Exact Algorithms for Finding Sumsets

Tim Randolph

April 24, 2024

Abstract

In this explanatory note, we consider the problem of *Sumset Containment (SSC)*: given an integer set S and an integer parameter k, does there exist a set A with $|A+A| \geq k$ such that the sumset $A+A = \{a+b \mid a, b \in A\}$ is a subset of S? We prove that this problem is $\mathsf{W}[1]$-complete (in k) and present a natural exact algorithm that runs in time $O^*(3^{n/6}) = O^*(1.201^n)$. As an extension, we use techniques from exact algorithms and additive combinatorics to improve the exponent when $k = \rho n$ for any $\rho \in [0, 1)$.

Our work suggests the following open problem for future research: does there exist an $O^*(3^{(1/6-\varepsilon)n})$-time exact algorithm for Sumset Containment for any constant $\varepsilon > 0$?

1 Introduction

The *sumset* of an integer set A, defined as

$$A + A := \{a + b \mid a, b \in A\}$$

provides a variety of information about A's additive structure. A central problem in additive combinatorics is that of characterizing this structure in sets for which $|A+A|$ is small: for example, Freiman's celebrated theorem states that if $|A+A| \leq K|A|$, there exist functions d and f such that A is contained within a generalized arithmetic progression of dimension $d(K)$ and volume $f(K)|A|$, where d and f are functions that depend on K but not the size of the set $|A|$ [Fre99]. The sumset has proved an immensely fruitful object; see [TV06], Chapter 2 for a broad overview.

Given the importance of the sumset to mathematics, it is natural to consider the algorithmic question of determining *whether* a set is a sumset. In 2023, Chen, Nadimpalli, Randolph, Servedio, and Zamir considered this problem (on the Boolean cube) from the perspective of property testing and proved hardness results [CNR+]. A complementary question is that of determining the smallest sumset that *contains* the input set, which has been previously studied under the name *2-Sumset Cover* [FFV09, BFRV15]. In this note, we consider a fundamental question that complements 2-Sumset Cover: given an input set, what is the largest sumset that it contains?

Formally, our problem is as follows.

Problem 1: Sumset Containment (SSC)

Input. A set of integers $S = \{s_1, s_2, \ldots, s_n\}$ and a parameter $k \leq n$.
Output. A set of integers A (hereafter referred to as an "additive root") such that $A + A \subseteq S$ and $|A + A| \geq k$, or "NO" if no such set exists.

When $k = n$, the question reduces to, "Does there exist an integer set A such that $A + A = S$?" We refer to this problem as *Sumset Equivalence* (SSE).

In this note, we first reduce SSC to a graph problem concerning cliques in edge-colored graphs and the use this reduction to solve SSC and SSE in time $O^*(3^{n/6})$ (Section 3). All three results follow via a reduction from k-Clique. This leads to a natural open question, to which we do not know the answer:

Open Problem 1. *Does there exist an exact algorithm for Sumset Containment (or Sumset Equivalence) that runs in time $O^*(3^{(1/6-\varepsilon)n})$ for some constant $\varepsilon > 0$?*

We also show that SSC is NP-complete, W[1]-complete, and cannot be solved in time $2^{o(\sqrt{n})}$ (Section 4).

The most difficult instances of SSC may be those in which $k = n$: when $n - k = \Omega(n)$, we can improve the runtime of our algorithms exponentially via a careful branch-and-bound algorithm. We prove this extension in Appendix A.

1.1 Related Work

As mentioned above, SSC complements the existing problem *2-Sumset Cover*: given a set S, does there exist a set A of cardinality at most k such that $S \subseteq A + A$? This problem was first studied by Fagnot, Fertin, and Vialette, who showed APX-hardness and a $5^{\frac{k^2(k+3)}{2}} poly(k)$-time algorithm [FFV09]. This was later improved to $2^{(3\log k - 1.4)k} poly(k)$ by Bulteau, Fertin, Rizzi, and Vialette [BFRV15].[1] 2-Sumset Cover is itself a specialization of *Generating Set*, in which the goal is to find a minimal set A such that $S \subseteq \{\sum_{i \in I} i \ ; \ I \subseteq A\}$ [CKSY07].

