Fine-Grained Complexity and Conditional Hardness for Sparse Graphs

Udit Agarwal* and Vijaya Ramachandran*

November 22, 2016

Abstract

There is a large class of path and cycle problems on graphs that currently have $\tilde{O}(n^3)$[1] time algorithms. Graphs encountered in practice are typically sparse, with the number of edges m being close to linear in n, the number of vertices, or at least with $m << n^2$. When considering sparsity, the current time complexities of these problems split into two classes: the $\tilde{\Theta}(mn)$ class, which includes APSP, Betweenness Centrality, and Minimum-Weight-Cycle, among several other problems, and the $\Theta(m^{3/2})$ class, which includes all problems relating to enumerating and detecting triangles. Here n and m are the number of vertices and edges in the graph. We investigate the fine-grained complexity of these problems on sparse graphs, and our main results are the following:

1. **Reductions and Algorithms.** We define the notion of a sparse reduction that preserves graph sparsity, and we present several such reductions for graph problems in the $\tilde{O}(mn)$ class. This gives rise to a rich partial order on graph problems with $\tilde{O}(mn)$ time algorithms, with the Minimum-Weight-Cycle problem as a major source in this partial order, and APSP a major sink. Surprisingly, very few of the known subcubic results are sparse reductions (outside of a few reductions that place Centrality problems in the sub-cubic equivalence class [1]). We develop new techniques in order to preserve sparsity in our reductions, many of which are nontrivial and intricate. Some of our reductions also lead to improved algorithms for various problems on finding simple cycles in undirected graphs.

2. **Conditional Hardness.** We establish a surprising conditional hardness result for sparse graphs: We show that if the Strong Exponential Time Hypothesis (SETH) holds, then several problems in the $\tilde{O}(mn)$ class, including certain problems that are also in the sub-cubic equivalence class such as Betweenness Centrality and Eccentricities, *cannot* have 'sub-mn' time algorithms, i.e., algorithms that run in $O(m^{\alpha} \cdot n^{2-\alpha-\epsilon})$ time, for constants $\alpha \geq 0$, $\epsilon > 0$. In particular, this result means that under SETH, the sub-cubic equivalence class is split into at least two classes when sparsity is taken into account, with triangle finding problems having faster algorithms than Eccentricities or Betweenness Centrality. This hardness result for the $\tilde{O}(mn)$ class is also surprising because a similar hardness result for the sub-cubic class is considered unlikely [5] since this would falsify NSETH (Nondeterministic SETH).

*Dept. of Computer Science, University of Texas, Austin TX 78712. Email: udit@cs.utexas.edu, vlr@cs.utexas.edu. This work was supported in part by NSF Grant CCF-1320675. The first author's research was also supported in part by a Calhoun Fellowship.

[1] $\tilde{O}$ hides polylog factors. For APSP on dense graphs, we use it to also hide a larger, but sub-polynomial factor [26].

1 Introduction

In recent years there has been considerable interest in determining the fine-grained complexity of problems in P [26, 9, 4]. For instance, the 3SUM problem has been central to the fine-grained complexity of several problems with quadratic time algorithms in computational geometry and other areas [7]. The 3SUM problem has a quadratic time algorithm and no sub-quadratic (i.e., $O(n^{2-\epsilon})$ for some constant $\epsilon > 0$) time algorithm is known for it. It has been shown that a sub-quadratic time algorithm for any of a large number of problems would imply a sub-quadratic time for 3SUM. In a similar vein it has been shown that a sub-quadratic time algorithm for LCS or Edit Distance would imply a sub-quadratic time algorithm for finding orthogonal vectors (OV) [4] and the latter would falsify SETH (Strong Exponential Time Hypothesis) [24].

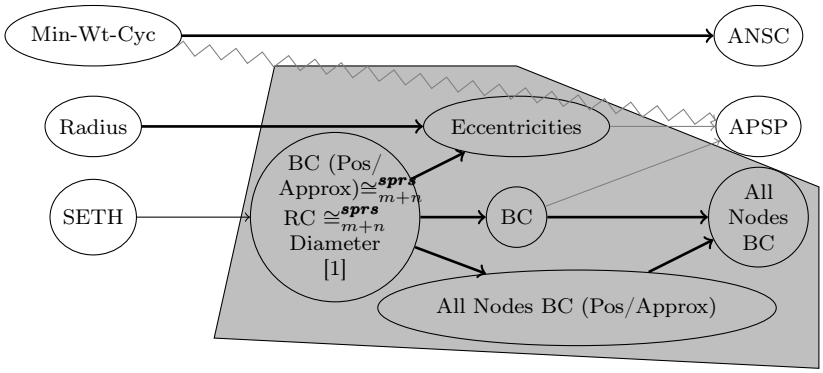
For several graph problems related to shortest paths that currently have $\tilde{O}(n^3)$ time algorithms, *equivalence* under sub-cubic reductions[2] has been shown: between all pairs shortest paths (APSP) in either a directed or undirected weighted graph, finding the second simple shortest path from u to v, for given vertices u and v (2-SiSP) in a weighted directed graph, finding a minimum weight cycle (Min-Wt-Cycle) in a directed or undirected graph, finding a minimum weight triangle (Min-Wt-Δ), and a host of other problems (see, e.g., [26]). This gives compelling evidence that a large class of graph problems is unlikely to have sub-cubic algorithms as a function of n, the number of vertices, unless fundamentally new algorithmic techniques are developed.

