Fine-Grained Complexity for Sparse Graphs*

Udit Agarwal[†]

Dept. of Computer Science
University of Texas at Austin
Austin, TX
udit@cs.utexas.edu

Vijaya Ramachandran

Dept. of Computer Science
University of Texas at Austin
Austin, TX
vlr@cs.utexas.edu

ABSTRACT

We consider the fine-grained complexity of sparse graph problems that currently have $\tilde{O}(mn)$ time algorithms, where m is the number of edges and n is the number of vertices in the input graph. This class includes several important path problems on both directed and undirected graphs, including APSP, MWC (Minimum Weight Cycle), Radius, Eccentricities, BC (Betweenness Centrality), etc.

We introduce the notion of a sparse reduction which preserves the sparsity of graphs, and we present near linear-time sparse reductions between various pairs of graph problems in the $\tilde{O}(mn)$ class. There are many sub-cubic reductions between graph problems in the $\tilde{O}(mn)$ class, but surprisingly few of these preserve sparsity. In the directed case, our results give a partial order on a large collection of problems in the $\tilde{O}(mn)$ class (along with some equivalences), and many of our reductions are very nontrivial. In the undirected case we give two nontrivial sparse reductions: from MWC to APSP, and from unweighted ANSC (all nodes shortest cycles) to unweighted APSP. We develop a new ‘bit-sampling’ method for these sparse reductions on undirected graphs, which also gives rise to improved or simpler algorithms for cycle finding problems in undirected graphs.

We formulate the notion of *MWC hardness*, which is based on the assumption that a minimum weight cycle in a directed graph cannot be computed in time polynomially smaller than mn . Our sparse reductions for directed path problems in the $\tilde{O}(mn)$ class establish that several problems in this class, including 2-SiSP (second simple shortest path), s - t Replacement Paths, Radius, Eccentricities and BC are MWC hard. Our sparse reductions give MWC hardness a status for the $\tilde{O}(mn)$ class similar to 3SUM hardness for the quadratic class, since they show sub- mn hardness for a large collection of fundamental and well-studied graph problems that have maintained an $\tilde{O}(mn)$ time bound for over half a century. We also identify Eccentricities and BC as key problems in the $\tilde{O}(mn)$ class which are simultaneously MWC-hard, SETH-hard and k -DSH-hard, where SETH is the Strong Exponential Time Hypothesis, and

k -DSH is the hypothesis that a dominating set of size k cannot be computed in time polynomially smaller than n^k .

Our framework using sparse reductions is very relevant to real-world graphs, which tend to be sparse and for which the $\tilde{O}(mn)$ time algorithms are the ones typically used in practice, and not the $\tilde{O}(n^3)$ time algorithms.

CCS CONCEPTS

• **Theory of computation** → Shortest paths;

KEYWORDS

Graph Algorithms, Fine-Grained Reductions, Sparse Graphs, Minimum Weight Cycle

ACM Reference Format:

Udit Agarwal and Vijaya Ramachandran. 2018. Fine-Grained Complexity for Sparse Graphs. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC’18)*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3188745.3188888>

1 INTRODUCTION

In recent years there has been considerable interest in determining the fine-grained complexity of problems in P, see e.g. [40]. For instance, the 3SUM [14] and OV (Orthogonal Vectors) [11, 41] problems have been central to the fine-grained complexity of several problems with quadratic time algorithms, in computational geometry and related areas for 3SUM and in edit distance and related areas for OV. APSP (all pairs shortest paths) has been central to the fine-grained complexity of several path problems with cubic time algorithms on dense graphs [42].

For several graph problems related to shortest paths that currently have $\tilde{O}(n^3)$ ¹ time algorithms, *equivalence* under sub-cubic reductions has been shown in work starting with [42] between APSP, finding a minimum weight cycle (MWC), finding a second simple shortest path from a given vertex u to a given vertex v (2-SiSP) in a weighted directed graph, finding a minimum weight triangle (Min-Wt- Δ), etc. This gives compelling evidence that a large class of problems on dense graphs is unlikely to have sub-cubic algorithms as a function of n , the number of vertices, unless fundamentally new algorithmic techniques are developed.

In this paper we consider a central collection of graph problems related to APSP, which refines the subcubic equivalence class. We let n be the number of vertices and m the number of edges. All of the sub-cubic equivalent graph problems mentioned above (and several others) have $\tilde{O}(mn)$ time algorithms; additionally, many sub-cubic

*This work was supported in part by NSF Grant CCF-1320675.

[†]Research was also supported in part by a Calhoun Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

STOC’18, June 25–29, 2018, Los Angeles, CA, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5559-9/18/06...\$15.00

<https://doi.org/10.1145/3188745.3188888>

¹ $\tilde{O}$ and $\tilde{\Theta}$ can hide sub-polynomial factors; in our new results they only hide polylog factors.

equivalent problems related to minimum triangle detection and triangle listing have lower $O(m^{3/2})$ time complexities for sparse graphs [22]. When a graph is truly sparse with $m = O(n)$ the $\tilde{O}(mn)$ APSP bound is essentially optimal or very close to optimal, since the size of the output for APSP is n^2 . Thus, a cubic in n bound for APSP does not fully capture what is currently achievable by known algorithms, especially since graphs that arise in practice tend to have m close to linear in n or at least are *sparse*, i.e., have $m = O(n^{1+\delta})$ for $\delta < 1$. This motivates our study of the fine-grained complexity of graph path problems in the $\tilde{O}(mn)$ class.

Another fundamental problem in the $\tilde{O}(mn)$ class is MWC (Minimum Weight Cycle). In both directed and undirected graphs, MWC can be computed in $\tilde{O}(mn)$ time using an algorithm for APSP. Recently Orlin and Sedeno-Noda [29] gave an improved $O(mn)$ time algorithm for directed MWC. This is an important result but the bound still remains $\Omega(mn)$. Finding an MWC algorithm that runs polynomially faster than mn time is a long-standing open problem in graph algorithms.

In this paper we present both fine-grained reductions and hardness results for graph problems in the $\tilde{O}(mn)$ class, most of which are equivalent under sub-cubic reductions on dense graphs, but now taking sparseness of edges into consideration. We use the current long-standing upper bound of $\tilde{O}(mn)$ for these problems as our reference, both for our fine-grained reductions and for our hardness results. Our results give a partial order on hardness of several problems in the $\tilde{O}(mn)$ class, with equivalence within some subsets of problems, and gives rise to a hardness conjecture (MWC hardness) for this class. Most of the sub-cubic reductions in previous work (including all in [42]) created dense intermediate problems and hence are not fine-grained reductions for the $\tilde{O}(mn)$ class. Fine-grained reductions and hardness results with respect to time bounds that consider only n or only m , such as bounds of the form $m^2, n^2, n^c, m^{1+\delta}$, are given in [3–5, 20, 34]. One exception is in [2] where some reductions that preserve graph sparsity are given for problems with $\tilde{O}(mn)$ time algorithms, such as diameter and some betweenness centrality problems. However, these are either reductions that start from a triangle finding problem that can be solved in $\tilde{O}(m^{3/2})$, hence not fine-grained reductions for the $\tilde{O}(mn)$ class, or start from a problem, such as Diameter, that is not known to be sub-cubic equivalent to APSP. In contrast, we give sparse fine-grained results for the $\tilde{O}(mn)$ class that refine the sub-cubic equivalence class. Several other related results on fine-grained complexity are in [3–5, 10, 20, 23, 26, 30, 40, 43].

In very recent work [27], the authors formulated the Min-Weight- k -Clique hypothesis, which states that a minimum weight k -clique cannot be found in time smaller than $n^{k-o(1)}$ for sufficiently large edge weights. They show that for all sparsities of the form $m = \Theta(n^{1+1/l})$ where l is a constant, MWC cannot be computed in $O(mn^{1-\epsilon} + n^2)$ time under Min-Weight- k -Clique hypothesis. Our sparse reduction results for directed graphs in conjunction with their result show that a large class of problems in the mn class including Second Shortest Path, Replacement Paths, Radius, and Betweenness Centrality also do not have $O(mn^{1-\epsilon} + n^2)$ time algorithm under Min-Weight- k -Clique hypothesis. Similarly our sparse reduction result from MWC to APSP in undirected graphs in conjunction with this result also establishes that there is no $O(mn^{1-\epsilon} + n^2)$

time algorithm for undirected APSP under Min-Weight- k -Clique hypothesis.

2 DEFINITIONS OF GRAPH PROBLEMS

We will deal with either an unweighted graph $G = (V, E)$ or a weighted graph $G = (V, E, w)$, where the weight function is $w : E \rightarrow \mathbb{R}^+$. We assume that the vertices have distinct labels with $\lceil \log n \rceil$ bits. Let $W_{\max}$ and $W_{\min}$ denote the largest and the smallest edge weight, and let the *edge weight ratio* be $\rho = W_{\max} / W_{\min}$. Let $d_G(x, y)$ denote the length (or weight) of a shortest path from x to y in G , and for a cycle C in G , let $d_C(x, y)$ denote the length of the shortest path from x to y in C . We deal with only simple graphs in this paper.

We present results for the following graph problems in this paper.

APSP. This is the problem of computing the shortest path distances for every pair of vertices in G together with a concise representation of the shortest paths, which in our case is an $n \times n$ matrix, $Last_G$, that contains, in position (x, y) , the predecessor vertex of y on a shortest path from x to y .

All Pairs Shortest Distances (APSD). Given a graph $G = (V, E)$, the APSD problem only involves computing the shortest path distances for every pair of vertices in G . Most of the currently known APSD algorithms, including matrix multiplication based methods for small integer weights [37, 38, 45], can compute APSP in the same bound as APSD.

Minimum Weight Cycle (MWC). Given a graph $G = (V, E)$, the minimum weight cycle problem is to find the weight of a minimum weight cycle in G .

All Nodes Shortest Cycles (ANSC). Given a graph $G = (V, E)$, the ANSC problem is to find the weight of a shortest cycle through each vertex in G .

Replacement Paths. Given a graph $G = (V, E)$ and a pair of vertices s, t , the replacement paths problem is to find, for each edge e on the shortest path from s to t , a shortest path from s to t avoiding e .

k -SiSP. Given a graph $G = (V, E)$ and $s, t \in V$, the k -SiSP problem is to find the k shortest simple paths from s to t : the i -th path must be different from first $(i - 1)$ paths and must have weight greater than or equal to the weight of any of these $(i - 1)$ paths.

k -SiSC. The corresponding cycle version of k -SiSP is known as k -SiSC, where the goal is to compute the k shortest simple cycles through a given vertex x , such that the i -th cycle generated is different from all previously generated $(i - 1)$ cycles and has weight greater than or equal to the weight of any of these $(i - 1)$ cycles.

