SETH-Based Lower Bounds for Subset Sum and Bicriteria Path*

Amir Abboud† Karl Bringmann‡ Danny Hermelin§ Dvir Shabtay¶

Abstract

Subset Sum and k-SAT are two of the most extensively studied problems in computer science, and conjectures about their hardness are among the cornerstones of fine-grained complexity. An important open problem in this area is to base the hardness of one of these problems on the other.

Our main result is a tight reduction from k-SAT to Subset Sum on dense instances, proving that Bellman's 1962 pseudo-polynomial $O^*(T)$-time algorithm for Subset Sum on n numbers and target T cannot be improved to time $T^{1-\varepsilon} \cdot 2^{o(n)}$ for any $\varepsilon > 0$, unless the Strong Exponential Time Hypothesis (SETH) fails.

As a corollary, we prove a "Direct-OR" theorem for Subset Sum under SETH, offering a new tool for proving conditional lower bounds: It is now possible to assume that deciding whether one out of N given instances of Subset Sum is a YES instance requires time $(NT)^{1-o(1)}$. As an application of this corollary, we prove a tight SETH-based lower bound for the classical Bicriteria s,t-Path problem, which is extensively studied in Operations Research. We separate its complexity from that of Subset Sum: On graphs with m edges and edge lengths bounded by L, we show that the $O(Lm)$ pseudo-polynomial time algorithm by Joksch from 1966 cannot be improved to $\tilde{O}(L+m)$, in contrast to a recent improvement for Subset Sum (Bringmann, SODA 2017).

*We would like to thank Jesper Nederlof for an inspiring discussion on Subset Sum. A.A. was supported by the grants of Virginia Vassilevska Williams: NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, and BSF Grant BSF:2012338. D.H. has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, and by the ISRAEL SCIENCE FOUNDATION (grant No. 551145/).

†IBM Almaden Research Center

‡Max Planck Institute for Informatics, Saarland Informatics Campus, Germany.

§Department of Industrial Engineering and Management, Ben-Gurion University, Israel

¶Department of Industrial Engineering and Management, Ben-Gurion University, Israel

1 Introduction

The field of fine-grained complexity is anchored around certain hypotheses about the exact time complexity of a small set of core problems. Due to dozens of reductions, we now know that the current algorithms for many important problems are optimal unless breakthrough algorithms for the core problems exist. A central challenge in this field is to understand the connections and relative difficulties among these core problems. In this work, we discover a new connection between two core problems: a tight reduction from k-SAT to Subset Sum.

In the first part of the introduction we discuss this new reduction and how it affects the landscape of fine-grained complexity. Then, in Section 1.2, we highlight a corollary of this reduction which gives a new tool for proving conditional lower bounds. As an application, in Section 1.3, we prove the first tight bounds for the classical Bicriteria s,t-Path problem from Operations Research.

Subset Sum. Subset Sum is one of the most fundamental problems in computer science. Its most basic form is the following: given n integers $x_1, \ldots, x_n \in \mathbb{N}$, and a target value $T \in \mathbb{N}$, decide whether there is a subset of the numbers that sums to T. The two most classical algorithms for the problem are the pseudo-polynomial $O(Tn)$ algorithm using dynamic programming [28], and the $O(2^{n/2} \cdot \mathrm{poly}(n, \log T))$ algorithm via "meet-in-the-middle" [67]. A central open question in Exact Algorithms [114] is whether faster algorithms exist, *e.g.*, can we combine the two approaches to get a $T^{1/2} \cdot n^{O(1)}$ time algorithm? Such a bound was recently found in a Merlin-Arthur setting [94].

Open Question 1. *Is* Subset Sum *in time* $T^{1-\varepsilon} \cdot 2^{o(n)}$ *or* $2^{(1-\varepsilon)\frac{n}{2}} \cdot T^{o(1)}$, *for some* $\varepsilon > 0$*?*

The status of Subset Sum as a major problem has been established due to many applications, deep connections to other fields, and educational value. The $O(Tn)$ algorithm from 1957 is an illuminating example of dynamic programming that is taught in most undergraduate algorithms courses, and the NP-hardness proof (from Karp's original paper [78]) is a prominent example of a reduction to a problem on numbers. Interestingly, one of the earliest cryptosystems by Merkle and Hell-

man was based on Subset Sum [92], and was later extended to a host of *Knapsack-type* cryptosystems[1] (see [106, 31, 97, 46, 69] and the references therein).

The version of Subset Sum where we ask for k numbers that sum to zero (the k-SUM problem) is conjectured to have $n^{\lceil k/2 \rceil \pm o(1)}$ time complexity. Most famously, the $k = 3$ case is the 3-SUM conjecture highlighted in the seminal work of Gajentaan and Overmars [57]. It has been shown that this problem lies at the core and captures the difficulty of dozens of problems in computational geometry. Searching in Google Scholar for "3sum-hard" reveals more than 250 papers (see [80] for a highly partial list). More recently, these conjectures have become even more prominent as core problems in fine-grained complexity since their interesting consequences have expanded beyond geometry into purely combinatorial problems [102, 113, 38, 72, 12, 6, 83, 7, 62, 72]. Note that k-SUM inherits its hardness from Subset Sum, by a simple reduction: to answer Open Question 1 positively it is enough to solve k-SUM in $T^{1-\varepsilon} \cdot n^{o(k)}$ or $n^{k/2-\varepsilon} \cdot T^{o(1)}$ time.

Entire books [91, 79] are dedicated to the algorithmic approaches that have been used to attack Subset Sum throughout many decades, and, quite astonishingly, major algorithmic advances are still being discovered in our days, *e.g.*, [88, 68, 26, 51, 14, 108, 77, 61, 44, 15, 16, 85, 56, 22, 94, 82, 33], not to mention the recent developments on generalized versions (see [24]) and other computational models (see [107, 41]). At STOC'17 an algorithm was presented that beats the trivial 2^n bound while using polynomial space, under certain assumptions on access to random bits [22]. At SODA'17 we have seen the first improvements (beyond log factors [100]) over the $O(Tn)$ algorithm, reducing the bound to $\tilde{O}(T+n)$ [82, 33]. And a few years earlier, a surprising result celebrated by cryptographers [68, 26] showed that $2^{0.499}$ algorithms are possible on random instances. All this progress leads to the feeling that a positive resolution to Open Question 1 might be just around the corner.

SETH. k-SAT is an equally fundamental problem (if not more) but of a Boolean rather than additive nature, where we are given a k-CNF formula on n variables and m clauses, and the task is to decide whether it is satisfiable. All known algorithms have a running time of the form $O(2^{(1-c/k)n})$ for some constant $c > 0$ [99, 49, 8], and the Strong Exponential Time Hypothesis (SETH) of Impagliazzo and Paturi [70, 71, 39] states that no $O(2^{(1-\varepsilon)n})$ time algorithms are possible for k-SAT, for some $\varepsilon > 0$ independent of k. Refuting SETH implies advances in circuit complexity [73], and is known to be impossible with popular techniques like resolution [25].

A seminal paper of Cygan, Dell, Lokshtanov, Marx, Nederlof, Okamoto, Paturi, Saurabh, and Wahlström [48] strives to classify the exact complexity of important NP-hard problems under SETH. The authors design a large collection of ingenious reductions and conclude that $2^{(1-\varepsilon)n}$ algorithms for problems like Hitting Set, Set Splitting, and Not-All-Equal SAT are impossible under SETH. Notably, Subset Sum is not in this list nor any problem for which the known algorithms are non-trivial (*e.g.*, require dynamic programming). As the authors point out: "*Of course, we would also like to show tight connections between SETH and the optimal growth rates of problems that* **do** *have non-trivial exact algorithms.*"

Since the work of Cygan et al. [48], SETH has enjoyed great success as a basis for lower bounds in Parameterized Complexity [87] and for problems within P [112]. Some of the most fundamental problems on strings (*e.g.*, [9, 17, 2, 35, 18, 34]), graphs (*e.g.*, [86, 104, 6, 58]), curves (*e.g.*, [32]), vectors [110, 111, 19, 29] and trees [1] have been shown to be *SETH-hard*: a small improvement to the running time of these problems would refute SETH. Despite the remarkable quantity and diversity of these results, we are yet to see a (tight) reduction from SAT to any problem like Subset Sum, where the complexity comes from the hardness of analyzing a search space defined by addition of numbers. In fact, all hardness results for problems of a more *number theoretic* or *additive combinatoric* flavor are based on the conjectured hardness of Subset Sum itself.

In this paper, we address an important open questions in the field of fine-grained complexity: *Can we prove a tight SETH-based lower bound for* Subset Sum*?*

The standard NP-hardness proofs imply loose lower bounds under SETH (in fact, under the weaker ETH) stating that $2^{o(\sqrt{n})}$ algorithms are impossible. A stronger but still loose result rules out $2^{o(n)} \cdot T^{o(1)}$-time algorithms for Subset Sum under ETH [75, 37]. Before that, Patrascu and Williams [98] showed that if we solve k-SUM in $n^{o(k)} \cdot T^{o(1)}$ time, then ETH is false. These results leave the possibility of $O(T^{0.001})$ algorithms. While it is open whether such algorithms imply new SAT algorithms, it has been shown that they would imply new algorithms for other famous problems. Bringmann [33] recently observed that a an $O(T^{0.78})$ algorithm for Subset Sum implies a new algorithm for k-Clique, via a reduction of Abboud, Lewi, and Williams [5]. Cygan et al. [48] ruled out $O(T^{1-\varepsilon}\text{poly}(n))$ algo-

[1] Cryptographers usually refer to Subset Sum as Knapsack.

rithms for Subset Sum under the conjecture that the Set Cover problem on m sets over a universe of size n cannot be solved in $O(2^{(1-\varepsilon)n} \cdot \text{poly}(m))$ time. Whether this conjecture can be replaced by the more popular SETH remains a major open question.

1.1 Main Result We would like to show that SETH implies a negative resolution to Open Question 1. Our main result accomplishes half of this statement, showing a tight reduction from SAT to Subset Sum on instances where $T = 2^{\delta n}$, also known as *dense* instances[2], ruling out $T^{1-\varepsilon} \cdot 2^{o(n)}$ time algorithms under SETH.

Theorem 1.1. *Assuming SETH, for any $\varepsilon > 0$ there exists a $\delta > 0$ such that* Subset Sum *is not in time $O(T^{1-\varepsilon}2^{\delta n})$, and k-*Sum *is not in time $O(T^{1-\varepsilon}n^{\delta k})$.*

Thus, Subset Sum is yet another SETH-hard problem. This is certainly a major addition to this list. This also adds many other problems that have reductions from Subset Sum, *e.g.*, the famous Knapsack problem, or from k-SUM (*e.g.*, [54, 30, 4, 43, 81]). For some of these problems, to be discussed shortly, this even leads to better lower bounds.

