq N^{1-r}$, we define the sets

$$S_{t,N^r} = \{b_i : \lceil i/N^r \rceil = t\}.$$

Not every π is suitable for our lower bound. Consider, for example, the degenerate case that $\pi(b_i) = a_i$ for all $i \in [N]$. If $\pi(b_i)$ is directly computable from a_i with no I/Os, then the theorem does not hold. We can insert a deletion message into both indices and write optimization works just fine.

Hence, to prove the theorem, we cannot choose π arbitrarily. We will pick a π that satisfies the following two conditions.

Ca. $c(b_i) \neq c(b_j)$ if $b_i \neq b_j$ and $b_i, b_j \in S_{t,\sqrt{N}}$ for some $t \in [\sqrt{N}]$.

In other words, the permutation must be compatible with the following: Break $\mathcal{B}$ into chunks of size $\sqrt{N}$ in order. Take the elements of each of these chunks and map them to some element in $\mathcal{A}$, so that no two elements in the same chunk of $\mathcal{B}$ fall within the same chunk of size $\sqrt{N}$ in $\mathcal{A}$.

Cb. $|c(S_{t,N^\delta}) \cap c(S_{t',N^\delta})| = \mathcal{O}(1)$ for every $t \neq t' \in [N^{1-\delta}]$, for some fixed constant $\delta \in (0, 1/4)$.

This is a crucial property. Note here that $|c(S_{t,\sqrt{N}}) \cap c(S_{t',\sqrt{N}})| = \sqrt{N}$ given Ca. This condition requires that if we break $\mathcal{B}$ into finer chunks of size N^δ, that we call ***subchunks***,

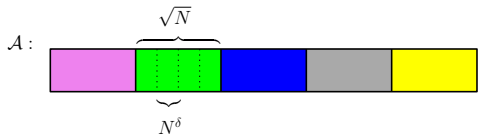

Figure 4.1: We decompose the ordered list of elements in $\mathcal{A}$ into $\sqrt{N}$ chunks of size $\sqrt{N}$. The elements in the same chunk are assigned the same color. Each chunk is further decomposed into subchunks of size N^{δ} for some parameter δ to be fixed later.

then the pairwise intersections must have constant size. See Figure 4.1. The following lemma shows the existence of such a π.

Lemma 4.3. *For every N and $\delta \in (0, 1/4)$, there exists some π that satisfies both* CA *and* CB.

Proof. To satisfy CA, it requires that $c(S_{t,\sqrt{N}}) = \{1, 2, \ldots, \sqrt{N}\}$ for every $t \in \{1, 2, \ldots, \sqrt{N}\}$. Hence, $c(b_1), c(b_2), \ldots, c(b_N)$ is a concatenation of $\sqrt{N}$ permutations of $\{1, 2, \ldots, \sqrt{N}\}$.

There are $(\sqrt{N})!$ such permutations but, to satisfy CB, some permutations cannot be placed together in the concatenation. We construct a graph $G = (V, E)$ to describe which permutations cannot be placed together. Each node in G denotes a permutation and thus $|V| = (\sqrt{N})!$. If two permutations cannot be placed together in the concatenation due to CB, then we connect the representative nodes by an edge. Because of symmetry, the graph is regular.

Here we upper bound the degree of each node. Let π_0 be a permutation specified by some fixed node and π_{rand} be a permutation specified by the node picked uniformly at random. Let T be the threshold constant in CB (i.e. $|c(S_{t,N^\delta}) \cap c(S_{t',N^\delta})| < T$). Let further $\mathbb{P}[[] i_1, i_2, \ldots, i_T; \pi_{rand}]$ be the probability that $i_1, i_2, \ldots, i_T$ fall within the same subchunk of π_{rand}. Then, each node in G has degree

$$
\begin{aligned}
d &= \left(\sqrt{N}\right)! \cdot \mathbb{P}[[] \pi_0 \text{ and } \pi_{rand} \text{ can't be placed together}] \\
&\le \left(\sqrt{N}\right)! \sum_{\substack{i_1, i_2, \ldots, i_T \text{ distinct, and} \\ \text{are in the same } \pi_0\text{'s subchunk}}} \mathbb{P}[[] i_1, i_2, \ldots, i_T; \pi_{rand}] \\
&\le \left(\sqrt{N}\right)! \left(N^{1/2-\delta}\right) \binom{N^\delta}{T} \left(\frac{1}{N^{1/2-\delta}}\right)^T \left(N^{1/2-\delta}\right) \\
&\le \left(\sqrt{N}\right)! \left(N^{1-2\delta-T(1/2-2\delta)}\right) \qquad \text{(note that } \delta < 1/4) \\
&\le \left(\sqrt{N}\right)!/N \qquad \text{(pick a sufficiently large } T)
\end{aligned}
$$

By Turán's Theorem, the graph $G = (V, E)$ has an independent set of size at least

$$\frac{|V|^2}{|V| + 2|E|} \geq \frac{\left(\left(\sqrt{N}\right)!\right)^2}{\left(\sqrt{N}\right)! + \left(\left(\sqrt{N}\right)!\right)^2 / N} = N - o(1),$$

meaning that some carefully chosen $\sqrt{N}$ permutations can be placed together in the concatenation without violating CB. As a result, the desired π exists. □

Given a mapping π that satisfies the both conditions, the adversary conducts the following ***adversarial sequence of insertions and deletions***:

- Insertion Phase: The adversary inserts, in any order, N tuples $(\pi(b_i), b_i)$ for every $i \in [N]$.
- Deletion Phase: The adversary deletes, in any order, $\sqrt{N}$ tuples $(d_k, *)$ for every $k \in [\sqrt{N}]$, where $d_k = a_i$ for some a_i whose color is k.

At the beginning of the deletion phase, each d_k, for $k \in [\sqrt{N}]$, might match any a_i whose color is k. We say that d_k has ***uncertainty*** u, abbreviated as $U(d_k) = u$, if the number of a_i's that can match d_k equals u. While performing the I/Os for deletions, some comparisons between a's and d's are made and thus the uncertainly $U(d_k)$ of any d_k might shrink, but we claim that not by too much, in aggregate, of all k. Here we prove a quantitative bound for the sum of the $U(d_k)$ at the end of the deletion phase.

Lemma 4.4. *Any algorithm for 2DD over an adversarial sequence that uses amortized $\mathcal{O}(\log_B N)$ I/Os per insertion and $\mathcal{O}(\log_B N/B^\alpha)$ I/Os per deletion, for any constant $\alpha \in (2/3, 1)$, has*

$$\sum_{k \in [\sqrt{N}], U(d_k) > 1} U(d_k) = \Omega(N/B^{1-\alpha})$$

at the end of the deletion phase.

Proof. It suffices to show that the desired lower bound holds even if the adversary reveals some information for each d_k at the beginning of the deletion phase. For each $k \in [\sqrt{N}]$, the adversary partitions the range $[(k-1)\sqrt{N} + 1, k\sqrt{N}]$ into B^r equal-sized consecutive subranges for some constant r determined later. See Figure 4.3. It then randomly picks a

subrange and reveals to the algorithm that d_k equals some a_i in the subrange. After such a revelation,

$$\sum_{k\in[\sqrt{N}],U(d_k)>1} U(d_k) = \Omega(N/B^r).$$

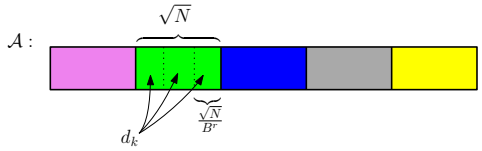

Figure 4.2: Partitioning a chunk into B^r subranges. d_k will be chosen inside the some subrange inside the k-th chunk.

Claim 4.5. *For some combination of randomly picked subranges and $r \in (0, 1/3)$, the sum of uncertainty $\Omega(N/B^r)$ cannot be further narrowed down by any superconstant factor at the end of the deletion phase.*

Proof. Let us consider the I/Os performed during the deletion phase. These I/Os can bring a's from disk to memory for subsequent comparisons with d's, and thus the uncertainty of d's can be reduced. For each a that is brought into memory, it is only possible to reduce the uncertainty of one d. We assume that the adversary reduces the uncertainty of the appropriate d, even if that d isn't in memory. Thus, we give the algorithm more power than any actual algorithm could have.

We say that some a is ***fresh*** if it has not yet been brought into memory since the beginning of the deletion phase, and therefore only fresh a's can be used to reduce the uncertainty further given the assumption. We note that only fetching the disk pages written in the insertion phase can give the algorithm fresh a's. Those written in the deletion phase cannot, since all uncertainty is maximally reduced when an a is fetched during the deletion phase. There are $\mathcal{O}(N \log_B N)$ disk pages written in the insertion phase, and therefore the number of disk pages that can contain some fresh a's is $\mathcal{O}(N \log_B N)$.

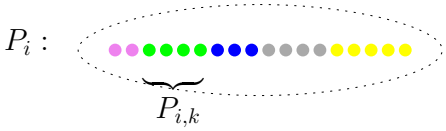

Figure 4.3: The i-th block written during the insertion phase. It may contain elements of different color with $P_{i,k}$ referring to the elements that have color k.

Let $P_i \subseteq \{a_j : j \in [N]\}$ denote the a_j's contained in the ith disk page written in the insertion phase. Note that $|P_i| \leq B$. Let $P_{i,k} = \{a_j \in P_i : c(a_j) = k\}$ (see Figure 4.3). We partition P_i into two disjoint sets H_i and L_i, where $L_i = P_i \setminus H_i$ and

$$H_i = \{a_j \in P_i : |P_{i,c(a_j)}| \geq B^r\}.$$

That is H_i is the set of elements in P_i whose color is frequently represented in P_i. Let R_k be the randomly picked subrange for d_k. Let $X_{i,k}$ denote the random variable $|R_k \cap P_i|$. Then $X_{i,k} \in [0, |P_{i,k}|]$, $\mathbb{E}[X_{i,k}] = |P_{i,k}|/B^r$, and all $X_{i,k}$'s are independent for every fixed i. Let

$$Y_i = \sum_{k \in [\sqrt{N}], P_{i,k} \subseteq L_i} X_{i,k},$$

and from linearity of expectation

$$\mathbb{E}[Y_i] = \sum_{k \in [\sqrt{N}], P_{i,k} \subseteq L_i} \mathbb{E}[X_{i,k}] = \frac{|L_i|}{B^r}.$$

By Hoeffding's inequality [53], we have

$$\begin{aligned}\mathbb{P}\left[Y_i - \mathbb{E}[Y_i] \geq B^{1-r}\right] &\leq \exp\left(-\frac{2(B^{1-r})^2}{\sum_{\substack{P_{i,k} \subseteq L_i \\ k \in [\sqrt{N}]}} |P_{i,k}|^2}\right) \\ &\leq e^{-\Omega\left(\frac{B^{2-2r}}{B^{1+r}}\right)},\end{aligned}$$

which is $e^{-B^{\Omega(1)}} = e^{-\log^{\Omega(\tau)} N} = 1/N^2$ if we pick any constant $r \in (0, 1/3)$ and pick τ to be sufficiently large. By the union bound, we know that for some combination of R_k for $k \in [\sqrt{N}]$,

$$Y_i \leq \mathbb{E}[Y_i] + B^{1-r} \leq 2B^{1-r}$$

for every disk page written in the insertion phase.

The adversary picks some such combination of R_k, and reveals the information to the algorithm. No matter what $\mathcal{O}(\sqrt{N} \log_B N/B^\alpha)$ I/Os are fetched by the algorithm in the deletion phase — w.l.o.g. let them be $P_1, P_2, \ldots, P_T$ for $T = \mathcal{O}(\sqrt{N} \log_B N/B^\alpha)$ — we have:

- The number of colors contributed by H_i for $i \in [T]$ is at most $(\sqrt{N} \log_B N/B^\alpha)(B/B^r)$.