A time bound of n^3 is a full factor of n above the trivial lower bound of n^2 for the time complexity of some of the above graph problems such as APSP where the output has size n^2; $\Omega(n^2)$ also applies to all of these problems on truly dense graphs, where the number of edges m is $\Theta(n^2)$. However, most graphs that arise in practice are sparse, with the number of edges m typically $O(n^{1+\delta})$, for $\delta << 1$. For APSP there is an $O(mn + n^2 \log\log n)$ time algorithm for directed graphs [15] and a slightly faster $O(mn \log \alpha(m, n))$ time algorithm for undirected graphs [16] (where α is a certain natural inverse of the Ackermann's function). Both of these time bounds are very close to optimal for truly sparse graphs with $m = O(n)$. Thus the conditional hardness implied by the sub-cubic equivalences is not very relevant for sparse graphs that arise in practice.

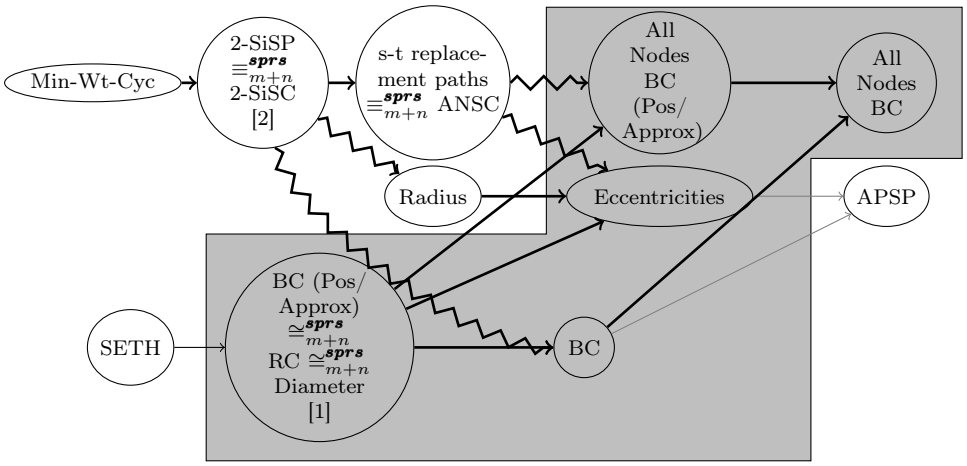
In this paper, we re-visit the fine-grained complexity of graph problems but we consider now the dependence on both n and m. All of the graph problems mentioned above (and several others known to be equivalent under sub-cubic reductions) have $\tilde{O}(mn)$ time algorithms; in fact all problems related to triangle detection and listing have an $O(m^{3/2})$ time algorithm, which is even faster for sparse graphs, though both mn and $m^{3/2}$ converge to n^3 for truly dense graphs.

For truly sparse graphs where $m = O(n)$ some fine-grained reductions and sub-quadratic hardness results with respect to m^2 are given in [17], but $m^2 = \Theta(n^4)$ when the graph is truly dense, and these results do not refine the sub-cubic hardness class. In this paper we derive results for arbitrary graphs taking sparseness into consideration, and we present novel reductions and hardness results that give a finer-grained insight into problems equivalent under sub-cubic reductions.

[2]Informally, a *sub-cubic* reduction from A to B implies that an algorithm running in $O(n^{3-\delta})$ for B would imply an $O(n^{3-\epsilon})$ time algorithm for A, where $\delta, \epsilon > 0$ are constants.



(a) Sparse Reductions for Weighted Undirected Graphs



(b) Sparse Reductions for Weighted Directed Graphs

Figure 1: Here BC stands for Betweenness Centrality, Pos stands for Positive, Approx stands for Approximate and RC stands for Reach Centrality. The bold edges represent sparse $O(m+n)$ reductions, the bold, squiggly edges represent tilde-sparse $O(m+n)$ reductions, the gray edges represent sparse $O(n^2)$ reductions and the gray, squiggly edges represent tilde-sparse $O(n^2)$ reductions. The shaded region indicates problems hard for sub-mn computations under SETH.

2 Our Contributions

Let $G = (V, E, w)$ be a weighted graph with $|V| = n$, $|E| = m$ and w the weight function $w : E \to \mathcal{R}^+$. Some of our results are for unweighted graphs where $w(e) = 1$ for all edges e. We let W_{max} and W_{min} denote the largest and the smallest edge weight in G, and let $\rho = W_{max}/W_{min}$. We assume that the vertices are numbered from 1 to n and each vertex is represented by $\lceil \log n \rceil$ bits. We 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 will say that a graph $G = (V, E)$ is sparse if $|E| = O(n^{1+\delta})$ for some positive constant $\delta < 1$. We deal with only simple graphs in this paper. We now give an overview of our results.

1. **Sparse Reductions and the mn Partial Order.**

Definition 2.1. (Sparse Reduction) *Given graph problems P and Q, there is an $f(m, n)$* sparse reduction from P to Q, *denoted by $P \leq^{sprs}_{f(m,n)} Q$, 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.*

Similarly, we will say that P *tilde-$f(m,n)$ sparse reduces to* Q, denoted by $P \lesssim^{\boldsymbol{sprs}}_{f(m,n)} Q$, 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, where $\tilde{O}$ hides polylog factors. We will also use $\equiv^{\boldsymbol{sprs}}_{f(m,n)}$ and $\cong^{\boldsymbol{sprs}}_{f(m,n)}$ in place of $\leq^{\boldsymbol{sprs}}_{f(m,n)}$ and $\lesssim^{\boldsymbol{sprs}}_{f(m,n)}$ 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 rich partial order of problems that are known to be sub-cubic equivalent, and currently have $\tilde{O}(mn)$ time algorithms. Surprisingly, very few of the known sub-cubic reductions carry over to the sparse case due to one or both of the following features. To start with, many of the known sub-cubic reductions convert a sparse graph into a dense one, which is fine when the complexity measure depends only on n as is the case for sub-cubic reductions, but this does not preserve dependence on both m and n. Secondly, a central technique used in those reductions has been to reduce from a suitable triangle finding problem. As noted above, triangle finding can be computed in $\tilde{O}(m^{3/2})$, which is a better bound than mn. In view of this, we present a suite of new sparse reductions for the $\tilde{O}(mn)$ time class. Many of our reductions are quite intricate, and we introduce a new technique of 'bit-fixing' for some of our reductions. In this technique we place an edge present in the input graph into a derived graph only if a certain bit pattern exists for the vertex labels on its endpoints, or on its weight. This technique is different from a method in [1] that uses $2 \log n$ new vertices and edges based on their bit patterns in order to create a sparse graph where the naive method would have given $\Theta(n^2)$ edges. (We also use this method from [1] in some of our reductions).