Instead of parameterizing SSC in the size of the sumset $|A+A|$, one could parameterize Sumset Containment in the size of the additive root $|A|$: given S, does there exist A of size $|A| \geq k$ with $A + A \subseteq S$? This problem is equivalent to k-Clique on the Cayley sum graph of S. k-Clique on Cayley graphs is known to be NP-hard [GR17], but the problem can be solved by existing algorithms for k-Clique in time at most $O^*(1.888^n)$ [TT77, Jia86, Rob86, Rob01].

Chen et al. have studied the problem of determining whether a subset of the Boolean hypercube $\mathbb{F}_2^n$ is (close to) a sumset in the *property testing* model, in which an algorithm's goal is to distinguish between sets containing a large sumset and sets that are ε-far from containing a large sumset. Here ε represents a constant fraction of the universe from which elements are drawn and the complexity of the algorithm is measured in terms of the number of queries to the indicator function of the input set. These authors prove $2^{\Omega(n)}$ lower bounds but no nontrivial upper bounds on the problem [CNR+].

More broadly, our work falls into the growing body of work at the intersection of theoretical computer science and additive combinatorics: for surveys, see [Bib13, Lov17, Tre09, Vio11].

2 Preliminaries

Big-O Notation. We supplement standard big-O notation with an asterisk (e.g., O^* and Ω^*) to indicate the suppression of $\mathsf{poly}(n)$ factors, regardless of the size of the argument. Thus $O^*(2^n)$ indicates $2^n \cdot \mathsf{poly}(n)$ and $O^*(1)$ indicates $\mathsf{poly}(n)$.

Sets. We write $[m]$ as shorthand for the integer set $\{1, 2, \ldots, m\}$ and $[\ell : m]$ as shorthand for the integer set $\{\ell, \ell+1, \ldots, m-1, m\}$. Given an integer set X and integer a, we write $X + a$ (resp. $X - a$) as shorthand for $\{x + a \mid x \in X\}$ (resp. $\{x - a \mid x \in X\}$) and X/a for $\{x/a \mid x \in X\}$.

[1]To prevent confusion, we note that although 2-Sumset Cover is FPTin k, and Sumset Containment is W[1]-complete, the two results are not quite comparable: in 2-Sumset Cover, the parameter k measures the size of the additive root and, most importantly, $k = \Omega(\sqrt{n})$ is guaranteed. In Sumset Containment k measures the size of the sumset and may be arbitrarily small.

Vectors and Norms. Given a vector $v \in \mathbb{Z}^n$, for $i \in \mathbb{Z}$ we let $\#_i(v)$ denote the number of i's appearing as components of v. The ℓ_1-norm $||v||_1$ is the sum of the absolute value of each component of v. Given a function f and a subset S of the domain of f, we adopt the convention $f(S) = \{f(s) \mid s \in S\}$.

3 An $O^*(3^{n/6})$-Time Algorithm for Sumset Containment

In this section, we reduce SSC to the problem of finding a clique in a certain graph which covers at least k differently-colored edges. We can then solve the problem by enumerating the maximal cliques of this graph.

3.1 Reduction to a Promise Problem

Because the correctness of a solution to SSC can be checked in polynomial time, we may assume without loss of generality that our algorithms never return false positives. Equivalently, we can ignore the "No" case and consider the promise problem in which we are guaranteed a hidden additive root A satisfying $A + A \in S$ and $|A + A| \geq k$ without loss of generality.

Let (S, k) be an instance of SSC for which there exists a solution A. Let $m := |A|$ and let a_1 and a_m denote the smallest and largest elements of A, respectively. Our algorithms will take as input "guessed" values for a_1 and a_m and will succeed (with high probability, if the algorithm is randomized) if the guesses are correct. Because $a_1 + a_1, a_m + a_m \in S$, we know that $a_1, a_m \in S/2$. As a result, we can guess every possible pair (a_1, a_m) in exchange for an $O(n^2)$-factor increase in runtime which is absorbed by the O^* notation. Thus we assume our algorithm receives the smallest and largest elements a_1 and a_m of a certain hidden solution A as part of its input without loss of generality.

