Losing Weight by Gaining Edges

Amir Abboud* abboud@cs.stanford.edu
Kevin Lewi* klewi@cs.stanford.edu
Ryan Williams*† rrw@cs.stanford.edu

Abstract

We present a new way to encode weighted sums into unweighted pairwise constraints, obtaining the following results.

- Define the k-SUM problem to be: given n integers in $[-n^{2k}, n^{2k}]$ are there k which sum to zero? (It is well known that the same problem over *arbitrary* integers is equivalent to the above definition, by linear-time randomized reductions.) We prove that this definition of k-SUM remains W[1]-hard, and is in fact W[1]-*complete*: k-SUM can be reduced to $f(k) \cdot n^{o(1)}$ instances of k-Clique.
- The maximum node-weighted k-Clique and node-weighted k-dominating set problems can be reduced to $n^{o(1)}$ instances of the *unweighted* k-Clique and k-dominating set problems, respectively. This implies a strong *equivalence* between the time complexities of the node weighted problems and the unweighted problems: any polynomial improvement on one would imply an improvement for the other.
- A triangle of weight 0 in a node weighted graph with m edges can be deterministically found in $m^{1.41}$ time.

1 Introduction

One of the most basic problems over integers, studied in geometry, cryptography, and combinatorics, is k-SUM, the parameterized version of the classical NP-complete problem SUBSET-SUM.

Definition 1.1 (k-SUM)**.** The (k, M)-SUM problem is to determine, given n integers $x_1, \ldots, x_n \in [0, M]$ and a target integer $t \in [0, M]$, if there exists a subset $S \subseteq [n]$ of size $|S| = k$ such that $\sum_{i \in S} x_i = t$.[1] We define k-SUM $\triangleq (k, n^{2k})$-SUM.

Our definition of k-SUM is justified via the following known proposition:

Proposition 1.1. Every instance S of (k, M)-SUM can be randomly reduced in $O(kn \log M)$ time to an instance S' of k-SUM as defined above.

That is, there is an efficient randomized reduction from k-SUM over arbitrary integers, which we call k-SUM-$\mathbb{Z}$, to our definition of k-SUM (Appendix B). Furthermore, we show in Appendix B that this reduction can be made deterministic under standard hardness assumptions.

*Computer Science Department, Stanford University
†Supported in part by a David Morgenthaler II Faculty Fellowship, and NSF CCF-1212372.
[1]Without loss of generality, the range of integers can be $[-M, M]$ and the target integer can be zero.

A classical meet-in-the-middle algorithm solves k-SUM in $\tilde{O}(n^{\lceil k/2 \rceil})$ time and it has been a longstanding open problem to obtain an $O(n^{\lceil k/2 \rceil - \varepsilon})$ algorithm for any integer $k \geq 3$ and constant $\varepsilon > 0$. Logarithmic improvements are known for the $k = 3$ case [BDP08, GP14] (that is, the famous 3-SUM problem). The k-SUM conjecture [Pat10, AL13] states that k-SUM requires $n^{\lceil k/2 \rceil - o(1)}$ time and is known to imply tight lower bounds for many problems in computational geometry [GO95, HB96, BHP01] (and many more) and has recently been used to show conditional lower bounds for discrete problems as well [Pat10, VW09, JV13, AW14]. A matching $\Omega(n^{\lceil k/2 \rceil})$ lower bound for k-SUM was shown for a restricted model of computation called k-linear decision trees (LDTs) [Eri95, AC05], although it was recently shown that depth $O(n^{k/2}\sqrt{\log n})$ suffices for $(2k-2)$-LDTs [GP14]. It is also known that if there is an unbounded function $s : \mathbb{N} \to \mathbb{N}$ such that for infinitely many k, k-SUM is in $n^{k/s(k)}$, then the Exponential Time Hypothesis is false [PW10].

Despite intensive research on this simple problem, our understanding is still lacking in many ways, one of which is from the viewpoint of parameterized complexity. In their seminal work on parameterized intractability, Downey and Fellows [DF92, DF95] proved that k-SUM-$\mathbb{Z}$ is W[1]-hard and is contained in W[P]. The even simpler Perfect Code problem was conjectured to lie between the classes W[1] and W[2] [DF95] until Cesati proved it was W[1]-complete in 2002 [Ces02]. Classifying k-SUM-$\mathbb{Z}$ within a finite level of the W-hierarchy was open until in 2007, when Buss and Islam [BI07] proved that k-SUM-$\mathbb{Z} \in$ W[3].

The primary contribution of this work is a novel and generic way to efficiently convert problems concerning sums of numbers into problems on unweighted pairwise constraints. We call this technique "Losing weight by gaining edges" and report several interesting applications of it. One application is a new parameterized reduction from k-SUM to k-Clique and therefore the resolution of the parameterized complexity of k-SUM (for numbers in $[-n^{2k}, n^{2k}]$). Under standard lower bound hypotheses, we also obtain a deterministic reduction from k-SUM-$\mathbb{Z}$ to k-Clique as well.

Theorem 1.2. *k-SUM is* W[1]*-complete.*

The significance of showing W[1]-hardness for a problem is well known (as it rules out FPT algorithms). The significance of showing that a problem is in W[1] is less obvious, so let us provide some motivation. First, although W[1]-complete problems are probably not FPT, prominent problems in W[1] (such as k-Clique) can still be solved substantially faster than exhaustive search over all $\binom{n}{k}$ subsets [NP85]. In contrast, analogous problems in W[2] (such as k-Dominating Set) do not have such algorithms unless CNF Satisfiability is in $2^{\delta n}$ time for some $\delta < 1$ [PW10], which is a major open problem in exact algorithms. Therefore, understanding which parameterized problems lie in W[1] is closely related to understanding which problems can be solved faster than exhaustive search. Second, showing that a problem is in W[1] rather than W[3] means that it can be expressed in an apparently weaker logic than before, with fewer quantifiers [FG06]. That is, putting a problem in W[1] decreases the descriptive complexity of the problem.

Theorem 1.2 has applications to parameterized complexity, yielding a new characterization of the class W[1] as the problems FPT-reducible to k-SUM. Since k-SUM is quite different in nature from the previously known W[1]-complete problems, we are able to put other such "intermediate" problems in W[1], including weighted graph problems and problems with application to coding theory such as Weight Distribution [CP12].

To show that k-SUM $\in$ W[1], we prove a very tight reduction from k-SUM to k-Clique. Given an instance of k-SUM on n numbers, we generate $f(k) \cdot n^{o(1)}$ instances of k-Clique on n node graphs, such that one of these graphs contains a k-clique if and only if our k-SUM instance has

a solution. This implies that any algorithm for k-Clique running in time $O(n^c)$ for some $c \geq 2$ yields an algorithm for k-SUM running in time $n^{c+o(1)}$. Hence, the k-SUM conjecture implies an $n^{\lceil k/2 \rceil - o(1)}$ lower bound for k-Clique as well.

Generalizing our ideas further, we are able to prove surprising consequences regarding other weighted problems.

Removing node weights. Two fundamental graph problems are k-Clique and k-Dominating Set. Natural extensions of these problems allow the input graph to have weights on its nodes. The problem can then be to find a k-clique or a k-dominating set of minimum or maximum sum of node weights (the *min* and *max* versions), or to find a k-clique or a k-dominating set with total weight exactly 0 (the *exact* version, defined below).

Definition 1.3 (The Node-Weight k-Clique-Sum Problem)**.** For integers $k, M > 0$, the (k, M)-`NW-CLIQUE` problem is to determine, given a graph G, a node-weight function $w : V(G) \to [0, M]$, and a target weight $t \in [0, M]$, if there is a set S of k nodes which form a clique such that $\sum_{v \in S} w(v) = t$. We define the Node-Weight k-Clique-Sum problem as (k, n^{2k})-`NW-CLIQUE`.

Definition 1.4 (Node-Weight k-Dominating-Set-Sum)**.** For an integer $k > 0$, the Node-Weight k-Dominating-Set-Sum problem is to determine, given a graph G, a node-weight function $w : V(G) \to [0, n^{2k}]$, and a target weight $t \in [0, n^{2k}]$, if there is a set S of k nodes which form a dominating set such that $\sum_{v \in S} w(v) = t$.

These additional node weights increase the expressibility of the problem and allow us to capture more applications. How much harder are these node weighted versions compared to the unweighted versions? By weight scaling arguments, one can show that the "exact" version is harder than the max and min versions, in the sense that any algorithm for "exact" implies an algorithm for max or min with only a logarithmic overhead [NvLvdZ12] (and Theorem 3.3 in [VW09]). But how much harder is (for example) Node-Weight k-Clique-Sum than the case where there are no weights at all?