Radius. For a given graph $G = (V, E)$, the Radius problem is to compute the value $\min_{x \in V} \max_{y \in V} d_G(x, y)$. The *center* of a graph is the vertex x which minimizes this value.

Diameter. For a given graph $G = (V, E)$, the Diameter problem is to compute the value $\max_{x, y \in V} d_G(x, y)$.

Eccentricities. For a given graph $G = (V, E)$, the Eccentricities problem is to compute the value $\max_{y \in V} d_G(x, y)$ for each vertex $x \in V$.

Betweenness Centrality (BC). For a given graph $G = (V, E)$ and a node $v \in V$, the Betweenness Centrality of v , $BC(v)$, is the value $\sum_{s, t \in V, s, t \neq v} \frac{\sigma_{s, t}(v)}{\sigma_{s, t}}$, where $\sigma_{s, t}$ is the number of shortest paths

from s to t and $\sigma_{s,t}(v)$ is the number of shortest paths from s to t passing through v .

As in [2] we assume that the graph has unique shortest paths, hence $BC(v)$ is simply the number of s, t pairs such that the shortest path from s to t passes through v .

All Nodes Betweenness Centrality (ANBC). The all-nodes version of Betweenness Centrality: determine $BC(v)$ for all vertices.

Positive Betweenness Centrality (Pos BC). Given a graph $G = (V, E)$ and a vertex v , the Pos BC problem is to determine if $BC(v) > 0$.

All Nodes Positive Betweenness Centrality (Pos ANBC). The all-nodes version of Positive Betweenness Centrality.

Reach Centrality (RC). For a given graph $G = (V, E)$ and a node $v \in V$, the Reach Centrality of v , $RC(v)$, is the value $\max_{s,t \in V: d_G(s,v)+d_G(v,t)=d_G(s,t)} \min(d_G(s,b), d_G(b,t))$

3 OUR CONTRIBUTIONS

I. Sparse Reductions and the mn Partial Order. In Definition 3.1 below, we define the notion of a sparsity preserving reduction (or *sparse reduction* for short) from a graph problem P to a graph problem Q that allows P to inherit Q 's time bound for the graph problem as a function of both m and n , as long as the reduction is efficient. Our definition is in the spirit of a Karp reduction [24], but slightly more general, since we allow a constant number of calls to Q instead of just one call in a Karp reduction (and we allow polylog calls for $\tilde{O}(\cdot)$ bounds).

One could consider a more general notion of a sparsity preserving reduction in the spirit of Turing reductions as in [16] (which considers functions of a single variable n). However, for all of the many sparsity preserving reductions we present here, the simpler notion defined below suffices. It should also be noted that the simple and elegant definition of a Karp reduction suffices for the vast majority of known NP-completeness reductions. The key difference between our definition and other definitions of fine-grained reductions is that it is fine-grained with regard to both m and n , and respects the dependence on *both parameters*.

It would be interesting to see if some of the open problems left by our work on fine-grained reductions for the mn class can be solved by moving to a more general sparsity preserving reduction in the spirit of a Turing reduction applied to functions of both m and n . We do not consider this more general version here since we do not need it for our reductions.

Definition 3.1 (Sparsity Preserving Graph Reductions). Given graph problems P and Q , there is a *sparsity preserving* $f(m, n)$ reduction from P to Q , denoted by $P \leq_{f(m,n)}^{sprs} Q$, if given an algorithm for Q that runs in $T_Q(m, n)$ time on graphs with n vertices and m edges, we can solve P in $O(T_Q(m, n) + f(m, n))$ time on graphs with n vertices and m edges, by making a constant number of oracle calls to Q .

For simplicity, we will refer to a sparsity preserving graph reduction as a *sparse reduction*, and we will say that P *sparse reduces* to Q . Similar to Definition 3.1, we will say that P *tilde- $f(m, n)$ sparse reduces* to Q , denoted by $P \lesssim_{f(m,n)}^{sprs} Q$, if, given an algorithm for Q that runs in $T_Q(m, n)$ time, we can solve P in $\tilde{O}(T_Q(m, n) + f(m, n))$ time (by making polylog oracle calls to Q on graphs with $\tilde{O}(n)$

vertices and $\tilde{O}(m)$ edges). We will also use $\equiv_{f(m,n)}^{sprs}$ and $\cong_{f(m,n)}^{sprs}$ in place of $\leq_{f(m,n)}^{sprs}$ and $\lesssim_{f(m,n)}^{sprs}$ when there are reductions in both directions. In a weighted graph we allow the $\tilde{O}$ term to have a $\log \rho$ factor. (Recall that $\rho = W_{max}/W_{min}$.)

We present several sparse reductions for problems that currently have $\tilde{O}(mn)$ time algorithms. This gives rise to a partial order on problems that are known to be sub-cubic equivalent, and currently have $\tilde{O}(mn)$ time algorithms. For the most part, our reductions take $\tilde{O}(m+n)$ time (many are in fact $O(m+n)$ time), except reductions to APSP take $\tilde{O}(n^2)$ time. This ensures that any improvement in the time bound for the target problem will give rise to the same improvement to the source problem, to within a polylog factor. Surprisingly, very few of the known sub-cubic reductions for the problems we consider carry over to the sparse case (and none from [42]). This is due to one or both of the following features.

- (1) A central technique used in many of the reductions that show sub-cubic equivalence to APSP is to reduce from triangle finding problems such as Min-Wt- Δ (e.g., [2, 42]). These reductions start with a triangle finding problem and proceed by constructing suitable tripartite graphs where the property of the target problem can be used to detect a desired triangle. However, a cycle can have $\Theta(n)$ vertices, and using such an approach starting with MWC would create n -partite graphs with $\Theta(n^2)$ vertices. Hence this approach does not work in our sparse setting, and this highlights the need to develop new techniques to reduce from MWC. Further as noted above, in the sparse setting, all triangle finding and enumeration problems are in $\tilde{O}(m^{3/2})$ time, which is an asymptotically smaller bound than mn for graphs with $m = o(n^2)$. Hence reductions to triangle finding problems (e.g., [35] in the dense case) are not relevant in the sparse setting (unless the mn time bound can be improved).
- (2) Many of the known sub-cubic reductions convert a sparse graph into a dense one (e.g., [35, 42]), and again these are not relevant in the sparse setting.

The definitions of the problems we consider are in Section 2. Tables 1 and 2 summarize the improvements our reductions achieve over prior results. We now give some highlights of our results.

(a) Undirected Graphs: Finding the weight of a minimum weight cycle (MWC) is a fundamental problem. A simple sparse $O(m+n)$ reduction from MWC to APSD is known for directed graph but it does not work in the undirected case mainly because an edge can be traversed in either direction in an undirected graph, and known algorithms for the directed case would create non-simple paths when applied to an undirected graph. Roditty and Williams [35], in a follow-up to [42], pointed out the challenges of reducing from undirected MWC to APSD in sub- n^ω time, where ω is the matrix multiplication exponent, and then gave a $\tilde{O}(n^2)$ reduction from undirected MWC to undirected Min-Wt- Δ in a *dense* bipartite graph. But a reduction that increases the density of the graph is not helpful in our sparse setting. Instead, in this paper we give a sparse $\tilde{O}(n^2)$ time reduction from undirected MWC to APSD. Similar techniques allow us to obtain a sparse $\tilde{O}(n^2)$ time reduction from undirected ANSC (All Nodes Shortest Cycles) [36, 44], which asks for a shortest cycle through every vertex) to APSP.

Table 1: Our sparse reduction results for undirected graphs. Note that Min-Wt- Δ can be solved in $m^{3/2}$ time.

REDUCTION	PRIOR RESULTS (UNDIRECTED)	NEW RESULTS (UNDIRECTED)
MWC $\leq$ APSD (Sec. 4)	Dense $\tilde{O}(n^2)$ MWC $\leq$ Min-Wt- Δ [35] ($\Theta(n^2)$ edges in reduced graphs) Min-Wt- $\Delta \leq$ APSD (trivial)	Sparse $\tilde{O}(n^2)$ Reduction ($\Theta(m)$ edges in reduced graphs)
ANSC (Unweighted) $\leq$ APSP (Sec. 4)	Sparse $\tilde{O}(mn^{\frac{3-\omega}{2}})$ reduction [44] a. randomized b. polynomial calls to APSP c. gives randomized $\tilde{O}(n^{\frac{\omega+3}{2}})$ time algorithm [44]	Sparse $\tilde{O}(n^2)$ Reduction a. deterministic b. $\tilde{O}(1)$ calls to APSP c. gives deterministic $\tilde{O}(n^\omega)$ time algorithm

Table 2: Our sparse reduction results for directed graphs. The definitions for these problems are in Section 2. Note that Min-Wt- Δ can be solved in $m^{3/2}$ time.

REDUCTION	PRIOR RESULTS (DIRECTED)	NEW RESULTS (DIRECTED)
MWC $\leq$ 2-SiSP	Dense $\tilde{O}(n^2)$ MWC $\leq$ Min-Wt- Δ [35] Dense $O(n^2)$ Min-Wt- $\Delta \leq$ 2-SiSP [42]	Sparse $O(m)$ Reduction
2-SiSP $\leq$ Radius; $\leq$ BC	Sparse $O(n^2)$ 2-SiSP $\leq$ APSP [17] but Dense sub- n^3 APSP $\leq$ Min-Wt- Δ [42] Sparse $\tilde{O}(m)$ Min-Wt- $\Delta \leq$ Radius, BC [2]	Sparse $\tilde{O}(m)$ Reductions
Replacement paths $\leq$ ANSC; $\leq$ Eccentricities	Sparse $O(n^2)$ Rep. Paths $\leq$ APSP [17] but Dense sub- n^3 APSP $\leq$ Min-Wt- Δ [42] Sparse $O(m)$ Min-Wt- $\Delta \leq$ ANSC (trivial) Sparse $\tilde{O}(m)$ Min-Wt- $\Delta \leq$ Eccentricities [2]	Sparse $\tilde{O}(m)$ Reductions
ANSC $\leq$ ANBC	Sparse $O(m)$ ANSC $\leq$ APSP (trivial) but Dense sub- n^3 APSP $\leq$ Min-Wt- Δ [42] Sparse $\tilde{O}(m)$ Min-Wt- $\Delta \leq$ ANBC [2]	Sparse $\tilde{O}(m)$ Reduction

Figure 1: Some of our sparse reductions for weighted directed graphs (More details are in Fig. 7).

This reduction improves the running time for unweighted ANSC in *dense* graphs [44], since we can now solve it in $\tilde{O}(n^\omega)$ time using the unweighted APSP algorithm in [8, 37]. Our ANSC reduction and resulting improved algorithm is only for unweighted graphs and extending it to weighted graphs appears to be challenging.