- The number of a_j's contributed by L_i for $i \in [T]$ is at most $(\sqrt{N} \log_B N/B^\alpha)(2B^{1-r})$.
- The number of a_j's is in memory at the beginning of the deletion phase is at most $M = o(N^\delta)$.

If we pick $\alpha > 1 - r$, there are $o(\sqrt{N})$ d_k's whose uncertainty can be further narrowed down by the a_j's fetched by some H_i. Furthermore, the number of a_j's contained in some L_i and in memory at the beginning of the deletion phase is bounded by $o(\sqrt{N})$, which means that few d_k's can have a comparison with a_j in some L_i to further narrow down the uncertainty. As a result, $\Omega(\sqrt{N})$ d_k's have the uncertainty unchanged since the revelation, yielding the total uncertainty $\Omega(N/B^r)$. □

Since r can be any constant in $(0, 1/3)$, then α can be any constant in $(2/3, 1)$. By Claim 4.5, we complete the proof of 4.4. □

After performing all deletions, the number of disk pages that contain some d_i for $i \in [\sqrt{N}]$ is at most $(\sqrt{N} \log_B N)/B^\alpha$ (i.e. no more than the budget of I/Os for deletions).

We are now in a position to prove the existence of a range query that requires superlinear number of I/Os. Observe that if a range query contains some b_j whose $\pi(b_j)$ still might match some d_i, to answer the range query correctly, the algorithm must, due to CA: (1) fetch some disk page that contains d_i, and (2) compare b_j with d_i to see whether b_j is deleted. By Lemma 4.4, we know that there are $\Omega(N/B^{1-\alpha})$ such b_j's and thus some range query of size N^δ has $\Omega(N^\delta/B^{1-\alpha})$ such b_j's, which is more than the claimed budget $\mathcal{O}(\log_B N + N^\delta/B)$. We note here that the number of d's that are already in memory at the beginning of the range query phase is $M = \mathcal{O}(N^\mu) = o(N^\delta/B^{1-\alpha})$, and thus are insufficient to change the bound. By the Markov inequality, we can say something stronger:

Lemma 4.6. *There are $\Omega(N^{1-\delta}/B^{1-\alpha})$ range queries of size N^δ so that, to answer any of the queries correctly, any algorithm needs to fetch $\Omega(N^\delta/B^{1-\alpha})$ different d_i's for the required comparisons.*

However, the observation is not sufficient to prove Theorem 4.2 because a single I/O might fetch back multiple d_i's for the required comparisons. That is the reason why we need CB. Observe further that every such expensive query RANGE$(2, (i-1)N^\delta + 1, iN^\delta)$ needs

the existence of some disk page that contain $\Omega(B^\alpha)$ different d_j's for required comparisons, denoted by the set D_i. Note that $c(D_i) \subseteq c(S_{i,N^\delta})$ and therefore $|c(D_i) \cap c(D_j)| \leq |c(S_{i,N^\delta}) \cap c(S_{j,N^\delta})| = \mathcal{O}(1)$ for every $i \neq j$. Since there are at most $o(\sqrt{N})$ disk pages containing some d_i, and each of the disk page can be a superset of $\mathcal{O}(B^{1-\alpha})$ different D_i's because $|c(D_i) \cap c(D_j)| = \mathcal{O}(1)$ for $i \neq j$, $B^\alpha > B^{2/3} > \sqrt{B}$ and the following lemma:

Lemma 4.7. *Let $T_1, T_2, \ldots, T_C$ be the subsets of S, where $|T_i \cap T_j| = \mathcal{O}(1)$ for every $i \neq j \in [C]$ and $|T_i| = \Delta = \omega(\sqrt{|S|})$ for each $i \in [C]$, then $C = \mathcal{O}(|S|/\Delta)$.*

Proof. We prove this by a counting argument. Consider the elements in $D_i = T_i \setminus \bigcup_{j<i} T_j$. Since $|T_i \cap T_j| = \mathcal{O}(1)$ for every $j < i$, $|D_i| = \Delta - \mathcal{O}(i)$. We are done because

$$|S| \geq \sum_{i \in [C]} |D_i| = \sum_{i \in [C]} |T_i| - \mathcal{O}(i)$$

and thus $C = \mathcal{O}(|S|/\Delta)$, where the last equality holds due to the fact that $\Delta = \omega(\sqrt{|S|})$. □

The number of different D_i's that are subsets of some disk page is only $o(N^{1/2}B^{1-\alpha})$. However, the number of different D_i's that are needed for the expensive range queries is $\Omega(N^{1-\delta}/B^{1-\alpha})$, implying that some range query requires $\omega(\log_B N + K/B)$ I/Os. This establishes 4.2.

4.4 Conclusion

In this chapter, we consider issues of both practical and theoretical importance in implementations of and algorithms for dictionaries. The development of write optimization has reduced the cost of insertions and deletions. As this tide of insertion/deletion cost recedes, the cost of queries becomes significant in many settings.

We show that natural operations, including count in single dictionaries and delete in compound dictionaries, limit the applicability of write optimization. Our lower bounds correspond to our experience, that these operations do, in fact, sometimes reduce the benefits of write optimization and can become bottlenecks of actual systems.

In addition to showing lower bounds that start to put a boundary around the applicability of write optimization and that provide an explanation for the difficulty of implementing

fast versions of some operations in databases and file systems, we consider one of our contributions to be the introduction of a set of problems around compound dictionaries, which are a heretofore poorly studied aspects of dictionaries, despite being one of the most common ways in which they are used.

We leave one major open question: can the lower bound for deletes in 2-dictionaries be extended to the entire write-optimization range and raised to show that deletes take $\Omega(\log_B N)$ time, in compound dictionaries with optimal range queries? In other words, can it be shown that each delete requires a search?

In this chapter we worked in the external memory comparison model. One natural question to investigate is whether hashing allows for write-optimization in a cross-referenced dictionary. By an easy application of hashing, we can show that one can achieve write-optimized bounds for insertions, $\mathcal{O}(1)$ update time for deletions, and answer range queries optimally. Thus the picture is already different since in the comparison model we conjecture an $\Omega(\log_B N)$ update I/O cost for our problem. We believe there is potential to prove non-trivial cell probe lower bounds for our problem but we remark that doing so very likely is going to require investigating the batched predecessor problem [22] in the cell-probe model, which is a difficult problem. We leave this for future work.

Chapter 5

rSUM-hardness yields APX-hardness

5.1 Problem Definition and Background

Recently, surprising connections have been established between exponential hardness and polynomial hardness. Specifically, the strong exponential time hypothesis (SETH) has been used to prove super-linear lower bounds for problems in P [1, 14, 26, 74, 76]. In this chapter, we establish complementary results: that hardness of a predicate in P can imply APX-hardness of the corresponding maximization problem.

Consider the ***3SUM*** problem of deciding if a set of integers has a multisubset of size 3 that sums to 0 [48]. This problem has a natural generalization to the set of ***rSUM*** problems. It is known that rSUM-hard problems have an $\Omega(n^{\lceil r/2 \rceil})$ lower bound in some models [7, 45], and the ***rSUM Conjecture*** [1, 73] states that these problems cannot be solved in $\mathcal{O}(n^{\lceil r/2 \rceil - \varepsilon})$ time for any constant $\varepsilon > 0$, on a RAM.

We define Max-rSUM to be the maximization version of these problems: Given a set S of integers, find the largest $T \subseteq S$ so that T has no multisubset of size r that sums to 0, that is, the largest $T \subseteq S$ that ***fails*** the rSUM test. Our first result establishes the hardness of Max-rSUM:

Theorem 5.1. Max-rSUM *is* APX*-complete for every fixed* $r \geq 3$.

This theorem implies the APX-hardness of many problems, because:

Observation 5.2. *There are many* rSUM*-hard decision problems* $\mathcal{P}$ *whose hardness reduction can be directly restated as an L-reduction from* Max-rSUM *to Max-*$\mathcal{P}$.

Examples [17, 25, 48] include:

- Max-$\mathcal{P}_{3P1L}$: Given $S \subset \mathbb{R}^2$, find the largest $T \subseteq S$ so that T contains no three (distinct) colinear points.
- Max-$\mathcal{P}_{\delta\Delta\text{-}free}$: Given $S \subset \mathbb{R}^2$, find the largest $T \subseteq S$ so that T contains no three (distinct) points that form a triangle with area at least δ for any constant δ.
- Max-$\mathcal{P}_{3AP\text{-}free}$: Given $S \subset \mathbb{Z}$, find the largest $T \subseteq S$ so that T contains no three (distinct) integers that form an arithmetic progression.
- Max-$\mathcal{P}_{3L1P}$: Given S, a set of lines in $\mathbb{R}^2$, find the largest $T \subseteq S$ so that T contains no three (distinct) lines that intersect at a point.

Theorem 5.1 implies these problems Max-$\mathcal{P}$ are APX-hard.

Some rSUM-hard problems $\mathcal{P}$ have no known hardness reduction from the corresponding rSUM problem that can be restated as an L-reduction from Max-rSUM to Max-$\mathcal{P}$, and therefore Theorem 5.1 does not directly establish their APX-hardness. For example, deciding if a set is a Golomb ruler (i.e. whether all pairwise differences are distinct) is 4SUM-hard but its hardness reduction does not translate into an L-reduction for the maximization problem. The problem of finding a maximum-size subset that is a Golomb ruler was not previously known to be APX-hard, though an additive inapproximability result was known [67].

In short, Observation 5.2 is limited. One of the main contributions of this chapter is to generalize the techniques used to prove Theorem 5.1 in order to establish the APX-hardness of a larger class of problems.

Reformulating rSUM as Root-free Subset Problems rSUM problems are sometimes encoded as root-testing problems in polynomials [31,59], and the natural polynomials encoding rSUM have been generalized to encode a wider class of problems [16]. We take a similar approach here.

Let $f \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ be an r-variate polynomial, and let S be ***f-root-free*** if for all distinct $x_1, x_2, \ldots, x_r \in S$, $f(x_1, x_2, \ldots, x_r) \neq 0$. We denote by $\mathcal{P}_f$ the predicate of deciding if a given set S *has* a root for f whose coordinates are distinct, i.e. S is ***not*** f-root-free, and

by Max-RF(f) the problem of finding the largest subset of a given $S \subset \mathbb{Z}$ that is f-root-free. We call f the ***characteristic polynomial*** of $\mathcal{P}_f$ and Max-RF(f).

Consider $f_3(x, y, z) = 3x(2x + y)(x + y + z)$. Then, $\mathcal{P}_{f_3}$ = 3SUM and Max-RF(f_3) = Max-3SUM. The simpler polynomial, $x + y + z$, is *not* the characteristic polynomial of 3SUM, because items in 3SUM are allowed to be repeated, whereas our root-free formalism requires distinct choices. The polynomial f_3 covers triple, double or no repetitions. We observe in Section 5.6 that, for every $r \geq 3$, there exists an f_r that is the characteristic polynomial of rSUM.

In the following, we present some classes of polynomials whose root-free-maximization problems are APX-hard. For example, the characteristic polynomial for the Golomb ruler is $f_{GR} = (x-2y+z)(x-y-z+w)$, which is a product of linear terms of at least 3 variables. Theorem 5.5 below covers this simple syntactic criterion and we conclude:

Corollary 5.3. *It is* APX*-hard to find the cardinality of the maximum-size subset of a given $S \subset \mathbb{Z}$ that is a Golomb ruler. That is,* Max-RF(f_{GR}) *is* APX*-hard.*

Not all characteristic polynomials yield APX-hard maximization problems. For example, if $f(x) = x$, then Max-RF(f) is trivial. Other polynomials, including all those in $\mathcal{F}_r = \{f_r \mid r \geq 3\}$, induce maximization problems that are APX-hard. In the following, our goal is to find a large class $\mathcal{F}$ of polynomials, with $\mathcal{F}_r \subseteq \mathcal{F}$, such that, for every $f \in \mathcal{F}$, Max-RF(f) is APX-hard. We begin with the following restricted class of polynomials that does not yet include $\mathcal{F}_r$.