For the most part, our reductions take $\tilde{O}(m+n)$ time (many are in fact $O(m+n)$ time, though a few are $\tilde{O}(n^2)$ when output size is n^2), thus ensuring 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.

Figure 1 gives a schematic of our partial order of sparse reductions for weighted directed and undirected graphs. In Section 6 we give a related partial order for unweighted graphs. The full definitions of these problems are deferred to Section 10 (since most of them are well-known). As noted there, for most of these problems, the reductions that establish sub-cubic equivalences (excluding the Diameter equivalence class under sparse reductions) either go through a triangle finding problem or create a dense graph. We now summarize our results on sparse reductions.

(a) *Undirected Graphs:* APSP$'$ is the problem of finding both the weights of the shortest paths as well as the last edge on each shortest path (see Section 10 for definitions of all problems we consider). Most of the currently known APSP algorithms, including matrix multiplication based methods for small integer weights [21, 22, 28], can compute APSP$'$ in the same bound as APSP.

Finding the weight of a minimum weight cycle (Min-Wt-Cycle) is another fundamental problem. There is a simple sparse $O(m+n)$ reduction to APSP for directed graphs, but it does not work for undirected graphs. For the sub-cubic case, Roditty and Williams [18], in a follow-up paper to [26], pointed out the challenges of finding such a sub-cubic reduction, and then followed up with a sub-cubic reduction from undirected Min-Wt-Cycle to undirected Min-Wt-Triangle, using a dense bipartite graph. But a reduction that increases the density of the graph is disallowed in our sparse setting. Instead, in this paper we give a sparse $\tilde{O}(n^2)$ time reduction from undirected Min-Wt-Cycle to APSP. This reduction uses the new bit-fixing technique mentioned above. Similar techniques allow us to obtain a sparse $\tilde{O}(n^2)$ time reduction from ANSC (All Nodes Shortest Cycles),

which asks for a shortest cycle through every vertex in an undirected graph to APSP′. Interestingly, this reduction improves the running time for ANSC in *dense* graphs [27], since we can now solve it in $\tilde{O}(n^\omega)$ time using the APSP′ algorithm [21, 3]. Our reduction and resulting improved algorithm is only for unweighted graphs and extending it to weighted graphs appears to be challenging.

Min-Wt-Cycle is a major starting (source) problem in our partial order of sparse reductions. In Section 3 we present a brief overview of the proof of the following theorem, which is our main fine-grained result for undirected graphs. The full result is in Section 6.

Theorem 2.2. *In weighted undirected graphs, Min-Wt-Cycle* $\lesssim_{n^2}^{\boldsymbol{sprs}}$ *APSP*

(b) Directed Graphs. We give several sparse reductions starting from Min-Wt-Cycle for directed graphs. In contrast to the undirected case, many sparse reductions here are fairly trivial, but we give several new nontrivial sparse reductions as noted in the following theorem.

Theorem 2.3. (Directed Graphs.) *In weighted directed graphs:*
(i) Min-Wt-Cyc $\leq_{m+n}^{\boldsymbol{sprs}}$ *2-SiSP* $\leq_{m+n}^{\boldsymbol{sprs}}$ *s-t replacement paths* $\equiv_{m+n}^{\boldsymbol{sprs}}$ *ANSC* $\lesssim_{m+n}^{\boldsymbol{sprs}}$ *Eccentricities*
(ii) 2-SiSP $\lesssim_{m+n}^{\boldsymbol{sprs}}$ *Radius* $\leq_{m+n}^{\boldsymbol{sprs}}$ *Eccentricities*
(iii) 2-SiSP $\lesssim_{m+n}^{\boldsymbol{sprs}}$ *Betweenness Centrality*

In Section 4 we present a brief overview of our sparse reduction from 2-SiSP to Radius for directed graphs. The remaining reductions are in Section 7.

Our results give evidence that Min-Wt-Cycle is a central problem for graph problems (undirected and directed) that currently have $\tilde{O}(mn)$ time algorithms, similar to 3SUM for quadratic time algorithms: A sub-mn time algorithm for any problem in the large collection of problems for which we have shown a chain of sparse sub-mn reductions starting from Min-Wt-Cycle would imply a sub-mn time algorithm for Min-Wt-Cycle, which has been open for many decades. At the same time, our results also offer APSP as a problem to which many problems have sparse reductions. Thus we show chains of sparse reductions that refine the class of sub-cubic equivalent problems, with Min-Wt-Cycle as an important problem at the source end (the 'easier' end) and APSP at the sink end. We also have reductions starting from the Radius problem in undirected graphs, and there are also known sparse reductions starting from the Diameter problem [1], which has an $\tilde{O}(mn)$ time algorithm but is not known to be sub-cubic equivalent to APSP.

2. **Reductions and Algorithms.** Through our sparse reductions we are able to obtain faster algorithms for several problems related to finding cycles in undirected graphs as stated in the following theorem. These results are presented in Section 8. Algorithms for the corresponding problems on directed graphs were presented recently in [2]. Note that the k-SiSP and k-SiSC problems have considerably faster algorithms for undirected graphs than for directed graphs, hence the results for undirected graphs do not follow from the directed versions. Part (a) in the following theorem resolves an open question in [27], and also gives the first sub-cubic reduction from ANSC to APSP'.