Getting a reduction that also rules out $2^{(1-\varepsilon)n/2} \cdot T^{o(1)}$ algorithms under SETH is still a fascinating open question. Notably, the strongest possible reduction, ruling out $n^{\lceil k/2 \rceil - \varepsilon} \cdot T^{o(1)}$ algorithms for k-SUM, is provably impossible under the *Nondeterministic SETH* of Carmosino et al. [42], but there is no barrier for an $n^{k/2-o(1)}$ lower bound.

A substantial technical barrier that we had to overcome when designing our reduction is the fact that there was no clear understanding of what the hard instances of Subset Sum should look like. Significant effort has been put into finding and characterizing the instances of Subset Sum and Knapsack that are hard to solve. This is challenging both from an experimental viewpoint (see the study of Pisinger [101]) and from the worst-case analysis perspective (see the discussion of Austrin et al. [16]). Recent breakthroughs refute the common belief that random instances are maximally hard [68, 26], and show that better upper bounds are possible for various classes of inputs. Our reduction is able to generate hard instances by crucially relying on a deep result on the combinatorics of numbers: the existence of dense average-free sets. A surprising construction of these sets from 1946 due to Behrend [27] (see also [52, 96]) has already lead to breakthroughs in various areas of theoretical computer science [45, 47, 10, 65, 50, 3]. These are among the most non-random-like structures in combinatorics, and therefore allow our instances to bypass the easyness of random inputs. This leads us to a candidate distribution of hard instances for Subset Sum, which could be of independent interest: Start from hard instances of SAT (*e.g.*, random formulas around the threshold) and map them with our reduction (the obtained distribution over numbers will be highly structured).

Recently, it was shown that the security of certain cryptographic primitives can be based on SETH [20, 21]. We hope that our SETH-hardness for an already popular problem in cryptography will lead to further interaction between fine-grained complexity and cryptography. In particular, it would be exciting if our hard instances could be used for a new Knapsack-type cryptosystem. Such schemes tend to be much more computationally efficient than popular schemes like RSA [31, 97, 69], but almost all known ones are not secure (as famously shown by Shamir [106]). Even more recently, Bennett, Golovnev, and Stephens-Davidowitz [29] proved SETH hardness for another central problem from cryptography, the Closest-Vector-Problem (CVP). While CVP is a harder problem than Subset Sum, their hardness result addresses a different regime of parameters, and rules out $O(2^{(1-\varepsilon)n})$ time algorithms (when the *dimension* is large). It would be exciting to combine the two techniques and get a completely tight lower bound for CVP.

1.2 A Direct-OR Theorem for Subset Sum Some readers might find the above result unnecessary: What is the value in a SETH-based lower bound if we already believe the Set Cover Conjecture of Cygan et al.? The rest of this introduction discusses new lower bound results that, to our knowledge, would not have been possible without our new SETH-based lower bound. To clarify what we mean, consider the following "Direct-OR" version of Subset Sum: Given N different and independent instances of Subset Sum, each on n numbers and each with a different target $T_i \leq T$, decide whether any of them is a YES instance. It is natural to expect the time complexity of this problem to be $(NT)^{1-o(1)}$, but how do we formally argue that this is the case? If we could assume that this holds, it would be a very useful tool for conditional lower bounds (as we show in Section 1.3).

Many problems, like SAT, have a simple self-reduction proving that the "Direct-OR" version is hard, assuming the problem itself is hard: To solve a SAT instance on n variables, it is enough to solve 2^x instances on $n-x$ variables. This is typically the case for problems where the best known algorithm is essentially matched with a brute force algorithm. But what about Subset Sum or Set Cover? Can we use an algorithm that solves

[2] The density of an instance is the ratio $\frac{n}{\log_2 \max x_i}$.

N instances of Subset Sum in $O(N^{0.1} \cdot T)$ time to solve Subset Sum in $O(T^{1-\varepsilon})$ time? We cannot prove such statements; however, we can prove that such algorithms would refute SETH.

Corollary 1.1. *Assuming SETH, for any $\varepsilon > 0$ and $\gamma > 0$ there exists a $\delta > 0$ such that no algorithm can solve the OR of N given instances of* Subset Sum *on target values $T_1, \ldots, T_N = O(N^\gamma)$ and at most $\delta \log N$ numbers each, in total time $O(N^{1+\gamma-\varepsilon})$.*

1.3 The Fine-Grained Complexity of Bicriteria Path The Bicriteria s,t-Path problem is the natural bicriteria variant of the classical s,t-Path problem where edges have two types of weights and we seek an s,t-path which meets given demands on both criteria. More precisely, we are given a directed graph G where each edge $e \in E(G)$ is assigned a pair of non-negative integers $\ell(e)$ and $c(e)$, respectively denoting the *length* and *cost* of e, and two non-negative integers L and C representing our budgets. The goal is to determine whether there is an s,t-path $e_1, \ldots, e_k$ in G, between a given source and a target vertex $s,t \in V(G)$, such that $\sum_{i=1}^{k} \ell(e_i) \leq L$ and $\sum_{i=1}^{k} c(e_i) \leq C$.

This natural variant of s,t-Path has been extensively studied in the literature, by various research communities, and has many diverse applications in several areas. Most notable of these are perhaps the applications in the area of transportation networks [60], and the *quality of service (QoS) routing problem* studied in the context of communication networks [89, 115]. There are also several applications for Bicriteria s,t-Path in Operations Research domains, in particular in the area of scheduling [36, 84, 93, 105], and in column generation techniques [66, 116]. Additional applications can be found in road traffic management, navigation systems, freight transportation, supply chain management and pipeline distribution systems [60].

A simple reduction proves that Bicriteria s,t-Path is at least as hard as Subset Sum (see Garey and Johnson [59]). In 1966, Joksch [76] presented a dynamic programming algorithm with pseudo-polynomial running time $O(Lm)$ (or $O(Cm)$) on graphs with m edges. Extensions of this classical algorithm appeared in abundance since then, see *e.g.*, [13, 63, 103] and the various FPTASs for the optimization variant of the problem [53, 60, 64, 90, 109]. The reader is referred to the survey by Garroppo et al. [60] for further results on Bicriteria s,t-Path.

Our SETH-based lower bound for Subset Sum easily transfers to show that a $O(L^{1-\varepsilon}2^{o(n)})$ time algorithm for Bicriteria s,t-Path refutes SETH. However, after the $O(Tn)$ algorithm for Subset Sum from 1960 was improved last year to $\tilde{O}(T+n)$, it is natural to wonder if the similar $O(Lm)$ algorithm for Bicriteria s,t-Path from 1966 can also be improved to $\tilde{O}(L+m)$ or even just to $O(Lm^{0.99})$. Such an improvement would be very interesting since the pseudo-polynomial algorithm is commonly used in practice, and since it would speed up the running time of the approximation algorithms. We prove that Bicriteria s,t-Path is in fact a harder problem than Subset Sum, and an improved algorithm would refute SETH. The main application of Corollary 1.1 that we report in this paper is a tight SETH-based lower bound for Bicriteria s,t-Path, which (conditionally) separates the time complexity of Bicriteria s,t-Path and Subset Sum.

Theorem 1.2. *Assuming SETH, for any $\varepsilon > 0$ and $\gamma > 0$ no algorithm solves* Bicriteria s,t-Path *on sparse n-vertex graphs and budgets $L, C = \Theta(n^\gamma)$ in time $O(n^{1+\gamma-\varepsilon})$.*

Intuitively, our reduction shows how a single instance of Bicriteria s,t-Path can simulate multiple instances of Subset Sum and solve the "Direct-OR" version of it.

Our second application of Corollary 1.1 concerns the number of different edge-lengths and/or edge-costs in our given input graph. Let λ denote the former parameter, and χ denote the latter. Note that λ and χ are different from L and C, and each can be quite small in comparison to the size of the entire input. In fact, in many of the scheduling applications for Bicriteria s,t-Path discussed above it is natural to assume that one of these is quite small. We present a SETH-based lower bound that almost matches the $O(n^{\min\{\lambda,\chi\}+2})$ upper bound for the problem.

Theorem 1.3. Bicriteria s,t-Path *can be solved in $O(n^{\min\{\lambda,\chi\}+2})$ time. Moreover, assuming SETH, for any constants $\lambda, \chi \geq 2$ and $\varepsilon > 0$, there is no $O(n^{\min\{\lambda,\chi\}-1-\varepsilon})$ time algorithm for the problem.*

Finally, we consider the case where we are searching for a path that uses only k internal vertices. This parameter is naturally small in comparison to the total input length in several applications of Bicriteria s,t-Path, for example the packet routing application discussed above. We show that this problem is equivalent to the k-Sum problem, up to logarithmic factors. For this, we consider an intermediate exact variant of Bicriteria s,t-Path, the Zero-Weight-k-Path problem, and utilize the known bounds for this variant to obtain the first improvement over the $O(n^k)$-time brute-force algorithm, as well as a matching lower bound.

Theorem 1.4. Bicriteria s,t-Path *can be solved in $\tilde{O}(n^{\lceil (k+1)/2 \rceil})$ time. Moreover, for any $\varepsilon > 0$, there is no*

$\tilde{O}(n^{\lceil (k+1)/2 \rceil - \varepsilon})$*-time algorithm for the problem, unless* k-SUM *has an* $\tilde{O}(n^{\lceil k/2 \rceil - \varepsilon})$*-time algorithm.*

2 Preliminaries

For a fixed integer p, we let $[p]$ denote the set of integers $\{1, \ldots, p\}$. All graphs in this paper are, unless otherwise stated, simple, directed, and without self-loops. We use standard graph theoretic notation, *e.g.*, for a graph G we let $V(G)$ and $E(G)$ denote the set of vertices and edges of G, respectively. Throughout the paper, we use the $O^*(\cdot)$ and $\tilde{O}(\cdot)$ notations to suppress polynomial and logarithmic factors.

Hardness Assumptions: The Exponential Time Hypothesis (ETH) and its strong variant (SETH) are conjectures about running time of any algorithm for the k-SAT problem: Given a boolean CNF formula ϕ, where each clause has at most k literals, determine whether ϕ has a satisfying assignment. Let $s_k = \inf\{\delta : k\text{-SAT can be solved in } O^*(2^{\delta n}) \text{ time}\}$. The Exponential Time Hypothesis, as stated by Impagliazzo, Paturi and Zane [71], is the conjecture that $s_3 > 0$. It is known that $s_3 > 0$ if and only if there is a $k \geq 3$ such that $s_k > 0$ [71], and that if ETH is true, the sequence $\{s_k\}_{k=1}^{\infty}$ increases infinitely often [70]. The Strong Exponential Time Hypothesis, coined by Impagliazzo and Paturi [40, 70], is the conjecture that $\lim_{k\to\infty} s_k = 1$. In our terms, this can be stated in the following more convenient manner:

CONJECTURE 2.1. *For any* $\varepsilon > 0$ *there exists* $k \geq 3$ *such that* k-SAT *on* n *variables cannot be solved in time* $O(2^{(1-\varepsilon)n})$.