Theorem 5.4. *If $f \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ is a homogeneous linear polynomial consisting of $r \geq 3$ variables, then* Max-RF(f) *is* APX*-hard and cannot be approximated to within $1-\varepsilon_r$ unless* P = NP *for any*

$$\varepsilon_r < \Delta_r \equiv \frac{1}{744.64 + 601.44(r-3)}.$$

This class is not general enough to encode rSUM, which motivates our quest to consider larger classes of polynomials.

Combining Linear Polynomials Theorem 5.4 restricts the characteristic polynomials to be linear. We show two ways to combine linear polynomials that yield APX-hard

maximization problems.

Theorem 5.5. *Let* $f = \prod_{i=1}^{k} \ell_i$ *where* k *is a constant and each* $\ell_i \in \mathbb{Z}[x_1, x_2, \ldots, x_{r_i}]$ *is a homogeneous linear polynomial consisting of* r_i *variables. If* $r = \max_{1 \le i \le k} r_i$ *is at least* 3*, then* Max-RF(f) *is* APX*-hard, and cannot be approximated to within* $1 - \varepsilon_r$ *unless* P = NP *for any* $\varepsilon_r < \Delta_r$*.*

This class of polynomials includes $\mathcal{F}_r$, as desired, as well as f_{GR}. We can think of the products of polynomials as a disjunctive combination, in that an r-tuple is a root of f if it is a root of any of the constituent linear polynomials. We also have a ***conjunctive*** generalization, as follows. We say an r by k matrix M is ***strongly full rank*** if $k \le r$ and every $k \times k$ submatrix of M is full rank. Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ be vectors of a same dimensionality, and let $\mathbf{M} = (\mathbf{v_1}|\mathbf{v_2}|\ldots|\mathbf{v_k})$ be the matrix where $M_{ij} = \mathbf{v_j}[i]$. We call $\mathbf{M}$ the ***aggregation*** of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$. We say a vector space is in ***general position*** if it has a set of basis vectors whose aggregation is strongly full rank.

Theorem 5.6. *Let* $f = \sum_{i=1}^{k} \ell_i^2$ *be a* r*-variate polynomial*[1] *where* k *is a constant and each* $\ell_i \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ *is a homogeneous linear polynomial. If the solution set of* $f = 0$ *is in general position and has dimension* d *at least* 2*, then* Max-RF(f) *is* APX*-hard, and cannot be approximated to within* $1 - \varepsilon_r$ *unless* P = NP*, for any*

$$\varepsilon_r < \Gamma_r \equiv \frac{1}{140(6\lceil (r-2)/6 \rceil + 1)}.$$

As promised, Theorem 5.6 can be viewed as the ***conjunctive*** generalization of Theorem 5.4, because the f-root-freeness of some set S implies that S is ℓ_i-root-free for every i. A natural problem in this subclass is the Max-rAP problem for any $r \ge 3$, finding the largest subset containing no r-term arithmetic progression.

Related Work. A similar generalization from rSUM problems [31,59] to a wider class of problems (replacing sum polynomials with more general polynomials) has also been found useful in studies of the time complexity of rSUM-hard problems in P, because the sum polynomial may be not sufficient to encode an rSUM-hard problem but a more general

[1] *We note here that* $r - 1 \ge d$ *and* $d \ge 2$ *together yield* $r \ge 3$*.*

polynomial may [16]. Our focus, in considering larger classes of polynomials, is to understand the APX-hardness of Max-rSUM-hard problems.

We present a class of approximation problems that are APX-hard because of a simple syntactic criterion. In that respect, there is some similarity to prior work on the MaxOnes problem. In [61], syntactic criteria were presented for certain MaxOnes problems, that also imply APX-hardness. Related topics were also discussed in [15, 60]. Our results are not closely related to [15, 60, 61]; the full version of this chapter will compare and contrast our results in more detail.

Max-RF(f)			Hardness to Compute		Applications
Problem	f	Lower Bound	Known	This chapter	
Max-3SUM	$3x(2x+y)(x+y+z)$	$\geq \lvert S\rvert/2$	-	$(1-\varepsilon)$Opt	Max-3SUM-hard [17, 48]
3AP-free subset	$(x+y-2z)$	$\geq \lvert S\rvert^{1-o(1)}$ [19, 42, 68, 79]	-	$(1-\varepsilon)$Opt	Matrix Mult [35, 36, 63, 99]
Golomb ruler (aka Sidon set)	$(x-2y+z)(x-y-z+w)$	$\sqrt{\lvert S\rvert}(1-o(1))$ [44]	Opt $-\,\delta$ [67]	$(1-\varepsilon)$Opt	Telecom Engineer [37, 40, 67, 84, 90].
rAP-free subset for fixed $r \geq 3$	$\sum_{i=1}^{r-2}(x_i - 2x_{i+1} + x_{i+2})^2$	$\geq \lvert S\rvert^{1-o(1)}$ [70]	-	$(1-\varepsilon)$Opt	van der Waerden no. [82, 96]

Table 5.1: Improved results for the problems that have a polynomial encoding.

Implications. We summarize the improved hardness results for the problems that have a polynomial encoding in Table 5.1, and for the problems that do not have a polynomial encoding as follows. The dependence diagram of all results is illustrated in Figure 5.1.

- A QPTAS algorithm for finding the largest subset of non-intersecting segments in the plane is known [2]. Unless the ETH fails, this problem cannot be APX-hard. Theorem 5.1 and Observation 5.2 together yield the APX-hardness of Max-$\mathcal{P}_{3L1P}$, implying that the problem Max-$\mathcal{P}_{3S1P}$ of finding a largest subset of a set of line segments so that no three line segments intersect at a point is APX-hard. This gives a sharp separation between the hardness of excluding pairs of intersecting line segments versus triples.

  We note that the separation is new because the latter problem was only known to be NP-hard via a reduction from finding an Mis (maximum independent set) of intersection graphs on segments [32, 49].

- Some of the problems we consider, notably Max-3AP, and Max-RF(f_{GR}), appear to be more general than the problem that is usually studied. For example, one of the most important uses of 3AP-free sets arises in matrix multiplication [35, 36, 63, 99], where $S = [n] = \{1, 2, \ldots, n\}$. There is a long line of research [19, 42, 43, 68, 79] on how to construct large 3AP-free subsets of $[n]$, but the hardness of computing an optimal subset itself remains open, and our results say nothing about the hardness of this problem. Similarly, our results say nothing about computing the maximum Golomb-ruler subset of $[n]$ [67].

  To see why, notice that these problems have tally representations that are ***sparse languages***. For example, to compute the largest 3AP-free subset of $[n]$, the input is n, and takes $\mathcal{O}(\log n)$ bits, whereas the output is an 3AP-free set, which is exponentially larger. The standard trick of computing the size of the 3AP-free subset and encoding the input $[n]$ in the tally representation yields a sparse language in NP. Such sparse languages, which arise in many setting, such as in lattice problems in statistical physics (survey in [97]) or in determining Ramsey numbers (survey in [81]), are known not to be NP-complete unless P = NP [72].

  On the other hand, the fastest known algorithms for computing, say, the largest 3AP-free subsets of $[n]$ rely on branch-and-bound [41, 50], and hence require good bounding conditions. That is, they rely on algorithms for approximating the cardinality of the largest 3AP-free subset of a given set S. So until a fast algorithm for computing the largest 3AP-free subset of $[n]$ directly is devised, the generalization to arbitrary sets will remain important to many applications.

- Our techniques used to prove the APX-hardness of Max-rSUM can provide better hardness results for optimization problems in graph theory. Specifically, we slightly modify the many-one reduction used in Lemma 5.8 to prove Lemma 5.19, the APX-hardness of finding MIS for graphs of girth at least any constant r, which was known to be NP-hard [69]. Furthermore, Lemma 5.19 implies Corollary 5.15, the APX-hardness of finding the largest (in terms of the number of nodes) node-induced r-cycle-free subgraph for any $r \geq 3$, denoted by Max-C_r-free, which was also known to be NP-hard [64].

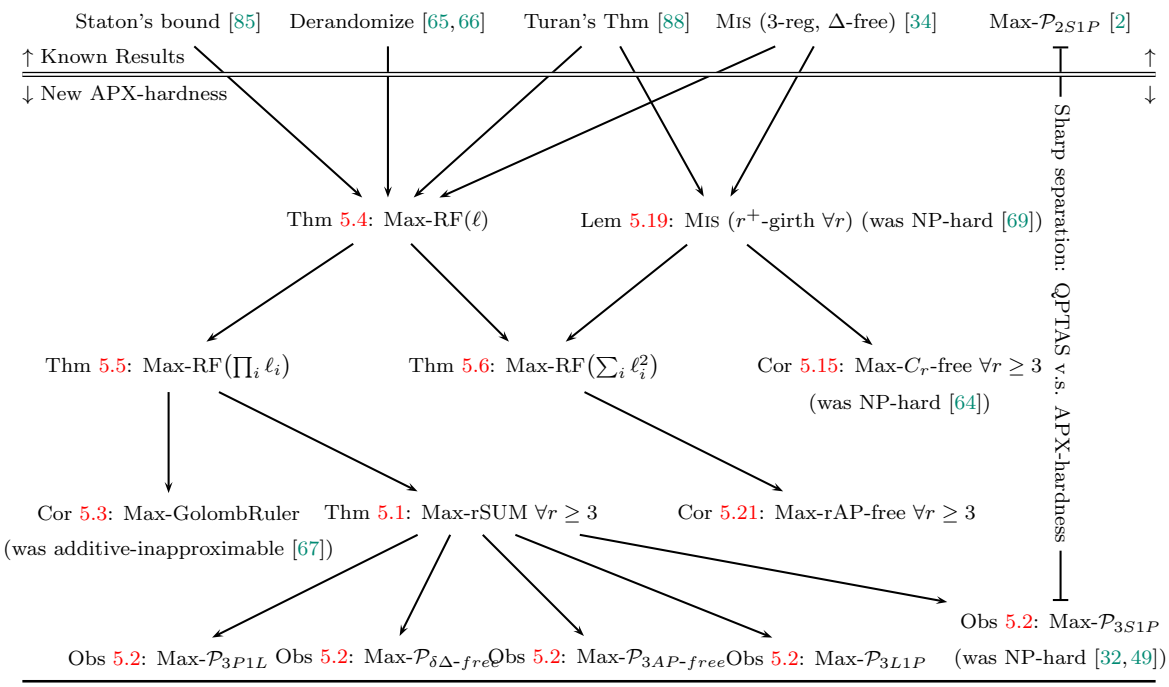

Figure 5.1: The dependence diagram of the results in this chapter.

Our Techniques and Organization. In Section 5.2, we prove the APX-hardness of Max-3SUM, as a warmup to the full proof. First, we show the existence of an NP-hardness reduction from MIS to Max-RF($x+y+z$) by a probabilistic proof, relying on the Schwartz-Zippel Lemma [80, 103]. Then, the probabilistic proof is derandomized [65, 66] so as to construct a deterministic, polynomial-time NP-hardness reduction. However, such a reduction is not approximation-preserving. To remedy this issue, we restrict MIS to 3-regular graphs, because every $\mathcal{O}(n)$-edge graph has independence ratio above some positive constant due to Turán's Theorem [88], thereby yielding a PTAS reduction. Lastly, we extend the APX-hardness of Max-RF($x+y+z$) to Max-3SUM by appealing to the Erdös construction of 3AP-free sets [43].