We introduce a new *bit-sampling technique* in these reductions. This technique contains a simple construction with exactly $\log n$ hash functions for Color Coding [9] with 2 colors (described in detail

in Section 4). Our bit-sampling method also gives the first near-linear time algorithm for k -SiSC in weighted undirected graphs. k -SiSC is the cycle variant of k -SiSP [25], and our reduction that gives a fast algorithm for k -SiSC in weighted undirected graphs is given in the full paper [7](Sec. A.2).

The full proofs of Theorems 3.2 and 3.3 are in Section 4.1 and full paper [7](Sec. 7) respectively.

THEOREM 3.2. *In a weighted undirected n -node m -edge graph with edge weight ratio ρ , MWC can be computed with $2 \cdot \log n \cdot \log \rho$ calls*

to APSD on graphs with $2n$ nodes, at most $2m$ edges, and edge weight ratio at most ρ , with $O(n + m)$ cost for constructing each reduced graph, and with additional $O(n^2 \cdot \log n \cdot \log(n\rho))$ processing time. Additionally, every edge in the reduced graph retains its corresponding edge weight from the original graph. Hence, $MWC \lesssim_{n^2}^{sprs} APSD$.

In undirected graphs with integer weights at most W_{max} , APSD can be computed in $\tilde{O}(W_{max} \cdot n^\omega)$ time [37, 38]. In [35], the authors give an $\tilde{O}(W_{max} \cdot n^\omega)$ time algorithm for undirected MWC in such graphs by preprocessing using a result in [28] and then making $\tilde{O}(1)$ calls to an APSD algorithm. By applying our sparse reduction in Theorem 3.2 we can get an alternate simpler $\tilde{O}(W_{max} \cdot n^\omega)$ time algorithm for MWC in undirected graphs.

The following result gives an improved algorithm for ANSC in undirected unweighted graphs.

THEOREM 3.3. *In undirected unweighted graphs, $ANSC \lesssim_{n^2}^{sprs}$ APSP and ANSC can be computed in $\tilde{O}(n^\omega)$ time.*

(b) Directed Graphs. We give several nontrivial sparse reductions starting from MWC in directed graphs, as noted in the following theorem (also highlighted in Figure 1).

THEOREM 3.4 (DIRECTED GRAPHS.). *In weighted directed graphs:*

- (1) $MWC \lesssim_{m+n}^{sprs} 2\text{-SiSP} \lesssim_{m+n}^{sprs} s\text{-}t \text{ replacement paths} \equiv_{m+n}^{sprs} ANSC \lesssim_{m+n}^{sprs} \text{Eccentricities}$
- (2) $2\text{-SiSP} \lesssim_{m+n}^{sprs} \text{Radius} \lesssim_{m+n}^{sprs} \text{Eccentricities}$, and
- (3) $2\text{-SiSP} \lesssim_{m+n}^{sprs} BC$

We present these sparse reductions in Section 5.

Sparse Reductions for Directed versus Undirected Graphs.

We present fewer fine-grained reductions for undirected graphs than for directed graphs. This is due to the following reasons: First, the 2-SiSC, 2-SiSP and Replacement Paths problem can be solved in near-linear time in undirected graphs and hence they are not part of our partial order for undirected graphs. Second, as highlighted before, an edge can be traversed in either direction in an undirected graph hence a shortest cycle problem, especially when the size of the cycle is unrestricted, cannot be readily reduced to a shortest path problem in undirected graphs. We confront this problem with our bit-sampling technique (see Section 4), which we use in all of our reductions involving cycle problems in undirected graphs. This technique also gives a new near-linear time algorithm for weighted $k\text{-SiSC}$, a new $\tilde{O}(n^\omega)$ time algorithm for unweighted ANSC, and a simple $\tilde{O}(W_{max} \cdot n^\omega)$ algorithm for MWC with integer weights in $[1 \dots W_{max}]$.

II. Conditional Hardness Results. Conditional hardness under fine-grained reductions falls into several categories such as 3SUM hardness [14], OV hardness [11, 40, 41], SETH hardness [21, 40, 41] and sub-cubic APSP hardness for graph problems on dense graphs [42].

In this paper our focus is on the mn class, the class of graph path problems for which the current best algorithms run in $\tilde{O}(mn)$ time. This class differs from all previous classes considered for fine-grained complexity since it depends on two parameters of the input, m and n . To formalize hardness results for this class we first make

precise the notion of a sub- mn time bound, which we capture with the following definition.

Definition 3.5 (Sub- mn). A function $g(m, n)$ is sub- mn if $g(m, n) = O(m^\alpha \cdot n^\beta)$, where α, β are constants such that $\alpha + \beta < 2$.

A straightforward application of the notions of sub-cubic and sub-quadratic to the two-variable function mn would have resulted in a simpler but less powerful definition for sub- mn , namely that of requiring a time bound $O((mn)^\delta)$, for some $\delta < 1$. Another weaker form of the above definition would have been to require $\alpha \leq 1$ and $\beta \leq 1$ with at least one of the two being strictly less than 1. The above definition is more general than either of these. It considers a bound of the form m^3/n^2 to be sub- mn even though such a bound for the graph problem with be larger than n^3 for dense graphs. Thus, it is a very strong definition of sub- mn when applied to hardness results. (Such a definition could be abused when giving sub- mn reductions, but as noted in part I above, all of our sub- mn reductions are linear or near-linear in the sizes of input and output, and thus readily satisfy the sub- mn definition while being very efficient.)

We will say that a problem is MWC-hard if there is a sub- mn reduction from Minimum Weight Cycle to this problem. Our fine-grained reductions are based on this MWC hardness assumption, i.e. there is no sub- mn time algorithm for MWC (Minimum Weight Cycle).

Definition 3.6 (MWC Hardness). A problem Q is said to be MWC-Hard if there is a sub- mn reduction from MWC (Minimum Weight Cycle) to Q .

The following observation follows from this definition.

OBSERVATION 3.7. *If a problem Q is MWC-Hard, then there is no sub- mn time algorithm for Q unless MWC (Minimum Weight Cycle) can be computed in sub- mn time.*

Directed MWC is a natural candidate for hardness for the mn class since it is a fundamental problem for which a simple $\tilde{O}(mn)$ time algorithm has been known for many decades, and very recently, an $O(mn)$ time algorithm [29]. But a sub- mn time algorithm remains as elusive as ever. Further, through the fine-grained reductions that we present in this paper, many other problems in the mn class have MWC hardness for sub- mn time as noted in the following theorem.

THEOREM 3.8. *The following problems on directed graphs do not have sub- mn time algorithms: 2-SiSP, 2-SiSC, $s\text{-}t$ Replacement Paths, ANSC, Radius, BC, and Eccentricities assuming that MWC cannot be computed in sub- mn time.*

In recent work, Lincoln et al. [27] showed that MWC is hard under the Min-Wt- k -Clique Conjecture. This result in conjunction with Theorem 3.8 shows that the problems in the $\tilde{O}(mn)$ class including 2-SiSP, 2-SiSC, Replacement Paths, Radius, BC, Eccentricities and ANSC are also Min-Wt- k -Clique hard.

We now discuss the SETH and $k\text{-DSH}$ hardness of Eccentricities and BC, which are also MWC and Min-Wt- k -Clique hard.

SETH and $k\text{-DSH}$ hardness for Diameter, Eccentricities and BC. Another fundamental graph problem in the $\tilde{O}(mn)$ class is Diameter, which has a trivial sub- mn reduction to Eccentricities. Even though Diameter is in the $\tilde{O}(mn)$ class, we do not have an

MWC-hardness result for it (nor is a sub-cubic reduction from MWC known). However computing Diameter in sub- m^2 time in graphs with $m = O(n)$ edges was shown to be SETH-hard in [34] for both directed and undirected graphs. The *Strong Exponential Time Hypothesis (SETH)* [21] states that for every $\delta < 1$ there exists a k such that there is no $2^{\delta n}$ time algorithm for k -SAT. On the other hand, a SETH-based conditional hardness is not known for either MWC or APSP. The construction in [34] that gives sub- m^2 hardness for Diameter on very sparse graphs also gives a sub- mn hardness under SETH.

The *k Dominating Set Hypothesis (k-DSH)* [31] states that there exists k_0 such that for all $k \geq k_0$, a dominating set of size k in an undirected graph on n vertices cannot be found in $O(n^{k-\epsilon})$ for any constant $\epsilon > 0$. It is known that k -DSH hardness implies SETH-hardness, and a sub- m^2 hardness result for Diameter under k -DSH for even values of k was shown in [34] for graphs with $\tilde{O}(n)$ edges. In Section 7 we show that Diameter is k -DSH hard for sub- mn time for all values of k , both odd and even, thus strengthening the SETH and k -DSH hardness results for Diameter and Eccentricities.

Eccentricities as a Central Problem for mn . We observe that directed Eccentricities is a central problem in the mn class: If a sub- mn time algorithm is obtained for directed Eccentricities, not only would it refute k -DSH, SETH and MWC hardness (Definition 3.6) it would also imply sub- mn time algorithms for several MWC-hard problems: 2-SiSP, 2-SiSC, s -t Replacement Paths, ANSC and Radius in directed graphs as well as Radius and Eccentricities in undirected graphs.

APSP. APSP has a special status in the mn class. Since its output size is n^2 , it has near-optimal algorithms [18, 32, 33, 39] for graphs with $m = O(n)$. Also, the n^2 size for the APSP output means that any inference made through sparse reductions to APSP will not be based on a sub- mn time bound but instead on a sub- $mn + n^2$ time bound. It also turns out that the SETH and k -DSH hardness results for Diameter depend crucially on staying with a purely sub- mn bound, and hence even though Diameter has a simple sparse n^2 reduction to APSP, we do not have SETH or k -DSH hardness for computing APSP in sub- $mn + n^2$ time.

Betweenness Centrality (BC). We discuss this problem in Sec. 6. Sparse reductions for several variants of BC were given in [2] but none established MWC-hardness. We give nontrivial sparse reductions to establish MWC hardness for some important variants of BC. Our results show that BC and Eccentricities are key problems in the mn class that are MWC-hard and SETH/ k -DSH hard while being sub-cubic equivalent to APSP. Fig. 7 in Section 6 is an enhancement of Fig. 1 that includes our results for BC variants.