Theorem 2.4. (Reductions and Algorithms.) *In undirected graphs:*
(a) For unweighted graphs, ANSC $\lesssim_{n^2}^{\boldsymbol{sprs}}$ APSP′ *and we can solve ANSC in* $\tilde{O}(n^\omega)$ *time.*
(b) For weighted graphs, k-SiSC $\cong_{m+n}^{\boldsymbol{sprs}}$ *k-SiSP, and we can compute k-SiSC in* $\tilde{O}(m+n)$ *time, k-ANSiSC in* $\tilde{O}(mn)$ *time and k-All-SiSC in* $\tilde{O}(m \cdot (n+k))$ *time.*

3. **Conditional Hardness.** The *Strong Exponential Time Hypothesis (SETH)* states that satisfiability of CNF-SAT on n variables cannot be computed in $O(2^{(1-\epsilon)n})$ time for any constant $\epsilon > 0$.

The *k Dominating Set Hypothesis (k-DSH)* states that 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 was shown in [14] that falsifying the k Dominating Set Hypothesis would falsify SETH.

Both SETH and k-DSH have remained unrefuted for several years. We complement our sparse reduction results with hardness results relative to SETH and k-DSH for improving the current time bounds for certain problems on sparse graphs.

Definition 2.5. (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 $\alpha + \beta < 2$.*

Theorem 2.6. (Conditional Hardness) *Under either SETH or k-DSH, none of the following problems have sub-mn algorithms in either undirected or directed graphs: Diameter, Eccentricities, and all versions of weighted Centrality problems.*

Our hardness results contrast with results known for dense graph problems that are sub-cubic equivalent to APSP: There is evidence (under NSETH [5]) that it is not possible to establish hardness under SETH for the sub-cubic equivalence class of problems, and yet we are able to establish a natural hardness result under SETH for several problems in the sub-*mn* partial order.

4. **Sparse Time Bounds and Their Separation Under SETH.** We define the notion of a sparse time bound and a relative ordering between certain pairs of these bounds.

Definition 2.7. (Sparse Time Bounds for Graph Problems) *Let $T(m,n)$ be a time bound for a graph algorithm where m is the number of edges and n the number of vertices in the input graph, and let $m = \Omega(n)$.*
(a) $T(m,n)$ is a sparse time bound *if it is strictly increasing in m.*
(b) $T(m,n)$ is a truly sparse time bound *if it is a sparse time bound and given values n, m, m' with $m' > m$, $\frac{T(m',n)}{T(m,n)} = \Omega\left(\left(\frac{m'}{m}\right)^{\epsilon}\right)$ for some constant $\epsilon > 0$.*

The above definition of a sparse time bound is meant to capture our intuitive notion that we want the time bound for a graph problem to decrease as the graph become more sparse. The goal of defining a *truly* sparse time bound is to differentiate between a polynomial dependence on m and a sub-polynomial dependence such as polylog. This is in the spirit of all previous results on conditional hardness and equivalances for fine-grained complexity (e.g., for APSP, 3SUM, LCS and their related problems). This definition requires a polynomial dependence on m but not on n since most graph problems can assume that $m = \Omega(n)$ by computing on (weakly) connected components. A sub-*mn* function (Definition 2.5) is a truly sparse time bound if $\alpha > 0$.

Next we define the notion of a sparse time bound $T(m,n)$ being smaller than another time bound $T'(m,n)$.

Definition 2.8. *Let $T'(m,n)$ be a time bound and let $T(m,n)$ be a truly sparse time bound. Then,*
(a) $T(m,n)$ is a smaller sparse time bound than $T'(m,n)$ *if there exist constants $\gamma, \epsilon > 0$ such that for all values of $m = O(n^{1+\gamma})$, $T(m,n) = O\left(\frac{1}{m^{\epsilon}} \cdot T'(m,n)\right)$.*
(b) $T(m,n)$ is a weakly smaller sparse time bound than $T'(m,n)$ *if there exists a positive constant γ such that for any constant δ with $\gamma > \delta > 0$, there exists an $\epsilon > 0$ such that $T(m,n) = O\left(\frac{1}{m^{\epsilon}} \cdot T'(m,n)\right)$ for all values of m in the range $m = O(n^{1+\gamma})$ and $m = \Omega(n^{1+\delta})$.*

Part (a) above definition requires a polynomially smaller (in m) bound for $T(m,n)$ relative to $T'(m,n)$ for sufficiently sparse graphs. For example, $\frac{m^3}{n^2}$ is a smaller sparse time bound than $m^{3/2}$,

which in turn is a smaller sparse time bound than mn; m^2 is a smaller sparse time bound than n^3 (note that n^3 is not a sparse bound). A time bound of $n\sqrt{m}$ is a weakly smaller sparse bound than $m\sqrt{n}$ by part (b) but not a smaller sparse bound since the two bounds coincide when $m = O(n)$.

Definitions 2.7 and 2.8 can be generalized by allowing for restricting the range of edge densities, e.g., $m\sqrt{n} + n^2$ is a truly sparse time bound when $m = \Omega(n^{3/2})$. We do not consider this here.

Theorem 2.9. (Conditional Split of Sparse Time Bounds.)
(a) Under either SETH or k-DSH, the triangle finding problems in the sub-cubic equivalence class have a smaller truly sparse time bound than computing Eccentricities or computing Betweenness Centrality for a single node in a graph with unique shortest paths.
(b) Under the 3SUM conjecture, the maximum matching problem has a weakly smaller sparse time bound than triangle enumeration problems.

Part (a) of the above theorem is established in Section 5, where we show a more general result. Part (b) follows from the conditional lower bound for triangle enumeration in [11] and the well-known $O(m\sqrt{n})$ time bound for maximum matching.