For k divisible by 3, the best k-Clique algorithms reduce the problem to 3-Clique on $n^{k/3}$ nodes, then use an $O(n^\omega)$ time algorithm for triangle detection [IR77] for a running time of $O(n^{\omega k/3})$ [NP85]. This reduction to the $k = 3$ case works for the node weighted case as well; combined with the recent $n^{\omega+o(1)}$ algorithms for node weighted triangle (3-clique) problems [CL09, VW09], we obtain $n^{\omega k/3+o(1)}$ running times for node weighted k-clique problems. The best k-Dominating Set algorithms reduce the problem to a rectangular matrix multiplication of matrices of dimensions $n^{k/2} \times n$ and $n \times n^{k/2}$ and run in time $n^{k+o(1)}$ [EG04]. These algorithms allow us to find all k-dominating sets in the graph and therefore can also solve the node weighted versions without extra cost.

Therefore, the state of the art algorithms for k-Clique and k-Dominating Set suggest that adding node weights does not make the problems much harder. Is that due to our current algorithms for the unweighted problems, or is there a deeper connection? Using the "Losing weight by gaining edges" ideas, we show that the node weighted versions of k-Clique and k-Dominating Set (and, in fact, *any* problem that allows us to implement certain "pairwise constraints") are essentially "equivalent" to the unweighted versions.

Theorem 1.5. *If k-Clique on n node and m edge graphs can be solved in time $T(n, m, k)$, then Node-Weight k-Clique-Sum on n node and m edge graphs can be solved in time $n^{o(1)} \cdot T(kn, k^2 m, k)$.*

If k-Dominating Set on n node graphs can be solved in time $T(n,k)$, then Node-Weight k-Dominating-Set on n node graphs can be solved in time $n^{o(1)} \cdot T(k^2 n, k)$.

Interestingly, Theorem 1.5 yields a short and simple $n^{\omega+o(1)}$ algorithm for the node-weighted triangle problems, while a series of papers were required to recently conclude the same upper bound using different techniques [VW06, VWY06, CL09, VW09]. Moreover, unlike the previous techniques, our approach extends to $k > 3$ and applies to more problems like k-Dominating Set.

Applying the result of Theorem 1.5 to the $O(m^{\frac{2\omega}{\omega+1}})$ triangle detection algorithm of Alon, Yuster and Zwick [AYZ97], we obtain a *deterministic* algorithm for Node-Weight Triangle-Sum in sparse graphs, improving the previous $n^{\omega+o(1)}$ upper bound [VW09] and matching the running time of the best randomized algorithm [VW09].

Corollary 1.1. *Node-Weight Triangle-Sum can be solved deterministically in $m^{1.41+o(1)}$ time.*

1.1 Overview of the Proofs

Let us give some intuition for Theorem 1.2. Both the containment in W[1] and the hardness for W[1] require new technical ideas. Downey and Fellows [DF92, DF95] proved that k-SUM-$\mathbb{Z}$ is W[1]-hard by a reduction requiring fairly large numbers: they are exponential in n, but can still be generated in an FPT way. To prove that k-SUM is W[1]-hard even when the numbers are only exponential in $k \log n$, we need a much more efficient encoding of k-Clique instances. We apply some machinery from additive combinatorics, namely a construction of large sets of integers avoiding trivial solutions to the linear equation $\sum_{i=1}^{k-1} x_i = (k-1)x_k$ [O'B11]. These sets allow us to efficiently "pack" a k-Clique instance into a $(\binom{k}{2}+k)$-SUM instance on small numbers.

Proving that k-SUM is in W[1] takes several technical steps. We provide a parameterized reduction from k-SUM on n numbers to only $f(k) \cdot n^{o(1)}$ graphs on $O(kn)$ nodes, such that some graph has a k-clique if and only if the original n numbers have a k-SUM. To efficiently reduce from numbers to graphs, we first reduce the numbers to an analogous problem on vectors. We define an intermediate problem (k,M)-`VECTOR-SUM`, in which one is given a list of n vectors from $\{-kM, \ldots, 0, \ldots, kM\}^d$, and is asked to determine if there are k vectors which sum to the all-zero vector. We give an FPT reduction from k-SUM to (k,M)-`VECTOR-SUM` where M and d are "small" (such that M^d is approximately equal to the original weights of the k-SUM instance). Next, we "push" the weights in these vectors onto the edges of a graph connecting the vectors, where the edge weights are much smaller than the original numbers: we reduce from (k,M)-`VECTOR-SUM` to edge-weighted k-clique-sum using a polynomial "squaring trick" which creates a graph with "small" edge weights, closely related in size to M. Finally, we reduce from the weighted problem to the unweighted version of the problem by brute-forcing all feasible weight combinations on the edges; as the edge weights are small, this creates $f(k) \cdot n^{o(1)}$ unweighted k-Clique instances for some function f.

Combining all these steps into one, one can view our approach as follows. We enumerate over all $\binom{k}{2}$-tuples of numbers $t = (\alpha_{i,j})_{i,j \in [k]}$ such that $\sum_{i,j} \alpha_{i,j} = 0$ where $\alpha_{i,j} \in [-M, M]$ for $M = f(k) \cdot \operatorname{poly}\log n$, and for each such tuple t we generate an instance of the unweighted problem. In this instance, two nodes are allowed to both be a part of our final solution (e.g. there is an edge between them in the k-clique case) if and only if some expression on the weights of the objects v_i and v_j evaluates to $F(w(v_i), w(v_j)) = \alpha_{i,j}$. The formulas are defined, via the "squaring trick", in such a way that there are k nodes satisfying these $\binom{k}{2}$ equations for some $\binom{k}{2}$-tuple t if and only if the sum of the weights of these k nodes is 0.

To implement our approach for k-Dominating Set we follow similar steps, except that we cannot implement the constraints on having a certain pair of objects in our solution by removing the edge between them anymore, since this does not prevent them from being in a feasible k-dominating set. This can be done, however, by adding extra nodes X to the graph such that the inclusion of pairs of nodes v_i, v_j in the solution S that do not satisfy our equations, $F(w(v_i), w(v_j)) \neq \alpha_{i,j}$, will prevent S from dominating all the nodes in X.

1.2 Related Work

There has been recent work in relating the complexity of k-SUM and variations of k-Clique for the specific case of $k = 3$. Pătraşcu [Pat10] shows a tight reduction from 3-SUM to listing 3-cliques; a reduction from listing 3-cliques to 3-SUM is given by Jafargholi and Viola [JV13]. Vassilevska and Williams [VW09] consider the exact edge-weight 3-clique problem and give a tight reduction from 3-SUM. For the case of $k > 3$, less is known, as the techniques used for the case of $k = 3$ do not seem to generalize easily. Abboud and Lewi [AL13] give reductions between k-SUM and various exact edge-weighted graph problems where the goal is to find an instance of a specific subgraph whose edge weights sum to 0.

2 Preliminaries

For $i < j \in \mathbb{Z}$, define $[i, j] \triangleq \{i, \ldots, j\}$. As shorthand, we define $[n] \triangleq [1, n]$. For a vector $\mathbf{v} \in \mathbb{Z}^d$, we denote by $\mathbf{v}[j]$ the value in the j^{th} coordinate of $\mathbf{v}$. We let $\mathbf{0}$ denote the all zeros vector. The default domain and range of a function is $\mathbb{N}$.

We define the k-Clique problem as follows.

Definition 2.1 (The k-Clique Problem)**.** For integers $k, n, m > 0$, the k-clique problem is to determine, given a graph G, if there is a size-k subset $S \subseteq [n]$ such that S is a clique in G.

The following problems are referred to in Corollary 3.2. They are simply the unparameterized versions of k-SUM and Exact Edge-Weight k-Clique, respectively.

Definition 2.2 (The Subset-SUM Problem)**.** The Subset-SUM problem is to determine, given a set of integers $x_1, \ldots, x_n, t$, if there exists a subset $S \subseteq [n]$ such that $\sum_{i \in S} x_i = t$.

Definition 2.3 (The Exact Edge-Weight Clique Problem)**.** For integers $n, m, M > 0$, the Exact Edge-Weight k-Clique problem is to determine, given an instance of a graph G on n vertices and m edges, a weight function $w : E(G) \to [-M, M]$, if there exists a set of nodes which form a clique with total weight 0.

3 From Numbers to Edges

Our results begin by showing how to reduce k-SUM to k-Clique. To do this, we first give a new reduction from k-SUM to k-Vector-Sum on n vectors in C^d for a set C which is relatively small compared to the numbers in the original instance. From k-Vector-Sum, we give a reduction to Edge-Weight k-Clique-Sum with small weights. Then, we can brute-force all possibilities for the $\binom{k}{2}$ edge weights for k-SUM and reduce to the (unweighted) k-Clique problem. Altogether, we conclude that k-SUM is in $\mathsf{W}[1]$.

3.1 Reducing k-SUM to k-Vector-Sum

We present a generic way to map numbers into vectors over small numbers such that the k-sums are preserved. We define the k-Vector-Sum problem as follows.