We use the following standard tool by Impagliazzo, Paturi and Zane:

LEMMA 2.1. (SPARSIFICATION LEMMA [71]) *For any* $\varepsilon > 0$ *and* $k \geq 3$*, there exists* $c_{k,\varepsilon} > 0$ *and an algorithm that, given a* k-SAT *instance* ϕ *on* n *variables, computes* k-SAT *instances* $\phi_1, \ldots, \phi_\ell$ *with* $\ell \leq 2^{\varepsilon n}$ *such that* ϕ *is satisfiable if and only if at least one* ϕ_i *is satisfiable. Moreover, each* ϕ_i *has* n *variables, each variable in* ϕ_i *appears in at most* $c_{k,\varepsilon}$ *clauses, and the algorithm runs in time* $\text{poly}(n)2^{\varepsilon n}$.

The k-SUM Problem: In k-SUM we are given sets $Z_1, \ldots, Z_k$ of non-negative integers and a target T, and we want to decide whether there are $z_1 \in Z_1, \ldots, z_k \in Z_k$ such that $z_1 + \ldots + z_k = T$. This problem can be solved in time $O(n^{\lceil k/2 \rceil})$ [67], and it is somewhat standard by now to assume that this is essentially the best possible [4]. This assumption, which generalizes the more popular assumption of the $k = 3$ case [57, 102], remains believable despite recent algorithmic progress [14, 23, 44, 77, 108].

CONJECTURE 2.2. k-SUM *cannot be solved in time* $\tilde{O}(n^{\lceil k/2 \rceil - \varepsilon})$ *for any* $\varepsilon > 0$ *and* $k \geq 3$.

3 From SAT to Subset Sum

In this section we present our main result, the hardness of SUBSET SUM and k-SUM under SETH. Our reduction goes through three main steps: We start with a k-SAT formula ϕ that is the input to our reduction. This formula is then reduced to subexponentially many Constraint Satisfaction Problems (CSP) with a restricted structure. The main technical part is then to reduce these CSP instances to equivalent SUBSET SUM instances. The last part of our construction, reducing SUBSET SUM to k-SUM, is rather standard. In the final part of the section we provide a proof for Corollary 1.1, showing that SUBSET SUM admits the "Direct-OR" property discussed in Section 1.2.

3.1 From k-SAT to Structured CSP

We first present a reduction from k-SAT to certain structured instances of Constraint Satisfaction Problems (CSP). This is a standard combination of the Sparsification Lemma with well-known tricks.

LEMMA 3.1. *Given a* k-SAT *instance* ϕ *on* n *variables and* m *clauses, for any* $\varepsilon > 0$ *and* $a \geq 1$ *we can compute in time* $\text{poly}(n)2^{\varepsilon n}$ *CSP instances* $\psi_1, \ldots, \psi_\ell$*, with* $\ell \leq 2^{\varepsilon n}$*, such that* ϕ *is satisfiable if and only if some* ψ_i *is satisfiable. Each* ψ_i *has* $\hat{n} = \lceil n/a \rceil$ *variables over universe* $[2^a]$ *and* $\hat{m} = \lceil n/a \rceil$ *constraints. Each variable is contained in at most* $\hat{c}_{k,\varepsilon} \cdot a$ *constraints, and each constraint contains at most* $\hat{c}_{k,\varepsilon} \cdot a$ *variables, for some constant* $\hat{c}_{k,\varepsilon}$ *depending only on* k *and* ε.

Proof. Let ϕ be an instance of k-SAT with n variables and m clauses. We start by invoking the Sparsification Lemma (Lemma 2.1). This yields k-SAT instances $\phi_1, \ldots, \phi_\ell$ with $\ell \leq 2^{\varepsilon n}$ such that ϕ is satisfiable if and only if some ϕ_i is satisfiable, and where each ϕ_i has n variables, and each variable in ϕ_i appears in at most $c_{k,\varepsilon}$ clauses of ϕ_i, for some constant $c_{k,\varepsilon}$. In particular, the number of clauses is at most $c_{k,\varepsilon} n$.

We combine multiple variables to a super-variable and multiple clauses to a super-constraint, which yields a certain structured CSP. Specifically, let $a \geq 1$, and partition the variables into $\lceil n/a \rceil$ blocks of length a. We replace each block of a variables by one super-variable over universe $[2^a]$. Similarly, we partition the clauses into $\lceil n/a \rceil$ blocks, each containing $\gamma := ac_{k,\varepsilon}$ clauses. We replace each block of γ clauses $C_1, \ldots, C_\gamma$ by one super-constraint C that depends on all super-variables containing variables appearing in $C_1, \ldots, C_\gamma$.

Clearly, the resulting CSP ψ_i is equivalent to ϕ_i. Since each variable appears in at most $c_{k,\varepsilon}$ clauses in ϕ_i,

and we combine a variables to obtain a variable of ψ_i, each variable appears in ψ_i in at most $ac_{k,\varepsilon}$ constraints. Similarly, each clause in ϕ_i contains at most k variables, and each super-constraint consists of $\gamma = ac_{k,\varepsilon}$ clauses, so each super-constraint contains at most $\hat{c}_{k,\varepsilon}a$ variables for $\hat{c}_{k,\varepsilon} = kc_{k,\varepsilon}$. This finishes the proof.

3.2 From Structured CSP to Subset Sum Next we reduce to Subset Sum. Specifically, we show the following.

Theorem 3.1. *For any $\varepsilon > 0$, given a k-SAT instance ϕ on n variables we can in time $\mathrm{poly}(n)2^{\varepsilon n}$ construct $2^{\varepsilon n}$ instances of Subset Sum on at most $\tilde{c}_{k,\varepsilon}n$ items and a target value bounded by $2^{(1+2\varepsilon)n}$ such that ϕ is satisfiable iff at least one of the Subset Sum instances is a YES-instance. Here $\tilde{c}_{k,\varepsilon}$ is a constant depending only on k and ε.*

As discussed in Section 1.1, our reduction crucially relies on a construction of average-free sets. For any $k \geq 2$, a set S of integers is *k-average-free* iff for all $k' \leq k$ and (not necessarily distinct) $x_1, \ldots, x_{k'+1} \in S$ with $x_1 + \ldots + x_{k'} = k' \cdot x_{k'+1}$ we have $x_1 = \ldots = x_{k'+1}$. A surprising construction by Behrend [27] has been slightly adapted in [5], showing the following.

Lemma 3.2. *There exists a universal constant $c > 0$ such that, given $\varepsilon \in (0,1)$, $k \geq 2$, and $n \geq 1$, a k-average-free set S of size n with $S \subset [0, k^{c/\varepsilon}n^{1+\varepsilon}]$ can be constructed in $\mathrm{poly}(n)$ time.*

While it seems natural to use this lemma when working with an additive problem like Subset Sum, we are only aware of very few uses of this result in conditional lower bounds [5, 55, 74]. One example is a reduction from k-Clique to k^2-SUM on numbers in $n^{k+o(1)}$ [5]. Our result can be viewed as a significant boosting of this reduction, where we exploit the power of Subset Sum further. Morally, k-Clique is like MAX-2-SAT, since faster algorithms for k-Clique imply faster algorithms for MAX-2-SAT [110]. We show that even MAX-d-SAT, for any d, can be reduced to k-SUM, which corresponds to a reduction from Clique on *hypergraphs* to k-SUM.

Proof. [of Theorem 3.1] We let $a \geq 1$ be a sufficiently large constant depending only on k and ε. We need a λ-average-free set, with $\lambda := \hat{c}_{k,\varepsilon}a$, where $\hat{c}_{k,\varepsilon}$ is the constant from Lemma 3.1. Lemma 3.2 yields a λ-average-free set S of size 2^a consisting of non-negative integers bounded by $B := \lambda^{c/\varepsilon}(2^a)^{1+\varepsilon}$, for some constant c depending only on ε. We let $f\colon [2^a] \to S$ be any injective function. Note that since a and B are constants constructing f takes constant time.

Run Lemma 3.1 to obtain CSP instances $\psi_1, \ldots, \psi_\ell$ with $\ell \leq 2^{\varepsilon n}$, each with $\hat{n} = \lceil n/a \rceil$ variables over universe $[2^a]$ and $\hat{m} = \hat{n}$ constraints, such that each variable is contained in at most λ constraints and each constraint contains at most λ variables. Fix a CSP $\psi = \psi_i$. We create an instance (Z, T) of Subset Sum, i.e., a set Z of positive integers and a target value T. We define these integers by describing blocks of their bits, from highest to lowest. (The items in Z are naturally partitioned, as for each variable x of ψ there will be $O_{k,\varepsilon}(1)$ items *of type x*, and for each clause C of ψ there will be $O_{k,\varepsilon}(1)$ items *of type C*.)

We first ensure that any correct solution picks exactly one item of each type. To this end, we start with a block of $O(\log \hat{n})$ bits where each item has value 1, and the target value is $\hat{n} + \hat{m}$, which ensures that we pick exactly $\hat{n} + \hat{m}$ items. This is followed by $O(\log \hat{n})$ many 0-bits to avoid overflow from the lower bits (we will have $O_{k,\varepsilon}(\hat{n})$ items overall). In the following $\hat{n} + \hat{m}$ bits, each position is associated to one type, and each item of that type has a 1 at this position and 0s at all other positions. The target T has all these bits set to 1. Together, these $O(\log \hat{n}) + \hat{n} + \hat{m}$ bits ensure that we pick exactly one item of each type. We again add $O(\log \hat{n})$ many 0-bits to avoid overflow from the lower bits.

The remaining $\hat{n}$ blocks of bits correspond to the variables of ψ. For each variable we have a block consisting of $\lceil \log(2\lambda B + 1) \rceil = \log B + \log \lambda + O(1)$ bits. The target number T has bits forming the number λB in each block of each variable.

Now we describe the items of type x, where x is a variable. For each assignment $\alpha \in [2^a]$ of x, there is an item $z(x, \alpha)$ of type x. In the block corresponding to variable x, the bits of $z(x, \alpha)$ form the number $\lambda B - d(x) \cdot f(\alpha)$, where $d(x)$ is the number of clauses containing x. In all blocks corresponding to other variables, the bits of $z(x, \alpha)$ are 0.

Next we describe the items of type C, where C is a constraint. Let $x_1, \ldots, x_s$ be the variables that are contained in C. For any assignment $\alpha_1, \ldots, \alpha_s \in [2^a]$ of $x_1, \ldots, x_s$ that satisfies the clause C, there is an item $z(C, \alpha_1, \ldots, \alpha_s)$ of type C. In the block corresponding to variable x_i the bits of $z(C, \alpha_1, \ldots, \alpha_s)$ form the number $f(\alpha_i)$, for any $1 \leq i \leq s$. In all blocks corresponding to other variables, the bits are 0.