Theorem 2.8 provides a second differentiation between dense sub-cubic equivalent problems and their sparse counterparts. Weighted triangle finding problems are sub-cubic equivalent to Eccentricities and Betweenness Centrality. However, the above theorem shows that under SETH these latter problems cannot attain the time bound of $O(m^{3/2})$ known for triangle finding algorithms. Thus, by accounting for graph sparsity, we have truly refined the sub-cubic equivalence class.

Our hardness proofs are fairly straightforward, and adapt earlier hardness results to our framework; the significance of our hardness results is in the new insights they give into our inability to make improvements to some long-standing time bounds for important problems on sparse graphs. On the other hand, some of our sparse reductions are highly intricate, and overall these reductions give a rich partial order on the large class of graph problems that currently have $\tilde{O}(mn)$ time algorithms.

Roadmap. The rest of the paper is organized as follows. The next three sections provide some highlight of our results: In section 3 we present a sparse reduction from Min-Wt-Cycle to APSP in undirected graphs, in Section 4 we describe a sparse reduction from 2-SiSP to Radius in directed graphs, and in Section 5 we present our sub-mn hardness result for Diameter and other problems. In the remaining sections we present the rest of our results: In Section 6 we present our other main result for undirected graphs, a sparse reduction from ANSC to APSP′ in unweighted undirected graphs, which also gives a faster algorithm for dense graphs. In Section 7 we give several sparse reductions for directed graphs to complement the 2-SiSP to Radius reduction that we gave in Section 4. We then present our algorithmic results for finding shortest cycles in undirected graphs in Section 8. Section 9 contains the proofs of some Lemmas introduced in Section 5. Finally in Section 10, we describe the formal definitions of the problems that we have covered in this paper.

3 Weighted Undirected Graphs: Min-Wt-Cycle $\lesssim_{n^2}^{sprs}$ APSP

We now sketch a sparse reduction from Min-Wt-Cycle in a weighted undirected graph to APSP′. Full details are in Section 6.1.

It is not difficult to see (see [18]) that in any cycle $C = \langle v_1, v_2, \ldots, v_l \rangle$ in a weighted undirected graph $G = (V, E, w)$ 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. The following observation holds for any edge on a minimum weight cycle but we will apply it specifically to its critical edge with respect to v_1.

Observation 3.1. *Let $G = (V, E, w)$ be a weighted undirected graph. Let $C = \langle v_1, v_2, \ldots, v_l \rangle$ 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 $P = \langle v_1, v_l, \ldots, v_{p+1}, v_p \rangle$ is a shortest path from v_1 to v_p in G'.*

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 Observation 3.1, in at least one of the graphs the edge (v_{p-1}, v_p) will not connect across the two copies of V and the path P 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.

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. It uses two forms of our bit-fixing method: 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 $\frac{W_{max}}{2^k} < w(u, v) \leq \frac{W_{max}}{2^{k-1}}$. The edges connecting vertices in V_1 retain the weights from G, while for each edge connecting between V_1 and V_2, we add a large weight $Q = nW_{max}$ to its weight in G. Here, $1 \leq i \leq \lceil \log n \rceil$, $j \in \{0, 1\}$ and $k \in \{1, 2, \ldots, \lceil \log \rho \rceil\}$, so we have $O(\log n \cdot \log \rho)$ 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^1_{p-1}, v^2_p) is absent and the edge (v^1_{p+1}, v^2_p) is present (as v_{p-1} and v_{p+1} differ on at least 1 bit). 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^1_{p+1}, v^2_p) is present and edge (v^1_{p-1}, v^2_p) is absent as noted by the first condition, but also the shortest path from v^1_1 to v^2_p is in fact the path P in Observation 3.1, and does not correspond to a false path where an edge in G is duplicated. In particular, we show that this second condition allows us to exclude a shortest path from v^1_1 to v^2_p 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^1_p, x^1), and then the edge (x^1, v^2_p). 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 $Q + W_{max}/2^{k-1}$. We show below 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.

It is not difficult to see that $d_{G_{i,j,k}}(u^1, v^2) \geq d_G(u, v) + Q$ for all graphs $G_{i,j,k}$ (see Section 6.1). In the following lemma 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 (a) will be satisfied by the path P in Observation 3.1 for $x = v^1_1$ and $y = v^2_p$ in some $G_{i,j,k}$, and (b) will cause a simple cycle in G to be contained in the concatenation of π with the shortest path from x to y computed by APSP$'$. This gives us a method to find a minimum weight cycle in G 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.

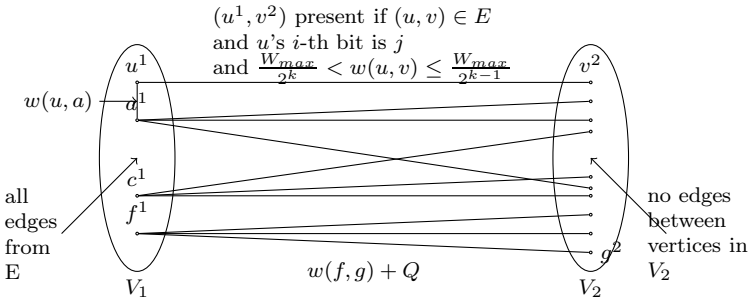


Figure 2: Construction of $G_{i,j,k}$. Here $Q = nW_{max}$.

Lemma 3.2. *Let $C = \langle v_1, v_2, \ldots, v_l \rangle$ 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. Assume $d_G(v_1, v_p) \geq d_G(v_1, v_{p+1})$. Then there exist $i \in \{1, \ldots, \lceil \log n \rceil\}$, $j \in \{0, 1\}$ and $k \in \{1, 2, \ldots, \lceil \log \rho \rceil\}$ such that the following conditions hold:*

1. $d_{G_{i,j,k}}(v_1^1, v_p^2) + d_G(v_1, v_p) = w(C) + Q$
2. $Last_{G_{i,j,k}}(v_1^1, v_p^2) \neq Last_G(v_1, v_p)$
3. $d_{G_{i,j,k}}(v_1^1, v_p^2) \leq d_G(v_1, v_p) + Q + \frac{W_{max}}{2^{k-1}}$

Further the converse also holds: If there exist vertices y, z in G with the above three properties satisfied for $y = v_1$, $z = v_p$ for one of the graphs $G_{i,j,k}$ using a weight wt in place of $w(C)$ in part 1, then there exists a cycle in G that passes through y (z) of weight at most wt.