Definition 3.1 (The k-Vector-Sum Problem)**.** For integers $k, n, M, d > 0$, the k-vector-sum problem (k, M)-VECTOR-SUM is to determine, given vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathbf{t} \in [0, kM]^d$, if there is a size-$k$ subset $S \subseteq [n]$ such that $\sum_{i \in S} \mathbf{v}_i = \mathbf{t}$.

Note that the problem was considered by Bhattacharyya et al. [BIW+11] and also by Cattaneo and Perdrix [CP12].

Lemma 3.1. *Let $k, p, d, s, M \in \mathbb{N}$ satisfy $k < p$, $p^d \geq kM + 1$, and $s = (k+1)^{d-1}$. There is a collection of mappings $f_1, \ldots, f_s : [0, M] \times [0, kM] \to [-kp, kp]^d$, each computable in time $O(\text{poly} \log M + k^d)$, such that for all numbers $x_1, \ldots, x_k \in [0, M]$ and targets $t \in [0, kM]$,*

$$\sum_{j=1}^{k} x_j = t \qquad \Leftrightarrow \qquad \exists\, i \in [s] \quad \textit{such that} \quad \sum_{j=1}^{k} f_i(x_j, t) = \mathbf{0}.$$

Proof. For $x \in [0, kM]$, define $\mathbf{v}_x \in [0, p-1]^d$ to be the representation of x in base p—that is, let $\mathbf{v}_x[1], \ldots, \mathbf{v}_x[d] \in [0, p-1]$ be unique integers such that $x = \mathbf{v}_x[1] \cdot p^0 + \cdots + \mathbf{v}_x[d] \cdot p^{d-1}$. Note that $p^d > kM$.

For every $(d-1)$-tuple of numbers $\gamma = (c_1, \ldots, c_{d-1}) \in [0, k]^{d-1}$ and a number $t \in [0, kM]$, define the vector $\boldsymbol{\tau}_\gamma$ to be such that

$$\boldsymbol{\tau}_\gamma[j] = \begin{cases} \mathbf{v}_t[1] + c_1 \cdot p, & \text{if } j = 1 \\ \mathbf{v}_t[j] - c_{j-1} + c_j \cdot p, & \text{if } 2 \leq j \leq d-1 \\ \mathbf{v}_t[d] - c_{d-1}, & \text{if } j = d \end{cases}$$

Intuitively, each tuple γ represents a possible choice of the "carries" obtained in each component when computing the sum of their corresponding base-p numbers, and $\boldsymbol{\tau}_\gamma$ corresponds to the target vector in this base-p representation. Note that there are at most $s = (k+1)^{d-1}$ such vectors. We arbitrarily number these vectors as $\mathbf{t}_1, \ldots, \mathbf{t}_s$, and for all $i \in [s]$ we define the mapping

$$f_i(x, t) = k \cdot \mathbf{v}_x - \mathbf{t}_i.$$

Consider a set of k numbers $S = \{x_1, \ldots, x_k\} \subseteq [0, M]$. We claim that $\sum_{j=1}^{k} x_j = t$ if and only if there exists some $i \in [s]$ such that $\sum_{j=1}^{k} f_i(x_j, t) = \mathbf{0}$. First observe that for any vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k, \mathbf{t} \in \mathbb{Z}^d$, $\sum_{j=1}^{k} (k \cdot \mathbf{u}_j - \mathbf{t}) = \mathbf{0}$ if and only if $\sum_{j=1}^{k} \mathbf{u}_j = \mathbf{t}$. It suffices to show that $\sum_{j=1}^{k} x_j = t$ if and only if there exists some $\gamma \in [0, k]^{d-1}$ such that $\sum_{j=1}^{k} \mathbf{v}_{x_j} = \boldsymbol{\tau}_\gamma$.

For the first direction, $t = \sum_{i=1}^{k} x_i$. We can write this equation in base-p representation as

$$\sum_{j=0}^{d-1} \mathbf{v}_t[j+1] \cdot p^j = \sum_{i=1}^{k} \left(\sum_{j=0}^{d-1} \mathbf{v}_{x_i}[j+1] \cdot p^j \right) = \sum_{j=0}^{d-1} \left(\sum_{i=1}^{k} \mathbf{v}_{x_i}[j+1] \right) \cdot p^j.$$

Therefore, $\sum_{i=1}^{k} \mathbf{v}_{x_i}[1] \in [0, k(p-1)] = \mathbf{v}_t[1] + c_1 \cdot p$ for some $c_1 \in [0, k]$, $c_1 + \sum_{i=1}^{k} \mathbf{v}_{x_i}[2] = c_2 \cdot p + \mathbf{v}_t[2]$ for some $c_2 \in [0, k]$, and so on. Finally, $c_{d-1} + \sum_{i=1}^{k} \mathbf{v}_{x_i}[d] = \mathbf{v}_t[d]$. Letting $\gamma = (c_1, \ldots, c_{d-1}) \in [0, k]^{d-1}$, these equations imply exactly that $\sum_{i=1}^{k} \mathbf{v}_{x_i} = \boldsymbol{\tau}_\gamma$.

For the other direction, suppose $\sum_{i=1}^{k} \mathbf{v}_{x_i} = \boldsymbol{\tau}_\gamma$ where $\gamma = (c_1, \ldots, c_{d-1}) \in [0,k]^{d-1}$. Again, converting from base-p representation we obtain $\sum_{i=1}^{k} x_i = \sum_{j=0}^{d-1} \left(\sum_{i=1}^{k} \mathbf{v}_{x_i}[j+1]\right) \cdot p^j$, and using the definition of $\boldsymbol{\tau}_\gamma$ we can write the sum of the variables in S as

$$\begin{aligned}\sum_{i=1}^{k} x_i &= (\mathbf{v}_t[1] + c_1 \cdot p^1) + \left(\sum_{i=1}^{k} \mathbf{v}_{x_i}[1]\right) \cdot p^1 + \cdots + \left(\sum_{i=1}^{k} \mathbf{v}_{x_i}[d]\right) \cdot p^{d-1} \\ &= \mathbf{v}_t[1] + (\mathbf{v}_t[2] + c_2 \cdot p^1) \cdot p^1 + \cdots + \left(\sum_{i=1}^{k} \mathbf{v}_{x_i}[d]\right) \cdot p^{d-1} \\ &= \cdots = \mathbf{v}_t[1] + \mathbf{v}_t[2] \cdot p^1 + \cdots + \mathbf{v}_t[d] \cdot p^{d-1} = t,\end{aligned}$$

which completes the proof. □

Corollary 3.1. *Let $k, p, d, M, n > 0$ be integers with $k < p$ and $p^d \geq kM + 1$. k-SUM on n integers in the range $[0, M]$ can be reduced to $O(k^d)$ instances of $(k, p-1)$-*`VECTOR-SUM` *on n vectors in $[0, p-1]^d$.*

3.2 Reducing to k-Clique

Here, we consider a generalization of the k-SUM problem—namely, the Node-Weight k-Clique-Sum problem. We give a reduction from Node-Weight k-Clique-Sum to Edge-Weight k-Clique-Sum (defined below), where the new edge weights are much smaller than the original node weights. We then show how to reduce to many instances of the unweighted version of the problem, where each instance corresponds to a possible setting of edge weights. Then, we give an application of this general reduction to the Node-Weight k-Clique-Sum problem.

Definition 3.2 (The Edge-Weight k-Clique-Sum Problem)**.** For integers $k, M > 0$, the edge-weight k-clique-sum problem (k, M)-`EW-CLIQUE` is to determine, given a graph G, an edge-weight function $w : E(G) \to [0, M]$, and a target weight $t \in [0, M]$, if there is a set S of k nodes which form a clique such that $\sum_{(u,v)\in S} w(u, v) = t$.

Lemma 3.2. *Let $k, p, d, M > 0$ be integers such that $k < p$ and $p^d \geq kM + 1$, and let $M' = O(k^3 d p^2)$. The (k, M)-*`NW-CLIQUE` *problem can be deterministically reduced to $O(k^d)$ instances of (k, M')-*`EW-CLIQUE` *in time $O(k^d \cdot n^2 \cdot \operatorname{poly}\log M)$.*

Thinking of $p + d$ as "small", but $\operatorname{poly}(p, d) \approx kM$ as "large", we get a substantial reduction in the weights of the problem by "spreading" the node weights over the edges.

