

Losing Weight by Gaining Edges

Amir Abboud, Kevin Lewi, and Ryan Williams

Computer Science Department
Stanford University

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 $x_1, \dots, x_n \in [0, M]$ 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 1.2. *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.*

¹ Without loss of generality, the range of integers can be $[-M, M]$ and the target integer can be zero.

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. Furthermore, we show in the full version that this reduction can be made deterministic under standard hardness assumptions.

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 [5,18] (that is, the famous 3-SUM problem). The k -SUM conjecture [25,1] 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 [17,19,6] (and many more) and has recently been used to show conditional lower bounds for discrete problems as well [25,29,21,2]. 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) [15,3], although it was recently shown that depth $O(n^{k/2} \sqrt{\log n})$ suffices for $(2k - 2)$ -LDTs [18]. It is also known that if there is an unbounded function $s : \mathbb{N} \rightarrow \mathbb{N}$ such that for infinitely many k , k -SUM is in $n^{k/s(k)}$, then the Exponential Time Hypothesis is false [26].

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 [12,13] 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 $W[2]$ [13] until Cesati proved it was $W[1]$ -complete in 2002 [10]. Classifying k -SUM- \mathbb{Z} within a finite level of the W -hierarchy was open until in 2007, when Buss and Islam [8] 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 1.3. *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 [23]. 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 $\delta < 1$ [26], 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 [16]. That is, putting a problem in $W[1]$ decreases the descriptive complexity of the problem.

Theorem 1.3 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 [9] (for which the details are given in the full version).

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 1.4 (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 $w : V(G) \rightarrow [0, M]$, 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 1.5 (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 $w : V(G) \rightarrow [0, n^{2k}]$, 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 [22] (and Theorem 3.3 in [29]). 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 [20] for a running time of $O(n^{\omega k/3})$ [23]. 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 [11,29], 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 $n^{k+o(1)}$ [14]. 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 1.6. *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 $n^{o(1)} \cdot T(kn, k^2m, k)$. 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 $n^{o(1)} \cdot T(k^2n, k)$.*

Interestingly, Theorem 1.6 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 [28,30,11,29]. Moreover, unlike the previous techniques, our approach extends to $k > 3$ and applies to more problems like k -Dominating Set.

Applying Theorem 1.6 to the $O(m^{\frac{\omega}{\omega+1}})$ triangle detection algorithm of Alon, Yuster and Zwick [4], we obtain a *deterministic* algorithm for Node-Weight Triangle-Sum in sparse graphs, improving the previous $n^{\omega+o(1)}$ upper bound [29] and matching the running time of the best randomized algorithm [29].

Corollary 1.7. *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 1.3. Both the containment in $W[1]$ and the hardness for $W[1]$ require new technical ideas. Downey and Fellows [12,13] 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 $\sum_{i=1}^{k-1} x_i = (k-1)x_k$ [24]. 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 $\{-kM, \dots, 0, \dots, kM\}^d$, 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 $M = f(k) \cdot \text{poly log } n$, 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 [25] 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 [21]. Vassilevska and Williams [29] 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 [1] 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 $[i, j] \triangleq \{i, \dots, j\}$. 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 3.8. 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, \dots, 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 $w : E(G) \rightarrow [-M, M]$, 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 $W[1]$.

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 $\mathbf{v}_1, \dots, \mathbf{v}_n, \mathbf{t} \in [0, kM]^d$, 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. [7] and also by Cattaneo and Perdrix [9].

Lemma 3.2. *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 $f_1, \dots, f_s : [0, M] \times [0, kM] \rightarrow [-kp, kp]^d$, each computable in time $O(\text{poly log } M + k^d)$, such that for all numbers $x_1, \dots, x_k \in [0, M]$ and targets $t \in [0, kM]$,*

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

The idea is simple: in a natural translation of numbers into vectors, to preserve k -sums we have to keep track of the carries that may occur. These f_i 's effectively try “all possible” carries there can be among a sum of k numbers. The proof is given in the full version.

Corollary 3.3. *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 3.4 (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 $w : E(G) \rightarrow [0, M]$, 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 3.5. *Let $k, p, d, M > 0$ be integers such that $k < p$ and $p^d \geq kM + 1$, and let $M' = O(k^3 dp^2)$. (k, M) -NW-CLIQUE can be deterministically reduced to $O(k^d)$ instances of (k, M') -EW-CLIQUE in time $O(k^d \cdot n^2 \cdot \text{poly log } M)$.*

Thinking of $p + d$ as “small”, but $\text{poly}(p, d) \approx kM$ 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 $w : V \rightarrow [0, M]$ and a target number $t \in [0, kM]$. Recall the mappings $f_i : [0, M] \times [0, kM] \rightarrow [-kp, kp]^d$ for $i \in [s]$ from Lemma 3.2, 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 $w_i : E \rightarrow [-M', M']$. For $(u, v) \in E$, let $\mathbf{u} = f_i(w(u), t)$ and $\mathbf{v} = f_i(w(v), t)$, and define

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

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 $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{Z}^d$,

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

Consider a set $S = \{u_1, \dots, u_k\} \subseteq V$ 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

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

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 3.2, 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 3.5 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. The proof of the following lemma is given in the full version.