Example: Suppose $a = 1$ and $\hat{c}_{k,\varepsilon} = 2$, and consider a CSP with variables x_1, x_2, x_3 over the universe $[2^a] = \{1, 2\}$, and constraints $C_1 = (x_1 = x_2)$, $C_2 = (x_2 \neq x_3)$, and $C_3 = (x_1 = 1 \Rightarrow x_3 = 1)$. Note that $\lambda = 2$. We construct the 2-average-free set $S = \{1, 2\}$; in particular, we may set $B = 2$, and use the injective mapping $f : [2^a] \to S$ defined by $f(x) = x$.

The following items correspond to the CSP variables (the |-symbols mark block boundaries and have no other meaning):

$$z(x_1,1) = 1|000|100000|000|0010|0000|0000|$$

$$z(x_1,2) = 1|000|100000|000|0000|0000|0000|$$

$$z(x_2,1) = 1|000|010000|000|0000|0010|0000|$$

$$z(x_2,2) = 1|000|010000|000|0000|0000|0000|$$

$$z(x_3,1) = 1|000|001000|000|0000|0000|0010|$$

$$z(x_3,2) = 1|000|001000|000|0000|0000|0000|$$

And the following items correspond to the constraints:

$$z(C_1,1,1) = 1|000|000100|000|0001|0001|0000|$$

$$z(C_1,2,2) = 1|000|000100|000|0010|0010|0000|$$

$$z(C_2,1,2) = 1|000|000010|000|0000|0001|0010|$$

$$z(C_2,2,1) = 1|000|000010|000|0000|0010|0001|$$

$$z(C_3,1,1) = 1|000|000001|000|0001|0000|0001|$$

$$z(C_3,2,1) = 1|000|000001|000|0010|0000|0001|$$

$$z(C_3,2,2) = 1|000|000001|000|0010|0000|0010|$$

We set the target to

$$T = 110|000|111111|000|0100|0100|0100|.$$

One can readily verify that T sums up to $z(x_1,2) + z(x_2,2) + z(x_2,1) + z(C_1,2,2) + z(C_2,2,1) + z(C_3,2,1)$, and that no other subset sums up to T. That is, the subsets summing to T are in one-to-one correspondence to the satisfying assignments of the CSP.

Correctness: Recall that the first $O(\log \hat{n}) + \hat{n} + \hat{m}$ bits ensure that we pick exactly one item of each type. Consider any variable x and the corresponding block of bits. The item of type x picks an assignment α, resulting in the number $\lambda B - d(x) \cdot f(\alpha)$, where $d(x)$ is the degree of x. The $d(x)$ constraints containing x pick assignments $\alpha_1, \ldots, \alpha_{d(x)}$ and contribute $f(\alpha_1) + \ldots + f(\alpha_{d(x)})$. Hence, the total contribution in the block is

$$f(\alpha_1) + \ldots + f(\alpha_{d(x)}) - d(x) \cdot f(\alpha) + \lambda B,$$

where $d(x) \leq \lambda$. Since f maps to a λ-average-free set, we can only obtain the target λB if $f(\alpha_1) = \ldots = f(\alpha_{d(x)}) = f(\alpha)$. Since f is injective, this shows that any correct solution picks a coherent assignment α for variable x. Finally, this coherent choice of assignments for all variables satisfies all clauses, since clause items only exist for assignments satisfying the clause. Hence, we obtain an equivalent Subset Sum instance.

Note that the length of blocks corresponding to variables is set so that there are no carries between blocks, which is necessary for the above argument. Indeed, the degree $d(x)$ of any variable x is at most λ, so the clauses containing x can contribute at most $\lambda \cdot B$ to its block, while the item of type x also contributes $0 \leq \lambda B - d(x) \cdot f(\alpha) \leq \lambda B$, which gives a number in $[0, 2\lambda B]$.

Size Bounds: Let us count the number of bits in the constructed numbers. We have $O(\log \hat{n}) + \hat{n} + \hat{m}$ bits from the first part ensuring that we pick one item of each type, and $\hat{n} \cdot (\log B + \log \lambda + O(1))$ bits from the second part ensuring to pick coherent and satisfying assignments. Plugging in $B = \lambda^{c/\varepsilon}(2^a)^{1+\varepsilon}$ and $\lambda = \hat{c}_{k,\varepsilon} a$ and using $\hat{n} = \hat{m} = \lceil n/a \rceil$ yields a total number of bits of

$$\begin{aligned}\log T &= O(\log \hat{n}) + \hat{n} + \hat{m} + \hat{n} \cdot (\log B + \log \lambda + O(1)) \\ &= (1+\varepsilon)n + O_{k,\varepsilon}(n \log(a)/a),\end{aligned}$$

where the hidden constant depends only on k and ε. Since $\log(a)/a$ tends to 0 for $a \to \infty$, we can choose a sufficiently large, depending on k and ε, to obtain $\log T \leq (1+\varepsilon)n + \varepsilon n \leq (1+2\varepsilon)n$.

Let us also count the number of constructed items. We have one item for each variable x and each assignment $\alpha \in [2^a]$, amounting to $2^a \hat{n} \leq 2^a n$ items. Moreover, we have one item for each clause C and all assignments $\alpha_1, \ldots, \alpha_s \in [2^a]$ that jointly satisfy the clause C, where $s \leq \lambda$ is the number of variables contained in C. This amounts to up to $2^{a\lambda}\hat{m} \leq 2^{a\lambda} n \leq 2^{\hat{c}_{k,\varepsilon} a^2} n$ items. Note that both factors only depend on k and ε, since a only depends on k and ε. Thus, the number of items is bounded by $\tilde{c}_{k,\varepsilon} n$, where $\tilde{c}_{k,\varepsilon}$ only depends on k and ε.

In total, we obtain a reduction that maps an instance ϕ of k-SAT on n variables to $2^{\varepsilon n}$ instances of Subset Sum with target at most $2^{(1+2\varepsilon)n}$ on at most $\tilde{c}_{k,\varepsilon} n$ items. The running time of the reduction is clearly $\mathrm{poly}(n) 2^{\varepsilon n}$.

Our main result (Theorem 1.1) now follows.

Proof. [of Theorem 1.1] *Subset Sum:* For any $\varepsilon > 0$ set $\varepsilon' := \varepsilon/5$ and let k be sufficiently large so that k-SAT has no $O(2^{(1-\varepsilon')n})$ algorithm; this exists assuming SETH. Set $\delta := \varepsilon'/\tilde{c}_{k,\varepsilon'}$, where $\tilde{c}_{k,\varepsilon'}$ is the constant from Theorem 3.1. Now assume that Subset Sum can be solved in time $O(T^{1-\varepsilon} 2^{\delta n})$. We show that this contradicts SETH. Let ϕ be a k-SAT instance on n variables, and run Theorem 3.1 with ε' to obtain $2^{\varepsilon' n}$ Subset Sum instances on at most $\tilde{c}_{k,\varepsilon'} n$ items and target at most $2^{(1+2\varepsilon')n}$. Using the assumed $O(T^{1-\varepsilon} 2^{\delta n})$ algorithm on each Subset Sum instance, yields a total

time for k-SAT of

$$\begin{aligned}&\text{poly}(n)2^{\varepsilon' n} + 2^{\varepsilon' n} \cdot \left(2^{(1+2\varepsilon')n}\right)^{1-\varepsilon} 2^{\delta \cdot \tilde{c}_{k,\varepsilon'} n} \\ &= 2^{(\varepsilon' + (1+2\varepsilon')(1-5\varepsilon') + \varepsilon')n} \leq 2^{(1-\varepsilon')n},\end{aligned}$$

where we used the definitions of ε' and δ as well as $(1+2\varepsilon')(1-5\varepsilon') \leq 1-3\varepsilon'$. This running time contradicts SETH, yielding the lower bound for Subset Sum.

k-SUM: The lower bound $O(T^{1-\varepsilon}n^{\delta k})$ for k-Sum now follows easily from the lower bound for Subset Sum. Consider a Subset Sum instance (Z, T) on $|Z| = n$ items and target T. Partition Z into sets $Z_1, \ldots, Z_k$ of of equal size, up to ± 1. For each set Z_i, enumerate all subset sums S_i of Z_i, ignoring the subsets summing to larger than T. Consider the k-Sum instance $(S_1, \ldots, S_k, T)$, where the task is to pick items $s_i \in S_i$ with $s_1 + \ldots + s_k = T$. Since $|S_i| \leq O(2^{n/k})$, an $O(T^{1-\varepsilon}n^{\delta k})$ time algorithm for k-Sum now implies an $O(T^{1-\varepsilon}2^{\delta n})$ algorithm for Subset Sum, thus contradicting SETH.

3.3 Direct-OR Theorem for Subset Sum We now provide a proof for Corollary 1.1. We show that deciding whether at least one of N given instances of Subset Sum is a YES-instance requires time $(NT)^{1-o(1)}$, where T is a common upper bound on the target. Here we crucially use our reduction from k-SAT to Subset Sum, since the former has an easy self-reduction allowing us to tightly reduce one instance to multiple subinstances, while such a self-reduction is not known for Subset Sum.

Proof. [of Corollary 1.1] Let $\varepsilon > 0$ and $\gamma > 0$, we will fix $\delta > 0$ later. Assume that the OR of N given instances of Subset Sum on target values $T_1, \ldots, T_N = O(N^\gamma)$ and at most $\delta \log N$ numbers each, can be solved in total time $O(N^{(1+\gamma)(1-\varepsilon)})$. We will show that SETH fails.

Let ϕ be an instance of k-SAT on n variables. Split the set of variables into X_1 and X_2 of size n_1 and n_2, such that $n_2 = \gamma \cdot n_1$ up to rounding. Specifically, we can set $n_1 := \lceil \frac{n}{1+\gamma} \rceil$ and $n_2 := \lfloor \frac{\gamma n}{1+\gamma} \rfloor$ and thus have $n_2 \leq \gamma n_1$. Enumerate all assignments of the variables in X_1. For each such assignment α let ϕ_α be the resulting k-SAT instance after applying the partial assignment α.

For each ϕ_α, run the reduction from Theorem 3.1 with $\varepsilon' = \min\{1/2, 1/\gamma\} \cdot \varepsilon/2$, resulting in at most $2^{\varepsilon' n_2}$ instances of Subset Sum on at most $\tilde{c}_{k,\varepsilon'} n_2$ items and target at most $2^{(1+2\varepsilon')n_2}$. In total, we obtain at most $2^{n_1 + \varepsilon' n_2}$ instances of Subset Sum, and ϕ is satisfiable iff at least one of these Subset Sum instances is a YES-instance. Set $N := 2^{(1+\varepsilon/2)n_1}$ and note that the number of instances is at most $2^{n_1+\varepsilon' n_2} \leq 2^{(1+\gamma\varepsilon')n_1} \leq N$, and that the target bound is at most $2^{(1+2\varepsilon')n_2} \leq 2^{(1+2\varepsilon')\gamma n_1} \leq N^\gamma$. Thus, we constructed at most N instances of Subset Sum on target at most N^γ, each having at most $\tilde{c}_{k,\varepsilon'} n_2 \leq \tilde{c}_{k,\varepsilon'} n$ items.