Proof. Let $G = (V, E)$ be a graph with a node weight function $w : V \to [0, M]$ and a target number $t \in [0, kM]$. Recall the mappings $f_i : [0, M] \times \times [0, kM] \to [-kp, kp]^d$ for $i \in [s]$ from Lemma 3.1, which maps numbers from $[0, M]$ into a collection of $s = O(k^d)$ length-d vectors with entries in $[-kp, kp]$. We translate the node-weight vector problem into an edge-weight problem via a "squaring trick," as follows. For each $i \in [s]$, we define an edge weight function $w_i : E \to [-M', M']$. For $(u, v) \in E$, let $\mathbf{u} = f_i(w(u), t)$ and $\mathbf{v} = f_i(w(v), t)$, and define

$$w_i(u, v) \triangleq \sum_{j=1}^{d} \left(\mathbf{u}[j]^2 + \mathbf{v}[j]^2 + 2(k-1)\mathbf{u}[j] \cdot \mathbf{v}[j]\right).$$

Note that for $M' = O(kdp^2)$, $w_i(u,v) \in [-M', M']$. We show that there is a k-clique in (G,w) of node-weight t if and only if for some $i \in [s]$, the edge-weighted graph (G, w_i) contains a k-clique of edge-weight 0. First, observe that for any k vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{Z}^d$,

$$\sum_{i=1}^{k} \mathbf{v}_i = \mathbf{0} \iff \sum_{j=1}^{d} \left(\sum_{i=1}^{k} \mathbf{v}_i[j] \right)^2 = 0.$$

Consider a set $S = \{u_1, \ldots, u_k\} \subseteq V$ that forms a k-clique in G. For any $i \in [s]$ and $u_a, u_b \in S$, let $\mathbf{u}_a = f_i(w(u_a), t)$ and $\mathbf{u}_b = f_i(w(u_b), t)$. Then, the edge-weight of S in (G, w_i) is

$$\sum_{1 \le a < b \le k} w_i(u_a, u_b) = (k-1) \sum_{a=1}^{k} \sum_{j=1}^{d} \mathbf{u}_a[j]^2 + 2(k-1) \sum_{1 \le a < b \le k} \sum_{j=1}^{d} \mathbf{u}_a[j] \cdot \mathbf{u}_b[j].$$

Since the sum is evaluated over all pairs $a, b \in [k]$ where $a < b$, the above quantity is equal to

$$(k-1) \cdot \sum_{j=1}^{d} \left(\sum_{u \in S} f_i(w(u), t)[j] \right)^2.$$

Therefore, for all $i \in [s]$, the edge-weight of S in (G, w_i) equals 0 if and only if the sum of the vectors $\sum_{u \in S} f_i(w(u), t)$ equals $\mathbf{0}$. And, by the properties of the mappings f_i from Lemma 3.1, the latter occurs for some $i \in [s]$ if and only if the node-weight of S in (G, w), $\sum_{u \in S} w(u)$, is equal to t, as desired. □

We observe that in the graphs produced by the above reduction, all k-cliques have non-negative weight. Therefore, Lemma 3.2 can also be viewed as a reduction to the "minimum-weight" k-Clique problem with edge weights, where the edge sum is minimized.

Finally, small weights on edges can simply be eliminated using a brute-force step.

Lemma 3.3. *For all integers $k, M > 0$, there is an $O(M^{\binom{k}{2}} \cdot n^2)$ time reduction from the problem (k, M)-*`EW-CLIQUE`* to $O(M^{\binom{k}{2}})$ instances of k-Clique on n nodes and $m \cdot \binom{k}{2}$ edges.*

Proof. Let $G = (V, E)$ be an edge-weighted graph with weight function $w : E \to [-M, M]$. For all possible $\binom{k}{2}$-tuples $\alpha = (\alpha_{1,2}, \alpha_{1,3}, \ldots, \alpha_{k-1,k})$ of integers from $[-M, M]$ such that $\sum_{i<j} \alpha_{i,j} = 0$, we make a k-partite graph G_α with $|V|$ nodes in each part—that is, $V(G_\alpha) = \{(v, i) \mid i \in [k], v \in V\}$. We put an edge in G_α between node (u, i) and node (v, j) if and only if $i < j$, $(u, v) \in E$, and $w(u, v) = \alpha_{i,j}$. Observe that $O(M^{\binom{k}{2}})$ graphs G_α are generated. We claim that G has a k-clique of weight 0 if and only if at least one of the unweighted graphs G_α contains a k-clique.

For the first direction, assume that for some $\alpha = (\alpha_{1,2}, \alpha_{1,3}, \ldots, \alpha_{k-1,k})$, the graph G_α contains a k-clique. Then, by the k-partite construction, it must have the form $\{(v_1, 1), \ldots, (v_k, k)\} \subseteq V(G_\alpha)$. Now let $S = \{v_1, \ldots, v_k\} \subseteq V$ and note that by the construction of G_α, S must form a k-clique in G with weight equal to $\sum_{i<j} \alpha_{i,j} = 0$. For the other direction, assume $\{u_1, \ldots, u_k\} \subseteq V$ is a k-clique in G of weight 0, then consider the $\binom{k}{2}$-tuple $\alpha = (w(u_1, u_2), w(u_1, u_3), \ldots, w(u_{k-1}, u_k))$. Then, by definition, G_α contains the k-clique $\{(u_1, 1), \ldots, (u_k, k)\}$. □

3.3 k-SUM is in W[1]

Using the above lemmas, we can efficiently reduce k-SUM to k-Clique. Consider a k-SUM instance (S,t) where $S = \{x_1, \ldots, x_n\} \subseteq [0, M]$ and $t \in [0, kM]$ with $M = n^{2k}$. Let $G = (V, E)$ be a node-weighted clique on n nodes $V = \{v_1, \ldots, v_n\}$ with weight function $w : V \to [0, M]$ such that $w(v_i) = x_i$ for all $i \in [n]$. Clearly, (S,t) has a k-SUM solution if and only if the instance (G, w, t) of (k, M)-NW-CLIQUE has a solution.

Set $d = \lceil \log n / \log\log n \rceil$ and $p = \lceil \log^{4k} n \rceil$, so that $p^d \geq (n)^{4k} > kM$. Using Lemma 3.2 the instance (G, w, t) of (k, M)-NW-CLIQUE can be reduced to $O(k^d) = O(n^{\log k / \log\log n})$ instances of (k, M')-EW-CLIQUE, where $M' = O(k^3 \cdot \log^{8k+1} n / \log\log n)$. Then, using Lemma 3.3, we can generate $g(n,k) = O(n^{\frac{\log k}{\log\log n}} \cdot k^{3k^2} \log^{8k^2+k} n)$ graphs on n nodes and $O(n^2)$ edges such that some graph has a k-Clique if and only if the original k-SUM instance has a solution.

For constant k, note that $g(n,k) = n^{o(1)}$, and hence:

Theorem 3.3. *For any $c > 2$, if k-Clique can be solved in time $O(n^c)$, then k-SUM can be solved in time $n^{c+o(1)}$.*

Furthermore, we remark that by applying the above reduction from k-SUM to k-Clique to the respective *unparameterized* versions of these problems, we obtain a reduction from Subset-SUM on arbitrary weights to Exact Edge-Weight Clique with small edge weights.

Corollary 3.2. *For any $\varepsilon > 0$, Subset-SUM on n numbers in $[-2^{O(n)}, 2^{O(n)}]$ can be reduced to $2^{\varepsilon n}$ instances of Exact Edge-Weight Clique on n nodes with edge weights are in $[-n^{O(1/\varepsilon)}, n^{O(1/\varepsilon)}]$.*

Note that Subset-SUM on n numbers in $[-2^{O(n)}, 2^{O(n)}]$ is as hard as the general case of Subset-SUM (by Lemma B.1), and the fastest known algorithm for Subset-SUM on n numbers runs in time $O(2^{n/2})$. The unweighted Max-Clique problem, which asks for the largest clique in a graph on n nodes, can be solved in time $O(2^{n/4})$ [Rob01]. Corollary 3.2 shows that even when the edge weights are small, the edge-weighted version of Max-Clique requires time $\Omega(2^{n/2})$ unless Subset-SUM can be solved faster.

An FPT Reduction. We show how to make the reduction fixed-parameter tractable. We can modify the oracle reduction for k-Clique above to get a many-one reduction to k-Clique if we simply take the disjoint union of the $g(n,k)$ k-Clique instances as a single k-Clique instance. The resulting graph has $n \cdot g(n,k)$ nodes, $O(n^2 \cdot g(n,k))$ edges, and has a k-clique if and only if one of the original graphs has a k-Clique. Then, we make the following standard argument to appropriately bound $g(n,k)$ via case analysis. If $k < \lceil \log\log n \rceil$, then $g(n,k) \leq n^{o(1)} \cdot 2^{f(k) \cdot \mathrm{poly}(k)}$. If $k \geq \lceil \log\log n \rceil$, then since $n \leq 2^{2^k}$, we have that $g(n,k) \leq 2^{2^k + f(k) \cdot \mathrm{poly}(k)}$. Therefore, $g(n,k) \leq n^{o(1)} \cdot h(k)$ for some computable $h : \mathbb{N} \to \mathbb{N}$, and we have shown the following:

Lemma 3.4. *k-SUM is in W[1].*

In Appendix B, we show how to obtain a randomized FPT reduction from the k-SUM problem over the integers to k-Clique, and how under plausible circuit lower bound assumptions, we can derandomize this reduction to show that k-SUM over the integers is in W[1]. This yields the first half of Theorem 1.2 (and we show the remainder, that k-SUM is W[1]-hard, in the next section).