The proof of the above lemma is in Section 6.1 (Lemmas 6.5 and 6.6). Section 6.1 also shows how to refine this reduction to APSP instead of APSP′, and in Section 6.2 we adapt it to obtain a sparse reduction from unweighted undirected ANSC to APSP′. Lemma 3.2 establishes that the following algorithm computes the weight of a minimum weight cycle by a sparse reduction to APSP′.

MWC-to-APSP′

1: $wt \leftarrow 0$
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** $d_{G_{i,j,k}}(y^1, z^2) \leq d_G(y, z) + Q + \frac{W_{max}}{2^{k-1}}$ **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** $wt \leftarrow min(wt, d_{G_{i,j,k}}(y^1, z^2) + d_G(y, z) - Q)$
7: **return** wt

4 Weighted Directed Graphs: 2-SiSP $\leq^{sprs}_{m+n}$ Radius

Both 2-SiSP and Radius are known to be subcubic equivalent to APSP [26, 1]. However in the absence of equivalence for sparse graphs, our aim is to refine the sub-mn partial order. A sparse $O(n^2)$ reduction from 2-SiSP to APSP was given in [8] and this led to a slightly faster k-SiSP algorithm using [15]. Our sparse reduction from 2-SiSP to Radius refines this result and the sub-mn partial order by plugging the Radius problem in the sparse reduction chain 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 ($s = v_0 \to v_1 \rightsquigarrow v_{l-1} \to v_l = t$), and we need to compute a second simple shortest path from s to t.

In our reduction, we construct a graph G'' where we first map every edge (v_j, v_{j+1}) lying on P to the vertices z_{j_o} and z_{j_i} such that the shortest path from z_{j_o} to z_{j_i} corresponds to the shortest path

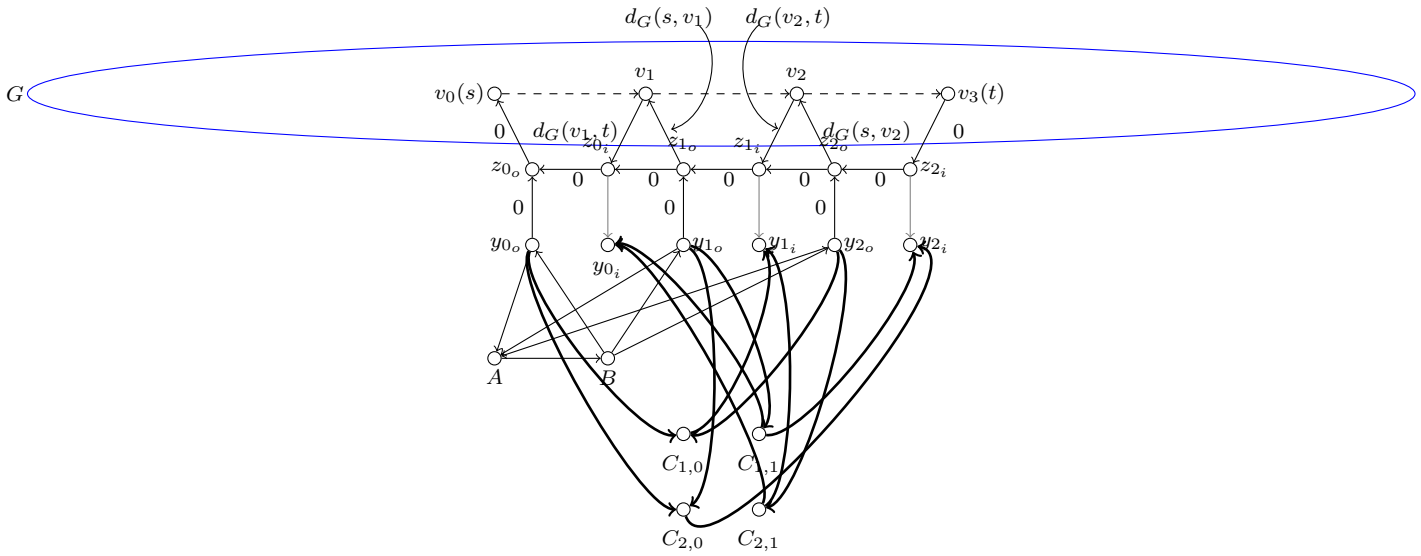


Figure 3: G'' for $l = 3$. The gray and the bold edges have weight $\frac{11}{9}M'$ and $\frac{1}{3}M'$ respectively. All the outgoing (incoming) edges from (to) A have weight 0 and the outgoing edges from B have weight M'.

from s to t avoiding the edge (v_j, v_{j+1}). We then add vertices y_{j_o} and y_{j_i} in the graph and connect them to vertices z_{j_o} and z_{j_i} such that the longest shortest path from y_{j_o} is to the vertex y_{j_i}, which in turn corresponds to the shortest path from z_{j_o} to z_{j_i}. We have a interconnection from each y_{j_o} vertex to all y_{k_i} vertices (except for $k = j$) with a sparse construction by using $O(\log n)$ additional vertices $C_{r,s}$ in a manner similar to a technique in [1], and we have two additional vertices A, B with suitable edges to induce connectivity among the y_{j_o} vertices. In our construction, we ensure that the center is among one of the vertices y_{j_o} and hence computing the Radius in the reduced graph gives the minimum among all the shortest paths from z_{j_o} to z_{j_i}. This corresponds to a shortest replacement path from s to t, which by definition is the second simple shortest path from s to t. Details of this reduction are in Section 7 and Figure 3 gives an example.