Using the assumed algorithm, the OR of these instances can be solved in total time $O(N^{(1+\gamma)(1-\varepsilon)})$. Since $(1+\gamma)n_1 = (1+\gamma)\lceil \frac{n}{1+\gamma} \rceil \leq n + 1 + \gamma = n + O(1)$ and $(1+\varepsilon/2)(1-\varepsilon) \leq 1 - \varepsilon/2$, this running time is

$$\begin{aligned}O\left(N^{(1+\gamma)(1-\varepsilon)}\right) &= O\left(\left(2^{(1+\varepsilon/2)n_1}\right)^{(1+\gamma)(1-\varepsilon)}\right) \\ &= O\left(2^{(1-\varepsilon/2)n}\right),\end{aligned}$$

which contradicts SETH. Specifically, for some $k = k(\varepsilon)$ this running time is less than the time required for k-SAT. Setting $\delta := \tilde{c}_{k,\varepsilon'}$ finishes the proof.

4 The Bicriteria s, t-Path Problem

In this section we apply the results of the previous section to the Bicriteria s,t-Path problem. We will show that the Bicriteria s,t-Path problem is in fact harder than Subset Sum, by proving that the classical pseudo-polynomial time algorithm for the problem cannot be improved on sparse graphs assuming SETH. We also prove Theorem 1.3 concerning a bounded number of different edge-lengths λ and edge-costs χ in the input network, and Theorem 1.4 concerning a bounded number k of internal vertices in a solution path.

4.1 Sparse networks We begin with the case of sparse networks; *i.e.* input graphs on n vertices and $O(n)$ edges. We embed multiple instances of Subset Sum into one instance of Bicriteria s,t-Path to prove Theorem 1.2, namely that there is no algorithm for Bicriteria s,t-Path on sparse graphs faster than the well-known $O(\min\{nL, nC\})$-time algorithm.

Proof. [of Theorem 1.2] We show that for any $\varepsilon > 0$, $\gamma > 0$, an algorithm solving Bicriteria s,t-Path on sparse n-vertex graphs and budgets $L, C = \Theta(n^\gamma)$ in time $O(n^{(1+\gamma)(1-\varepsilon)})$ contradicts SETH. As in Corollary 1.1, let $(Z_1, T_1), \ldots, (Z_N, T_N)$ be instances of Subset Sum on targets $T_i \leq N^\gamma$ and number of items $|Z_i| \leq \delta \log N$ for all i. Without loss of generality, we can assume that all sets Z_i have the same size $k = \delta \log N$ (*e.g.*, by making Z_i a multiset containing the number 0 multiple times).

Fix an instance (Z_i, T_i) and let $Z_i = \{z_1, \ldots, z_k\}$. We construct a graph G_i with vertices $s, v_1, \ldots, v_k, t$. Writing $v_0 := s$ for simplicity, for each $j \in [k]$ we add an edge from v_{j-1} to v_j with length z_j and cost $N^\gamma - z_j$, and we add another[3] edge from v_{j-1} to v_j with length 0 and cost N^γ. Finally, we add an edge

[3]Note that parallel edges can be avoided by subdividing all constructed edges.

from v_k to t with length $N^\gamma - T_i$ and cost T_i. Then the set of s,t-paths corresponds to the power set of Z_i, and the s,t-path corresponding to $Y \subseteq Z_i$ has total length $N^\gamma - T_i + \sum_{y\in Y} y$ and cost $kN^\gamma + T_i - \sum_{y\in Y} y$. Hence, setting the upper bound on the length to $L = N^\gamma$ and on the cost to $C = kN^\gamma$, there is an s,t-path respecting these bounds iff there is a subset Y of Z_i summing to T_i, i.e., iff (Z_i, T_i) is a YES-instance.

We combine the graphs $G_1, \ldots, G_N$ into one graph G by identifying all source vertices s, identifying all target vertices t, and then taking the disjoint union of the remainder. With the common length bound $L = N^\gamma$ and cost bound $C = kN^\gamma$, there is an s,t-path respecting these bounds in G iff some instance (Z_i, T_i) is a YES-instance. Furthermore, note that G has $n = \Theta(N \log N)$ vertices, is sparse, and can be constructed in time $O(N \log N)$. Hence, an $O(n^{(1+\gamma)(1-\varepsilon)})$ time algorithm for Bicriteria s,t-Path would imply an $O(N^{(1+\gamma)(1-\varepsilon)}\text{polylog}N) = O(N^{(1+\gamma)(1-\varepsilon/2)})$ time algorithm for deciding whether at least one of N Subset Sum instances is a YES-instance, a contradiction to SETH by Corollary 1.1.

Finally, let us ensure that $L, C = \Theta(n^\gamma)$. Note that the budgets L and C are both bounded by $O(N^\gamma \log N)$. If $\gamma \geq 1$, then add a supersource s' and one edge from s' to s with length and cost equal to $N^\gamma \log^\gamma N$, and add $N^\gamma \log^\gamma N$ to L and C. This results in an equivalent instance, and the new bounds L, C are $\Theta(N^\gamma \log^\gamma N) = \Theta(n^\gamma)$. If $\gamma < 1$, then do the same where the length and cost from s' to s is $N^\gamma \log N$, and then add $N \log^{1/\gamma} N$ dummy vertices to the graph to increase n to $\Theta(N \log^{1/\gamma} N)$. Again we obtain budgets $L, C = \Theta(N^\gamma \log N) = \Theta((N \log^{1/\gamma} N)^\gamma) = \Theta(n^\gamma)$. In both cases, the same running time analysis as in the last paragraph goes through. This completes the proof of Theorem 1.2.

4.2 Few different edge-lengths or edge-costs

We next consider the parameters λ (the number of different edge-lengths) and χ (the number of different edge-costs). We show that Bicriteria s,t-Path can be solved in $O(n^{\min\{\lambda,\chi\}+2})$ time, while its unlikely to be solvable in $O(n^{\min\{\lambda,\chi\}-1-\varepsilon})$ for any $\varepsilon > 0$, providing a complete proof for Theorem 1.3. The upper bound of this theorem is quite easy, and is given in the following lemma.

Lemma 4.1. Bicriteria s,t-Path *can be solved in* $O(n^{\min\{\lambda,\chi\}+2})$ *time.*

Proof. It suffices to give an $O(n^{\lambda+2})$ time algorithm, as the case of time $O(n^{\chi+2})$ is symmetric, and a combination of these two algorithms yields the claim. Let $\tilde{\ell}_1, \ldots, \tilde{\ell}_\lambda$ be all different edge-length values. We compute a table $T[v, i_1, \ldots, i_\lambda]$, where $v \in V(G)$ and $i_1, \ldots, i_\lambda \in \{0, \ldots, n\}$, which stores the minimum cost of any s,v-path that has exactly i_j edges of length $\tilde{\ell}_j$, for each $j \in \{1, \ldots, \lambda\}$. For the base case of our computation, we set $T[s, 0, \ldots, 0] = 0$ and $T[s, i_1, \ldots, i_\lambda] = \infty$ for entries with some $i_j \neq 0$. The remaining entries are computed via the following recursion:

$$T[v, i_1, \ldots, i_\lambda] = \min_{1\leq j\leq \lambda} \min_{\substack{(u,v)\in E(G),\\ \ell((u,v))=\tilde{\ell}_j.}} T[u, i_1, \ldots, i_j - 1, \ldots, i_\lambda] + c((u,v)).$$

It is easy to see that the above recursion is correct, since if $e_1, \ldots, e_k$ is an optimal s,v-path corresponding to an entry $T[v, i_1, \ldots, i_\lambda]$ in T, with $e_k = (u,v)$ and $\ell(e_k) = \tilde{\ell}_j$ for some $j \in \{1, \ldots, \lambda\}$, then $e_1, \ldots, e_{k-1}$ is an optimal s,u-path corresponding to the entry $T[u, i_1, \ldots, i_j - 1, \ldots, i_\lambda]$. Thus, after computing table T, we can determine whether there is a feasible s,t-path in G by checking whether there is an entry $T[t, i_1, \ldots, i_\lambda]$ with $\sum_{j=1}^{\lambda} i_j \cdot \tilde{\ell}_j \leq L$ and $T[t, i_1, \ldots, i_\lambda] \leq C$. As there are $O(n^{\lambda+1})$ entries in T in total, and each entry can be computed in $O(n)$ time, the entire algorithm requires $O(n^{\lambda+2})$ time.

We now turn to proving the lower-bound given in Theorem 1.3. The starting point is our lower bound for k-Sum ruling out $O(T^{1-\varepsilon}n^{\delta k})$ algorithms (Theorem 1.1). We present a reduction from k-Sum to Bicriteria s,t-Path, where the resulting graph in the Bicriteria s,t-Path instance has few different edge-lengths and edge-costs.

Let $(Z_1, \ldots, Z_k, T)$ be an instance of k-Sum with $Z_i \subset [0, T]$ and $|Z_i| \leq n$ for all i, and we want to decide whether there are $z_1 \in Z_1, \ldots, z_k \in Z_k$ with $z_1 + \ldots + z_k = T$. We begin by constructing an acyclic multigraph G^*, using similar ideas to those used for proving Theorem 1.2. The multigraph G^* has $k+1$ vertices $s = v_0, \ldots, v_k = t$, and is constructed as follows: For each $i \in \{1, \ldots, k\}$, we add at most n edges from v_{i-1} to v_i, one for each element in Z_i. The length of an edge $e \in E(G^*)$ corresponding to element $z_i \in Z_i$ is set to $\ell(e) = z_i$, and its cost is set to $c(e) = T - z_i$.

Lemma 4.2. $(Z_1, \ldots, Z_k, T)$ *has a solution iff* G^* *has an* s,t*-path of length at most* $L = T$ *and cost at most* $C = T(k-1)$.

Proof. Suppose there are $z_1 \in Z_1, \ldots, z_k \in Z_k$ that sum to T. Consider the s,t-path $e_1, \ldots, e_k$ in G^*, where e_i is the edge from v_{i-1} to v_i corresponding to z_i. Then $\sum_{i=1}^{k} \ell(e_i) = \sum_{i=1}^{k} z_i = T = L$, and

$\sum_{i=1}^{k} c(e_i) = \sum_{i=1}^{k} T - z_i = kT - T = C$. Conversely, any s,t-path in G^* has k edges $e_1, \ldots, e_k$, where e_i is an edge from v_{i-1} to v_i. If such a path is feasible, meaning that $\sum_{i=1}^{k} \ell(e_i) \leq L = T$ and $\sum_{i=1}^{k} c(e_i) \leq C = T(k-1)$, then these two inequalities must be tight because $c(e_i) = T - \ell(e_i)$ for each $i \in [k]$. This implies that the integers $z_1, \ldots, z_k$ corresponding to the edges $e_1, \ldots, e_k$ of G^*, sum to T.