In Section 5.3, we prove the APX-hardness of Max-RF(f) for any homogeneous, 3^+-variate, linear polynomial f by the same approach as that in Section 5.2, but pay more attention to the structure of f. Since we are generalizing f, we need to carefully avoid letting f be zeroed by substituting the variables in f with a set of dependent random variables used in the probabilistic proof. In addition, we discuss the sharpness of the three constraints by showing that, for any constraint we leave out, there exists a polynomial f so

that Max-RF(f) is in P.

In Section 5.4, we give a proof of Theorem 5.5. We begin by defining the canonical representation of f and the equivalence classes of polynomials under Max-RF($\cdot$). Then, we pick a suitable polynomial ℓ_* from the ℓ_i's, where a tricky case appears when f is 3-variate. In the tricky case, ℓ_* is selected from the equivalence class of $(x + y - z)$. Then, our approach conducts multiple hardness reductions used for Max-RF(ℓ_*); a probabilistic argument then shows that, for one of these reductions, the solutions for Max-RF($f = \prod_i \ell_i$) and Max-RF(ℓ_*) coincide.

In Section 5.5, we give a proof of Theorem 5.6. To prove the case for $d = r - 1$, Theorem 5.4 suffices. For $d < r-1$ in general, dependence increases between the coordinates in a solution, thus requiring the solution set of $f = 0$ to be in general position. To further avoid the undesired dependence, we prove a helper lemma showing that Mis for graphs of girth $\geq r$ for any r is APX-hard, from which we are able to conduct a PTAS reduction to Max-RF($f = \sum_i \ell_i^2$).

Lastly, in Section 5.6, we study the inapproximability constant. We first appeal to Turán's Theorem [88] to obtain an initial bound. We also discuss (see also [8, 83]) why the AKS Theorem [9], which is stronger than Turán's Theorem for triangle-free graphs, does not yield a better inapproximability constant. Then we provide an improved inapproximability constant using an extremal graph result by Staton [85].

5.2 Hardness of Max-3SUM

In this section, we prove the hardness of Max-RF(f^*) where $f^*(x, y, z) = x+y+z$ and defer a proof for the general case to Section 5.3. The proof of the hardness of approximating Max-RF(f^*) will serve as intuition for the general case. We modify the proof for the hardness of Max-RF(f^*) to show the hardness of Max-3SUM, which implies the hardness of the maximization version of numerous 3SUM-hard problems whose hardness reduction satisfies Observation 5.2.

NP-hardness. We claim the existence of a polynomial-time many-one reduction from instances of Mis to instances of Max-RF(f^*). Let n-node m-edge graph $G = (V, E)$ be an

instance of MIS. We need a mapping h from $V \cup E$ to a set S of $n+m$ integers so that G has an independent set of size k iff S has a f^*-root-free subset of size $k+m$. We show that such a set S exists by the probabilistic method [11] and present how to construct S deterministically in time polynomial in n, using derandomization [65, 66].

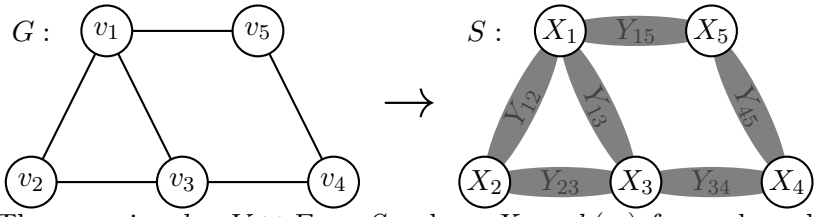

Figure 5.2: The mapping $h : V \cup E \to S$, where $X_i = h(v_i)$ for each node $v_i \in V$ and $Y_{ij} = h(\{v_i, v_j\})$ if $\{v_i, v_j\} \in E$.

Theorem 5.7. Max-RF$(x+y+z)$ *is* NP*-hard.*

Proof. To implement a mapping $h : V \cup E \to S$, we will use an ***n-order superposable set*** w.r.t. the polynomial f^*, which we define as follows. For any set B of n integers $X_1, X_2, \ldots, X_n$, we define the auxiliary set A_{f^*} induced by B and f^* to be $\{Y_{ij} : f^*(X_i, X_j, Y_{ij}) = 0, i \neq j\}$. We say B is an n-order superposable set if $|B \cup A_{f^*}| = n + \binom{n}{2}$, and for every three distinct elements $a_1, a_2, a_3 \in B \cup A_{f^*}$, $f^*(a_1, a_2, a_3) = 0$ iff $\{a_1, a_2, a_3\} = \{X_i, X_j, Y_{ij}\}$ for some $i \neq j$.

Given the superposable set B, one can realize a mapping $h : V \cup E \to S$, where $h(v_i) = X_i$ for $v_i \in V$ and $h(\{v_i, v_j\}) = Y_{ij}$ for $\{v_i, v_j\} \in E$, as illustrated in Figure 5.2. The following lemma will establish that the image set S and graph G preserve the relation required in the many-one reduction.

Lemma 5.8. *An n-node m-edge graph $G = (V, E)$ has an independent set of size k iff the set $S = h(V \cup E)$ has a f^*-root-free subset of size $k+m$.*

Proof. ($\Rightarrow$) Let I be an independent set. Every $I \cup E$ corresponds to a set $T = h(I \cup E)$, a subset of S. Since I is an independent set, for every edge $\{v_i, v_j\}$, the two integers X_i, X_j are not simultaneously contained in T. By the definition of a superposable set, T is f^*-root-free since it does not contain all three of X_i, X_j, Y_{ij}.

($\Leftarrow$) Let T be a f^*-root-free subset of S. For each edge $\{v_i, v_j\} \in E$, if both $X_i, X_j \in T$, then $Y_{ij} \notin T$ because T is f^*-root-free. In that case, one can modify T by replacing X_i with

Y_{ij}. Such a modification does not change the size of T but reduces the number of pairs of (X_i, X_j) in T where the corresponding nodes v_i, v_j are adjacent. One can repeat the change until there is no such (X_i, X_j) pair. Hence, G has an independent set of size at least k. □

Let $R_p(n)$ be a set of n integers $r_1, r_2, \ldots, r_n$ sampled uniformly at random from the universe $U = \mathbb{Z}_p$, for some prime p. In Lemma 5.9, we prove that, for sufficiently large p, $R_p(n)$ is a superposable set with positive probability. We choose $\mathbb{Z}_p$ to facilitate the derandomization. However, if a set is superposable under $\mathbb{Z}_p$, then it is superposable under $\mathbb{Z}$. After the construction, we use this superposable set under $\mathbb{Z}$.

Lemma 5.9. *The probability that $R_p(n)$ is an n-order superposable set is $1 - \mathcal{O}(n^6/p)$.*

Proof. We note that for any pair of different linear polynomials, assigning an integer sampled uniformly at random from a universe U to each variable in the polynomials makes the two polynomials equal in $\mathbb{Z}_p$ with probability $p_{eq} = 1/|U|$, by a simple version of the Schwartz-Zippel Lemma [80, 103]. Here $U = \mathbb{Z}_p$ and $1/|U| = 1/p$. In subsequent sections, we will replace U with another set and will rely more heavily on the Schwartz-Zippel Lemma.

To show $B = R_p(n)$ is superposable, we consider the two probabilities:

$$\mathbb{P}\left[|B \cup A_{f^*}| < n + \binom{n}{2}\right] \leq \sum_{X_i, X_j \in B} p_{eq} + \sum_{X_i \in B, Y_{ij} \in A_{f^*}} p_{eq} + \sum_{Y_{ij}, Y_{i'j'} \in A_{f^*}} p_{eq} = \mathcal{O}(n^4/p)$$

and

$$\mathbb{P}\left[f^*(a_1, a_2, a_3) = 0 \text{ for some } \{a_1, a_2, a_3\} \neq \{X_i, X_j, Y_{ij}\}\right] \leq \sum_{a_1, a_2, a_3 \in B \cup A_{f^*}} p_{eq} = \mathcal{O}(n^6/p).$$

We are done by applying the Union bound to the two failure probabilities. □

Observe that a fully random assignment to the variables of the polynomials is not necessary to make the two polynomials equal with probability as small as $1/p$. Instead, if the variables are assigned 6-wise-independently, the probability p_{eq} is still $1/p$. This observation yields a polynomial-time construction of the superposable set, as follows.

Lemma 5.10. *One can construct an n-order superposable set in time polynomial in n.*

Proof. Exhaustively explore the polynomial-size probability space of 6-wise independence to find the superposable set, which we know to exist [65, 66]. □

We complete the proof of Theorem 5.7 by combining Lemmas 5.8, 5.9, and 5.10. □

Inapproximability. In the NP-hardness reduction, we present a mapping $h : V \cup E \to S$, so that every n-node m-edge graph G has an independent set of size k iff the set S has a f^*-root-free subset of size $k+m$. Here we demonstrate the inapproximability of Max-RF(f^*) by a PTAS reduction. Indeed we show that if one has a $(1-\varepsilon)$-approximation for Max-RF(f^*) for any constant $\varepsilon > 0$, then one can approximate the MIS of some types of graphs to within a $(1 - \mathcal{O}(\varepsilon))$ factor. If MIS is hard to approximate in the graph class, then Max-RF(f^*) similarly has no PTAS unless P = NP.

To make the NP-hardness reduction be a PTAS reduction as well, we restrict the MIS problem to graphs whose maximum independent set has size $\Omega(m)$. The graph class we select is 3-regular graphs, for which MIS has no PTAS unless P = NP [23], and by Turán's Theorem [88] every such graph has a maximum independent set of size at least $m/6 = \Omega(m)$. We now give a PTAS reduction from MIS for 3-regular graphs to Max-RF(f^*).

Lemma 5.11. *If there exists a polynomial-time algorithm that, for every 3-regular graph $G = (V, E)$, can find a f^*-root-free subset of $h(V \cup E)$ whose size approximates the maximum possible to within a factor of $(1-\varepsilon)$, then an independent set of G whose size approximates the maximum possible to within a factor of $(1 - 7\varepsilon)$ can be found in polynomial time.*

Proof. Every $(1-\varepsilon)$-approximation algorithm for Max-RF(f^*) can find a f^*-root-free subset of size $(1-\varepsilon)(m+k)$, which corresponds to an independent set of size $(1-\varepsilon)k-\varepsilon m \geq (1-7\varepsilon)k$, where the last inequality follows from the fact that $k \geq m/6$ for every 3-regular graph due to Turán's Theorem [88]. □

From the PTAS reduction in Lemma 5.11 and the APX-hardness of MIS for 3-regular graphs [23], we have Theorem 5.12.

Theorem 5.12. Max-RF($x + y + z$) *is* APX*-hard.*

Max-3SUM***-hardness.*** In Theorem 5.12, we prove the inapproximability for finding a maximum f^*-root-free subset, i.e. containing no three distinct elements a, b, c such that $a + b + c = 0$. This problem is equivalent to the Max-3SUM problem, except that in the

Max-3SUM problem, the three elements are allowed to repeat. To handle duplicates using the Max-RF($\cdot$) scheme, we consider the inapproximability of Max-RF(g) for the polynomial $g(x, y, z) = 3x(2x + y)(x + y + z)$, so that a set S is g-root-free if and only if it contains no three (perhaps repeating) elements whose sum equals zero. Note that the 3 in the definition of g is superfluous, since we are only interested in the roots. We leave it in to emphasize that we are considering the three cases of no repetition ($x + y + z$), one element selected twice ($2x + y$), and one element selected three times ($3x$).