3.4 Node-Weight k-Clique-Sum

The reduction of Section 3.3 shows that the Node-Weight k-Clique-Sum problem can be reduced to $n^{o(1)}$ instances of k-Clique, when k is a fixed constant. We observe that if the input graph has m edges, then the graphs generated by the reduction have no more than k^2m edges. Therefore, we have a tight reduction from node-weight clique to k-Clique.

This concludes the proof of the first half of Theorem 1.5 referencing k-Clique. We defer the proof of the second half of Theorem 1.5 concerning k-Dominating Set to Appendix C.

4 From k-Clique to k-SUM

In this section, we give a new reduction from k-clique to k-SUM in which the numbers generated are all in the interval $[-n^{2k}, n^{2k}]$. This proves that k-SUM is in fact W[1]-hard. We can view the result as an alternate proof for the W[1]-hardness of k-SUM without use of the Perfect Code problem, as done by Downey and Fellows [DF92]. The reduction is given from k-Clique to k-Vector-Sum (recall Definition 3.1), and then from k-Vector-Sum to k-SUM.

Lemma 4.1. *For an integer $k > 1$, k-Clique on n nodes and m edges reduces to an instance of $(k + \binom{k}{2}, k \cdot n^{1+o(1)})$-*`VECTOR-SUM` *deterministically in time $O(n^2)$.*

Proof. Given a graph $G = (V, E)$ we reduce an instance of k-Clique to an instance of k-Vector-Sum. First, we construct a k-sum-free set $D \subseteq [Q]$ of size n, where Q can be bounded by $n^{1+o(1)}$, and uniquely associate each vertex $v \in V$ with an element $q_v \in D$. By Lemma A.1, the set D can be constructed in time poly(n).

Let $\{e_1, \ldots, e_d\}$ be the standard basis of $\mathbb{R}^d$. Let $T = Q(k-1)+1$, $d = k^2 + k + 1$, and define $j \cdot e_i$ to be the length-d vector (1-indexed) with entry i consisting of the value j and all other entries being 0. We define a mapping $\boldsymbol{\eta} : (V \times [k]) \cup (E \times [k] \times [k]) \to [0, T]^d$ from the vertices and edges of the graph to length-d vectors over $[0, T]$. For each vertex $v \in V$ and $i \in [k]$, define

$$\boldsymbol{\eta}(v, i) \triangleq (T - (k-1)q_v) \cdot e_i + e_d$$

and for each edge $e = (u, v) \in E$ and $i, j \in [k]$ with $i < j$, define

$$\boldsymbol{\eta}(e, i, j) \triangleq q_u \cdot e_i + q_v \cdot e_j + e_{k \cdot i + j}.$$

Define the target vector $\mathbf{t} = k \cdot e_d + (\sum_{i=1}^{k} T \cdot e_i) + (\sum_{i,j \in [k], i<j} e_{k \cdot i + j})$. Then, there is a k-clique in the instance G of k-Clique on n nodes and m edges if and only if there is a solution to the instance $I = (\{\boldsymbol{\eta}(v, i) \mid v \in V,\ i \in [k]\} \cup \{\boldsymbol{\eta}(e, i, j) \mid e \in E,\ i, j \in [k],\ i < j\}, \mathbf{t})$ of $(k + \binom{k}{2}, T)$-`VECTOR-SUM`.

We now show correctness of the reduction. First, assume $S = \{u_1, \ldots, u_k\} \subseteq V$ forms a k-clique in G. Let $E(S)$ be the set of edges between vertices in S. We claim the set

$$A = \{\boldsymbol{\eta}(u_j, j) \mid u_j \in S\} \cup \{\boldsymbol{\eta}(e, i, j) \mid e = (u_i, u_j) \in E(S),\ i < j\},$$

with $|A| = k + \binom{k}{2}$, is a solution to the instance I. To see this, consider $\sum_{\mathbf{v} \in A} \mathbf{v} = \sum_{u_j \in S} \boldsymbol{\eta}(u_j, j) + \sum_{e=(u_i,u_j) \in E(S)} \boldsymbol{\eta}(e, i, j)$ and note that it equals

$$\sum_{j=1}^{k} \left((T - (k-1)q_{v_j}) \cdot e_j + e_d\right) + \sum_{j=1}^{k} (k-1)q_{v_j} \cdot e_j + \sum_{i,j \in [k], i<j} e_{k \cdot i + j} = \mathbf{t}.$$

For the other direction, consider some subset of vectors $A \subseteq \{\boldsymbol{\eta}(v,i) \mid v \in V,\ i \in [k]\} \cup \{\boldsymbol{\eta}(e,i,j) \mid e \in E,\ i,j \in [k],\ i \neq j\}$ of size $k + \binom{k}{2}$ for which the sum $\mathbf{z} = \sum_{\mathbf{v} \in A} \mathbf{v}$ equals the target vector $\mathbf{t}$. Let $A_V \subseteq V \times [k]$ represent the set of pairs (u,i) for $u \in V$ and $i \in [k]$ such that $\boldsymbol{\eta}(u,i) \in A$. Similarly, let $A_E \subseteq E \times [k] \times [k]$ represent the set of triples (e,i,j) for $e \in E$ and $i,j \in [k]$ such that $\boldsymbol{\eta}(e,i,j) \in A$.

Recall that $\mathbf{t}[d] = k$ and therefore by the definition of the mapping $\boldsymbol{\eta}$ from vertices to vectors, $|A_V|$ must equal k. Since $|A| = |A_V| + |A_E|$, it follows that $|A_E| = \binom{k}{2}$. Moreover, for all $i,j \in [k]$ with $i < j$, since we defined $\mathbf{t}[k \cdot i + j] = 1$, there is exactly one triple in A_E of the form (e,i,j). This implies that for every $j \in [k]$, there is at least one pair $(u,j) \in A_V$, for otherwise $\mathbf{z}[j]$ would be the sum of only $(k-1)$ numbers from $D \subseteq [Q]$, and since $\mathbf{t}[j] = T > Q(k-1)$, this sum cannot equal $\mathbf{t}[j]$. Note that $|A_V| = k$, and so for each $j \in [k]$ there is exactly one pair $(u,j) \in A_V$. Denote this vertex u by u_j.

For each $j \in [k]$, let $e_1, \ldots, e_{k-1}$ be the $k-1$ edges in E such that for each $i \in [k-1]$, either $(e_i, y, j) \in A_E$ or $(e_i, j, y) \in A_E$ for some $y \in [k]$. If the former holds, then let $e_i = (x, v_i)$ and if the latter holds, let $e_i = (v_i, x)$ for some $x, v_i \in V$. We claim that for $i \in [k-1]$, v_i must be identical to u_j. To see this, note that by the definition of the vectors, for all $j \in [k]$, $\mathbf{z}[j] = \boldsymbol{\eta}(u_j, j)[j] + \sum_{i=1}^{k-1} \boldsymbol{\eta}(e_i, i, j)[j] = T - (k-1)q_{u_j} + \sum_{i=1}^{k-1} q_{v_i}$, and since $\mathbf{z}[j] = \mathbf{t}[j] = T$, it must be the case that $\sum_{i=1}^{k-1} q_{v_i} = (k-1)q_{u_j}$. Then, since D is k-sum-free, this implies that $u_j = v_i$ for all $i \in [k-1]$, which concludes the claim.

Therefore, the vectors in A_V correspond to k vertices $V(A) \subseteq V$ and the vectors in A_E correspond to $\binom{k}{2}$ edges $E(A) \subseteq E$ such that precisely $k-1$ edges are incident to each vertex in $V(A)$. Thus, each edge must be incident to two vertices in $V(A)$, and so the vertices in $V(A)$ along with the edges in $E(A)$ form a k-clique in G. □

The following lemma gives a simple reduction from k-Vector-Sum to k-SUM, by the usual trick of converting from vectors to integers (via a Freiman isomorphism of order k).