3.2 Bounds on the Candidate Set

Our first task is to bound the search space: given a SSC instance (S, k), we will construct a finite *candidate set* C that is a superset of any hidden solution A. We can then search within C for A. We begin with the following observation:

Observation 1 (Supersets of the Candidate Set)**.** Let (S, k) be an instance of SSC with a hidden solution $A = \{a_1, \ldots, a_m\}$. A satisfies the following expressions:

1. $A \subseteq S - a_1$.
2. $A \subseteq S - a_m$.
3. $A \subseteq S/2$.
4. $A \subseteq [a_1 : a_m]$.

Proof. Given $A + A \subseteq S$, expressions (1) and (2) follow from the more general observation that $A + a \subseteq A + A$ for any $a \in A$. Expression 3 follows from the fact that $\{2a \mid a \in A\} \subseteq A + A$. Expression 4 follows immediately from the definition of a_1 and a_m. □

With respect to an instance (S, k) of SSC, a_1, and a_m, we define the *initial candidate set*

$$C_0(S) := (S - a_1) \cap (S - a_m) \cap S/2 \cap [a_1 : a_m]. \tag{1}$$

By Observation 1 we have $A \subseteq C_0(S)$. Carefully examining the definition of $C_0(S)$ allows us to bound its cardinality:

Observation 2 (Cardinality of the Candidate Set)**.** Define $C_0(S)$ with respect to an input set S and values $a_1 < a_m$ as above. We have

$$|C_0(S)| \leq \lceil n/2 \rceil.$$

Proof. First, consider the fact that $C_0(S) \subseteq (S - a_1) \cap [a_1 : a_m]$ by definition. The largest element $s \in S$ such that $s - a_1 \in [a_1 : a_m]$ is $a_1 + a_m$, which in turn implies

$$|(S - a_1) \cap [a_1 : a_m]| \leq |\{s \in S \mid s \leq a_1 + a_m\}|. \tag{2}$$

By symmetry, the fact that $C_0(S) \subseteq (S - a_m) \cap [a_1 : a_m]$ implies that the smallest element $s \in S$ such that $s - a_m \in [a_1 : a_m]$ is $a_1 + a_m$, which gives

$$|(S - a_m) \cap [a_1 : a_m]| \leq |\{s \in S \mid s \geq a_1 + a_m\}|. \tag{3}$$

Combining Equations (1) to (3), we have

$$\begin{aligned}
|C_0(S)| &\leq \min(|(S - a_1) \cap [a_1 : a_m]|, |(S - a_m) \cap [a_1 : a_m]|) && (4)\\
&\leq \min(|\{s \in S \mid s \leq a_1 + a_m\}|, |\{s \in S \mid s \geq a_1 + a_m\}|) && (5)\\
&\leq \lceil |S|/2 \rceil, && (6)
\end{aligned}$$

as claimed. □

Observation 2 implies that the following simple algorithm for SSC runs in time $O^*(2^{n/2})$:

1. Fix an SSC input (S, k) and solution elements a_1 and a_m.
   (a) Construct $C_0(S)$ with respect to a_1 and a_m.
   (b) For every $B \subseteq C_0(S)$, test if $B + B \in S$ and $|B + B| \geq k$.

3.3 Candidate Graphs

In order to improve on our brute force algorithm, it will helpful to frame the problem in terms of a graph on the vertex set $C_0(S)$. The idea of constructing such a graph on possible elements of A is due to Lev [Lev]; here, we improve on the approach by using a smaller candidate set.