Separation of Time Bounds for Sparse Graphs. It is readily seen that the $\tilde{O}(m^{3/2})$ bound for triangle finding problems is a better bound than the $\tilde{O}(mn)$ bound for the mn class. But imposing a total ordering on functions of two variables requires some care. For example, maximal 2-connected subgraphs of a given directed graph can be computed in $O(m^{3/2})$ time [13] as well as in $O(n^2)$ time [19]. with $m^{3/2}$ a better bound for very sparse graphs and n^2 for very dense graphs. In the full paper ([7], Sec. 6) we motivate a natural definition of what it means for one time bound to be smaller than another time bound for sparse graphs. By our

definitions, $m^{3/2}$ is a smaller time bound than both mn and n^2 for sparse graphs. Our definitions establish that the problems related to triangle listing must have *provably smaller* time bounds for sparse graphs than the mn class under the hardness conjectures and fine-grained reductions for the MWC class.

Roadmap. Sections 4 and 5 present our sparse reductions for undirected and directed graphs. Section 6 present our sparse reductions for centrality problems in directed graphs. In Section 7 we present our k -DSH hardness result.

4 WEIGHTED UNDIRECTED GRAPHS

4.1 Reducing MWC to APSD

In undirected graphs, the only known sub-cubic reduction from MWC to APSD [35] uses a dense reduction to Min-Wt- Δ . Described in [35] for integer edge weights of value at most W_{max} , it first uses an algorithm in [28] to compute, in $O(n^2 \cdot \log n \log n W_{max})$ time, a 2-approximation W to the weight of a minimum weight cycle as well as shortest paths between all pairs of vertices with pathlength at most $W/2$. The reduced graph for Min-Wt- Δ is constructed as a (dense) bipartite graph with edges to represent all of these shortest paths, together with the edges of the original graph in one side of the bipartition. This results in each triangle in the reduced graph corresponding to a cycle in the original graph, and with a minimum weight cycle guaranteed to be present as a triangle. An MWC is then constructed using a version of Color Coding [9] with 2 colors.

The approach in [35] does not work in our case, as we are dealing with sparse reductions. Instead, we give a sparse reduction directly from MWC to APSD. In contrast to [35], where finding a minimum weight 3-edge triangle gives the MWC in the original graph, in our reduction the MWC is constructed as a path P in a reduced graph followed by a shortest path in the original graph.

One may ask if we can sparsify the dense reduction from MWC to Min-Wt- Δ in [35] but such a reduction, though very desirable, would immediately refute MWCC and would achieve a major breakthrough by giving an $\tilde{O}(m^{3/2})$ time algorithm for undirected MWC.

We now sketch our sparse reduction from undirected MWC to APSD. We start with stating from [35] the notion of a ‘critical edge’ and then present some additional properties we will use.

LEMMA 4.1 ([35]). *Let $G = (V, E, w)$ be a weighted undirected graph, where $w : E \rightarrow \mathbb{R}^+$, and let $C = \langle v_1, v_2, \dots, v_l \rangle$ be a cycle in G . There exists an edge (v_i, v_{i+1}) on C such that $\lceil \frac{w(C)}{2} \rceil - w(v_i, v_{i+1}) \leq d_C(v_1, v_i) \leq \lfloor \frac{w(C)}{2} \rfloor$ and $\lceil \frac{w(C)}{2} \rceil - w(v_i, v_{i+1}) \leq d_C(v_{i+1}, v_1) \leq \lfloor \frac{w(C)}{2} \rfloor$. The edge (v_i, v_{i+1}) is called the critical edge of C with respect to the start vertex v_1 .*

LEMMA 4.2. *Let C be a minimum weight cycle in weighted undirected graph G . Let x and y be two vertices on C and let $\pi_{x,y}^1$ and $\pi_{x,y}^2$ be the paths from x to y in C . WLOG assume that $w(\pi_{x,y}^1) \leq w(\pi_{x,y}^2)$. Then $\pi_{x,y}^1$ is a shortest path from x to y and $\pi_{x,y}^2$ is a second simple shortest path from x to y , i.e. a path from x to y that is shortest among all paths from x to y that are not identical to $\pi_{x,y}^1$.*

PROOF. Assume to the contrary that $\pi_{x,y}^3$ is a second simple shortest path from x to y of weight less than $w(\pi_{x,y}^2)$. Let $\pi_{x,y}^3$ deviate from $\pi_{x,y}^1$ at vertex u and then merge back at vertex v .

Then the subpaths from u to v in $\pi_{x,y}^1$ and $\pi_{x,y}^3$ together form a cycle of weight strictly less than $w(C)$, resulting in a contradiction as C is a minimum weight cycle in G . $\square$

OBSERVATION 4.3. *Let $G = (V, E, w)$ be a weighted undirected graph. Let $C = \langle v_1, v_2, \dots, v_l \rangle$ be a minimum weight cycle in G , and let (v_p, v_{p+1}) be its critical edge with respect to v_1 . WLOG assume that $d_G(v_1, v_p) \geq d_G(v_1, v_{p+1})$. If G' is obtained by removing edge (v_{p-1}, v_p) from G , then the path $P = \langle v_1, v_l, \dots, v_{p+1}, v_p \rangle$ is a shortest path from v_1 to v_p in G' .*

Observation 4.3 holds since the path P there must be either a shortest path or a second simple shortest path in G by the above lemma, so in G' it must be a shortest path.

In our reduction we construct a collection of graphs $G_{i,j,k}$, each with $2n$ vertices (containing 2 copies of V) and $O(m)$ edges, with the guarantee that, for the minimum weight cycle C , in at least one of the graphs the edge (v_{p-1}, v_p) (in Observation 4.3) will not connect across the two copies of V and the path P of Observation 4.3 will be present. Then, if a call to APSP computes P as a shortest path from v_1 to v_p (across the two copies of V), we can verify that edge (v_p, v_{p+1}) is not the last edge on the computed shortest path from v_1 to v_p in G , and so we can form the concatenation of these two paths as a possible candidate for a minimum weight cycle. The challenge is to construct a small collection of graphs where we can ensure that the path we identify in one of the derived graphs is in fact the simple path P in the input graph. We overcome this challenge by using our new *bit-sampling* technique which allows us to selectively sample edges from G to place in these constructed graphs $G_{i,j,k}$ such that P is guaranteed to be identified in one of these graphs.

Each $G_{i,j,k}$ has two copies of each vertex $u \in V$, $u^1 \in V_1$ and $u^2 \in V_2$. All edges in G are present on the vertex set V_1 , but there is no edge that connects any pair of vertices within V_2 . In $G_{i,j,k}$ there is an edge from $u^1 \in V_1$ to $v^2 \in V_2$ iff there is an edge from u to v in G , and u 's i -th bit is j , and $\frac{W_{\max}}{2^k} < w(u, v) \leq \frac{W_{\max}}{2^{k-1}}$. All the edges in $G_{i,j,k}$ retain their weights from G . Thus, the edge (u^1, v^2) is present in $G_{i,j,k}$ with weight w only if (u, v) is an edge in G with the same weight w and further, certain conditions (as described above) hold for the indices i, j, k . This is our bit-sampling method. Here, $1 \leq i \leq \lceil \log n \rceil$, $j \in \{0, 1\}$ and $k \in \{1, 2, \dots, \lceil \log \rho \rceil\}$, so we have $2 \cdot \log n \cdot \log \rho$ graphs. Figure 2 depicts the construction of graph $G_{i,j,k}$.

The first condition for an edge (u^1, v^2) to be present in $G_{i,j,k}$ is that u 's i -th bit must be j . This ensures that there exist a graph where the edge (v_{p-1}^1, v_p^2) is absent and the edge (v_{p+1}^1, v_p^2) is present (as v_{p-1} and v_{p+1} differ on at least 1 bit). To contrast with a similar step in [35], we need to find the path P in the sparse derived graph, while in [35] it suffices to look for the 2-edge path that represents P in a triangle in their dense reduced graph.

The second condition — that an edge (u^1, v^2) is present only if $\frac{W_{\max}}{2^k} < w(u, v) \leq \frac{W_{\max}}{2^{k-1}}$ — ensures that there is a graph $G_{i,j,k}$ in which, not only is edge (v_{p+1}^1, v_p^2) present and edge (v_{p-1}^1, v_p^2) absent as noted by the first condition, but also the shortest path from v_1^1 to v_p^2 is in fact the path P in Observation 4.3, and does not correspond to a false path where an edge in G is traversed

twice. In particular, we show that this second condition allows us to exclude a shortest path from v_1^1 to v_p^2 of the following form: take the shortest path from v_1 to v_p in G on vertices in V_1 , then take an edge (v_p^1, x^1) , and then the edge (x^1, v_p^2) . Such a path, which has weight $d_C(v_1, v_p) + 2w(x, v_p)$, could be shorter than the desired path, which has weight $d_C(v_1, v_{p+1}) + w(v_{p+1}, v_p)$. In our reduction we avoid selecting this ineligible path by requiring that the weight of the selected path should not exceed $d_G(v_1, v_p)$ by more than $M/2^{k-1}$. We show that these conditions suffice to ensure that P is identified in one of the $G_{i,j,k}$, and no spurious path of shorter length is identified. Notice that, in contrast to [35], we do not estimate the MWC weight by computing a 2-approximation. Instead, this second condition allows us to identify the critical edge in the appropriate graph.

In the following two lemmas we identify three key properties of a path π from y^1 to z^2 ($y \neq z$) in a $G_{i,j,k}$ that (I) will be satisfied by the path P in Observation 4.3 for $y^1 = v_1^1$ and $z^2 = v_p^2$ in some $G_{i,j,k}$ (Lemma 4.4), and (II) will cause a simple cycle in G to be contained in the concatenation of π with the shortest path from y to z computed by APSP (Lemma 4.5). Once we have these two Lemmas in hand, it gives us a method to find a minimum weight cycle in G (described in Algorithm MWC-to-APSP) by calling APSP on each $G_{i,j,k}$ and then identifying all pairs y^1, z^2 in each graph that satisfy these properties. Since the path P is guaranteed to be one of the pairs, and no spurious path will be identified, the minimum weight cycle can be identified. We now fill in the details.

LEMMA 4.4. *Let $C = \langle v_1, v_2, \dots, v_l \rangle$ be a minimum weight cycle in G and let (v_p, v_{p+1}) be its critical edge with respect to the start vertex v_1 . WLOG assume that $d_G(v_1, v_p) \geq d_G(v_1, v_{p+1})$. Then there exists an $i \in \{1, \dots, \lceil \log n \rceil\}$, $j \in \{0, 1\}$ and $k \in \{1, 2, \dots, \lceil \log \rho \rceil\}$ such that the following conditions hold:*

- (1) $d_{G_{i,j,k}}(v_1^1, v_p^2) + d_G(v_1, v_p) = w(C)$
- (2) $\text{Last}_{G_{i,j,k}}(v_1^1, v_p^2) \neq \text{Last}_G(v_1, v_p)$
- (3) $d_{G_{i,j,k}}(v_1^1, v_p^2) \leq d_G(v_1, v_p) + \frac{W_{\max}}{2^{k-1}}$

PROOF. Let i, j and k be such that: v_{p-1} and v_{p+1} differ on i -th bit and j be the i -th bit of v_{p+1} and k be such that $\frac{W_{\max}}{2^k} < w(v_p, v_{p+1}) \leq \frac{W_{\max}}{2^{k-1}}$. Hence, edge (v_{p-1}^1, v_p^2) is not present and the edge (v_{p+1}^1, v_p^2) is present in $G_{i,j,k}$ and so $\text{Last}_{G_{i,j,k}}(v_1^1, v_p^2) \neq \text{Last}_G(v_1, v_p)$, satisfying part 2 of the lemma.