Let $\tau \geq 1$ be any constant and let $B := \lceil T^{1/\tau} \rceil$. We next convert G^* into a graph $\tilde{G}$ which has $\tau + 1$ different edge-lengths and $\tau + 1$ different edge-costs, both taken from the set $\{0, B^0, B^1, \ldots, B^{\tau-1}\}$. Recall that $V(G^*) = \{v_0, \ldots, v_k\}$, and the length and cost of each edge in G^* is non-negative and bounded by T. The vertex set of $\tilde{G}$ will include all vertices of G^*, as well as additional vertices.

For an edge $e \in E(G^*)$, write its length as $\ell(e) = \sum_{i=0}^{\tau-1} a_i B^i$, and its cost as $c(e) = \sum_{i=0}^{\tau-1} b_i B^i$, for integers $a_1, \ldots, a_{\tau-1}, b_1, \ldots, b_{\tau-1} \in \{0, \ldots, B-1\}$. We replace the edge e of G^* with a path in $\tilde{G}$ between the endpoints of e that has $\sum_{i=0}^{\tau-1}(a_i + b_i)$ internal vertices. For each $i \in \{0, \ldots, \tau-1\}$, we set a_i edges in this path to have length B^i and cost 0, and b_i edges to have length 0 and cost B^i. Replacing all edges of G^* by paths in this way, we obtain the graph $\tilde{G}$ which has $O(nB)$ vertices and edges (since k and τ are constant). As any edge in G^* between v_i and v_{i+1} corresponds to a path between these two vertices in $\tilde{G}$ with the same length and cost, we have:

LEMMA 4.3. *Any s,t-path in G^* corresponds to an s,t-path in $\tilde{G}$ with same length and cost, and vice-versa.*

LEMMA 4.4. *Assuming SETH, for any constant $\lambda, \chi \geq 2$ there is no $O(n^{\min\{\lambda,\chi\}-1-\varepsilon})$ algorithm for* BICRITERIA *s,t-*PATH *for any $\varepsilon > 0$.*

Proof. Suppose BICRITERIA s,t-PATH has a $O(n^{\min\{\lambda,\chi\}-1-\varepsilon})$ time algorithm. We use this algorithm to obtain a fast algorithm for k-SUM, contradicting SETH by Theorem 1.1. On a given input $(Z_1, \ldots, Z_k, T)$ of k-SUM on n items, for $\tau := \min\{\lambda, \chi\} - 1$ we construct the instance $(\tilde{G}, s, t, L, C)$ described above. Then $\tilde{G}$ is a directed acyclic graph with $\tau + 1 = \min\{\lambda, \chi\}$ different edge-lengths and edge-costs $\{0, B^0, B^1, \ldots, B^{\tau-1}\}$. Moreover, due to Lemmas 4.2 and 4.3, there are $z_1 \in Z_1, \ldots, z_k \in Z_k$ summing to T iff $\tilde{G}$ has a feasible s,t-path. Thus, we can use our assumed BICRITERIA s,t-PATH algorithm on $(\tilde{G}, s, t, L, C)$ to solve the given k-SUM instance. As $\tilde{G}$ has $O(nB)$ vertices and edges, where $B = \lceil T^{1/\tau} \rceil$, an $O(n^{\min\{\lambda,\chi\}-1-\varepsilon})$ algorithm runs in time $O((nB)^{\tau-\varepsilon}) = O(T^{1-\varepsilon/\tau} n^{\tau})$ time on $(\tilde{G}, s, t, L, C)$. For $\delta := \delta(\varepsilon/\tau)$ from Theorem 1.1 and k set to τ/δ, this running time is $O(T^{1-\varepsilon/\tau} n^{\delta(\varepsilon/\tau)k})$ and thus contradicts SETH by Theorem 1.1.

4.3 Solution paths with few vertices In this section we investigate the complexity of BICRITERIA s,t-PATH with respect to the number of internal vertices k in a solution path. Assuming k is fixed and bounded, we obtain a tight classification of the time complexity for the problem, up to sub-polynomial factors, under Conjecture 2.2.

Our starting point is the EXACT k-PATH problem: Given an integer $T \in \{0, \ldots, W\}$, and a directed graph G with edge weights, decide whether there is a simple path in G on k vertices in which the sum of the weights is exactly T. Thus, this is the "exact" variant of BICRITERIA s,t-PATH on graphs with a single edge criteria, and no source and target vertices. The EXACT k-PATH problem can be solved in $\tilde{O}(n^{\lceil (k+1)/2 \rceil})$ time by a "meet-in-the-middle" algorithm [4], where the $\tilde{O}(\cdot)$ notation suppresses poly-logarithmic factors in W. It is also known that EXACT k-PATH has no $\tilde{O}(n^{\lceil (k+1)/2 \rceil - \varepsilon})$ time algorithm, for any $\varepsilon > 0$, unless the k-SUM conjecture is false [4]. We will show how to obtain similar bounds for BICRITERIA s,t-PATH by implementing a very efficient reduction between the two problems.

To show that EXACT k-PATH can be used to solve BICRITERIA s,t-PATH, we will combine multiple ideas. The first is the observation that EXACT k-PATH can easily solve the EXACT BICRITERIA k-PATH problem, a variant which involves bicriteria edge weights: Given a pair of integers (T_1, T_2), and a directed graph G with two edge weight function $w_1(\cdot)$ and $w_2(\cdot)$, decide whether there is a simple path in G on k vertices in which the sum of the w_i-weights is exactly T_i for $i \in \{1, 2\}$.

LEMMA 4.5. *There is an $O(n^2)$ time reduction that reduces an instance of* EXACT BICRITERIA k-PATH *with edge weights in $\{0, 1, \ldots, W\}^2$ to an instance of* EXACT k-PATH *with edge weights in $\{0, 1, \ldots, 2kW^2 + W\}$.*

Proof. Define a mapping of a pairs in $\{0, 1, \ldots, W\}^2$ to single integers $\{0, \ldots, 2kW^2 + W\}$ by setting $f(w_1, w_2) = w_2 + w_1 \cdot 2kW$ for each $w_1, w_2 \in \{0, \ldots, W\}$. Observe that for any k pairs $(w_1^1, w_2^1), \ldots, (w_1^k, w_2^k)$, we have $(\sum_{i=1}^{k} w_1^i = T_1 \wedge \sum_{i=1}^{k} w_2^i = T_2)$ iff $\sum_{i=1}^{k} f(w_1^i, w_2^i) = f(T_1, T_2)$. Therefore, given a graph as in the statement, we can map each pair of edge weights into a single edge weight, thus reducing to EXACT k-PATH without changing the answer.

The next and more difficult step is to reduce BICRI-

teria s,t-Path to Exact Bicriteria k-Path. This requires us to reduce the question of whether there is a path of length and cost at most L and C, to questions about the existence of paths with length and cost *equalling exactly* T_1 and T_2. A naive approach would be to check if there is a path of exact length and cost (T_1, T_2) for all values $T_1 \leq L$ and $T_2 \leq C$. Such a reduction will incur a very large $O(LC)$ overhead. We will improve this to $O(\log^{O(1)}(L+C))$.

In the remainder of this section, let W be the maximum of L and C. The idea behind our reduction is to look for the smallest $x, y \in [\log W]$ such that if we restrict all edge lengths ℓ to the x most significant bits of ℓ, and all edge costs c to the y most significant bits of c, then there is a path that satisfies the threshold constraints with equality. To do this, we can check for every pair x, y, whether after restricting edge lengths and costs, there is a k-path with total weight *exactly* equal to the restriction of the vector (L, C), possibly minus the carry from the removed bits. Since the carry from summing k numbers can be at most k, we will not have to check more than $O(k^2)$ "targets" per pair $x, y \in [\log W]$.

To implement this formally, we will need the following technical lemma. The proof uses a bit scaling technique that is common in approximation algorithms. Previously, tight reductions that use this technique were presented by Vassilevska and Williams [113] (in a very specific setting), and by Nederlof *et al.* [95] (who proved a general statement). We will need a generalization of the result of [95] in which we introduce a parameter k, and show that the overhead depends only on k and W, and does not depend on n.

Lemma 4.6. *Let U be a universe of size n with two weight functions $w_1, w_2 : U \to \{0, \ldots, W\}$, and let $T_1, T_2 \in \{0, \ldots, W\}$ be two integers. Then, there is a polynomial time algorithm that returns a set of weight functions $w_1^{(i)}, w_2^{(i)} : U \to \{0, \ldots, W\}$ and integers $T_1^{(i)}, T_2^{(i)} \in \{0, \ldots, W\}$, for $i \in [q]$ and $q = O(k^2 \log^2 W)$, such that: For every subset $X \subseteq U$ of size $|X| = k$ it holds that $(w_1(X) \leq T_1 \wedge w_2(X) \leq T_2)$ iff $(w_1^{(i)}(X) = T_1^{(i)} \wedge w_2^{(i)}(X) = T_2^{(i)})$ for some $i \in [q]$.*

Proof. We will assume that a number in $\{0, \ldots, W\}$ is encoded in binary with $\log W$ bits in the standard way. For numbers $a \in \{0, \ldots, W\}$ and $x \in [\log W]$ we let $[a]_x$ be the x-bit number that is obtained by taking the x most significant bits in a. Alternatively, $[a]_x = \lfloor a / 2^{\log W - x} \rfloor$. In what follows, we will construct weight functions and targets for each dimension independently and in a similar way. We will present the construction for the w_1's.

First, we add the weight functions $w_1^{(i)} = w_1$, with target $T_1^{(i)} = L - a$ for any $a \in [4k]$. Call these the initial (i)'s. Then, for any $x \in [\log W]$ and $a \in [2k]$, we add the weight function $w_1^{(i)}(e) = [w_1(e)]_x$, and set the target to $T_1^{(i)} = [L]_x - k - a$. This defines $O(k \log W)$ new functions and targets, and we will show below that for any subset $X \subseteq U$ we have that $w_1(X) \leq L$ iff for some i we have $w_1^{(i)}(X) = T_1^{(i)}$. Then, we apply the same construction for w_2, and take every pair of constructed functions and targets, to obtain a set of $O(k^2 \log^2 W)$ functions and targets that satisfy the required property.