Definition 1 (Candidate Graph)**.** With respect to a SSC instance (S, k) and given solution elements a_1 and a_m, the *candidate graph* is the (pseudo)graph

$$G_S = (C_0(S), \{(c_i, c_j) | c_i + c_j \in S\}).$$

Note that we *include* the self-loop (c, c) if $c + c \in S$.

Given an edge (c_i, c_j), we refer to the sum $c_i + c_j$ as the *color* of the edge. The *frequency* $f(s)$ of integer s refers to the number of s-colored edges in G_S. By construction, there is a one-to-one mapping between solutions of (S, k) and cliques in G_S that include edges of at least k colors.

3.4 A Simple Exact Algorithm

Theorem 1. *SSC can be solved in time $O^*(3^{n/6}) = O^*(1.2010^n)$ and polynomial space.*

Proof. To solve a SSC instance (S, k), it suffices to find a clique in G_S that covers edges of at least k distinct colors. Because any solution clique is a subgraph of a *maximal* solution clique, it suffices to enumerate and check maximal cliques in the candidate graph G_S.

A graph on $|C_0(S)| \leq \lceil n/2 \rceil$ vertices contains at most $\Gamma := 3^{\lceil n/6 \rceil}$ maximal cliques [MM65], and these cliques can be enumerated in time $poly(n) \cdot \Gamma$ [TIAS77, LLRK80] and space $O(n^2)$ [TIAS77]. Checking each clique to see if it is a solution takes time and space $O(n^2)$. □

Obvious strategies to improve the runtime of this algorithm include further bounding the size of the candidate set and devising a strategy more efficient than enumerating all maximal cliques in the candidate graph. However, we have so far been unable to improve over $O^*(3^{n/6})$ in the worst case.

Candidate graphs with $\Omega^*(3^{n/6})$ maximal cliques correspond to SSC instances in which $k \approx n$. When $k \leq (1 - \Omega(1))n$, we can get exponential improvements on the runtime, although the algorithms are more involved. See Appendix A for details.

4 Hardness Results

Theorem 2 (Hardness of SSC)**.** *Our hardness results for SSC are as follows:*

1. *SSC is* NP*-complete.*
2. *SSC is* W[1]*-complete (parameterized in sumset size k).*
3. *SSC cannot be solved in time $2^{o(\sqrt{n})}$ under the Exponential Time Hypothesis.*

Proof. SSC is trivially in NP. Statements (1) and (2) thus follow immediately from Proposition 1 (reduction from k-Clique to SSC) and Proposition 2 (membership in W[1]), both proved below.

Our reduction transforms an instance of k-Clique on n vertices into an equivalent $O(n^2)$-item instance of $\frac{k(k+1)}{2}$-SSC. Thus a $2^{o(\sqrt{n})}$-time algorithm for SSC would allow us to solve k-Clique in time $2^{o(n)}$, which in turn would refute the ETH [IPZ01]. □

Proposition 1. *SSC is* NP*-hard and* W[1]*-hard.*

Proof. Given an instance $G = (V, E)$ of the NP-hard, W[1]-hard problem k-Clique, we transform it into an instance $(S, \frac{k(k+1)}{2})$ of SSC over $\mathbb{Z}^n$ as follows:

1. Convert G into a pseudograph by creating self-loops (v_i, v_i) for all $v_i \in V$.
2. Assign each vertex $v_i \in V$ the vector weight $w(v_i) = e_i \in \mathbb{Z}^n$, where e_i denotes the basis vector with a 1 in the i^{th} coordinate and zeroes elsewhere.
3. Assign each edge $(v_i, v_j) \in E$ the weight $w(v_i, v_j) = w(v_i) + w(v_j)$ and let S be the set of edge weights.

From SSC on $\mathbb{Z}^n$, we can transform the problem into an instance of SSC on $\mathbb{Z}$ using a standard conversion: each vertex v_i is assigned the integer weight $w(v_i) = 2^{2i}$, and edges are weighted with integer sums. For clarity of exposition, we complete the proof in $\mathbb{Z}^n$, but a straightforward analog of the following proof holds for SSC over $\mathbb{Z}$.