The following three claims are fairly straightforward and are proved in Section 7:

(*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}.

(*ii*) *A shortest path from* z_{j_o} *to* z_{j_i} *corresponds to a replacement path for the edge* (v_j, v_{j+1}).

(*iii*) *One of the vertices among* y_{j_o}*'s is a center of* G''.

Thus by computing the radius in G'' using (*i*), (*ii*), and (*iii*), we can compute the weight of a shortest replacement path from s to t, which is a second simple shortest path from s to t. The cost of this reduction is $O(m + n \log n)$.

The above result also gives us a sparse reduction from the s-t replacement paths problem to the Eccentricities problem. Details are in Section 7.

5 Conditional Hardness Results

The following lemma shows that a sub-mn time algorithm for Diameter in an unweighted graph, either undirected or directed, would falsify both k-DSH and SETH.

Lemma 5.1. *Suppose for some constant* α *there is an* $O(m^{\alpha} \cdot n^{2-\alpha-\epsilon})$ *time algorithm, for some*

$\epsilon > 0$, for solving Diameter in an unweighted m-edge n-node graph, either undirected or directed. Then there exists a $k' > 0$ such that for all $k \geq k'$, the k-Dominating Set problem can be solved in $O(n^{k-\epsilon})$ time.

The proof of this lemma is similar to a result in [17]. Consider determining if undirected graph $G = (V, E)$ has a k-dominating set. We form the graph $G' = (V', E')$, 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 a vertex $v \in V_1$ to a vertex $x \in V_2$ if the subset corresponding to v does not dominate x. We induce a clique in the vertex partition V_2. As shown in [17], G' has diameter 3 if G has a dominating set of size k and has diameter 2 otherwise, and this gives a reduction from k-Dominating Set to Diameter when k is even.

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 $V - V_x$. If we compute diameter in each G_x, $x \in V$ it is readily seen that we will detect a graph with diameter greater than 2 if and only if G has a dominating set of size k (see Section 9 for details).

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 remains the same for k even. In the directed case, we get the same result by replacing every edge in G' with two directed edges in opposite directions. This establishes Lemma 5.1.

This lemma, along with the reduction from SETH to k-Dominating Set in [14] and our sparse reductions in Section 3 and 4 gives us Theorem 2.6.

For Theorem 2.9, it suffices to show that $m^{3/2}$ is a faster sparse time bound than $m^\alpha \cdot n^{2-\alpha}$ for any $0 < \alpha \leq 2$ since Lemma 5.1, together with our sparse reductions, has shown that under SETH or k-DSH no $O(m^\alpha \cdot n^{2-\alpha-\epsilon})$ algorithm, for any $\epsilon > 0$, exists for Eccentricities or BC. This result follows from the following more general lemma.

Lemma 5.2. *Let $T_1(m, n) = O(m^{\alpha_1} n^{\beta_1})$ and $T_2(m, n) = O(m^{\alpha_2} n^{\beta_2})$ be two truly sparse time bounds, where $\alpha_1, \beta_1, \alpha_2, \beta_2$ are constants.*
(a) $T_1(m, n)$ is a smaller sparse time bound than $T_2(m, n)$ if $\alpha_2 + \beta_2 > \alpha_1 + \beta_1$, or
(b) $T_1(m, n)$ is a weakly smaller sparse time bound than $T_2(m, n)$ if $\alpha_2 + \beta_2 = \alpha_1 + \beta_1$, and $\alpha_2 > \alpha_1$.

Proof Sketch: (a) If $\alpha_2 + \beta_2 > \alpha_1 + \beta_1$, then for $\alpha_1 > \alpha_2$ we show that $T_1(m, n)$ is a smaller sparse time bound than $T_2(m, n)$ for any $\gamma < \frac{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)}{(\alpha_1-\alpha_2)}$ and $\epsilon = \frac{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)+(\alpha_2-\alpha_1)\gamma}{1+\gamma}$, and if $\alpha_1 \leq \alpha_2$, then the result holds for any $\gamma > 0$ and $\epsilon = \frac{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)}{1+\gamma}$.

(b) If $\alpha_2 + \beta_2 = \alpha_1 + \beta_1$, and $\alpha_2 > \alpha_1$, then we show that for any $\gamma > 0$, any constant $0 < \delta < \gamma$ (so $m = n^{1+\delta}$), and $\epsilon = \frac{(\alpha_2-\alpha_1)\delta}{1+\delta}$, $T_1(m, n)$ is a weakly smaller sparse time bound than $T_2(m, n)$.

The details of the derivations are in Section 9. □

Discussion.

Our results leave many avenues for further investigation. We list two here:

1. Can we extend our conditional hardness results under SETH for Eccentricities and Betweenness Centrality to a conditional hardness for sparse APSP under SETH? A sub-mn time bound is not

possible for APSP in truly sparse graphs since its output size is n^2. But a time bound of $O(g(m,n)+n^2)$ for APSP, with $g(m,n)$ a sub-mn bound with $\alpha > 0$, is a truly sparse time bound for APSP for a suitable range of edge densities, and our SETH-based conditional hardness result does not rule out a time bound of this form for APSP. More generally, can we obtain conditional hardness results for other problems in the mn partial order?

2. Obtaining further refinements of the mn partial order would be of interest, especially finding more equivalences within the mn partial order. In particular, can we place Min-Wt-Cycle and APSP in the same equivalence class under a suitable sparse reduction?

6 Fine-Grained Reductions for Undirected Graphs

In Section 3, we gave an overview of our sparse reduction from Min-Wt-Cycle to APSP. Here in Section 6.1, we provide full details of this reduction. We then describe a $\tilde{O}(n^2)$ sparse reduction from ANSC to APSP$'$ in unweighted undirected graphs in Section 6.2.