Lemma 4.2. *(k,M)-`VECTOR-SUM` can be reduced to k-SUM on n integers in the range $[0, (kM+1)^d]$ in $O(n \log M)$ time.*

Proof. Let $p = kM + 1$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the n vectors of a k-Vector-Sum instance with target $\mathbf{t}$. For each vector $\mathbf{v}_i = \langle \mathbf{v}_i[j] \rangle_{j=0}^{d-1}$, define the integer $x_i \triangleq \sum_{j=0}^{d-1} \mathbf{v}_i[j] \cdot p^j$, and define $t = \sum_{j=0}^{d-1} \mathbf{t}_j \cdot p^j$. Then the k-SUM instance $(\{x_i\}_{i \in [n]}, t)$ has a solution if and only if the k-Vector-Sum instance $(\{\mathbf{v}_i\}_{i \in [n]}, \mathbf{t})$ has a solution. For correctness, let $S = \{i_1, \ldots, i_k\}$ be a set of k indices. Then, $\sum_{i \in S} x_i = \sum_{i \in S} \sum_{j=0}^{\ell-1} \mathbf{v}_i[j] \cdot p^j$. By switching the order of the summations, we have that this quantity is $\sum_{j=0}^{\ell-1} \sum_{i \in S} \mathbf{v}_i[j] \cdot p^j$. Since $\sum_{i \in S} \mathbf{v}_i[j] \in [0, p-1]$, we conclude by the uniqueness of the representation of t that $\mathbf{t}_j = \sum_{i \in S} \mathbf{v}_i[j]$ for all integers $j \in [0, d-1]$. Hence, the k-Vector-Sum instance has a solution described by the index set S as well. For the other direction, if $\mathbf{t}_j = \sum_{i \in S} \mathbf{v}_i[j]$ for all $j \in [0, d-1]$, then $\sum_{j=0}^{d-1} \mathbf{t}_j \cdot p^j = \sum_{j=0}^{d-1} \sum_{i \in S} \mathbf{v}_i[j] \cdot p^j$. The left side is equal to t and, by a switch of the order of summations, the right side is equal to $\sum_{i \in S} x_i$. Therefore, $t = \sum_{i \in S} x_i$ and so the k-SUM instance has a solution described by the index set S, as desired. □

We remark that in some cases, the proof can be changed slightly to yield smaller numbers in the k-SUM instance produced by the reduction. In particular, when reducing k-Clique to k-Vector-Sum, only the numbers in the first k coordinates can be as large as $k \cdot n^{1+o(1)}$ while the numbers in the last k^2+1 coordinates are bounded by k, and therefore, when reducing to $(k + \binom{k}{2}, M)$-`SUM` on $kn + \binom{k}{2}m$

numbers, the numbers generated can be bounded by $M = k^d \cdot (kn^{1+o(1)})^k \cdot k^{k^2+1} = O(k^{2k^2} \cdot n^{k+o(k)})$. In other words, we have reduced k-Clique to k'-SUM with numbers in the range $\left[-n^{\sqrt{k'}}, n^{\sqrt{k'}}\right]$, where $k' = k + \binom{k}{2}$.

The composition of Lemma 4.1 and Lemma 4.2 yields an FPT reduction, and we have obtained:

Lemma 4.3. *k-SUM is* W[1]*-hard.*

This concludes the proof of Theorem 1.2.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by a David Morgenthaler II Faculty Fellowship, and NSF CCF-1212372.

References

[AC05] N. Ailon and B. Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.

[AL13] A. Abboud and K. Lewi. Exact Weight Subgraphs and the k-Sum Conjecture. In *ICALP (1)*, pages 1–12, 2013.

[AW14] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. *CoRR*, abs/1402.0054, 2014.

[AYZ97] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[BDP08] I. Baran, E. D. Demaine, and M. Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

[Beh46] F. A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *NAS of the United States of America*, 32(12):331, 1946.

[BHP01] 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.

[BI07] J. F. Buss and T. Islam. Algorithms in the W-Hierarchy. *Theory Comput. Syst.*, 41(3):445–457, 2007.

[BIW+11] A. Bhattacharyya, P. Indyk, D. P. Woodruff, and N. Xie. The complexity of linear dependence problems in vector spaces. In *ICS*, pages 496–508, 2011.

[Ces02] M. Cesati. Perfect Code is W[1]-complete. *Inf. Process. Lett.*, 81(3):163–168, 2002.

[CL09] A. Czumaj and A. Lingas. Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication. *SIAM J. Comput.*, 39(2):431–444, 2009.

[CP12] D. Cattanéo and S. Perdrix. The parameterized complexity of domination-type problems and application to linear codes. *CoRR*, abs/1209.5267, 2012.

[DF92] R. G. Downey and M. R. Fellows. Fixed-parameter intractability. In *Structure in Complexity Theory Conference*, pages 36–49, 1992.

[DF95] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995.

[EG04] F. Eisenbrand and F. Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004.

[Eri95] J. Erickson. Lower bounds for linear satisfiability problems. In *SODA*, pages 388–395, 1995.

[FG06] J. Flum and M. Grohe. Parameterized Complexity Theory. Springer Verlag, 2006.

[GO95] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165 – 185, 1995.

[GP14] A. Grønlund and S. Pettie. Threesomes, Degenerates, and Love Triangles. *CoRR*, abs/1404.0799, 2014.

[HB96] A. Hernández-Barrera. Finding an $O(n^2 \log n)$ Algorithm Is Sometimes Hard. In *CCCG*, pages 289–294, 1996.

[IR77] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. STOC '77, pages 1–10, New York, NY, USA, 1977. ACM.

[JV13] Z. Jafargholi and E. Viola. 3SUM, 3XOR, Triangles. *CoRR*, abs/1305.3827, 2013.

[KvM02] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 2002.

[NP85] J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[NSS95] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *FOCS*, pages 182–191, 1995.

[NvLvdZ12] J. Nederlof, E. J. van Leeuwen, and R. van der Zwaan. Reducing a target interval to a few exact queries. In *MFCS*, pages 718–727, 2012.

[O'B11] K. O'Bryant. Sets of integers that do not contain long arithmetic progressions. *Electr. J. Comb.*, 18(1), 2011.

[Pat10] M. Patrascu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610, 2010.

[PW10] M. Pătraşcu and R. Williams. On the Possibility of Faster SAT Algorithms. In *SODA*, pages 1065–1075, 2010.

[Rob01] J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical report, 1251-01, LaBRI, Université de Bordeaux I, 2001.

[VW06] V. Vassilevska and R. Williams. Finding a maximum weight triangle in $n^{3\text{-delta}}$ time, with applications. In *STOC*, pages 225–231, 2006.

[VW09] V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. In *STOC*, pages 455–464, 2009.

[VWY06] V. Vassilevska, R. Williams, and R. Yuster. Finding the smallest h-subgraph in real weighted graphs and related problems. In *ICALP (1)*, pages 262–273, 2006.

A Additional Preliminaries

We review some useful combinatorial results that are needed in Section 4.

Definition A.1 ((n,k)-perfect hash function family)**.** For integers $n, k, s > 0$, a set of functions $F = \{f_i\}_{i\in[s]}$ where $f_i : [n] \to [k]$ is a (n,k)*-perfect hash function family of size* s if and only if for every size-k subset $S \subseteq [n]$, there exists some $f_i \in F$ such that $\cup_{j\in S} f_i(j) = [k]$.

Proposition A.1 (cf. [NSS95, Theorem 3])**.** An (n,k)-perfect hash function family of size $e^k \cdot k^{O(1)} \log n$ can be constructed deterministically in time $k^{O(k)} \cdot \text{poly}(n)$.

Lemma A.1. *For any $\varepsilon > 0$, there exists a $c > 0$ such that a k-sum-free set S of size n with $S \subset [0, (ckn)^{1+\varepsilon}]$ can be constructed in* poly(n) *time.*

Definition A.2 (k-sum-free set)**.** For any $k \geq 2$, a set $S \subseteq \mathbb{Z}$ is a k*-sum-free set* if and only if for all (not necessarily distinct) $x_1, \ldots, x_k \in S$, $\sum_{i\in[k-1]} x_i = (k-1) \cdot x_k$ implies that $x_1 = \cdots = x_k$.

We outline a construction of a k-sum-free set, derived from a small modification to Behrend's original construction [Beh46].

Constructing k-sum-free Sets. For a vector $\mathbf{v}$, let $\|\mathbf{v}\|$ represent the ℓ_2 norm of $\mathbf{v}$, and let $|\langle \mathbf{v}_1, \mathbf{v}_2\rangle|$ represent the dot product of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Let $p = (k-1)d - 1$. Consider the set $S_r(m,d)$ consisting of all integers x that can be written as $x = \sum_{i=0}^{m-1} a_i p^i$, where for all $i \in [0, m-1]$, $a_i \in [0, d-1]$ and $\|\langle a_0, \ldots, a_{m-1}\rangle\| = \sqrt{r}$. For an integer $x \in S_r(m,d)$ of the form $x = \sum_{i\in[0,m-1]} a_i \cdot p^i$, let $\|x\|$ represent the ℓ_2 norm of the vector $\langle a_i\rangle_{i\in[0,m-1]}$. By definition, $\|x\| = \sqrt{r}$.