Note the reduction takes $poly(n)$ time and space. We prove that the resulting instance of $\frac{k(k+1)}{2}$-SSC is a YES instance if and only if G contains a k-clique.

($\Rightarrow$) Let $H = (V_H, E_H)$ be a k-clique of G (with self-loops added). Letting $w(V_H)$ denote the set of weights assigned to the vertices in V_H, we have $w(V_H)+w(V_H) \subseteq S$ and $|w(V_H)+w(V_H)| = \frac{k(k+1)}{2}$ by construction.

($\Leftarrow$) We first prove that our transformation creates a set S whose additive roots contain only basis vectors.

Claim 1. *Let $S \subseteq \{0,1,2\}^n$ be a set such that every vector $s \in S$ has ℓ_1-norm $||s||_1 = 2$. Then any set $A \subseteq \mathbb{Z}^n$ such that $A + A \subseteq S$ contains only elements in $\{e_i\}_{i\in[n]}$.*

Proof. Let $S \subseteq \{0,1,2\}^n$ be a set of integer vectors of ℓ_1-norm 2. Most elements of $\mathbb{Z}^n$ cannot be included in a set A satisfying $A + A \subseteq S$ because, when doubled, they create elements that do not appear in S. We observe that:

1. Any $v \in A$ satisfies $v \in \{0,1\}^n$. (This is because, if an integer vector $v \notin \{0,1\}^n$, some component of $v + v$ falls outside the set $\{0,1,2\}$, and thus $v + v \notin S$.)

2. Any $v \in A$ satisfies $||v||_1 = 1$. (Otherwise, $||v + v||_1 \neq 2$, and thus $v + v \notin S$.)

Thus the only elements of $\mathbb{Z}^n$ that can appear in A are basis vectors. □

Let $(S, \frac{k(k+1)}{2})$ be a SSC instance created by transforming an instance $G = (V,E)$ of k-Clique. By construction, $S = w(E)$ contains two kinds of elements in $\mathbb{Z}^n$:

1. Each edge $(u,v) \in E$ with $u \neq v$ creates an element $s \in S$ with $\#_0(s) = n - 2$ and $\#_1(s) = 2$.

2. Each self-loop (v,v) creates an element of $s \in S$ with $\#_0(s) = n - 1$ and $\#_2(s) = 1$.

By Claim 1, any set A with $A + A \subseteq S$ contains only basis vectors. For A containing only basis vectors, $|A + A| = k(k+1)/2$ if and only if $|A| = k$ because no two pairs of basis vectors have the same sum. Thus if S is a YES instance of $(\frac{k(k+1)}{2})$-Sumset Containment, $S \supseteq A + A$ for some set $A = \{e_{i_1}, \ldots, e_{i_k}\}$.

Each element of $A + A$ certifies the existence of exactly one edge in G on the vertex set $\{i_1, i_2, \ldots, i_k\}$. Thus $\{i_1, i_2, \ldots, i_k\}$ induces a k-clique on G.

We conclude that solutions to the instance of $\frac{k(k+1)}{2}$-SSC created by our transformation correspond exactly to solutions of the initial k-Clique instance. As a result, any FPT(or $\mathsf{poly}(n)$-time) algorithm for k-SSC would imply an FPT(resp. $\mathsf{poly}(n)$-time) algorithm for k-Clique. □

Proposition 2. *SSC is in* $\mathsf{W}[1]$.

Proof. To show SSC $\in \mathsf{W}[1]$, we show that SSC is FPT-reducible to the $\mathsf{W}[1]$-complete problem Weighted Weft-1 SAT. In Weighted Weft-1 SAT, we are given as input a constant-depth Boolean circuit of weft 1[2] and an integer k and accept if and only if the circuit is satisfied by some input of weight k.

Fix a SSC instance (S,k) and construct the graph

$$G = (V,E) := (S/2, \{(s_1, s_2) \mid s_1 + s_2 \in S\}).$$

(We do not include self-edges in G.) This simplified vertex set has more vertices than $C_0(S)$, but eliminates the need to guess a_1 and a_m.