Let us map the path P in Observation 4.3 to the path P' in $G_{i,j,k}$, such that all vertices except v_p are mapped to V_1 and v_p is mapped to V_2 (bold path from v_1^1 to v_p^2 in Figure 3b). Then, if P' is a shortest path from v_1^1 to v_p^2 in $G_{i,j,k}$, both parts 1 and 3 of the lemma will hold. So it remains to show that P' is a shortest path. But if not, an actual shortest path from v_1^1 to v_p^2 in $G_{i,j,k}$ would create a shorter cycle in G than C , and if that cycle were not simple, one could extract from it an even shorter cycle, contradicting the fact that C is a minimum weight cycle in G . $\square$

LEMMA 4.5. *If there exists $i \in \{1, \dots, \lceil \log n \rceil\}$, $j \in \{0, 1\}$, $k \in \{1, 2, \dots, \lceil \log \rho \rceil\}$, and $y, z \in V$ such that the following conditions hold:*

- (1) $d_{G_{i,j,k}}(y^1, z^2) + d_G(y, z) = wt$ for some wt

Figure 2: Construction of $G_{i,j,k}$.

MWC-to-APSP(G); ▷ this gives a simpler $\tilde{O}(W_{max} \cdot n^\omega)$ time algorithm for MWC with small integer weights than [35]

```

1:  $wt \leftarrow \infty$ 
2: for  $1 \leq i \leq \lceil \log n \rceil$ ,  $j \in \{0, 1\}$ , and  $1 \leq k \leq \lceil \log \rho \rceil$  do
3:   Compute APSP on  $G_{i,j,k}$ 
4:   for  $y, z \in V$  do
5:     if  $d_{G_{i,j,k}}(y^1, z^2) \leq d_G(y, z) + \frac{W_{max}}{2^{k-1}}$  then check if  $Last_{G_{i,j,k}}(y^1, z^2) \neq Last_G(y, z)$ 
6:     if both checks in Step 5 hold then  $wt \leftarrow \min(wt, d_{G_{i,j,k}}(y^1, z^2) + d_G(y, z))$ 
7: return  $wt$ 

```

(a) MWC C in G . Shortest path from v_1 to v_p is highlighted. (b) Shortest path from y^1 to z^2 in $G_{i,j,k}$.

- i, j, k are such that:
 1. i -th bit of $v_{p+1}(v_{p-1})$ is $j(j)$
 2. $W_{max}/2^k < w(v_p, v_{p+1}) \leq W_{max}/2^{k-1}$.
- Alg. MWC-to-APSP' will compute $w(C)$ in this $G_{i,j,k}$ in wt (line 6)

Figure 3: Here Figure (a) represent the MWC C in G . The path $\pi_{y,z}$ (in bold) is the shortest path from y to z in G . The path π_{y^1,z^2} (in bold) in Figure (b) is the shortest path from y^1 to z^2 in $G_{i,j,k}$: where the edge (v_{p-1}^1, v_p^1) is absent due to i, j bits. The paths $\pi_{y,z}$ in G and π_{y^1,z^2} in $G_{i,j,k}$ together comprise the MWC C .

- (2) $Last_{G_{i,j,k}}(y^1, z^2) \neq Last_G(y, z)$
- (3) $d_{G_{i,j,k}}(y^1, z^2) \leq d_G(y, z) + \frac{W_{max}}{2^{k-1}}$

Then G has a simple cycle of weight at most wt that contains z .

PROOF. Let $\pi_{y,z}$ be a shortest path from y to z in G (see Figure 3a) and let π_{y^1,z^2} be a shortest path from y^1 to z^2 in $G_{i,j,k}$ (Figure 3b). Let $\pi'_{y,z}$ be the path corresponding to π_{y^1,z^2} in G .

Now we need to show that the path $\pi'_{y,z}$ is simple. Assume that $\pi'_{y,z}$ is not simple. It implies that the path π_{y^1,z^2} must contain x^1 and x^2 for some $x \in V$. Now if $x \neq z$, then we can remove the subpath from x^1 to x^2 (or from x^2 to x^1) to obtain an even shorter path from y^1 to z^2 .

It implies that the path π_{y^1,z^2} contains z^1 as an internal vertex. Let π_{z^1,z^2} be the subpath of π_{y^1,z^2} from vertex z^1 to z^2 . If π_{z^1,z^2} contains at least 2 internal vertices then this would be a simple cycle of weight less than wt , and we are done. Otherwise, the path π_{z^1,z^2} contains exactly one internal vertex (say x^1). Hence path π_{z^1,z^2} corresponds to the edge (z, x) traversed twice in graph G . But the weight of the edge (x, z) must be greater than $\frac{W_{max}}{2^k}$ (as the edge (x^1, z^2) is present in $G_{i,j,k}$). Hence $w(\pi_{z^1,z^2}) > \frac{W_{max}}{2^{k-1}}$ and hence $d_{G_{i,j,k}}(y^1, z^2) \geq d_G(y, z) + w(\pi_{z^1,z^2}) > d_G(y, z) + \frac{W_{max}}{2^{k-1}}$, resulting in a contradiction as condition 3 states otherwise. (It is for this property that the index k in $G_{i,j,k}$ is used.) Thus path π_{y^1,z^2} does not contain z^1 as an internal vertex and hence $\pi'_{y,z}$ is simple.

ANSC-to-APSP

```

1: for each vertex  $z \in V$  do  $wt[z] \leftarrow \infty$ 
2: for  $1 \leq i \leq \lceil \log n \rceil, j \in \{0, 1\}$  do
3:   Compute APSP' on  $G_{i,j}$ ,
4:   for  $y, z \in V$  do
5:     if  $d_{G_{i,j}}(y^1, z^2) \leq d_G(y, z) + 1$  then check if  $Last_{G_{i,j}}(y^1, z^2) \neq Last_G(y, z)$ 
6:     if both checks in Step 5 hold then  $wt[z] \leftarrow \min(wt[z], d_{G_{i,j}}(y^1, z^2) + d_G(y, z))$ 
7: return  $wt$  array

```

If the paths $\pi_{y,z}$ and $\pi'_{y,z}$ do not have any internal vertices in common, then $\pi_{y,z} \circ \pi'_{y,z}$ corresponds to a simple cycle C in G of weight wt that passes through y and z . Otherwise, we can extract from $\pi_{y,z} \circ \pi'_{y,z}$ a cycle of weight smaller than wt . This establishes the lemma. $\square$

PROOF OF THEOREM 3.2: To compute the weight of a minimum weight cycle in G in $\tilde{O}(n^2 + T_{APSP})$, we use procedure MWC-to-APSP. By Lemmas 4.4 and 4.5, the value wt returned by this algorithm is the weight of a minimum weight cycle in G . $\square$

An $\tilde{O}(W_{max} \cdot n^\omega)$ time algorithm for MWC with small integer weights is given in [35]. By using the $\tilde{O}(W_{max} \cdot n^\omega)$ algorithm for APSP with small integer edge weights [38], Alg. MWC-to-APSP gives a simpler $\tilde{O}(W_{max} \cdot n^\omega)$ time algorithm for this problem.

Sparse Reduction to APSD: We now describe how to avoid using the $Last$ matrix in the reduction. A 2-approximation algorithm for finding a cycle of weight at most $2t$, where t is such that the minimum-weight cycle's weight lies in the range $(t, 2t]$, as well as distances between pairs of vertices within distance at most t , was given by Lingas and Lundell [28]. This algorithm can also compute the last edge on each shortest path it computes, and its running time is $\tilde{O}(n^2 \log(np))$. For a minimum weight cycle $C = \langle v_1, v_2, \dots, v_l \rangle$ where the edge (v_p, v_{p+1}) is a critical edge with respect to the start vertex v_1 , the shortest path length from v_1 to v_p or to v_{p+1} is at most t . Thus using this algorithm, we can compute the last edge on a shortest path for such pair of vertices in $\tilde{O}(n^2 \log(np))$ time.

In our reduction to APSD, we first run the 2-approximation algorithm on the input graph G to obtain the $Last(y, z)$ for certain pairs of vertices. Then, in Step 5 we check if $Last_{G_{i,j,k}}(y^1, z^2) \neq Last_G(y, z)$ only if $Last_G(y, z)$ has been computed (otherwise the current path is not a candidate for computing a minimum weight cycle). It appears from the algorithm that the $Last$ values are also needed in the $G_{i,j,k}$. However, instead of computing the $Last$ values in each $G_{i,j,k}$, we check for the shortest path from y to z only in those $G_{i,j,k}$ graphs where the $Last_G(y, z)$ has been computed, and the edge is not present in $G_{i,j,k}$. In other words, if $Last(y, z) = q$, we will only consider the shortest paths from y^1 to z^2 in those graphs $G_{i,j,k}$ where q 's i -th bit is not equal to j . Thus our reduction to APSD goes through without needing APSP to output the $Last$ matrix. This gives rise to an improved algorithm for MWC with small integer weights.

4.2 Reducing ANSC to APSP in Unweighted Undirected Graphs

For our sparse $\tilde{O}(n^2)$ reduction from ANSC to APSP in unweighted undirected graphs, we use the graphs from the previous section, but we do not use the index k , since the graph is unweighted.

Our reduction exploits the fact that in unweighted graphs, every edge in a cycle is a critical edge with respect to some vertex. Thus we construct $2\lceil \log n \rceil$ graphs $G_{i,j}$, and in order to construct a shortest cycle through vertex z in G , we will set $z = v_p^2$ in the reduction in the previous section. Then, by letting one of the two edges incident on z in the shortest cycle through z be the critical edge for the cycle, the construction from the previous section will allow us to find the length of a minimum length cycle through z , for each $z \in V$, with the post-processing algorithm ANSC-to-APSP. The correctness of this sparse reduction follows from lemmas similar to Lemma 4.4 and 4.5 and is available in the full paper [7](Sec. 7). Alg. ANSC-to-APSP gives a new $\tilde{O}(n^\omega)$ time algorithm for unweighted ANSC.