Finding a maximum f^*-root-free subset from a given set S is as hard as finding a maximum g-root-free subset if S does not contain 0 and S does not contain two elements where one is twice the negative of the other. We present how to extend the PTAS reduction for Max-RF(f^*) to that for Max-RF(g). To impose the needed restriction on S, one can replace the universe U used in Lemma 5.9 with an 3AP-free subset of $\mathbb{Z}_p \setminus \{0, 2, 4, \ldots\}$. We note here that replacing the universe U is simpler than constructing a superposable set w.r.t. g from scratch. This technique will be more useful when the structure of the function becomes much more complicated in subsequent sections. Then, the maximum f^*-root-free subset of S has the same size of the maximum g-root-free subset, as follows:

Lemma 5.13. *For every three distinct integers a, b, c in $B \cup A_{f^*}$ where B is any 3AP-free subset of $\mathbb{Z}_p \setminus \{0, 2, 4, \ldots\}$ and $A_{f^*} = \{x : f^*(x, y, z) = 0, y, z \in B, y \neq z\}$, $g(a, b, c) = 0$ iff $f^*(a, b, c) = 0$.*

Proof. If $f^*(a, b, c) = 0$, then $g(a, b, c) = 0$. On the other hand, if for some a, b, c, $g(a, b, c) = 0$ but $f^*(a, b, c) \neq 0$, then either one of a, b, c is zero or one of the a, b, c is twice the negative of another. We note that B only contains positive integers and A_{f^*} only contains negative integers. Hence, $B \cup A_{f^*}$ does not contain zero, which makes the former case impossible. As for the latter, if one of $\{a, b, c\}$ is twice the negative of the another, say w.l.o.g. that $a = -2b$, then a and b come from different sets B and A_{f^*}. Then $a \in A_{f^*}$ due to a even and thus $a = -(w+z)$ for $w, z \in B, w \neq z$. However, this means $-(w+z)+2b = 0$ contradicting the 3AP-freeness of B. □

To complete the proof for showing the inapproximability of Max-RF(g), we appeal to the known polynomial-time construction of $p^{\Omega(1)}$-size 3AP-free subsets due to Erdös [43] or

any of the subsequent works [19,42,68,79] that improve the size of the constructed 3AP-free subset. Now the universe U has size $p^{\Omega(1)}$, so to make the existence proof work, one needs to choose a p that is polynomially larger than before. As a consequence of Theorem 5.12, Lemma 5.13, and the polynomial-time construction due to Erdös, Max-3SUM is APX-hard. To show the APX-completeness of Max-3SUM, we note that picking the larger of the subset of all positive integers and the subset of all negative integers gives a 2-approximation for Max-3SUM because a solution cannot include 0.

Theorem 5.14. Max-3SUM *is* APX*-complete.*

5.3 Hardness of Max-RF(f)

We generalize the hardness result of Max-RF($x+y+z$) in Section 5.2 to Max-RF(f) for every homogeneous 3^+-variate linear polynomial f.

We defined superposable sets for 3-variate polynomials. Here we extend the definition and define an ***n-order superposable set*** w.r.t. an $(r+3)$-variate polynomial f as follows. For any set B of $n + r\binom{n}{2}$ integers X_i, X_{ijk} for $i < j \in [n], k \in [r]$, we define the auxiliary set A_f induced by B and f to be $\{Y_{ij} : f(X_i, X_j, X_{ij1}, \ldots, X_{ijr}, Y_{ij}) = 0, i \neq j\}$. Let $\mathcal{S}_{ij} = \{X_i, X_j, X_{ij1}, \ldots, X_{ijr}, Y_{ij}\}$ for every $i \neq j$. We say B is an n-order superposable set if $|B \cup A_f| = n + (r+1)\binom{n}{2}$, and for every $(r+3)$ distinct elements $a_1, \ldots, a_{r+3} \in B \cup A_f$, $f(a_1, \ldots, a_{r+3}) = 0$ only if $\{a_1, \ldots, a_{r+3}\} = \mathcal{S}_{ij}$ for some $i \neq j$.

Given a superposable set B, one can realize a mapping $h : V \cup E \to 2^S$, the power set of S, where $h(v_i) = \{X_i\}$ for $v_i \in V$ and $h(\{v_i, v_j\}) = \{X_{ij1}, \ldots, X_{ijr}, Y_{ij}\}$ for $\{v_i, v_j\} \in E$. As in the proof of Theorem 5.7, if an n-order superposable set B can be constructed in time polynomial in n for every n, then Max-RF(f) is NP-hard. Furthermore, the hardness-reduction can be generalized to a PTAS reduction for every constant-variate polynomial f simply by replacing $(1-7\varepsilon)$ with $(1-(7+6r)\varepsilon)$ in the statement Lemma 5.11. Hence, a polynomial-time construction of B yields a proof of Theorem 5.4, shown as follows.

Proof of Theorem 5.4. Let $f = \sum_{i=1}^{r+3} c_i x_i$, $c_i \neq 0$, and let $m = \ell cm(c_1, \ldots, c_{r+3})$. We construct an n-order superposable set B by sampling X_i, X_{ijk} for $i < j \in [n], k \in [r]$ from the following universe $U = \mathbb{Z}_p \cap m\mathbb{Z}$ for some prime p. We choose the universe U thusly

because no matter what the sampled X's are, they make all Y_{ij} integral. We claim that the sampled set B is an n-order superposable set with positive probability.

We sample each X_i, X_{ijk} in B uniformly at random from the universe U. Before the sampling is made, each X_i, X_{ijk} in B can be seen as an independent random variable and each Y_{ij} in A_f can be seen as some linear combination of these independent random variables.

A bound on the success rate of this random construction of B can be computed from the following two failure rates. The first failure rate $P_1 = \mathbb{P}\left[|B \cup A_f| < n + (r+1)\binom{n}{2}\right]$ is bounded by $n^{\mathcal{O}(1)}/(p/m)$. To see this, we note that every pair of distinct $a, b \in B \cup A_f$ is a pair of different linear combinations of the independent random variables. By the Schwartz-Zippel Lemma, the sampled value of a equals that of b with probability $1/(p/m)$. Hence, the claimed bound for P_1 holds by the Union bound.

To bound the second failure rate

$$\mathbb{P}\left[f(a_1, \ldots, a_{r+3}) = 0 \text{ for some } \{a_1, \ldots, a_{r+3}\} \notin \{\mathcal{S}_{ij} : i \neq j\}\right],$$

we observe that $f(a_1, \ldots, a_{r+3})$ is a non-zero polynomial for every $\{a_1, \ldots, a_{r+3}\} \notin \{\mathcal{S}_{ij} : i \neq j\}$ where $a_1, \ldots, a_{r+3}$ are distinct elements in $B \cup A_f$. Suppose that $f(a_1, \ldots, a_{r+3})$ were a zero-polynomial for some $\{a_1, \ldots, a_{r+3}\} \notin \{\mathcal{S}_{ij} : i \neq j\}$. If we express each a_ℓ for $\ell \in [r+3]$ as a linear combination of the random variables in B, then each variable in B either does not appear or appears at least twice in all of a_ℓ's expressions. This happens only when $r = 0$ and $\{a_1, a_2, a_3\} = \{Y_{ij}, Y_{jk}, Y_{ik}\}$ for some $i < j < k$. However,

$$f(a_1, a_2, a_3) = c_1\left(\frac{c_1X_i + c_2X_j}{-c_3}\right) + c_2\left(\frac{c_1X_j + c_2X_k}{-c_3}\right) + c_3\left(\frac{c_1X_i + c_2X_k}{-c_3}\right)$$

cannot be a zero polynomial because X_j's coefficient is non-zero, or

$$f(a_1, a_2, a_3) = c_1\left(\frac{c_1X_i + c_2X_j}{-c_3}\right) + c_2\left(\frac{c_1X_i + c_2X_k}{-c_3}\right) + c_3\left(\frac{c_1X_j + c_2X_k}{-c_3}\right)$$

cannot be a zero polynomial unless $(c_1, c_2, c_3) = (t, -t, t)$ for some $t \neq 0$, which is avoided by sorting the variables in f by their coefficients. The same argument works when (a_1, a_2, a_3) equals other permutations of (Y_{ij}, Y_{jk}, Y_{ik}). Hence, the observation is true. There are a polynomial number of linear polynomial $f(a_1, \ldots, a_{r+3})$ that cannot be zero given the random assignment of X_i, X_{ijk} for $i < j \in [n], k \in [r]$. Therefore the failure rate is $n^{\mathcal{O}(1)}/(p/m)$.

Given the bounds on failure probability, the randomly sampled B is an n-order superposable set with positive probability by picking p polynomially large in n. After a derandomization step similarly to that in Lemma 5.10, we have B constructed in deterministic polynomial time. This proves Theorem 5.1 in part, and we defer the discussion of inapproximability constants to Section 5.6. □

The above PTAS reductions are based on the hardness of Mis on 3-regular graphs, which specifies additional structure on the given input set S. Our reduction still works if the underlying graph is replaced with another graph class $\mathcal{G}$ as long as $\mathcal{G}$ has $\mathcal{O}(n)$ edges and Mis admits no PTAS on $\mathcal{G}$ unless P = NP. Such a replacement is useful for proving further hardness results. To illustrate, let $\mathcal{G}$ be 3-regular triangle-free, where both conditions hold by Turán's Theorem and [34]. Then, finding a maximum independent set on $G \in \mathcal{G}$ can be reduced to Max-Triangle-Free-Subgraph on H, which can be seen by replacing each edge in G with a triangle that connects a newly added node. Now let $\mathcal{G}$ have maximum degree at most 3 and girth at least r, for any r, where both conditions hold by Lemma 5.19. Let Max-$C_{r\text{-free}}$ by the problem of finding the largest (in terms of number of nodes) node-induced r-cycle-free subgraph. We get that:

Corollary 5.15. *Max-$C_{r\text{-free}}$ is* APX*-hard for any $r \geq 3$.*

Sharpness of the Three Conditions. Theorem 5.4 is our base result. It applies, as noted to f that are homogeneous, 3^+-variate, and linear. All three conditions are necessary to achieve inapproximability over the entire class of f, which we can see as follows. We say a function $f(\mathbf{x})$ is ***trivial*** if there are finitely many integral solutions to $f(\mathbf{x}) = 0$. In this case, one can compute the Max-RF(f) problem exactly in polynomial time based on a precomputed table. Therefore, to prove that Max-RF(f) is hard to approximate, trivial functions must be excluded. Some inhomogeneous functions are trivial. For example let $f(x) = c_0 + \sum_1^r c_i x_i^{t_i}$ and suppose $gcd(c_1, \ldots, c_r)$ does not divide c_0. Then $f(\mathbf{x}) = 0$ has no integral solution. Furthermore, some high-order polynomial functions are trivial. Notice that every homogeneous even-order polynomial whose coefficients are all non-negative has only one solution, $\mathbf{0}$, and from Fermat's Last Theorem [98] it follows that some homogeneous 3-variate 3^+-order polynomial $x^k + y^k - z^k$ for $k \geq 3$ has no integral solution other than

$(a, -a, 0)$. One can simply remove 0 to compute Max-RF$(x^k + y^k - z^k)$ optimally. Therefore, both homogeneity and linearity are necessary for the entire class to be inapproximable.

Now consider the requirement that f be 3^+-variate. One can observe that for every 2-variate quadratic or linear $f(x)$, the Max-RF(f) problem is equivalent to the maximum independent set problem on degenerate graphs where each node has degree at most 2 and is thus a union of cycles and paths. Solving Max-RF(f) for such a $f(x)$ takes linear time and therefore it is necessary to require that f be 3^+-variate.

5.4 Hardness of Max-RF$(\prod \ell_i)$

We extend the base inapproximability result of Theorem 5.4 to show the inapproximability result of Max-RF(f) for $f = \prod_{i \in [k]} \ell_i$ where each ℓ_i is a homogeneous and linear polynomial. Specifically, we will prove Theorem 5.5, that is, we will show that if any ℓ_i is 3^+-variate, then Max-RF(f) is APX-hard.