Our Boolean circuit has $O(n^2)$ inputs, one for each edge in G. The circuit itself is the logical AND ($\wedge$) of the following clauses:

[2]That is, there exists at most one gate with unbounded fan-in on any path from an input to the output.

1. A clause $\neg(c_1, c_2) \vee \neg(c_3, c_4)$ for every pair of edges (c_1, c_2), $(c_3, c_4) \in E$ satisfying $c_1 + c_2 = c_3 + c_4$.
2. Clauses $\neg(c_1, c_2) \vee \neg(c_3, c_4)$ if the subgraph induced by the vertex set $\{c_1, c_2, c_3, c_4\}$ is not complete. (For two adjacent edges, e.g., (c_1, c_2) and (c_2, c_3), we check whether the subgraph induced by $\{c_1, c_2, c_3\}$ is not complete.)

We observe that this circuit has weft 1 and depth 2.

We claim our circuit has a satisfying assignment of weight exactly k if and only if SSC is a YES instance. First, suppose our formula has a satisfying assignment of weight exactly k. The satisfying assignment identifies k edges with distinct colors (certified by Condition 1) such that every endpoint of the selected edges is connected to every endpoint of every other edge (certified by Condition 2). That is, the graph induced by the endpoints of the selected edges is a clique, and the endpoints are an additive root A of S. (Notice that our choice of vertex set implies $a + a \in S$ for every $a \in A$.)

Conversely, if our SSC instance admits a solution A, A induces a clique on G that covers edges of at least k colors. Selecting any k colors from the edges of the solution clique satisfies our circuit. Thus an FPT-algorithm for Weighted Weft-1 SAT implies an FPT-algorithm for SSC. □

5 Related Problems

Open Problem 1, the question of faster exact algorithms, is our main open question. However, the algorithmic study of sumsets is still new, and many other interesting open problems exist.

5.1 Sumset Containment on Other Groups

The question of finding contained sumsets applies to groups beyond the integers. We expect the problem to remain difficult: for example, our NP- and W[1]-completeness proofs can be extended to $\mathbb{Z}^p$ and $\mathbb{F}_2^n$.

However, the additive structure of other groups may affect the problem. For example, over the Boolean hypercube $\mathbb{F}_2^n$, it is unclear how to construct a candidate set of cardinality less than n. Instead, the best natural candidate set may be the input set S itself: this follows from the observation that over $\mathbb{F}_2^n$, $(A + \{v\}) + (A + \{v\}) = A + A$ for any $v \in \mathbb{F}_2^n$. Thus, taking v to be any vector in A, we may assume $0 \in A$ without loss of generality. If $0 \in A$, this implies $A = A + \{0\} \subseteq A + A \subset S$. However, because $\mathbb{F}_2^n$ is easily decomposed into subspaces, this could make the problem more tractable.

Open Problem 2. *Design nontrivial exact algorithms for Sumset Containment on* $\mathbb{F}_2^n$.

5.2 Sum and Difference Set Containment

Another natural generalization is the problem of finding other sum and difference sets: given a pair of integers (r, s), discover whether there exist large sets of the form $rA + sB$ contained in S. The following problem might be a useful starting point:

Problem 2: Additive Decomposition

Input. A set of integers $S = \{s_1, s_2, \ldots, s_n\}$ and a parameter $k < n$.
Output. YES if there exist $A, B \subset \mathbb{Z}$ with $|A|, |B| > 1$, $|A + B| \geq k$, and $A + B \subseteq S$ or 'NO' if no solution is possible.

Every set S is the sum of $S + \{0\}$, so we specify $|A|, |B| > 1$ to avoid the trivial possibility. We may make the assumption that $s_1 = a_1 = b_1 = 0$ without loss of generality: an input S has a solution (A, B) if and only if $S - s_1 = S - (a_1 + b_1)$ has the solution $(A - a_1, B - b_1)$.