4.3 Bit-Sampling

We use the bit-sampling technique in our reductions for undirected graphs: from weighted MWC to APSP (Sec. 4.1), unweighted ANSC to unweighted APSP (Sec. 4.2) and from weighted k -SiSC to k -SiSP (full paper [7], Sec. A.2). This technique is crucial to all of these reductions. Using this technique, we obtain a new near-linear time algorithm for undirected k -SiSC, a new $\tilde{O}(n^\omega)$ algorithm for unweighted undirected ANSC and a simpler $\tilde{O}(W_{max} \cdot n^\omega)$ algorithm for weighted MWC. Here we describe how this technique is different from the 'bit-encoding' technique in [2] and how it gives an explicit construction for Color Coding for 2 colors.

4.3.1 Bit-sampling and Color Coding. Color Coding is a method introduced by Alon, Yuster and Zwick [9]. For the special case of 2 colors, the method constructs a collection C of $O(\log n)$ different 2-colorings on an n -element set V , such that for every pair $\{x, y\}$ in V , there is a 2-coloring in C that assigns different colors to x and y . When the elements of V have unique $\log n$ -bit labels, e.g., by numbering them from 0 to $n - 1$, our bit-sampling method on index i (ignoring indices j and k) can be viewed as an explicit construction of exactly $\lceil \log n \rceil$ hash functions for the 2-perfect hash family: the i -th hash function assigns to each element the i -th bit in its label as its color.

In our construction we actually use $2 \log n$ functions (using both i and j) since we need a stronger version of color coding where, for any pair of vertices x, y , there is a hash function that assigns color 0 to x and 1 to y and another that assigns 1 to x and 0 to y . This is needed in order to ensure that when $x = v_{p-1}$ and $y = v_{p+1}$,

the edge (v_{p-1}^1, v_p^2) is absent and the edge (v_{p+1}^1, v_p^2) is present. A different variant of Color Coding with 2 colors is used in [35] in their dense reduction from undirected MWC to Min-Wt- Δ , and we do not immediately see how to apply our bit-sampling technique there.

Our bit-sampling method differs from a ‘bit-encoding’ technique used in some reductions in [1, 2], where the objective is to preserve sparsity in the constructed graph while also preserving paths from the original graph $G = (V, E)$. This technique creates paths between two copies of V by adding $\Theta(\log n)$ new vertices with $O(\log n)$ bit labels, and using the $O(\log n)$ bit labels on these new vertices to induce the desired paths in the constructed graph. The bit-encoding technique (from [2]) is useful for certain types of reductions, and we use it in our sparse reduction from 2-SiSP to Radius in Section 5, and from 2-SiSP to BC in Section 6.

The bit-sampling technique we use in our reduction here is different from this bit-encoding technique. Here the objective is to selectively sample the edges from the original graph to be placed in the reduced graph, based on the bit-pattern of the end points and the edge weight. In our construction, we create $\Theta(\log n)$ different graphs, where in each graph the copies of V are connected by single-edge paths, without requiring additional intermediate vertices.

In the full paper [7](Sec. A.2), we give another application of our *bit-sampling technique* to obtain a new near-linear time algorithm for k -SiSC in undirected graphs (see definition in Sec. 2). Note that this problem is not in the mn class and this result is relevant here as an application of our new bit-sampling technique.

5 WEIGHTED DIRECTED GRAPHS

5.1 Reducing 2-SiSP to Radius and s - t Replacement Paths to Eccentricities

A sparse $O(n^2)$ reduction from 2-SiSP to APSP was given in [17]. Our sparse reduction from 2-SiSP to Radius refines this result and the sub- mn partial order by plugging the Radius and Eccentricities problems within the sparse reduction chain from 2-SiSP to APSP. Also, in Sec 5.2 we show $\text{MWC} \leq_{m+n}^{\text{sprs}} 2\text{-SiSP}$, thus establishing MWC-hardness for both 2-SiSP and Radius. Our 2-SiSP to Radius reduction here is unrelated to the sparse reduction in [17] from 2-SiSP to APSP.

The input is $G = (V, E, w)$, with source s and sink t in V , and a shortest path P ($s = v_0 \rightarrow v_1 \rightsquigarrow v_{l-1} \rightarrow v_l = t$). We need to compute a second simple s - t shortest path. Figure 4 gives an example of our reduction to an input G'' to the Radius problem for $l = 3$. Our reduction differs from a sub-cubic reduction from Min-Wt- Δ to Radius in [2] which transforms minimum weight triangle to Radius by creating a 4-partite graph. However, since Min-Wt- Δ can be solved in $O(m^{3/2})$ time this is not relevant to us. Instead, we give a more complex reduction from 2-SiSP where the second shortest path can have $\Theta(n)$ edges and hence we cannot start with a k -partite graph for some constant k .

In G'' we first map every edge (v_j, v_{j+1}) lying on P to the vertices z_{j_0} and z_{j_i} such that the shortest path from z_{j_0} to z_{j_i} corresponds to the shortest path from s to t avoiding the edge (v_j, v_{j+1}) . We then add vertices y_{j_0} and y_{j_i} in the graph and connect them to vertices z_{j_0} and z_{j_i} by adding edges (y_{j_0}, z_{j_0}) and (y_{j_i}, z_{j_i}) , and

then additional edges from y_{j_0} to other y_{k_0} and y_{k_i} vertices such that the longest shortest path from y_{j_0} is to the vertex y_{j_i} , which in turn corresponds to the shortest path from z_{j_0} to z_{j_i} . In order to preserve sparsity, we have an interconnection from each y_{j_0} vertex to all y_{k_i} vertices (except for $k = j$) with a sparse construction by using $2 \log n$ additional vertices $C_{r,s}$ in a manner similar to a bit-encoding technique used in [2] in their reduction from Min-Wt- Δ to Betweenness Centrality (this technique however, is different from the new ‘bit-sampling’ technique used in Section 4), and we have two additional vertices A, B with suitable edges to induce connectivity among the y_{j_0} vertices. In our construction, we ensure that the center is one of the y_{j_0} vertices and hence computing the Radius in the reduced graph gives the minimum among all the shortest paths from z_{j_0} to z_{j_i} . This corresponds to a shortest replacement path from s to t .

LEMMA 5.1. *In weighted directed graphs, $2\text{-SiSP} \leq_{m+n}^{\text{sprs}} \text{Radius}$ and s - t Replacement Paths $\leq_{m+n}^{\text{sprs}} \text{Eccentricities}$*

PROOF. We are given an input graph $G = (V, E)$, a source vertex s and a sink/target vertex t and we wish to compute the second simple shortest path from s to t . Let P ($s = v_0 \rightarrow v_1 \rightsquigarrow v_{l-1} \rightarrow v_l = t$) be the shortest path from s to t in G .

Constructing the reduced graph G'' : We first create the graph G' , which contain G and l additional vertices $z_0, z_1, \dots, z_{l-1}$. We remove the edges lying on P from G' . For each $0 \leq i \leq l-1$, we add an edge from z_i to v_i of weight $d_G(s, v_i)$ and an edge from v_{i+1} to z_i of weight $d_G(v_{i+1}, t)$. Also for each $1 \leq i \leq l-1$, we add a zero weight edge from z_i to z_{i-1} .

Now form G'' from G' . For each $0 \leq j \leq l-1$, we replace vertex z_j by vertices z_{j_i} and z_{j_0} and we place a directed edge of weight 0 from z_{j_i} to z_{j_0} , and we also replace each incoming edge to (outgoing edge from) z_j with an incoming edge to z_{j_i} (outgoing edge from z_{j_0}) in G'' .

Let $W_{\max}$ be the largest edge weight in G and let $M' = 9nW_{\max}$. For each $0 \leq j \leq l-1$, we add additional vertices y_{j_i} and y_{j_0} and we place a directed edge of weight 0 from y_{j_0} to z_{j_0} and an edge of weight $\frac{11}{9}M'$ from z_{j_i} to y_{j_i} .

We add 2 additional vertices A and B , and we place a directed edge from A to B of weight 0. We also add l incoming edges to A (outgoing edges from B) from (to) each of the y'_{j_0} ’s of weight 0 (M').

We also add edges of weight $\frac{2M'}{3}$ from y_{j_0} to y_{k_i} (for each $k \neq j$). But due to the addition of $O(n^2)$ edges, graph G' becomes dense. To solve this problem, we add a gadget in our construction that ensures that $\forall 0 \leq j \leq l-1$, we have at least one path of length 2 and weight equal to $\frac{2M'}{3}$ from y_{j_0} to y_{k_i} (for each $k \neq j$) (similar to [2]). In this gadget, we add $2\lceil \log n \rceil$ vertices of the form $C_{r,s}$ for $1 \leq r \leq \lceil \log n \rceil$ and $s \in \{0, 1\}$. Now for each $0 \leq j \leq l-1$, $1 \leq r \leq \lceil \log n \rceil$ and $s \in \{0, 1\}$, we add an edge of weight $\frac{M'}{3}$ from y_{j_0} to $C_{r,s}$ if j ’s r -th bit is equal to s . We also add an edge of weight $\frac{M'}{3}$ from $C_{r,s}$ to y_{j_i} if j ’s r -th bit is not equal to s . So overall we add $2n \log n$ edges that are incident to $C_{r,s}$ vertices; for each y_{j_0} we add $\log n$ outgoing edges to $C_{r,s}$ vertices and for each y_{j_i} we add $\log n$ incoming edges from $C_{r,s}$ vertices.

We can observe that for $0 \leq j \leq l-1$, there is at least one path of weight $\frac{2M'}{3}$ from y_{j_0} to y_{k_i} (for each $k \neq j$) and the gadget does not add any new paths from y_{j_0} to y_{j_i} . The reason is that for every

Figure 4: G'' for $l = 3$. The gray and the bold edges have weight $\frac{11}{9}M'$ and $\frac{1}{3}M'$ respectively. All the outgoing (incoming) edges from (to) A have weight 0 and the outgoing edges from B have weight M' .

distinct j, k , there is at least one bit (say r) where j and k differ and let s be the r -th bit of j . Then there must be an edge from y_{j_o} to $C_{r,s}$ and an edge from $C_{r,s}$ to y_{k_i} , resulting in a path of weight $\frac{2M'}{3}$ from y_{j_o} to y_{k_i} . And by the same argument we can also observe that this gadget does not add any new paths from y_{j_o} to y_{k_i} .

We call this graph as G'' . Figure 4 depicts the full construction of G'' for $l = 3$. We now establish the following three properties.