We begin by defining a ***canonical representation*** for f. Observe that Max-RF$((x + y - z)(y + a - b))$ equals Max-RF$((x + y - z)^2)$, which also equals Max-RF$(\delta(x + y - z))$ for any $\delta \in \mathbb{Z} \setminus \{0\}$. Let Coef$(\ell_i)$ be the multi-set of coefficients in ℓ_i. We say that ℓ_i and ℓ_j are in the same equivalence class if Coef$(\ell_i) = \{\delta C : C \in \text{Coef}(\ell_j)\}$ for some non-zero integer δ. Then, we obtain the representation of f by removing ℓ_i from f if ℓ_i and ℓ_j are from the same equivalence class for some $j < i$. Given the representation, let ℓ_* be the ℓ_i that has the most number of variables. If there is a tie, then pick any of them, but if all ℓ_i's have at most 3 variables, pick ℓ_* from the same equivalence class as $(x + y - z)$ if any.

Claim 5.16. *There exists some superposable set B w.r.t. ℓ_*, whose auxiliary set is A_{ℓ_*}, such that the largest f-root-free subset of $B \cup A_{\ell_*}$ has the same cardinality as the largest ℓ_*-root-free subset of $B \cup A_{\ell_*}$.*

Given Claim 5.16, if such a superposable set B can be found, then one can use the APX-hardness of Max-RF(ℓ_*) to conclude that Max-RF(f) is APX-hard. However, we do not know how to find such a superposable set B directly. Our approach is to prove the existence of such a superposable set in a small pool $\mathcal{Q}$ of possible superposable sets, but

without knowing which one is the desired one. Then, we implement an algorithm for Max-RF(ℓ_*) by asking which $B \cup A_{\ell_*}$ for $B \in \mathcal{Q}$ has the largest f-root-free subset, which gives the answer for Max-RF(ℓ_*) and thus Max-RF(f) is APX-hard as well. To see why, the cardinality of the largest ℓ_*-root-free subset is an upper bound of the cardinality of f-root-free-subset, and by Claim 5.16 the upper bound is matched for some $B \cup A_{\ell_*}$, $B \in \mathcal{Q}$. As a result, the existence of such a pool $\mathcal{Q}$ implies the correctness of Theorem 5.5.

To demonstrate the existence of such a $\mathcal{Q}$, for an $(r+3)$-variate polynomial f we first construct a superposable set $B = \{X_i : i \in [n]\} \cup \{X_{ijk} : i, j \in [n], k \in [r]\}$ as before, noting that each element in B is a random variable, where will be later received a value. Similarly, define $A_{\ell_*} = \{Y_{ij} : \ell_*(X_i, X_j, X_{ij1}, \ldots, X_{ijr}, Y_{ij}) = 0\}$ where Y_{ij} is a linear combination of X's rather than a value. For every graph $G = (V, E)$, we construct a set Γ of random variables containing X_i's if $i \in V$ as well as Y_{ij}, X_{ijk} for $k \in [r]$ if $(i, j) \in E$. Then, we use the naive exponential-time algorithm to remove one of X_i, X_j from Γ for each $(i, j) \in E$ so that the number of removals is minimized. The exponential-time computation is allowed here because we are proving the existence of $\mathcal{Q}$ rather than performing a hardness reduction. Let $\bar{\Gamma}$ be the resulting set, a set of variables. We note here that, if the variables in $\bar{\Gamma}$ (rather than Γ) take values uniformly at random, then they are independent.

Lemma 5.17. *For every linear factor $\ell_i \neq \ell_*$ of f, replacing variables in ℓ_i with distinct variables in $\bar{\Gamma}$ cannot make ℓ_i a zero polynomial.*

Proof. Case 1: if ℓ_i has fewer variables than ℓ_*, then it is clear. Case 2: if ℓ_i has the same number of variables as ℓ_*, then ℓ_i could be a zero polynomial only when both ℓ_i and ℓ_* are 3-variate. Thus, let $\ell_i = a_1 x + a_2 y + a_3 z$ and $\ell_* = b_1 x + b_2 y + b_3 z$. Given ℓ_*, we have that $Y_{ij} = (b_1 X_i + b_2 X_j)/ - b_3$. The only case that ℓ_i could be a zero polynomial is thus

$$\ell_i = a_1 \left(\frac{b_1 X_i + b_2 X_j}{-b_3} \right) + a_2 \left(\frac{b_1 X_j + b_2 X_k}{-b_3} \right) + a_3 \left(\frac{b_1 X_i + b_2 X_k}{-b_3} \right).$$

Note that a's and b's are non-zero, and thus ℓ_i is a zero only if $\mathrm{Coef}(\ell_i) = \{a_1, a_2, a_3\} = \{1, 1, -1\}$. However, ℓ_* has the same equivalence class as $(x + y - z)$ if any ℓ_i does. That means ℓ_i, ℓ_* are in the same equivalence class, a contradiction on the representation of f. □

Since every linear factor ℓ_i of f cannot be a zero polynomial by such a replacement,

$f = \prod \ell_i$ cannot either. We claim that:

Lemma 5.18. *For any set U where $|U| = |\bar{\Gamma}|^c$ for some sufficiently large constant c, if the variables in $\bar{\Gamma}$ takes values from U uniformly at random, then the evaluated set of $\bar{\Gamma}$ is f-root-free with positive probability.*

Proof. We prove this by the Schwartz-Zippel Lemma. Let f be a k-variate polynomial with total degree d. By Lemma 5.17, f is not a zero polynomial no matter how the variables in f are replaced with distinct variables in $\bar{\Gamma}$. Therefore,

$$\mathbb{P}\left[f(W_1, W_2, \ldots, W_k) = 0\right] \leq \frac{d}{|U|} \text{ for every } W_1, W_2, \ldots, W_k \in \bar{\Gamma}.$$

There are $\mathcal{O}(|\bar{\Gamma}|^k)$ different combinations of $W_1, W_2, \ldots, W_k$. By the union bound on all possible combinations of W_i's, the evaluated set of $\bar{\Gamma}$ is f-root-free with probability

$$1 - \frac{d|\bar{\Gamma}|^k}{|U|}.$$

We are done because d and k are constants, and we can pick $c = k + 1$. □

Note that the largest ℓ_*-root-free subset of Γ has cardinality $|\bar{\Gamma}|$, and the largest f-root-free subset of Γ has cardinality no less than a f-root-free subset of $\bar{\Gamma}$ because $\bar{\Gamma} \subseteq \Gamma$. Since Lemma 5.18 states that the largest f-root-free subset of $\bar{\Gamma}$ has cardinality $|\bar{\Gamma}|$ for some $B \in U^n$, Claim 5.16 holds.

To make $\mathcal{Q}$ small, we replace the fully random sample space U^n with a k-wise independent sample space of size $|U|^k$. Consequently, Theorem 5.5 immediately follows, where the inapproximability constant is obtained from that of Max-RF(ℓ_*).

5.5 Hardness of Max-RF($\sum_i \ell_i^2$)

In this section, we will show that Max-RF(f) is APX-hard for any r-variate polynomial $f = \sum_{i \in [k]} \ell_i^2$ if the solution set of $f = 0$ is in general position and has dimension $d \geq 2$ where $\ell_1, \ell_2, \ldots, \ell_k \in \mathbb{Z}[x_1, x_2, \ldots, x_r]$ are homogeneous linear polynomials. That is, we prove Theorem 5.6.

To prove Theorem 5.6 for $d = t - 1$, one can appeal to the proof of Theorem 5.4. For $d < t - 1$ in general, the number of dependent random variables induced by the superposable

set B is no longer 1, thus requiring the solution set of $f = 0$ to be in general position. We need to modify the definition of the superposable set w.r.t. such an f, as described below.

We begin by proving a helper lemma that MIS for graphs of maximum degree at most 3, girth at least g is APX-hard for any constant g, which extends the NP-hardness result shown in [69].

Lemma 5.19. MIS *for graphs of maximum degree at most 3, girth at least g is* APX*-hard for any constant g, and cannot be approximated to within $1 - \varepsilon_g$ unless* P = NP *for any*

$$\varepsilon_g < \frac{1}{140(6\lceil (g-3)/6 \rceil + 1)}.$$

Proof. We prove this by giving a PTAS reduction from MIS for 3-regular graphs $G_{3r} = (V_{3r}, E_{3r})$ to MIS for graphs $G_{g^+} = (V_{g^+}, E_{g^+})$ of girth $\geq g$. We obtain G_{g^+} from G_{3r} by replacing each edge in E_{3r} with a path of length $2t + 1$ ($t \in \mathbb{Z}$), connecting $2t$ new nodes. Hence, the smallest cycle in G_{g^+} is $3 + 6t$. We pick $t = \lceil (g-3)/6 \rceil$ so that G_{g^+} has no cycle of length $< g$.

It is known [69] that G_{g^+} has an independent set of size $t|E_{3r}| + k$ iff G_{3r} has an independent set of size k. Every $(1 - \varepsilon)$-approximation algorithm for MIS of G_{g^+} can find an independent set of size $(1 - \varepsilon)(t|E_{3r}| + k)$, which corresponds to an independent set of size $(1 - \varepsilon)k - \varepsilon t|E_{3r}| \geq (1 - (6t + 1)\varepsilon)k$ in G_{3r}, where the last inequality follows from the fact that $k \geq |E_{3r}|/6$ for every 3-regular graph, due to Turán's Theorem [88].

Based on [23], MIS for 3-regular graphs cannot be approximated to within $1 - \varepsilon_{3r}$ for any $\varepsilon_{3r} < 1/140$. Thus, ε cannot be less than $\frac{1}{140(6t+1)} = \frac{1}{140(6\lceil (g-3)/6 \rceil + 1)}$. □

Proof of Theorem 5.6. For any n-node, m-edge $G = (V, E)$ that have maximum degree at most 3 and girth at least $r + 1$, we construct a set B of independent random variables and an auxiliary set A_f so that

$$B = \{X_i : i \in V\} \cup \{X_{ijk} : (i, j) \in E, i < j, k \in [1, d-2]\}, \text{ and}$$

$$A_f = \left\{Y_{ij1}, \ldots, Y_{ij(r-d)} : f\left(X_i, X_j, X_{ij1} \ldots X_{ij(d-2)}, Y_{ij1} \ldots Y_{ij(r-d)}\right) = 0, (i, j) \in E, i < j\right\},$$

where the solution space of $f = 0$ is in general position and has dimension $d \geq 2$. Hence, for every $(i, j) \in E, i < j$, $\left(Y_{ij1}, Y_{ij2}, \ldots, Y_{ij(r-d)}\right)$ is unique.