By analogy to Sumset Containment, we can view Additive Decomposition as the problem of finding a maximal biclique containing at least k colors on a candidate bipartite graph $G = (C_A \cup C_B, E)$, where C_A and C_B represent candidate elements for A and B, respectively. By using the assumption that $0 \in A, B$ and guessing the element $a_{max} = \max_{a \in A} a$, we can construct initial candidate sets of size $|C_{A,0}| = |C_{B,0}| \leq n/2 + 1$ by reasoning similarly to Observation 2. We can then solve the problem in time $O^*(2^{n/2})$ by enumerating all maximal bicliques [Pri00].

Open Problem 3. *Can k-Additive Decomposition be solved in time $O^*(2^{(1/2-\varepsilon)n})$ for some $\varepsilon > 0$?*

Acknowledgements. The author thanks Xi Chen, Shivam Nadimpalli and Rocco Servedio for a series of insightful discussions that informed this note, as well as anonymous reviewers who pointed out opportunities to clarify the presentation and fix mistakes.

References

[BFRV15] Laurent Bulteau, Guillaume Fertin, Romeo Rizzi, and Stéphane Vialette. Some algorithmic results for [2]-sumset covers. *Information Processing Letters*, 115(1):1–5, 2015.

[Bib13] Khodakhast Bibak. Additive combinatorics: With a view towards computer science and cryptography—an exposition. In Jonathan M. Borwein, Igor Shparlinski, and Wadim Zudilin, editors, *Number Theory and Related Fields*, pages 99–128, New York, NY, 2013. Springer New York.

[CKSY07] Michael J Collins, David Kempe, Jared Saia, and Maxwell Young. Nonnegative integral subset representations of integer sets. *Information Processing Letters*, 101(3):129–133, 2007.

[CNR$^+$] Xi Chen, Shivam Nadimpalli, Tim Randolph, Rocco A. Servedio, and Or Zamir. Testing sumsets is hard.

[FFV09] Isabelle Fagnot, Guillaume Fertin, and Stéphane Vialette. On finding small 2-generating sets. In *Computing and Combinatorics: 15th Annual International Conference, COCOON 2009 Niagara Falls, NY, USA, July 13-15, 2009 Proceedings 15*, pages 378–387. Springer, 2009.

[Fre99] Gregory A Freiman. Structure theory of set addition. *Asterisque-Societe Mathematique de France*, 258:1–20, 1999.

[GR17] Chris Godsil and Brendan Rooney. Hardness of computing clique number and chromatic number for cayley graphs. *European Journal of Combinatorics*, 62:147–166, 2017.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[Jia86] Tang Jian. An o (2 0.304 n) algorithm for solving maximum independent set problem. *IEEE Transactions on Computers*, 100(9):847–851, 1986.

[Lev] Seva Lev. Computational version of inverse sumset question. MathOverflow. URL:https://mathoverflow.net/q/358494 (version: 2020-04-25).

[LLRK80] Eugene L. Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9(3):558–565, 1980.

[Lov17] Shachar Lovett. *Additive Combinatorics and its Applications in Theoretical Computer Science.* Number 8 in Graduate Surveys. Theory of Computing Library, 2017.

[MM65] John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

[Pri00] Erich Prisner. Bicliques in graphs i: Bounds on their number. *Combinatorica*, 20(1):109–117, 2000.

[Rob86] John Michael Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986.

[Rob01] John M Robson. Finding a maximum independent set in time o (2n/4). Technical report, Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[TIAS77] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.

[Tre09] Luca Trevisan. Guest column: additive combinatorics and theoretical computer science. *ACM SIGACT News*, 40(2):50–66, 2009.

[TT77] Robert Endre Tarjan and Anthony E Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

[TV06] Terence Tao and Van H Vu. *Additive combinatorics*, volume 105. Cambridge University Press, 2006.

[Vio11] Emanuele Viola. *Selected Results in Additive Combinatorics: An Exposition.* Number 3 in Graduate Surveys. Theory of Computing Library, 2011.

A Improved Algorithms for $k < n$