The correctness will be based on the following bound, which follows because when summing k numbers the carry from removed least significant bits cannot be more than k. For any $x \in [\log W], a_1, \ldots, a_k \in \{0, \ldots, W\}$,

$$\left[\sum_{j=1}^{k} a_j\right]_x - k \leq \sum_{j=1}^{k} [a_j]_x \leq \left[\sum_{j=1}^{k} a_j\right]_x$$

Fix some $X \subset U$. For the first direction, assume that for some i, $w_1^{(i)}(X) = T_1^{(i)}$. If it is one of the initial (i)'s, then we immediately have that $w_1(X) \leq L$. Otherwise, let $X = \{v_1, \ldots, v_k\}$, then we get that

$$\begin{aligned}[w_1(X)]_x = \left[\sum_{j=1}^{k} w_1(v_j)\right]_x &\leq \sum_{j=1}^{k} [w_1(v_j)]_x + k \\ &= w_1^{(i)}(X) + k \\ &= T_1^{(i)} + k \leq [L]_x - 1\end{aligned}$$

which implies that $w_1(X) < L$.

For the other direction, assume that $w_1(X) < L$. If $w_1(X) \geq L - 4k$ then for one of the initial (i)'s we will have $w_1^{(i)}(X) = w_1(X) = L - a = T_1^{(i)}$ for some $a \in [4k]$. Otherwise, let x be the smallest integer in $[\log W]$ for which $[w_1(X)]_x \leq [L]_x - k$. Because x is the smallest, we also know that $[w_1(X)]_x \geq [L]_x - 2k$. Therefore,

$$w_1^{(i)}(X) = \sum_{j=1}^{k} [w_1(v_j)]_x \leq \left[\sum_{j=1}^{k} w_1(v_j)\right]_x \leq [L]_x - k$$

and

$$w_1^{(i)}(X) = \sum_{j=1}^{k} [w_1(v_j)]_x \geq \left[\sum_{j=1}^{k} w_1(v_j)\right]_x - k \geq [L]_x - 3k.$$

Therefore, for some $a \in [2k]$, we will have that $w_1^{(i)}(X) = [L]_x - k - a = T_1^{(i)}$.

We are now ready to present the main reduction of this section. Let (G, s, t, L, C) be a given instance of Bicriteria s,t-Path. Our reduction follows three general steps that proceed as follows:

1. *Color coding:* At the first step, we use the derandomized version of the color coding technique [11] to obtain $p' = O(\lg n)$ partitions of the vertex set $V(G) \setminus \{s,t\}$ into k classes $V_1^{(\alpha)}, \ldots, V_k^{(\alpha)}$, $\alpha \in [p']$, with the following property: If there is a feasible s,t-path P with k internal vertices in G, we are guaranteed that for at least one partition we will have $|V(P) \cap V_i^{(\alpha)}| = 1$ for each $i \in [k]$. By trying out all possible $O(1)$ orderings of the classes in each partition, we can assume that if $P = s, v_1, \ldots, v_k, t$, then $V(P) \cap V_i^{(\alpha)} = \{v_i\}$ for each $i \in [k]$.

   Let p denote the total number of ordered partitions. For each ordered partition $\alpha \in [p]$, we remove all edges between vertices inside the same class, and all edges (u,v) where $u \in V_i^{(\alpha)}$, $v \in V_j^{(\alpha)}$, and $j \neq i+1$. We also remove all edges from s to vertices not in $V_1^{(\alpha)}$, and all edges to t from vertices not in V_k. Let G_α denote the resulting graph, with $\alpha \in [p]$ for $p = O(\lg n)$.

2. *Removal of s and t:* Next, we next remove s and t from each G_α. For every vertex $v \in V_1^{(\alpha)}$, if v was connected with an edge from s of length ℓ and cost c, then we remove this edge and add this length ℓ and cost c to all the edges outgoing from v. Similarly, we remove the edge from $v \in V_k^{(\alpha)}$ to t, and add its length and cost to all edges ingoing to v. Finally, any vertex in $V_1^{(\alpha)}$ that was not connected with an edge from s is removed from the graph, and every vertex in $V_k^{(\alpha)}$ that was not connected to t is removed.

3. *Inequality to equality reduction:* Now, for each G_α, we apply Lemma 4.6 with the universe U being the edges of G_α, and $w_1, w_2 : U \to \{0, \ldots, W\}$ being the lengths and costs of the edges. We get a set of $q = O(\log^2 W)$ weight functions and targets. For $\beta \in [q]$, let $G_{\alpha,\beta}$ be the graph obtained from G by replacing the lengths and costs with new functions $w_1^{(\beta)}, w_2^{(\beta)}$. The final Exact Bicriteria k-Path is then constructed as $(G_{\alpha,\beta}, T_1^{(\beta)}, T_2^{(\beta)})$.

Thus, we reduce our Bicriteria s,t-Path instance to at most $O(\log n \log^2 W)$ instances of Exact Bicriteria k-Path. Note that if G contains a feasible s,t-path $P = s, v_1, \ldots, v_k, t$ of length $\ell_P \leq L$ and cost c_P, then by correctness of the color coding technique, there is some $\alpha \in [p]$ such that G_α contains P with $V(P) \cap V_i^{(\alpha)} = \{v_i\}$ for each $i \in [k]$. Moreover, the total weight of $v_1, \ldots, v_k$ in G_α is (ℓ_P, c_P). By Lemma 4.6, there is some $\beta \in [q]$ for which the total weight of $v_1, \ldots, v_k$ in $G_{\alpha,\beta}$ is $(T_1^{(\beta)}, T_2^{(\beta)})$. Thus, P is a solution for $(G_{\alpha,\beta}, T_1^{(\beta)}, T_2^{(\beta)})$. Conversely, by the same line of arguments, any solution path for some Exact Bicriteria k-Path instance $(G_{\alpha,\beta}, T_1^{(\beta)}, T_2^{(\beta)})$ corresponds to a feasible s,t-path in G with k internal vertices.

Thus, we have obtained a reduction from Bicriteria s,t-Path to Exact Bicriteria k-Path. Combining this with reduction from Exact Bicriteria k-Path to Exact k-Path given in Lemma 4.5, we obtain the following.

Lemma 4.7. *Fix $k \geq 1$, and let (G, s, t, L, C) be an instance of* Bicriteria s,t-Path *where G has n vertices. Set $W = \max\{L, C\}$. Then one can determine whether (G, s, t, L, C) has a solution with k internal vertices by solving $O(\log n \log^2 W)$ instances of* Exact k-Path *on graphs with $O(n)$ vertices and edge weights bounded by W^2.*

Corollary 4.1. *There is an algorithm solving* Bicriteria s,t-Path *in $\tilde{O}(n^{\lceil (k+1)/2 \rceil})$ time.*

Proof. By Lemma 4.7, an instance of Bicriteria s,t-Path can be reduced to $O(\log n \log^2 W)$ instances of Exact k-Path. Using the algorithm in [4], each of these Exact k-Path instances can be solved in $\tilde{O}(n^{\lceil (k+1)/2 \rceil})$ time.

We next turn to proving our lower bound for Bicriteria s,t-Path. For this, we show a reduction in the other direction, from Exact k-Path to Bicriteria s,t-Path.

Lemma 4.8. *Let $\varepsilon > 0$. There is no $\tilde{O}(n^{\lceil (k+1)/2 \rceil - \varepsilon})$ time algorithm for* Bicriteria s,t-Path *unless the* k-Sum *conjecture (Conjecture 2.2) is false.*

Proof. We show a reduction from Exact k-Path to Bicriteria s,t-Path. This proves the claim, as it is known that an $\tilde{O}(n^{\lceil (k+1)/2 \rceil - \varepsilon})$ time algorithm for Exact k-Path, for any $\varepsilon > 0$, implies that the k-Sum conjecture is false [4]. Let (G, T) be an instance of Exact k-Path, where G is an edge-weighted graph and $T \in \{0, \ldots, W\}$ is the target. We proceed as follows: As in the upper-bound reduction, we first apply the color-coding technique [11] to obtain $p = O(\log n)$ vertex-partitioned graphs $G_1, \ldots, G_p$, $V(G_\alpha) = V_1^{(\alpha)} \uplus \cdots \uplus V_k^{(\alpha)}$ for each $\alpha \in [p]$, such that G has a solution path

$P = v_1, \ldots, v_k$ iff for at least one graph G_α we have $V(P) = V_i^{(\alpha)} \cap \{v_i\}$ for each $i \in [k]$.

We then construct a new graph H_α from each graph G_α as follows: We first remove from G_α all edges inside the same vertex class $V_i^{(\alpha)}$, and all edges between vertices in $V_i^{(\alpha)}$ and vertices in $V_j^{(\alpha)}$ with $j \neq i+1$. We then replace each remaining edge with weight $x \in \{0, \ldots, W\}$ in G_α with an edge with length x and cost $W - x$ in H_α. Then, we add vertices s, t to H_α, connect s to all the vertices in $V_1^{(\alpha)}$, connect all the vertices in $V_k^{(\alpha)}$ to t, and set the length and cost of all these edges to 0. To complete the proof, we argue that G has a simple path of weight exactly T iff some H_α contains a feasible s,t-path for $L = T$ and $C = (k-1)W - T$.

Suppose $P = v_1, \ldots, v_k$ is a simple path in G with $w(P) = T$. Then there is some $\alpha \in [p]$ such that P is a path in G_α with $V(P) = V_i^{(\alpha)} \cap \{v_i\}$ for each $i \in [k]$. By construction of H_α, $P' = s, v_1, \ldots, v_k, t$ is a path in H_α, and it has total length $\ell(P') = w(P) = T \leq L$, and total cost $c(P') = (k-1)W - w(P) = (k-1)W - T \leq C$. Conversely, if $P' = s, v_1, \ldots, v_k, t$ is a feasible s,t-path in some H_α with length $\ell(P') \leq L$ and cost $c(P') \leq C$, then $P = v_1, \ldots, v_k$ is path in G. We know that the weight of P in G is bounded by above by $w(P) = \ell(P') \leq L = T$. Furthermore, we have $(k-1)W - w(P) = (k-1)W - \ell(P') = c(P') \leq C = (k-1)W - T$, implying that $w(P) \geq T$. These two inequalities imply $w(P) = T$, and thus P is a solution for (G, T).

Thus, we can solve (G, T) by solving $O(\log n)$ instances of Bicriteria s,t-Path. This means that an $\tilde{O}(n^{\lceil (k+1)/2 \rceil - \varepsilon})$ algorithm for Bicriteria s,t-Path, for $\varepsilon > 0$, would imply an algorithm with the same running time for Exact k-Path. By the reductions in [4], this refutes the k-Sum conjecture.

Theorem 1.4 now immediately follows from the upper and lower bounds given in Corollary 4.1 and Lemma 4.8 for finding a solution for a Bicriteria s,t-Path instance that has k internal vertices.

References

[1] Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1256–1271, 2016.

[2] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 59–78, 2015.

[3] Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In *Proc. of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 351–361, 2016.

[4] Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-Sum conjecture. In *Proc. of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12, 2013.

[5] Amir Abboud, Kevin Lewi, and R. Ryan Williams. Losing weight by gaining edges. In *Proc. of the 22th annual European Symposium on Algorithms (ESA)*, pages 1–12, 2014.