Here we define the Y_{ijk} explicitly. Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_d}$ be a set of basis vectors (column vectors) in $\mathbb{Z}^r$ of the solution set of $f = 0$. Let $\mathbf{A}$ be the aggregation of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_d}$ where $\mathbf{A} = (\mathbf{v_1}|\mathbf{v_2}|\cdots|\mathbf{v_d})$. Let $\mathbf{Q}$ be the square matrix composed of the upper d rows of $\mathbf{A}$. By the definition of general position, $\mathbf{A}$ is strongly full rank, $\mathbf{Q}$ is full rank, and thus $\mathbf{z}$ is uniquely defined by

$$\mathbf{Qz} = \begin{bmatrix} X_i \\ X_j \\ X_{ij1} \\ \vdots \\ X_{ij(d-2)} \end{bmatrix}, \text{ and we set } \mathbf{Az} = \mathbf{A}\left(\mathbf{Q}^{-1}\begin{bmatrix} X_i \\ X_j \\ X_{ij1} \\ \vdots \\ X_{ij(d-2)} \end{bmatrix}\right) = \begin{bmatrix} X_i \\ X_j \\ X_{ij1} \\ \vdots \\ X_{ij(d-2)} \\ Y_{ij1} \\ \vdots \\ Y_{ij(r-d)} \end{bmatrix}.$$

Hence, each of $Y_{ij1}, \ldots, Y_{ij(r-d)}$ is a linear combination of $X_i, X_j, X_{ij1}, \ldots, X_{ij(d-2)}$. Note that $\mathbf{AQ}^{-1}$ is also strongly full rank, yielding that a linear combination of any d variables from the set

$$\{X_i, X_j, X_{ij1}, \ldots, X_{ij(d-2)}, Y_{ij1}, \ldots, Y_{ij(r-d)}\}$$

cannot be a zero polynomial. We are ready to prove that B is superposable w.r.t. E, that is:

Lemma 5.20. *For any distinct $a_1, a_2, \ldots, a_r \in B \cup A_f$, $f(a_1, a_2, \ldots, a_r)$ is a zero polynomial iff $\{a_1, a_2, \ldots, a_r\} = \{X_i, X_j, X_{ij1}, \ldots, X_{ij(d-2)}, Y_{ij1}, \ldots, Y_{ij(r-d)}\}$ for some $(i, j) \in E, i < j$.*

Proof. If $\{a_1, a_2, \ldots, a_r\} \subset \{X_i : i \in [n]\}$, then $f(a_1, a_2, \ldots, a_r)$ cannot be a zero polynomial because the X_i's are independent variables and each X_i appears at most once in any linear polynomial ℓ_j that comprises f. Thus, to zero $f(a_1, a_2, \ldots, a_r)$ we may assume that

$$a_p \in \mathcal{S}_{ij} \equiv \{X_{ij1}, \ldots, X_{ij(d-2)}, Y_{ij1}, \ldots, Y_{ij(r-d)}\} \text{ for some } p \in [r], (i, j) \in E, i < j.$$

Say a_p appears in some homogeneous linear polynomial ℓ_q that comprises f. To make $f(a_1, \ldots, a_r)$ a zero polynomial, one must make ℓ_q a zero polynomial. We disprove the possibility of making ℓ_q zeroed as follows. If ℓ_q picks $\geq d$ variables from $\mathcal{S}_{ij}$, then each of

$a_1, \ldots, a_r$ can be represented by linear combination of random variables in $\mathcal{S}_{ij}$. In other words, $\{a_1, \ldots, a_r\} \subset (\mathcal{S}_{ij} \cup \{X_i, X_j\})$ because $d \geq 2$. If ℓ_q picks $d-1$ variables from $\mathcal{S}_{ij}$, then to make ℓ_q zeroed, ℓ_q needs to pick two variables a_w and a_z where a_w is from $\mathcal{S}_{ik} \cup \{X_i\}$ and a_z is from $\mathcal{S}_{j\ell} \cup \{X_j\}$. Note that $k \neq \ell$ because G has girth $r+1 \geq d+2 \geq 4$. This would lead to a contradiction since if we solve the system by the $d-1$ variables from $\mathcal{S}_{ij}$ as well as a_w, then a_z can be represented by linear combination of variables from $\mathcal{S}_{ij} \cup \mathcal{S}_{ik} \cup \{X_i\}$, contradicting that $a_z \in \mathcal{S}_{j\ell}$, $\ell \neq k$, and $d \geq 2$. If ℓ_q picks $\leq d-2$ variables from $\mathcal{S}_{ij}$, since the rest of variables can be partitioned into subsets, each of which sum to a multiple of X_i, or a multiple of X_j, but not a linear combination of X_i and X_j due to G having girth at least $r+1$, therefore ℓ_q cannot be zeroed since this effectively picks $\leq d$ variables from $\mathcal{S}_{ij} \cup \{X_i, X_j\}$. □

Lastly, the exact construction of the superposable set is similar to that in Theorems 5.4 and 5.5. By Lemma 5.20 and the Swartz-Zippel Lemma, we know that sampling $\{X_i : i \in [n]\} \cup \{X_{ijk} : (i,j) \in E, i < j, k \in [d-2]\}$ uniformly at random from $(\det(\mathbf{Q})\mathbb{Z})^{n+(d-2)m}$ yields a superposable set with positive probability. We pick every X_i and X_{ijk} as multiples of $\mathbf{det}(\mathbf{Q})$ to ensure that all dependent variables Y_{ijk}'s are in $\mathbb{Z}$. Then, after derandomization using techniques for constant-wise independence, the construction takes time polynomial time in n. Setting $g = r+1$, we get our inapproximability constant by Lemma 5.19. □

An implication of Theorem 5.6 is the APX-hardness of finding the maximum-cardinality r-term AP-free subset S for any fixed $r \geq 3$, noting that S may contain elements that form an c-term arithmetic progression for $c < r$ but not $c \geq r$. This problem can be encoded as Max-RF(f) for $f = \sum_{i=1}^{r-2}(x_i - 2x_{i+1} + x_{i+2})^2$, and the solution set of $f = 0$ is the plane

$$(x_1, x_2, \ldots, x_k) = \alpha(1, 3, \ldots, 2k-1) + \beta(2, 4, \ldots, 2k) \text{ for constant } \alpha, \beta \in \mathbb{R}.$$

Therefore,

$$B = \begin{bmatrix} 1 & 3 & \ldots & 2k-1 \\ 2 & 4 & \ldots & 2k \end{bmatrix} \text{ in which every } 2 \times 2 \text{ submatrix } \begin{bmatrix} 2i-1 & 2j-1 \\ 2i & 2j \end{bmatrix}$$

is full rank. By Theorem 5.6, we get:

Corollary 5.21. *Finding a maximum-cardinality r-term AP-free subset of a given integral set S for any fixed $r \geq 3$ is* APX*-hard.*

Sharpness of $d \geq 2$**.** Not every problem in the class Max-RF($\sum_i \ell_i^2$) is hard to approximate. If the solution set of $f = 0$ is a point[2], then it suffices to remove an integer in the set S that coincides with the coordinate of the point. If it is a line, for example $\alpha(1, 2, 4, 8)$ for $\alpha \in \mathbb{R}$, then a greedy algorithm can solve this case in P by removing the last coordinate for every tuple of 4 integers that are multiples of (1, 2, 4, 8).

5.6 Inapproximability Constant

Lastly, we determine an inapproximability constant $1 - \varepsilon_r$ for Max-RF(f), for every homogeneous, r-variate for $r \geq 3$, linear polynomial f. We use the facts that Mis on 3-regular graphs cannot be approximated to within the constant $C_3 = 139/140 + \varepsilon$ for any constant $\varepsilon > 0$ [23], and Mis on 3-regular triangle-free graphs can not be approximated to within the constant $C_{3\Delta} = 1422/1432 + \varepsilon$ for any constant $\varepsilon > 0$ [34].

We first apply Lemma 5.11 to bound a inapproximability constant $1 - \delta_r$ based on C_3 and then replace the use of Turán's Theorem in Lemma 5.11 with the AKS Theorem [9] and Staton's result [85] to bound the claimed inapproximability constant $1 - \varepsilon_r$ based on $C_{3\Delta}$.

Since Mis on 3-regular graphs cannot be approximated to within C_3, from Lemma 5.11 we have following theorem:

Theorem 5.22. *For every homogeneous, r-variate ($r \geq 3$), linear polynomial f,* Max-RF(f) *cannot be approximated to within any constant factor larger than $1-\delta_r$ in polynomial time unless* P = NP*, where* $\delta_r = \frac{1-C_3}{7+6(r-3)}$.

Simply replacing C_3 with $C_{3\Delta}$ cannot increase δ_r because $C_3 < C_{3\Delta}$ and such replacement in Theorem 5.22 makes δ_r smaller. Instead, we replace the use of Turán's Theorem, which applies to general graphs, with the AKS Theorem (see Theorem 5.23), which works for triangle-free graphs. In [8,83], the constant in the big-Omega notation in AKS Theorem

[2] $\mathbf{0}$ must be a solution of $f = 0$ because the ℓ_i are homogeneous.

is bounded above by 1/100 and 1/8, respectively. Though the size of an independent set guaranteed by the AKS theorem is asymptotically larger than that of Turán theorem, it is numerically smaller when $d = 3$.

Theorem 5.23 (AKS Theorem [9])**.** *Every d-regular triangle-free graph has an independent set of size $\Omega(n \log d/d)$.*

Note that the constant in the big-Omega notation is universal for every d. For a particular value of d the constant can be larger. In particular, in [85] Staton shows that every 3-regular triangle-free graph has an independent set of size $5m/21$, which is more than the $m/6$ guaranteed by Turán theorem. The constant 5/21 is tight due to Fajtlowicz [47]. Based on this improved guarantee of the size of an independent set, we obtain the following result.

Theorem 5.24. *For every homogeneous, r-variate, linear polynomial f,* Max-RF(f) *cannot be approximated to within any constant factor larger than $1 - \varepsilon_r$ in polynomial time unless* P = NP*, where $\varepsilon_r = 1 - \frac{1-C_{3\Delta}}{5.2+4.2(r-3)}$.*

The inapproximability constant of Max-rSUM ***problems.*** As noted in Section 5.1, we have $f_3 = 3x(2x+y)(x+y+z)$, and by Theorem 5.5 Max-3SUM has the inapproximability constant $1 - \varepsilon_3$ because the ℓ_* of f_3 has 3 variables. As for $r = 4$, $f_4 = 4x(3x + y)(2x + 2y)(2x+y+z)(x+y+z+w)$ and therefore Max-4SUM has the inapproximability constant $1-\varepsilon_4$. One can observe that each linear factor ℓ_i in f_r has Coef(ℓ_i) equal to some partition of the integer r, and thus for each r the ℓ_* for f_r is $\sum_{1\le i\le r} x_i$. This implies that Max-rSUM has inapproximability constant $1 - \varepsilon_r$ for every $r \ge 3$.

References

[1] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 434–443, Washington, DC, USA, 2014. IEEE Computer Society.

[2] A. Adamaszek and A. Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 400–409, Washington, DC, USA, 2013. IEEE Computer Society.

[3] P. Afshani, L. Arge, and K. D. Larsen. Orthogonal range reporting: Query lower bounds, optimal structures in 3-d, and higher-dimensional improvements. In *26th Annual Symposium on Computational Geometry (SoCG)*, pages 240–246, 2010.

[4] P. Afshani, L. Arge, and K. G. Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *28th Annual Symposium on Computational Geometry (SoCG)*, pages 323–332, 2012.

[5] P. Afshani, M. A. Bender, M. Farach-Colton, J. T. Fineman, M. Goswami, and M. Tsai. Cross-referenced dictionaries and the limits of write optimization. In *SODA*, pages 1523–1532, 2017.

[6] A. Aggarwal and S. Vitter, Jeffrey. The input/output complexity of sorting and related problems. *Commun. ACM*, 31:1116–1127, 1988.

[7] N. Ailon and B. Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, Mar. 2005.

[8] M. Ajtai, P. Erdös, J. Komlós, and E. Szemerédi. On Turán's theorem for sparse graphs. *Combinatorica*, 1(4):313–317, 1981.

[9] M. Ajtai, J. Komlós, and E. Szemerédi. A dense infinite Sidon sequence. *European Journal of Combinatorics*, 2(1):1–11, 1981.

[10] E. Allender, M. Farach-Colton, and M. Tsai. rSUM-hardness yields APX-hardness. In *sumbission*, 2017.

[11] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley, 1992.

[12] Apache. HBase. http://hbase.apache.org, Last Accessed May 16, 2015, 2015.