Claim A.1. For any vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$, if $\|\mathbf{v}_1\| = \cdots = \|\mathbf{v}_k\|$ and $\sum_{i\in[k]} \|\mathbf{v}_i\| = \left\|\sum_{i\in[k]} \mathbf{v}_i\right\|$, then $\mathbf{v}_1 = \cdots = \mathbf{v}_k$.

Proof. Note that

$$\left\|\sum_{i\in[k]} \mathbf{v}_i\right\|^2 = \sum_{i\in[k]} \langle \mathbf{v}_i, \mathbf{v}_i\rangle + 2 \sum_{i,j\in[k],i<j} |\langle \mathbf{v}_i, \mathbf{v}_j\rangle| = \sum_{i\in[k]} \|\mathbf{v}_i\|^2 + 2 \sum_{i,j\in[k],i<j} |\langle \mathbf{v}_i, \mathbf{v}_j\rangle|.$$

By the Cauchy-Schwarz inequality, $|\langle \mathbf{v}_i, \mathbf{v}_j\rangle| \leq \|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\|$. Thus, in order for the above quantity to be equal to $\left(\sum_{i\in[k]} \|\mathbf{v}_i\|\right)^2$, it must be the case that $|\langle \mathbf{v}_i, \mathbf{v}_j\rangle| = \|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\|$ for all $i, j \in [k]$. Hence,

also by Cauchy-Schwarz, $\mathbf{v}_i$ and $\mathbf{v}_j$ are linearly dependent. Then, using the fact that $\|\mathbf{v}_i\| = \|\mathbf{v}_j\|$, it follows that $\mathbf{v}_i = \mathbf{v}_j$ for all pairs $i, j \in [k]$, which completes the proof. □

Lemma A.2. *For all integers $k \geq 2$, for all $\ell \in [0, m(d-1)^2]$, S_ℓ is k-sum-free.*

Proof. Let $x_1, \ldots, x_k \in S_\ell(m,d)$ be such that $\sum_{i\in[k-1]} x_i = (k-1)\cdot x_k$. Then, $\|\sum_{i\in[k-1]} x_i\| = (k-1)\cdot\|x_k\|$. Since $\|x_1\| = \|x_2\| = \cdots = \|x_k\|$, it follows that

$$\sum_{i\in[k-1]} \|x_i\| = \left\| \sum_{i\in[k-1]} x_i \right\|.$$

Hence, by Claim A.1, it follows that $x_1 = \cdots = x_k$. □

There are d^m possible settings for $a_0, \ldots, a_{m-1}$ where each $a_i \in [0, d-1]$, and the value $\|\langle a_i\rangle_{i\in[0,m-1]}\|^2$ lies within the range $[0, m(d-1)^2]$. Hence, by the pigeonhole principle, there exists some $r \in [0, m(d-1)^2]$ such that $|S_r(m,d)| \geq \frac{d^m}{m(d-1)^2+1} \geq d^{m-2}/m$. Note that $S_r(m,d) \subset [0, p^m]$. Hence, to obtain a k-sum-free set of size n, we have that $d = (m\cdot n)^{\frac{1}{m-2}}$. Recall that $p = \Theta(kd)$, and so $p^m = O(kmn)^{\frac{m}{m-2}}$. For any $\varepsilon > 0$, we can set $m = 2/\varepsilon + 2$ so that $p^m = (ckn)^{1+\varepsilon}$ for some constant $c > 0$. Hence, $S_r(m,d) \subset [0, (ckn)^{1+\varepsilon}]$. For any n, we can compute, for each $\ell \in [0, m(d-1)^2]$, the exact size of $S_\ell(m,d)$, returning the ℓ for which $|S_\ell(m,d)|$ is maximized. Note that $m(d-1)^2 = O(n^\varepsilon)$, and the construction of $S_\ell(m,d)$ requires time linear in $|S_\ell(m,d)|$. Lemma A.1 follows.

A.1 Background on Parameterized Complexity

For an exposition of fixed parameter tractability, reducibility, and the W-hierarchy, we refer the reader to [DF92, DF95, FG06]. Here we give a brief overview.

A parameterized problem is defined to be a subset of $\{0,1\}^\star \times \mathbb{N}$. Let L, L' be parameterized problems. L is in the class FPT if there is an algorithm A, constant c, and computable function $f : \mathbb{N} \to \mathbb{N}$, such that on all inputs $y = (x,k)$, $A(y)$ decides whether y is in L and runs in time at most $f(k)\cdot|x|^c$. A *FPT reduction* from L to L' is an algorithm R mapping $\{0,1\}^\star\times\mathbb{N}$ to $\{0,1\}^\star\times\mathbb{N}$, such that for all $y = (x,k)$, $R(y) \in L'$ if and only if $y \in L$, $R(y)$ runs in $f(k)\cdot|x|^c$ time (for some c and f), and $R(y) = (x', k')$ where $k' \leq g(k)$ for some computable function g.

Although it is not the original definition, it is equivalent to define W[1] to be the class of parameterized problems which have an FPT reduction to k-Clique [DF95, FG06].

B A Deterministic FPT Reduction from k-SUM to k-Clique

In this section we show how, by assuming a plausible circuit lower bound assumption, we are able to strengthen our result from Section 3.3 to a reduction from general k-SUM to k-Clique (as opposed to only reducing k-SUM on numbers in $[-n^{2k}, n^{2k}]$ to k-Clique).

To do this, we give a randomized process which takes a k-SUM instance and outputs a collection of k-SUM instances on numbers in $[-n^{2k}, n^{2k}]$ which, with high probability, are such that a member of the collection contains a solution if and only if the original k-SUM instance contains a solution. Then, we apply a theorem which allows us to derandomize this reduction given the appropriate circuit lower bound.

A Randomized Weight Reduction. We first define a randomized process (cf. [KvM02, Definition 4.1]) consisting of an algorithm F along with a predicate π. In what follows, the algorithm F will represent a (randomized) oracle reduction from general k-SUM to k-SUM on numbers in $[-n^{2k}, n^{2k}]$ which succeeds on some random bit sequences, and the predicate π represents whether or not the input random bit sequence is such that F produces a valid oracle reduction. Let $z = O(k \cdot \operatorname{poly}\log n)$, and let $\mathcal{D}$ represent the domain of k-SUM instances on n integers.

Let $F : \mathcal{D} \times \{0,1\}^z \to \mathcal{D}^k$ be a process which takes as input x a k-SUM instance along with a random bit sequence $r \in \{0,1\}^z$. Let the instance x describe a k-SUM instance consisting of a set S of n integers in the range $[0, M]$ with target value 0 (without loss of generality). F uses the sequence of random bits r to choose a prime $p \in [2, dn^k \log n \log(kM)]$ by picking random integers within the interval until a prime is obtained. Let S' represent the set of integers derived by taking each integer in S modulo p. The output of F is a collection of k instances of k-SUM, each with base set S', and the i^{th} instance has target value $i \cdot p$, for $i \in [0, k-1]$.

Let $\pi : \mathcal{D} \times \{0,1\}^z \to \{0,1\}$ be a predicate which takes as input x a k-SUM instance along with a random bit sequence $r \in \{0,1\}^z$ and outputs a single bit. The predicate $\pi(x, r)$ runs $F(x, r)$ to obtain a collection C of k instances of k-SUM, and outputs 1 if and only if the k-SUM instance x has a solution and there exists some k-SUM instance in C which does not have a solution, or, the original k-SUM instance does not have a solution and none of the k-SUM instances in C have a solution. The following lemma is folklore, and also informally stated in [BDP08], but we provide a proof here for completeness, as it also justifies Proposition 1.1.

Lemma B.1. *Let $k, n, M, M' > 0$ be integers such that $M' = O(kn^k \log n \log(kM))$, and let $d > 0$ be a sufficiently large constant. Let C be the collection of k-SUM instances output by F on input x, a k-SUM instance on n integers in the range $[0, M]$, and a random bit sequence $r \in \{0,1\}^z$. Then, each k-SUM instance in C is on n integers in the range $[0, M']$, and $\pi(x, r) = 1$ with probability at least $1 - 1/(cd)$. for some fixed constant $c > 0$.*

Proof. Let S be the set of integers of the k-SUM instance. For a sufficiently large constant $d > 0$, for a randomly chosen prime $p \in [2, d \cdot n^k \log n \log(kM)]$, consider the new set S' where we take all integers of S modulo p. Clearly, if S has a k-SUM solution, then S' also has a solution, and hence, there exists a member of the collection of k-SUM instances output by F which contains a solution. For any k-tuple of integers in $[0, M]$ with nonzero sum s, we have $s \in [kM]$ so s has at most $\log(kM)$ prime factors. By the prime number theorem, for sufficiently large n, there are at least $cd \cdot n^k \log(kM)$ primes in this interval for a fixed constant $c > 0$, and so the probability that p is a prime factor of s is at most

$$\frac{\log(kM)}{cdn^k \log(kM)} \leq \left(\frac{1}{cd \cdot n^k}\right).$$

By a union bound over all k-tuples of numbers from S', the probability of a false positive in S' is at most $1/(cd)$. We have shown that if the k-SUM instance S with target value 0 does not contain a solution, then with probability at least $1 - 1/(cd)$ over the choice of a random prime p, S does not contain a k-tuple of integers which sum to a multiple of p, and hence, there does not exist a member of the collection of k-SUM instances output by F which contains a solution.