[6] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014.

[7] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. of the 47th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 41–50, 2015.

[8] Amir Abboud, R. Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 218–230, 2015.

[9] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 39–51, 2014.

[10] Noga Alon, Michael Krivelevich, Eldar Fischer, and Mario Szegedy. Efficient testing of large graphs. In *Proc. of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 656–666, 1999.

[11] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

[12] Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *Proc. of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 114–125, 2014.

[13] Yash P. Aneja and Kunhiraman P.K. Nair. The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25:549–553, 1978.

[14] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. Space–time tradeoffs for Subset Sum: An improved worst case algorithm. In *Proc. of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 45–56, 2013.

[15] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Subset Sum in the absence of concentration. In *Proc. of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 48–61, 2015.

[16] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper

Nederlof. Dense Subset Sum may be the hardest. In *Proc. of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 13:1–13:14, 2016.

[17] Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false). In *Proc. of the 47th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.

[18] Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *Proc. of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 457–466, 2016.

[19] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. *arXiv preprint arXiv:1704.02958*, 2017.

[20] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), to appear*, 2017.

[21] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of useful work. *IACR Cryptology ePrint Archive*, 2017:203, 2017.

[22] Nikhil Bansal, Shashwat Garg, Jesper Nederlof, and Nikhil Vyas. Faster space-efficient algorithms for Subset Sum, k-Sum and related problems. In *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), to appear*, 2017.

[23] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

[24] Luis Barba, Jean Cardinal, John Iacono, Stefan Langerman, Aurélien Ooms, and Noam Solomon. Subquadratic algorithms for algebraic generalizations of 3SUM. In *Proc. of the 15th international Workshop on Algorithms and Data Structures (WADS)*, pages 97–108, 2017.

[25] Christopher Beck and Russell Impagliazzo. Strong ETH holds for regular resolution. In *Proc. of the 45th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 487–494, 2013.

[26] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Proc. of 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 364–385, 2011.

[27] Felix A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences of the United States of America*, 32(12):331–332, 1946.

[28] Richard E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[29] Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. On the quantitive hardness of CVP. *CoRR*, abs/1704.03928, 2017.

[30] Arnab Bhattacharyya, Piotr Indyk, David P. Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In *Proc. of the 1st ACM Conference on Innovations in Theoretical Computer Science (ICS)*, pages 496–508, 2011.

[31] Ernest F Brickell and Andrew M Odlyzko. Cryptanalysis: A survey of recent results. *Proceedings of the IEEE*, 76(5):578–593, 1988.

[32] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014.

[33] Karl Bringmann. A near-linear pseudopolynomial time algorithm for Subset Sum. In *Proc. of of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1073–1084, 2017.

[34] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 307–318. IEEE, 2017.

[35] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015.

[36] Dirk Briskorn, Byung-Cheon Choi, Kangbok Lee, Joseph Y.-T. Leung, and Michael Pinedo. Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, 207:605–619, 2010.

[37] Harry Buhrman, Bruno Loff, and Leen Torenvliet. Hardness of approximation for knapsack problems. *Theory Comput. Syst.*, 56(2):372–393, 2015.

[38] Ayelet Butman, Peter Clifford, Raphaël Clifford, Markus Jalsenius, Noa Lewenstein, Benny Porat, Ely Porat, and Benjamin Sach. Pattern matching under polynomial transformation. *SIAM Journal on Computing*, 42(2):611–633, 2013.

[39] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *Proc. of 21st Conference on Computational Complexity (CCC)*, pages 252–260, 2006.

[40] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Proc. of the 4th International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 75–85, 2009.

[41] Jean Cardinal, John Iacono, and Aurélien Ooms. Solving k-SUM using few linear queries. In *Proc. of the 24th Annual European Symposium on Algorithms (ESA)*, pages 25:1–25:17, 2016.

[42] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for nonreducibility. In *Proc. of the 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 261–270, 2016.

[43] David Cattanéo and Simon Perdrix. The parameterized complexity of domination-type problems and application to linear codes. In *Proc. of the 11th International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 86–103, 2014.

[44] Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. of the 47th annual ACM Symposium on Theory Of Computing (STOC)*, pages 31–40, 2015.

[45] Ashok K. Chandra, Merrick L. Furst, and Richard J. Lipton. Multi-party protocols. In *Proc. of the 15th annual ACM Symposium on Theory Of Computing (STOC)*, pages 94–99, 1983.

[46] Benny Chor and Ronald R. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988.

[47] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.

[48] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41, 2016.

[49] Evgeny Dantsin and Edward A. Hirsch. Worst-case upper bounds. In *Handbook of Satisfiability*, pages 403–424. 2009.

[50] Holger Dell and Dieter Van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proc. of the 42th annual ACM Symposium on Theory Of Computing (STOC)*, pages 251–260, 2010.

[51] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *Proc. of the 32nd Annual Conference on Advances in Cryptology (CRYPTO)*, pages 719–740, 2012.

[52] Michael Elkin. An improved construction of progression-free sets. In *Proc. of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 886–905, 2010.

[53] Funda Ergun, Rakesh Sinha, and Lisa Zhang. An improved FPTAS for the restricted shortest path problem. *Information Processing Letters*, 83:287–291, 2002.

[54] Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM Journal on Computing*, 28(4):1198–1214, 1999.

[55] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM J. Comput.*, 43(5):1541–1563, 2014.

[56] Ari Freund. Improved subquadratic 3SUM. *Algorithmica*, 77(2):440–458, 2017.

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

[58] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and R. Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2162–2181, 2017.

[59] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP Completeness*. W.H. Freeman & Co., 1976.

[60] Rosario G. Garroppo, Stefano Giordano, and Luca Tavanti. A survey on multi-constrained optimal path computation: Exact and approximate algorithms. *Computer Networks*, 54:3081–3107, 2010.

[61] Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. In *Proc. of the 25th annual European Symposium on Algorithms (ESA)*, pages 42:1–42:13, 2017.

[62] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In *Proc. of the 24th annual European Symposium on Algorithms (ESA)*, pages 45:1–45:16, 2016.

[63] Pierre Hansen. Bicriterion path problems. In *Proc. of the 3rd confernece om Multiple Criteria Decision Making Theory and Application*, pages 109–127, 1980.

[64] Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17:36–42, 1992.

[65] Johan Håstad and Avi Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Structures & Algorithms*, 22(2):139–160, 2003.

[66] Kaj Holmberg and Di Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1):42–57, 2003.

[67] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292, 1974.

[68] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *Proc. of 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 235–256, 2010.

[69] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of cryptology*, 9(4):199–216, 1996.

[70] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

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

[72] Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, triangles. *Algorithmica*, 74(1):326–343, 2016.

[73] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Proc. of the 42nd International Colloquium on Automata, Languages, and Programming*

(ICALP), pages 749–760, 2015.

[74] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013.

[75] Klaus Jansen, Felix Land, and Kati Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM J. Discrete Math.*, 30(1):343–366, 2016.

[76] Hans C. Joksch. The shortest route problem with constraints. *Journal of Mathematical Analysis and Applications*, 14:191–197, 1966.

[77] Allan Grønlund Jørgensen and Seth Pettie. Threesomes, degenerates, and love triangles. In *Proc. of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 621–630, 2014.

[78] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[79] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[80] James King. A survey of 3SUM-hard problems. 2004.

[81] Tomasz Kociumaka, Solon P. Pissis, and Jakub Radoszewski. Parameterizing PWM-and profile-matching and knapsack by the feasible-weight solutions count. *arXiv:1604.07581*, 2016.

[82] Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for Subset Sum. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1062–1072, 2017.

[83] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016.

[84] Kangbok Lee, Byung-Cheon Choi, Joseph Y.-T. Leung, and Michael L. Pinedo. Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. *Information Processing Letters*, 109:913–917, 2009.

[85] Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic time-space trade-offs for k-SUM. In *Proc. of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 58:1–58:14, 2016.

[86] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *Proc. of the 27th 2nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 777–789, 2011.

[87] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[88] Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In *Proc. of the 42nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 321–330, 2010.

[89] Dean H. Lorenz and Ariel Orda. QoS routing on networks with uncertain parameters. *IEEE/ACM Transactions on Networking*, 6:768–778, 1998.

[90] Dean H. Lorenz and Danny Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28:213–219, 2001.

[91] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[92] Ralph Merkle and Martin Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE transactions on Information Theory*, 24(5):525–530, 1978.

[93] Joseph Naor, Hadas Shachnai, and Tami Tamir. Real-time scheduling with a budget. *Algorithmica*, 47:343–364.

[94] Jesper Nederlof. A short note on Merlin-Arthur protocols for subset sum. *Information Processing Letters*, 118:15–16, 2017.

[95] Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. Reducing a target interval to a few exact queries. In *Proc. of the 37th international symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 718–727, 2012.

[96] Kevin O'Bryant. Sets of integers that do not contain long arithmetic progressions. *Electronic Journal of Combinatorics*, 18(1):P59, 2011.

[97] Andrew M. Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

[98] Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1065–1075. SIAM, 2010.

[99] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *Journal of the ACM*, 52(3):337–364, 2005.

[100] David Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003.

[101] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

[102] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. of the 42nd Annual ACM Symposium on Theory Of Computing (STOC)*, pages 603–610, 2010.

[103] Andrea Raith and Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & OR*, 36(4):1299–1331, 2009.

[104] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. of the 45th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 515–524, 2013.

[105] Dvir Shabtay, Kfir Arviv, Yael Edan, and Helman Stern. A combined robot selection and scheduling problem for flow-shops with no-wait restrictions. *Omega*, 43:96–107, 2014.

[106] Adi Shamir. A polynomial-time algorithm for break-

ing the basic Merkle-Hellman cryptosystem. *IEEE transactions on information theory*, 30(5):699–704, 1984.

[107] Emanuele Viola. The communication complexity of addition. *Combinatorica*, 35(6):703–747, 2015.

[108] Joshua R. Wang. Space-efficient randomized algorithms for k-SUM. In *Proc. of the 22th annual European Symposium on Algorithms (ESA)*, pages 810–829, 2014.

[109] Arthur Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[110] R. Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2–3):357–365, 2005.

[111] R. Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1867–1877, 2014.

[112] Virginia Vassilevska Williams. Hardness of easy problems: basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 43, 2015.

[113] Virginia Vassilevska Williams and R. Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. of the 41st Annual ACM Symposium on Theory Of Computing (STOC)*, pages 455–464, 2009.

[114] Gerhard J. Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405, 2008.

[115] O. Younis and S. Fahmy. Constraint-based routing in the internet: Basic principles and recent research. *IEEE Communications Surveys and Tutorials*, 5(1):2–13, 2003.

[116] U. Zimmermann and M.E. Lübbecke. Computer aided scheduling of switching engines. In Willi Jäger and Hans-Joachim Krebs, editors, *Mathematics — Key Technology for the Future: Joint Projects between Universities and Industry*, pages 690–702. Springer Berlin Heidelberg, 2003.