[13] L. Arge. The buffer tree: A technique for designing batched external data structures. *Algorithmica*, 37(1):1–24, 2003.

[14] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.

[15] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, Jan 1994.

[16] L. Barba, J. Cardinal, J. Iacono, S. Langerman, A. Ooms, and N. Solomon. Subquadratic algorithms for algebraic generalizations of 3SUM. In *33st International Symposium on Computational Geometry, SoCG 2017, Brisbane, July 4-7, 2017, Brisbane, Australia*, page to appear, 2017.

[17] G. Barequet and S. Har-Peled. Polygon containment and translational min-Hausdorff-distance between segment sets are 3Sum-hard. *Int. J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

[18] R. Bayer and E. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1:173–189, 1972.

[19] F. A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proc. Natl. Acad. Sci. USA*, 32(12):331–332, 1946.

[20] M. A. Bender, M. Farach-Colton, J. T. Fineman, Y. Fogel, B. Kuszmaul, and J. Nelson. Cache-oblivious streaming B-trees. In *Proc. 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 81–92, 2007.

[21] M. A. Bender, M. Farach-Colton, M. Goswami, D. Medjedovic, P. Montes, and M. Tsai. The batched predecessor problem in external memory. In *ESA*, pages 112–124, 2014.

[22] M. A. Bender, M. Farach-Colton, M. Goswami, D. Medjedovic, P. Montes, and M. Tsai. The batched predecessor problem in external memory. In A. S. Schulz and D. Wagner, editors, *Proc. 22th Annual European Symposium (ESA)*, pages 112–124, 2014.

[23] P. Berman and M. Karpinski. On some tighter inapproximability results (extended abstract). In *the 26th International Colloquium in Automata, Languages and Programming (ICALP)*, pages 200–209, 1999.

[24] B. Bollobás and W. Fernandez de la Vega. The diameter of random regular graphs. *Combinatorica*, 2(2):125–134, 1982.

[25] P. Bose and S. Langerman. Weighted ham-sandwich cuts. In *Discrete and Computational Geometry, Japanese Conference, JCDCG 2004, Tokyo, Japan, October 8-11, 2004, Revised Selected Papers*, pages 48–53, 2004.

[26] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670, 2014.

[27] G. S. Brodal, E. D. Demaine, J. T. Fineman, J. Iacono, S. Langerman, and J. I. Munro. Cache-oblivious dynamic dictionaries with update/query tradeoffs. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1448–1456, 2010.

[28] G. S. Brodal and R. Fagerberg. Lower bounds for external memory dictionaries. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 546–554, 2003.

[29] G. S. Brodal and R. Fagerberg. Lower bounds for external memory dictionaries. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03)*, pages 546–554, Baltimore, MD, 2003.

[30] A. L. Buchsbaum, M. Goldwasser, S. Venkatasubramanian, and J. R. Westbrook. On external memory graph traversal. In *11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 859–860, 2000.

[31] J. Cardinal, J. Iacono, and A. Ooms. Solving k-SUM using few linear queries. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 25:1–25:17, 2016.

[32] J. Chalopin and D. Gonçalves. Every planar graph is the intersection graph of segments in the plane: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 631–638, New York, NY, USA, 2009. ACM.

[33] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[34] M. Chlebík and J. Chlebíková. Approximation hardness for small occurrence instances of NP-hard problems. In *5th Italian Conference on Algorithms and Complexity (CIAC)*, pages 152–164. Springer, 2003.

[35] A. Conway, M. Farach-Colton, and M.-T. Tsai. Preasymptotics of matrix multiplication. In submission, 2016.

[36] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Computational algebraic complexity editorial.

[37] C. Cotta, I. Dotú, A. J. Fernández, and P. Van Hentenryck. A memetic approach to Golomb rulers. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature*, PPSN'06, pages 252–261, Berlin, Heidelberg, 2006. Springer-Verlag.

[38] M. Dewey. *Decimal Classification and Relative Index for Libraries, Clippings, Notes, Etc.* Library Bureau, 1891.

[39] W. Dittrich, D. Hutchinson, and A. Maheshwari. Blocking in parallel multisearch problems (extended abstract). In *10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 98–107, 1998.

[40] A. Dollas, W. T. Rankin, and D. Mccracken. New algorithms for Golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory*, 44:379–382, 1998.

[41] J. Dybizbański. Sequences containing no 3-term arithmetic progressions. *Elec. J. of Comb.*, 19(2):15–19, 2012.

[42] M. Elkin. An improved construction of progression-free sets. In *SODA*, pages 886–905, 2010.

[43] P. Erdös and P. Turán. On some sequences of integers. *J. London Math. Soc*, 11:261–264, 1936.

[44] P. Erdös and P. Turán. On a problem of Sidon in additive number theory and on some related problems. *J. London Math. Soc*, 16:212–215, 1941.

[45] J. Erickson. Lower bounds for linear satisfiability problems. *Chicago Journal of Theoretical Computer Science*, 8:388–395, 1997.

[46] J. Esmet, M. A. Bender, M. Farach-Colton, and B. C. Kuszmaul. The TokuFS streaming file system. In *Proc. 4th USENIX Workshop on Hot Topics in Storage (HotStorage)*, Boston, MA, USA, June 2012.

[47] S. Fajtlowicz. The independence ratio for cubic graphs. In *8th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, pages 273–277. LSU, 1977.

[48] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5(3):165–185, Oct 1995.

[49] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem in NP complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

[50] W. I. Gasarch, J. Glenn, and C. P. Kruskal. Finding large 3-free sets I: the small n case. *J. Comput. Syst. Sci.*, 74(4):628–655, 2008.

[51] Google, Inc. LevelDB: A fast and lightweight key/value database library by Google. http://github.com/leveldb/, Last Accessed May 16, 2015, 2015.

[52] D. Heller-Roazen. Tradition's destruction: On the Library of Alexandria. *October*, 100:133–153, 2002.

[53] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[54] J. Hopkins. The 1791 French cataloging code and the origins of the card catalog. *Libraries & Culture*, pages 378–404, 1992.

[55] G. W. Houston. *Inside Roman Libraries: Book Collections and Their Management in Antiquity*. UNC Press Books, 2014.

[56] W. Jannen, J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. Betrfs: A right-optimized write-optimized file system. In *Proc. 13th USENIX Conference on File and Storage Technologies (FAST)*, pages 301–315, February 2015.

[57] W. Jannen, J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. Betrfs: Write-optimization in a kernel file system. *Transactions on Storage*, 11(4):18:1–18:29, October 2015.

[58] W. Johnson, Oct. 2015. Personal communication.

[59] A. G. Jørgensen and S. Pettie. Threesomes, degenerates, and love triangles. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 621–630, 2014.

[60] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 329–337. ACM, 1996.

[61] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, Dec 2001.

[62] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[63] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[64] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219 – 230, 1980.

[65] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[66] M. Luby and A. Wigderson. Pairwise independence and derandomization. *Found. Trends Theor. Comput. Sci.*, 1(4):237–301, Aug 2006.

[67] C. Meyer and P. A. Papakonstantinou. On the complexity of constructing Golomb rulers. *Discrete Appl. Math.*, 157(4):738–748, Feb 2009.

[68] L. Moser. On non-averaging sets of integers. *Canadian J. Math.*, 5:245–253, 1953.

[69] O. J. Murphy. Computing independent sets in graphs with large girth. *Discrete Appl. Math.*, 35(2):167–170, Jan. 1992.

[70] K. O'Bryant. Sets of integers that do not contain long arithmetic progressions. *Electr. J. Comb.*, 18(1), 2011.

[71] L. of Congress. Library of Congress classification outline, 2015. Viewed November 8, 2015.

[72] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM J. Comput.*, 20(3):471–483, 1991.

[73] M. Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 603–610, New York, NY, USA, 2010. ACM.

[74] M. Pătraşcu and R. Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1065–1075, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[75] K. Ren and G. A. Gibson. TABLEFS: Enhancing metadata efficiency in the local file system. In *USENIX Annual Technical Conference*, pages 145–156, 2013.

[76] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 515–524, New York, NY, USA, 2013. ACM.

[77] M. Rouse, Oct. 2015. Personal communication.

[78] R.-H. Rouse. The early library of the Sorbonne. *Scriptorium*, 21(1):42–71, 1967.

[79] R. Salem and D. C. Spencer. On sets of integers which contain no three terms in arithmetical progression. *Proc. Natl. Acad. Sci. USA*, 28(12):561–563, 1942.

[80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, Oct 1980.

[81] P. Schweitzer. Problems of unknown complexity – graph isomorphism and Ramsey theoretic numbers, 2009.

[82] Z. Shao, F. Deng, M. Liang, and X. Xu. On sets without k-term arithmetic progression. *Journal of Computer and System Sciences*, 78(2):610 – 618, 2012.

[83] J. B. Shearer. A note on the independence number of triangle-free graphs. *Discrete Mathematics*, 46(1):83–87, 1983.

[84] S. W. Soliday, A. Homaifar, and G. L. Lebby. Genetic algorithm approach to the search for Golomb rulers. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 528–535, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[85] W. Staton. Some Ramsey-type numbers and the independence ratio. *Transactions of the American Mathematical Society*, 256:353–370, 1979.

[86] S. Subramanian and S. Ramaswamy. The p-range tree: A new data structure for range searching in secondary memory. In *Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 378–387, 1995.

[87] T. Tao. A cheap version of the kabatjanskii-levenstein bound for almost orthogonal vectors. https://terrytao.wordpress.com/2013/07/18/a-cheap-version-of-the-kabatjanskii-levenstein-bound-for-almost-orthogonal-vectors/, 2013.

[88] T. Tao and V. H. Vu. *Additive Combinatorics*. Cambridge University Press, 2009.

[89] R. E. Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, 18(2):110–127, 1979.

[90] J. Tavares, F. B. Pereira, and E. Costa. Understanding the role of insertion and correction in the evolution of Golomb rulers. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA*, pages 69–76, 2004.

[91] Tokutek, Inc. TokuDB: MySQL Performance, MariaDB Performance . http://www.tokutek.com/products/tokudb-for-mysql/.

[92] Tokutek, Inc. TokuMX—MongoDB Performance Engine. http://www.tokutek.com/products/tokumx-for-mongodb/.

[93] Tokutek, Inc. TokuDB and TokuMX, 2014. http://www.tokutek.com.

[94] P. Turán. Egy gráfelméleti szélsőérték feladatról. *Matematikai és Fizikai Lapok*, 48:436–452, 1941.

[95] E. Verbin and Q. Zhang. The limits of buffering: A tight lower bound for dynamic membership in the external memory model. *SIAM J. Comput.*, 42(1):212–229, 2013.

[96] S. S. Wagstaff. On k-free sequences of integers. *Mathematics of Computation*, 26(119):767–771, 1972.

[97] D. J. A. Welsh. The computational complexity of some classical problems from statistical physics. In *In Disorder in Physical Systems*, pages 307–321. Clarendon Press, 1990.

[98] A. Wiles. Modular elliptic curves and Fermat's last theorem. *Annals of Mathematics*, 141(3):443–551, 1995.

[99] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA, 2012. ACM.

[100] F. J. Witty. The Pínakes of Callimachus. *The Library Quarterly: Information, Community, Policy*, 28(2):132–136, 1958.

[101] K. Yi. Dynamic indexability and the optimality of b-trees. *J. ACM*, 59(4):21:1–21:19, Aug. 2012.

[102] J. Yuan, Y. Zhan, W. Jannen, P. Pandey, A. Akshintala, K. Chandnani, P. Deo, Z. Kasheff, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. Optimizing every operation in a write-optimized file system. In *Proc. 14th USENIX Conference on File and Storage Technologies (FAST)*, pages 1–14, Santa Clara, CA, February 2016.

[103] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 216–226. Springer, 1979.