(i) For each $0 \leq j \leq l - 1$, the longest shortest path in G'' from y_{j_o} is to the vertex y_{j_i} . It is easy to see that the shortest path from y_{j_o} to any of the vertices in G or any of the z 's has weight at most nW_{max} . And the shortest paths from y_{j_o} to the vertices A and B have weight 0. For $k \neq j$, the shortest path from y_{j_o} to y_{k_o} and y_{k_i} has weight M' and $\frac{2}{3}M'$ respectively. Whereas the shortest path from y_{j_o} to y_{j_i} has weight at least $10nW_{max}$ as it includes the last edge (z_{j_i}, y_{j_i}) of weight $\frac{11}{9}M' = 11nW_{max}$. It is easy to observe that the shortest path from y_{j_o} to y_{j_i} corresponds to the shortest path from z_{j_o} to z_{j_i} .

(ii) The shortest path from z_{j_o} to z_{j_i} corresponds to the replacement path for the edge (v_j, v_{j+1}) lying on P . Suppose not and let P_j ($s \rightsquigarrow v_h \rightsquigarrow v_k \rightsquigarrow t$) (where v_h is the vertex where P_j separates from P and v_k is the vertex where it joins P) be the replacement path from s to t for the edge (v_j, v_{j+1}) . But then the path π_j ($z_{j_o} \rightarrow z_{j-1_i} \rightsquigarrow z_{h_o} \rightarrow v_h \circ P_j(v_h, v_k) \circ v_k \rightarrow z_{k_i} \rightarrow z_{k_o} \rightsquigarrow z_{j_i}$) (where $P_j(v_h, v_k)$ is the subpath of P_j from v_j to v_k) from z_{j_o} to z_{j_i} has weight equal to $wt(P_j)$, resulting in a contradiction as the shortest path from z_{j_o} to z_{j_i} has weight greater than that of P_j .

(iii) One of the vertices among y_{j_o} 's is a center of G'' . It is easy to see that none of the vertices in G could be a center of the graph G'' as there is no path from any $v \in V$ to any of the y_{j_o} 's in G'' . Using a similar argument, we can observe that none of the z 's, or the vertices y_{j_i} 's could be a potential candidate for the center of G'' . For vertices A and B , the shortest path to any of the y_{j_i} 's has weight exactly $\frac{5}{3}M' = 15nW_{max}$, which is strictly greater than the weight of the largest shortest path from any of the y_{j_o} 's. Thus one of the vertices among y_{j_o} 's is a center of G'' .

Thus by computing the radius in G'' , from (i), (ii), and (iii), we can compute the weight of the shortest replacement path from s to

t , which by definition of 2-SiSP, is the second simple shortest path from s to t . This completes the proof of $2\text{-SiSP} \lesssim_{m+n}^{sprs}$ Radius.

Now, if instead of computing Radius in G'' we compute the Eccentricities of all vertices in G'' , then from (i) and (ii) we can compute the weight of the replacement path for every edge (v_j, v_{j+1}) lying on P , thus solving the replacement paths problem. So s - t Replacement Paths $\lesssim_{m+n}^{sprs}$ Eccentricities.

Constructing G'' takes $O(m + n \log n)$ time since we add $O(n)$ additional vertices and $O(m + n \log n)$ additional edges, and given the output of Radius (Eccentricities), we can compute 2-SiSP (s - t Replacement Paths) in $O(1)$ ($O(n)$) time and hence the cost of both reductions is $O(m + n \log n)$. $\square$

5.2 Directed ANSC and Replacement Paths

We first describe a sparse reduction from directed MWC to 2-SiSP, which we will use for reducing ANSC to the s - t replacement paths problem. This reduction is adapted from a sub-cubic non-sparse reduction from Min-Wt- Δ to 2-SiSP in [42]. The reduction in [42] reduces Min-Wt- Δ to 2-SiSP by creating a tripartite graph. Since starting from Min-Wt- Δ is not appropriate for our results (as discussed in our sparse reduction to directed Radius), we start instead from MWC, and instead of the tripartite graph used in [42] we use the original graph G with every vertex v replaced with 2 copies, v_i and v_o .

In this reduction, as in [42], we first create a path of length n with vertices labeled from p_0 to p_n , which will be the initial shortest path. We then map every edge (p_i, p_{i+1}) to the vertex i in the original graph G such that the replacement path from p_0 to p_n for the edge (p_i, p_{i+1}) corresponds to the shortest cycle passing through i in G . Thus computing 2-SiSP (i.e., the shortest replacement path) from p_0 to p_n in the constructed graph corresponds to the minimum weight cycle in the original graph. Figure 5 gives an example of the constructed graph for $n = 3$.

LEMMA 5.2. In weighted directed graphs, $MWC \leq_{m+n}^{sprs} 2\text{-SiSP}$

The proof of Lemma 5.2 is available in full paper [7](Sec. 8).

We now establish the equivalence between ANSC and the s - t replacement paths problem under $(m + n)$ -reductions by first showing

Figure 5: G'' for $n = 3$ in the reduction: $\text{MWC} \leq_{m+n}^{\text{sprs}} \text{2-SiSP}$

an $(m+n)$ -sparse reduction from s - t replacement paths problem to ANSC. We then describe a sparse reduction from ANSC to the s - t replacement paths problem, which is similar to the reduction from MWC to 2-SiSP.

LEMMA 5.3. *In weighted directed graphs, s - t replacement paths $\equiv_{m+n}^{\text{sprs}} \text{ANSC}$*

The proof of Lemma 5.3 is available in full paper [7](Sec. 8).

6 BETWEENNESS CENTRALITY: REDUCTIONS

In this section, we consider sparse reductions for Betweenness Centrality and related problems. In its full generality, the Betweenness Centrality of a vertex v is the sum, across all pairs of vertices s, t , of the fraction of shortest paths from s to t that contain v as an internal vertex. This problem has a $\tilde{O}(mn)$ time algorithm due to Brandes [12]. Since there can be an exponential (in n) number of shortest paths from one vertex to another, this general problem can deal with very large numbers. In [2], a simplified variant was considered, where it is assumed that there is a unique shortest path for each pair of vertices, and the Betweenness Centrality of vertex v , $BC(v)$, is defined as the number of vertex pairs s, t such that v is an internal vertex on the unique shortest path from s to t . We will also restrict our attention to this variant here.

A number of sparse reductions relating to the following problems were given in [2].

- *Betweenness Centrality (BC)* of a vertex v , $BC(v)$.
- *Positive Betweenness Centrality (Pos BC)* of v : determine whether $BC(v) > 0$.
- *All Nodes Betweenness Centrality (ANBC)*: compute, for each v , the value of $BC(v)$.
- *Positive All Nodes Betweenness Centrality (Pos ANBC)*: determine, for each v , whether $BC(v) > 0$.
- *Reach Centrality (RC)* of v : compute $\max_{s,t \in V: d_G(s,v)+d_G(v,t)=d_G(s,t)} \min(d_G(s,v), d_G(v,t))$.

Figure 6 gives an overview of the previous fine-grained results given in [2] for Centrality problems. In this figure, BC is the only centrality problem that is known to be sub-cubic equivalent to APSP, and hence is shaded in the figure (along with Min-Wt- Δ). None of these sparse reductions in [2] imply MWC hardness for any of the centrality problems since Diameter is not MWC-hard (or even sub-cubic equivalent to APSP), and Min-Wt- Δ has an $\tilde{O}(m^{3/2})$ time algorithm, and so will give a sub- mn algorithm for MWC if it

is MWC-hard. On other hand, Diameter is known to be both SETH-hard [34] and k -DSH Hard (Sec. 7) and hence all these problems in Figure 6 (except Min-Wt- Δ) are also SETH and k -DSH hard.

In this section, we give a sparse reduction from 2-SiSP to BC, establishing MWC-hardness for BC. We also give a tilde-sparse reduction from ANSC to Pos ANBC, and thus we have MWC-hardness for both Pos ANBC and for ANBC, though neither problem is known to be in the sub-cubic equivalence class. (Both have $\tilde{O}(mn)$ time algorithms, and have APSP-hardness under sub-cubic reductions.)

Figure 7 gives an updated partial order of our sparse reductions for weighted directed graphs; this figure augments Figure 1 by including the sparse reductions for BC problems given in this section.

6.1 2-SiSP to BC

Our sparse reduction from 2-SiSP to BC is similar to the reduction from 2-SiSP to Radius described in Section 5. The input is $G = (V, E, w)$, with source s and sink t in V , and a shortest path P ($s = v_0 \rightarrow v_1 \rightsquigarrow v_{l-1} \rightarrow v_l = t$). We need to compute a second simple s - t shortest path. Figure 8 gives an example of our reduction to an input G'' to the BC problem for $l = 3$.

In our reduction, we first map every edge (v_j, v_{j+1}) to new vertices y_{j_o} and y_{j_i} such that the shortest path from y_{j_o} to y_{j_i} corresponds to the replacement path from s to t for the edge (v_j, v_{j+1}) . We then add an additional vertex A and connect it to vertices y_{j_o} 's and y_{j_i} 's. We also ensure that the only shortest paths passing through A are from y_{j_o} to y_{j_i} . We then do binary search on the edge weights for the edges going from A to y_{j_i} 's with oracle calls to the Betweenness Centrality problem, to compute the weight of the shortest replacement path from s to t , which by definition of 2-SiSP, is the second simple shortest path from s to t .

LEMMA 6.1. *In weighted directed graphs, $2\text{-SiSP} \leq_{m+n}^{\text{sprs}} \text{BC}$*

The proof of Lemma 6.1 is available in full paper [7](Sec. 9).

6.2 ANSC to Positive ANBC

We now describe a tilde-sparse reduction from the ANSC problem to the All Nodes Positive Betweenness Centrality problem (Pos ANBC). Sparse reductions from Min-Wt- Δ and from Diameter to Pos ANBC are given in [2]. However, Min-Wt- Δ can be solved in $O(m^{3/2})$ time, and Diameter is not known to be MWC-hard, hence neither of these reductions can be used to show hardness of the All Nodes Positive Betweenness Centrality problem relative to MWC hardness. (Recall that Pos ANBC is not known to be subcubic equivalent to APSP.)

Our reduction is similar to the reduction from 2-SiSP to the Betweenness Centrality problem, but instead of computing betweenness centrality through one vertex, it computes the positive betweenness centrality values for n different nodes. The input is $G = (V, E, w)$ and we wish to compute the ANSC in G .

In this reduction, we first split every vertex x into vertices x_o and x_i such that the shortest path from x_o to x_i corresponds to the shortest cycle passing through x in the original graph. We then add additional vertices z_x for each vertex x in the original graph and connect it to the vertices x_o and x_i such that the only shortest path passing through z_x is from x_o to x_i . We then perform binary search on the edge weights for the edges going from z_x to x_i with oracle

Figure 6: Known sparse reductions for centrality problems, all from [2]. The regular edges represent sparse $O(m+n)$ reductions, the squiggly edges represent tilde-sparse $O(m+n)$ reductions, and the dashed edges represent reductions that are trivial. BC and Min-Wt- Δ (shaded with gray) are known to be sub-cubic equivalent to APSP [2, 42].