Lemma 3.6. *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.*

3.3 k -SUM is in $\mathbf{W}[1]$

Using the above lemmas, we can efficiently reduce k -SUM to k -Clique. Consider a k -SUM instance (S, t) where $S = \{x_1, \dots, x_n\} \subseteq [0, M]$ and $t \in [0, kM]$ with $M = n^{2k}$. Let $G = (V, E)$ be a node-weighted clique on n nodes $V = \{v_1, \dots, v_n\}$ with weight function $w : V \rightarrow [0, M]$ 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 3.5 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 3.6, 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 3.7. *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 3.8. *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 ??), 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 $O(2^{n/4})$ [27]. Corollary 3.8 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 $g(n, k) \leq n^{o(1)} \cdot 2^{f(k) \cdot \text{poly}(k)}$. If $k \geq \lceil \log \log n \rceil$, then since $n \leq 2^{2^k}$, we have that $g(n, k) \leq 2^{2^k + f(k) \cdot \text{poly}(k)}$. Therefore, $g(n, k) \leq n^{o(1)} \cdot h(k)$ for some computable $h : \mathbb{N} \rightarrow \mathbb{N}$, and we have shown the following:

Lemma 3.9. *k -SUM is in $\mathbf{W}[1]$.*

In the full version, 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 1.3 (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 1.6 referencing k -Clique. We defer the proof of the second half of Theorem 1.6 concerning k -Dominating Set to the full version.

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 [12]. The reduction is given from k -Clique to k -Vector-Sum (recall Definition 3.1), and then from k -Vector-Sum to k -SUM.

The proof of the following lemma is given in the full version.

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 $O(n^2)$.*

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). We give the proof in the full version.

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.*

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 $[-n^{\sqrt{k'}}, n^{\sqrt{k'}}]$, 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 1.3.

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

1. Amir Abboud and Kevin Lewi. Exact Weight Subgraphs and the k-Sum Conjecture. In *ICALP (1)*, pages 1–12, 2013.
2. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. *CoRR*, abs/1402.0054, 2014.
3. Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.
4. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
5. Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.
6. Gill Barequet and Sariel 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.
7. Arnab Bhattacharyya, Piotr Indyk, David P. Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In *ICS*, pages 496–508, 2011.
8. Jonathan F. Buss and Tarique Islam. Algorithms in the W-Hierarchy. *Theory Comput. Syst.*, 41(3):445–457, 2007.
9. David Cattanéo and Simon Perdrix. The parameterized complexity of domination-type problems and application to linear codes. *CoRR*, abs/1209.5267, 2012.
10. Marco Cesati. Perfect Code is $W[1]$ -complete. *Inf. Process. Lett.*, 81(3):163–168, 2002.
11. Artur Czumaj and Andrzej Lingas. Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication. *SIAM J. Comput.*, 39(2):431–444, 2009.
12. Rodney G. Downey and Michael R. Fellows. Fixed-parameter intractability. In *Structure in Complexity Theory Conference*, pages 36–49, 1992.
13. Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995.
14. Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004.
15. Jeff Erickson. Lower bounds for linear satisfiability problems. In *SODA*, pages 388–395, 1995.

16. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer Verlag, 2006.
17. Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165 – 185, 1995.
18. A. Grönlund and S. Pettie. Threesomes, Degenerates, and Love Triangles. *CoRR*, abs/1404.0799, 2014.
19. Antonio Hernández-Barrera. Finding an $O(n^2 \log n)$ Algorithm Is Sometimes Hard. In *CCCG*, pages 289–294, 1996.
20. Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *STOC '77*, pages 1–10, New York, NY, USA, 1977. ACM.
21. Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, Triangles. *CoRR*, abs/1305.3827, 2013.
22. Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. Reducing a target interval to a few exact queries. In *MFCS*, pages 718–727, 2012.
23. Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.
24. Kevin O’Bryant. Sets of integers that do not contain long arithmetic progressions. *Electr. J. Comb.*, 18(1), 2011.
25. Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610, 2010.
26. Mihai Pătraşcu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In *SODA*, pages 1065–1075, 2010.
27. John M Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical report, 1251-01, LaBRI, Université de Bordeaux I, 2001.
28. Virginia Vassilevska and Ryan Williams. Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications. In *STOC*, pages 225–231, 2006.
29. Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *STOC*, pages 455–464, 2009.
30. Virginia Vassilevska, Ryan Williams, and Raphael Yuster. Finding the smallest h -subgraph in real weighted graphs and related problems. In *ICALP (1)*, pages 262–273, 2006.