Finally, we can remove the "modulo p" constraint from S' as follows. Treating all elements in S' as integers in $[0, p-1]$, for each $z \in \{-k \cdot p, \ldots, -p, 0, p, \ldots, k \cdot p\}$ we reduce to checking whether there are $x_1, \ldots, x_k \in S'$ such that $\sum_i x_i = z$ (where this sum is over the integers). Such a k-tuple exists if and only if $\sum_i x_i$ equals 0 mod p. □

Derandomizing the Reduction. Let B be an oracle for SAT, and assume there exists a function $f \in \mathsf{E}$ with B-oracle circuit complexity $2^{\varepsilon n}$ for some $\varepsilon > 0$. We note that, for all k-SUM instances x and random bit sequences r, $\pi(x, r)$ can be decided by a B-oracle circuit of size n^a for some positive constant a, since we can construct a circuit which creates and then solves the k-SUM instances to check the predicate. We also have by Lemma B.1 that for all x, for uniformly sampled $r \in \{0,1\}^z$, $\Pr[\pi(x, r) = 1]$ with probability $1 - 1/(cd)$, where c is a universal constant and d can be chosen to be arbitrarily large. We now apply the following derandomization theorem due to Klivans and van Melkebeek [KvM02].

Theorem B.1 (cf. [KvM02, Theorem 4.4])**.** *Let B be an oracle, b a positive constant, and $\ell : \mathbb{N} \to \mathbb{N}$ a constructible function. Let (F, π) be a randomized process using a polynomial number of random bits such that B can efficiently check (F, π). If there exists a Boolean function $f \in \mathsf{E}$ such that $C_f^B(\ell(n)) = \Omega(n)$, then there exists a function G computable in E and a constructible function $s(n) = O(\ell^2(n^{O(1)})/\log n)$ such that for any input x of length n,*

$$|\Pr_r[\pi(x, r) = 1] - \Pr_s[\pi(x, G(s)) = 1]| \leq O(1/n^b)$$

where r is uniformly distributed over $\{0,1\}^{r(n)}$ and s over $\{0,1\}^{s(n)}$.

By applying Theorem B.1 with $\ell(n) = \log n$, there exists a constant $s > 0$ and an efficiently computable function G computable in with seed length $s \log n$ such that for any input x of length n, $|\Pr[\pi(x, r) = 1] - \Pr[\pi(x, G(\sigma)) = 1]| = O(1/n^b)$. By choosing $d = \Omega(n)$, we have that $\Pr[\pi(x, G(\sigma)) = 1] \geq 1 - 1/n$.

The Deterministic FPT Reduction. Given an instance x of k-SUM, for each $\sigma \in \{0,1\}^{s \log n}$, let C_σ be the collection of instances of k-SUM output by $F(x, G(\sigma))$. Note that C_σ is in fact a collection of instances of k-SUM on numbers in $[-n^{2k}, n^{2k}]$ by Lemma B.1. Hence, as described in Section 3.3, we can reduce each member of C_σ to an instance of k-Clique—let S_σ represent this collection of k-Clique instances, obtained by applying the reduction to each member of C_σ. Let S^* be a family of k-Clique instances obtained by taking the union of S_σ over all $\sigma \in \{0,1\}^{s \log n}$. The original k-SUM instance x contains a solution if and only if at least a $1/n$ fraction of the members of S^* contain a k-Clique.

To see the correctness of this deterministic reduction, note that at most a $1/n$ fraction of $\sigma \in \{0,1\}^{s \log n}$ are such that $\pi(x, G(\sigma)) = 0$. For all other seeds σ, $\pi(x, G(\sigma)) = 1$ and the process F correctly reduces its input x to a collection of k-SUM instances. Therefore, if x contains a solution, then all members of S^* will contain a k-Clique, and if x does not contain a solution, then at most a $1/n$ fraction of the members of S^* could possibly contain a k-Clique. As the constant s in the seed length is fixed, we note that the process F also runs in time polynomial in its input and therefore our reduction is FPT. We have shown the following theorem.

Theorem B.2. *Let B be an oracle for SAT. If there is a function $f \in \mathsf{E}$ with B-oracle circuit complexity $2^{\varepsilon n}$ for some $\varepsilon > 0$, then k-SUM $\in \mathsf{W}[1]$.*

We note that the oracle B can be replaced with any oracle sufficient to efficiently compute the predicate π—for example, Theorem B.2 follows where B is an oracle for k-SUM.

C Node-Weight Dominating Set

Lemma C.1. *For every fixed $k \geq 2$, Node-Weight k-Dominating-Set-Sum can be reduced to $n^{o(1)}$ instances of k-Dominating Set on graphs on $O(k^2 n)$ nodes. The reduction runs in $n^{2+o(1)}$ time.*

Proof. Given a node-weighted graph $G = (V, E), w : V \to [-n^{2k}, n^{2k}]$, we first use Lemma 3.1 with $d = O(\log n / \log\log n)$ and $p = O(k2^k \log n)$ to get $s = O(n^{\log k / \log\log n}) = n^{o(1)}$ instances of the problem where the weight of every node is a vector in $[p]^d$ instead of a number in $[-n^{2k}, n^{2k}]$. We treat each instance $i \in [s]$ separately, as follows.

For a pair of nodes $u, v \in V$, let their weights be $\mathbf{u}, \mathbf{v} \in [p]^d$ and define the expression

$$F(u,v) = \sum_{j=1}^{d} \left(\mathbf{u}[j]^2 + \mathbf{v}[j]^2 + 2(k-1)\mathbf{u}[j] \cdot \mathbf{v}[j] \right),$$

which corresponds to the "squaring trick" from Lemma 3.2.

We enumerate over all $\binom{k}{2}$-tuples of numbers $t = (\alpha_{i,j})_{i,j \in [k]}$ such that $\sum_{i,j} \alpha_{i,j} = 0$ where $\alpha_{i,j} \in [-M, M]$ where $M = O(\text{poly}\log n)$ is an upper bound on the $F(u,v)$ values, and for each such tuple t we generate an instance of k-Dominating Set, an unweighted graph G_t. Note that the number of such tuples is $\log^{O(k)} n = n^{o(1)}$.

Let V_0, $V_1, \ldots, V_k$, and $V_{i,j}$ for every $i < j, i, j \in [k]$ be $\binom{k}{2} + k + 1$ copies of the node set V. The node set of G_t is $V_0 \cup \bigcup_i V_i \cup \bigcup_{i<j} V_{i,j}$. Let us denote the copy of node v in the set V_X by v_X. For every edge $(u, v) \in E$ of G we add the edges (u_i, v_0) and (v_i, u_0), for every $i \in [k]$, to G_t. We also make the copies $V_1, \ldots, V_k$ cliques by adding edges (v_i, u_i) for every $u, v \in V$ and $i \in [k]$. Note that this forces every k-dominating set in G_t to contain exactly one node v_i in every set V_i, and that such k nodes dominate every node in V_0 if and only if they correspond to a k-dominating set in G. Finally, we add the following edges that simulate "gaining edges" by allowing pairs of nodes u_i, v_j to be in a k-dominating set in G_t if and only if they are consistent with the tuple t which is chosen to guarantee that the total sum of weights is zero. For every pair of nodes $u \neq v$ in G and indices $i < j$ in $[k]$ we add the edge $(u_i, v_{i,j})$ to G_t and then check if $F(u,v) = \alpha_{i,j}$, and if so, we add the edge $(u_j, v_{i,j})$ to G_t.

We now claim that a subset $S \subseteq V$ of size k is k-dominating set of weight zero in G if and only if the nodes $S' = \{x_{i(x)} \mid x \in S\}$ where $i(x)$ is the index of x in an arbitrary ordering of S, are a k-dominating set in G_t, for some tuple t and some instance $i \in [s]$. To see this, first note that a set of k nodes $S[1]_1 \in V_1, \ldots, S[k]_k \in V_k$ are a dominating set in G_t if and only if $S[1], \ldots, S[k]$ are a dominating set in G and for every pair $i < j$ in $[k]$, the condition $F(w(S[i]), w(S[j])) = \alpha_{i,j}$ is satisfied. By definition of our formulas $F(\cdot,\cdot)$ and the proofs of Lemmas 3.1, 3.2, and 3.3, we get that $S[1]_1, \ldots, S[k]_k$ are a dominating set in G_t for some tuple t and instance $i \in [s]$ if and only if $S[1], \ldots, S[k]$ are a dominating set in G of total weight zero. □