Figure 7: Sparse reductions for weighted directed graphs. The regular edges represent sparse $O(m+n)$ reductions, the squiggly edges represent tilde-sparse $O(m+n)$ reductions, and the dashed edges represent reductions that are trivial. All problems except APSP are MWCC-hard. Eccentricities, BC, ANBC and Pos ANBC are also SETH/ k -DSH hard. ANBC and Pos ANBC (problems inside the dashed circles) are both not known to be subcubic equivalent to APSP. BC and Eccentricities (in bold) are MWC-hard, sub-cubic equivalent to APSP and SETH/ k -DSH hard.

Figure 8: G'' for $l = 3$ in the reduction: directed 2-SiSP $\lesssim_{m+n}^{sprs}$ Betweenness Centrality. The gray and the bold edges have weight M' and $\frac{1}{3}M'$ respectively. All the outgoing (incoming) edges from (to) A have weight $M' + q$ (0). Here $M' = 9nW_{max}$ where W_{max} is the largest edge weight in G and q is some value in the range from 0 to nW_{max} .

calls to the Positive Betweenness Centrality problem, to compute the weight of the shortest cycle passing through x in G .

LEMMA 6.2. In weighted directed graphs, $ANSC \lesssim_{m+n}^{sprs} Pos\ ANBC$

The proof of Lemma 6.2 is available in full paper [7](Sec. 9).

7 CONDITIONAL HARDNESS UNDER k -DSH

Here we improve on a result shown in [34] that a sub- m^2 algorithm for Diameter would refute k -DSH for even values of k by showing sub- mn hardness for Diameter for both odd and even values of k .

Since Diameter trivially reduces to Eccentricities and BC [2] and k -DSH hardness implies SETH hardness, this result also holds for Eccentricities and BC, and relative to both k -DSH and SETH.

Given an undirected graph $G = (V, E)$ we need to determine if G has a dominating set of size k . For k even we use [34] to construct $G' = (V', E')$, where $V' = V_1 \cup V_2$, with V_1 containing a vertex for each subset of V of size $k/2$ and $V_2 = V$. We add an edge from $v \in V_1$ to $x \in V_2$ if the subset corresponding to v does not dominate x and we induce a clique in the vertex partition V_2 . As shown in [34], G'

has diameter 3 if G has a dominating set of size k and has diameter 2 otherwise.

If k is odd, so $k = 2r + 1$, we make n calls to graphs derived from $G' = (V', E')$ as follows, where now each vertex in V_1 represents a subset of r vertices in V . For each $x \in V$ let V_x be the set $\{x\} \cup \{\text{neighbors of } x \text{ in } G\}$, and let G_x be the subgraph of G' induced on $V - V_x$. Now it is not difficult to see that G has a dominating set of size k that includes vertex x iff G_x has diameter greater than 2. (For details, see the proof in the full paper ([7], Sec. 5)).

Each graph G_x has $N = O(n^r)$ vertices and $M = O(n^{r+1})$ edges. If we now assume that Diameter can be computed in time $O(M^\alpha \cdot N^{2-\alpha-\epsilon})$, then the above algorithm for k Dominating Set runs in time $O(n \cdot M^\alpha \cdot N^{2-\alpha-\epsilon}) = O(n^{2r+1-\epsilon r+\alpha})$, which is $O(n^{k-\epsilon})$ time when $k \geq 3 + \frac{2\alpha}{\epsilon}$. The analysis is similar for k even, and has a straightforward extension to the directed case.

8 FURTHER WORK

We have given an extensive collection of sparse reductions for path problems in $\tilde{O}(mn)$ class and have established MWC as a key problem for this class. Several open problems remain of which we mention two.

Our reduction from ANSC to APSP in undirected graphs only works for the unweighted case. An open question here is to extend this reduction to the weighted case or to come up with an altogether different reduction.

The directed and undirected versions of most of these problems are known to be subcubic equivalent [42]. However such an equivalence is not known for the sparse case. For path problems, the undirected versions trivially reduces to their directed versions though nothing is known for the reverse case. Nothing is known in either direction for cycle problems. It will be interesting to see if one can establish sparse reductions for these problems.

REFERENCES

- [1] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. 2016. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Proc. ISDC*. Springer, 29–42.
- [2] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. 2015. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. SODA*. 1681–1697.
- [3] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. 2015. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. STOC*. ACM, 41–50.
- [4] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. FOCS*. IEEE, 434–443.
- [5] Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. 2016. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proc. SODA*. SIAM, 377–391.
- [6] Udit Agarwal and Vijaya Ramachandran. 2016. Finding k Simple Shortest Paths and Cycles. In *Proc. ISAAC*. 8:1–8:12.
- [7] Udit Agarwal and Vijaya Ramachandran. 2016. Fine-grained complexity and conditional hardness for sparse graphs. *arXiv preprint arXiv:1611.07008v3* (2016).
- [8] Noga Alon, Zvi Galil, Oded Margalit, and Moni Naor. 1992. Witnesses for boolean matrix multiplication and for shortest paths. In *Proc. FOCS*. IEEE, 417–426.
- [9] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-coding. *Journal of the ACM (JACM)* 42, 4 (1995), 844–856.
- [10] Amihoud Amir, Timothy M Chan, Moshe Lewenstein, and Noa Lewenstein. 2014. On hardness of jumbled indexing. In *Proc. ICALP*. Springer, 114–125.
- [11] Arturs Backurs and Piotr Indyk. 2015. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false). In *Proc. STOC*. ACM, 51–58.
- [12] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Jour. Math. Soc.* 25, 2 (2001), 163–177.
- [13] Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F Italiano, Veronika Loitzenbauer, and Nikos Parotsidis. 2017. Faster algorithms for computing maximal 2-connected subgraphs in sparse directed graphs. In *Proc. SODA*. SIAM, 1900–1918.
- [14] Anka Gajentaan and Mark H Overmars. 1995. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry* 5, 3 (1995), 165–185.
- [15] Zvi Galil and Oded Margalit. 1997. All pairs shortest distances for graphs with small integer length edges. *Information and Computation* 134, 2 (1997), 103–139.
- [16] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. 2017. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proc. SODA*. SIAM, 2162–2181.
- [17] Zvi Gotthilf and Moshe Lewenstein. 2009. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Inf. Proc. Lett.* 109, 7 (2009), 352–355.
- [18] Torben Hagerup. 2000. Improved shortest paths on the word RAM. In *Proc. ICALP*. Springer, 61–72.
- [19] Monika Henzinger, Sebastian Krinninger, and Veronika Loitzenbauer. 2015. Finding 2-Edge and 2-Vertex Strongly Connected Components in Quadratic Time. In *Proc. ICALP*. Springer, 713–724.
- [20] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. 2015. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. STOC*. ACM, 21–30.
- [21] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k -SAT. *Jour. Comput. Sys. Sci.* 62, 2 (2001), 367–375.
- [22] Alon Itai and Michael Rodeh. 1978. Finding a Minimum Circuit in a Graph. *SIAM Jour. Comput.* 7, 4 (1978), 413–423.
- [23] Zahra Jafargholi and Emanuele Viola. 2016. 3SUM, 3XOR, Triangles. *Algorithmica* 74, 1 (2016), 326–343.
- [24] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.
- [25] Naoki Katoh, Toshihide Ibaraki, and Hisashi Mine. 1982. An efficient algorithm for k shortest simple paths. *Networks* 12, 4 (1982), 411–427.
- [26] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. 2016. Higher lower bounds from the 3SUM conjecture. In *Proc. SODA*. SIAM, 1272–1287.
- [27] Andrea Lincoln, V. Vassilevska Williams, and Ryan Williams. 2018. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. In *Proc. SODA*. SIAM. To appear.
- [28] Andrzej Lingas and Eva-Marta Lundell. 2009. Efficient approximation algorithms for shortest cycles in undirected graphs. *Inf. Proc. Lett.* 109, 10 (2009), 493–498.
- [29] James B Orlin and Antonio Sedeno-Noda. 2017. An $O(nm)$ time algorithm for finding the min length directed cycle in a graph. In *Proc. SODA*. SIAM.
- [30] Mihai Patrascu. 2010. Towards polynomial lower bounds for dynamic problems. In *Proc. STOC*. ACM, 603–610.
- [31] Mihai Patrascu and Ryan Williams. 2010. On the possibility of faster SAT algorithms. In *Proc. SODA*. SIAM, 1065–1075.
- [32] Seth Pettie. 2004. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science* 312, 1 (2004), 47–74.
- [33] Seth Pettie and Vijaya Ramachandran. 2005. A shortest path algorithm for real-weighted undirected graphs. *SIAM Jour. Comput.* 34, 6 (2005), 1398–1431.
- [34] Liam Roditty and Virginia Vassilevska Williams. 2013. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. STOC*. ACM, 515–524.
- [35] Liam Roditty and Virginia Vassilevska Williams. 2011. Minimum weight cycles and triangles: Equivalences and algorithms. In *Proc. FOCS*. IEEE, 180–189.
- [36] Piotr Sankowski and Karol Węgrzycki. 2017. Improved Distance Queries and Cycle Counting by Frobenius Normal Form. In *Proc. STACS*. 56:1–56:14.
- [37] Raimund Seidel. 1995. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Jour. Comput. Sys. Sci.* 51, 3 (1995), 400–403.
- [38] Avi Shoshan and Uri Zwick. 1999. All pairs shortest paths in undirected graphs with integer weights. In *Proc. FOCS*. IEEE, 605–614.
- [39] Mikkel Thorup. 1997. Undirected single source shortest paths in linear time. In *Proc. FOCS*. IEEE, 12–21.
- [40] Virginia Vassilevska Williams. 2015. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPICs-Leibniz Intl. Proc. Informatics*, Vol. 43. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- [41] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science* 348, 2 (2005), 357–365.
- [42] Virginia Vassilevska Williams and Ryan Williams. 2010. Subcubic equivalences between path, matrix and triangle problems. In *Proc. FOCS*. IEEE, 645–654.
- [43] Virginia Vassilevska Williams and Ryan Williams. 2013. Finding, minimizing, and counting weighted subgraphs. *SIAM Jour. Comp.* 42, 3 (2013), 831–854.
- [44] Raphael Yuster. 2011. A shortest cycle for each vertex of a graph. *Inf. Proc. Lett.* 111, 21 (2011), 1057–1061.
- [45] Uri Zwick. 2002. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *JACM* 49, 3 (2002), 289–317.