We start with the following lemma from [18] that identifies the notion of a 'critical edge'.

Lemma 6.1 ([18])**.** *Let $G = (V, E, w)$ be a weighted undirected graph, where $w : E \to \mathcal{R}^+$, and let $C = \langle v_1, v_2, \ldots, v_l \rangle$ 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.*

6.1 Reducing Minimum Weight Cycle to APSP

Our sparse reduction from Min-Wt-Cycle in a weighted undirected graph initially will be to APSP$'$ (which also computes the *Last* matrix), but then we show how we can convert this to a reduction to APSP. We first describe a useful property of a minimum weight cycle; our reduction will be based on the property stated in the Corollary that follows.

Lemma 6.2. *Let C be a minimum weight cycle in weighted undirected graph G. Let x and y be two vertices lying on the cycle C and let $\pi^1_{x,y}$ and $\pi^2_{x,y}$ be the paths from x to y in C. WLOG assume that $w(\pi^1_{x,y}) \leq w(\pi^2_{x,y})$. Then $\pi^1_{x,y}$ is a shortest path from x to y and $\pi^2_{x,y}$ 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 $\pi^1_{x,y}$.*

Proof. Assume to the contrary that $\pi^3_{x,y}$ is a second simple shortest path from x to y of weight less than $w(\pi^2_{x,y})$. Let the path $\pi^3_{x,y}$ deviates from the path $\pi^1_{x,y}$ at some vertex u and then it merges back at some vertex v. Then the subpaths from u to v in $\pi^1_{x,y}$ and $\pi^3_{x,y}$ together form a cycle of weight strictly less than $w(C)$, resulting in a contradiction as C is a minimum weight cycle in G. □

The following corollary holds for any edge on a minimum weight cycle but we will apply it specifically to its critical edge with respect to v_1.

Corollary 6.3. *Let $G = (V, E, w)$ be a weighted undirected graph. Let $C = \langle v_1, v_2, \ldots, v_l \rangle$ 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 $P = \langle v_1, v_l, \ldots, v_{p+1}, v_p \rangle$ is a shortest path from v_1 to v_p in G'.*

We now use the above Corollary in our sparse reduction, Min-Wt-Cycle $\lesssim_{n^2}^{\textbf{sprs}}$ APSP$'$. In this reduction we construct a collection of graphs $G_{i,j,k}$, each with $2n$ vertices and $O(m)$ edges. 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 $\frac{W_{max}}{2^k} < w(u,v) \leq \frac{W_{max}}{2^{k-1}}$. The edges connecting vertices in V_1 retain the weights from G, while for each edge connecting between V_1 and V_2, we add a large weight $Q = nW_{max}$ to its weight in G. Here, $1 \leq i \leq \lceil \log n \rceil$, $j \in \{0,1\}$ and $k \in \{1,2,\ldots,\lceil \log \rho \rceil\}$, so we have $O(\log n \cdot \log \rho)$ 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). The second condition ensures that there is a graph $G_{i,j,k}$ in which edge (v_{p+1}^1, v_p^2) is present and edge (v_{p-1}^1, v_p^2) is absent, and the shortest path from v_1^1 to v_p^2 is in fact P, and does not correspond to a false path where an edge in G is duplicated. In particular, as shown below, 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 $Q + W_{max}/2^{k-1}$. We show below 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.

We now describe and analyze our reduction, starting with a reduction to APSP$'$.

Lemma 6.4. *For every $i \in \{1,\ldots,\lceil \log n \rceil\}$, $j \in \{0,1\}$ and $k \in \{1,2,\ldots,\lceil \log \rho \rceil\}$ and $u, v \in V$, $d_{G_{i,j,k}}(u^1, v^2) \geq d_G(u,v) + Q$*

Proof. (Sketch) If $d_{G_{i,j,k}}(u^1, v^2) < d_G(u,v) + Q$ then π_{u^1,v^2}, the shortest path from u^1 to v^2 in $G_{i,j,k}$, must contain x^1 and x^2 for some $x \in V$. We can then remove the subpath from x^1 to x^2 to obtain an even shorter path from u^1 to v^2. □

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 (a) will be satisfied by the path P in Corollary 6.3 for $x = v_1^1$ and $y = v_p^2$ in some $G_{i,j,k}$ (Lemma 6.5), and (b) will cause a simple cycle in G to be contained in the concatenation of π with the shortest path from x to y computed by APSP$'$ (Lemma 6.6). Once we have these two Lemmas in hand, it gives us a method to find a minimum weight cycle in G 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 6.5. *Let $C = \langle v_1, v_2, \ldots, v_l \rangle$ 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 $i \in \{1,\ldots,\lceil \log n \rceil\}$, $j \in \{0,1\}$ and $k \in \{1,2,\ldots,\lceil \log \rho \rceil\}$ such that the following conditions hold:*

1. $d_{G_{i,j,k}}(v_1^1, v_p^2) + d_G(v_1, v_p) = w(C) + Q$
2. $Last_{G_{i,j,k}}(v_1^1, v_p^2) \neq Last_G(v_1, v_p)$

3. $d_{G_{i,j,k}}(v_1^1, v_p^2) \leq d_G(v_1, v_p) + Q + \frac{W_{max}}{2^{k-1}}$

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 $Last_{G_{i,j,k}}(v_1^1, v_p^2) \neq Last_G(v_1, v_p)$, satisfying part 2 of the lemma.

Let us map the path P in Corollary 6.3 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. 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. □

Lemma 6.6. *If there exists an $i \in \{1, \ldots, \lceil \log n \rceil\}$, $j \in \{0, 1\}$ and $k \in \{1, 2, \ldots, \lceil \log \rho \rceil\}$ and $y, z \in V$ such that the following conditions hold:*

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

Then there exists a simple cycle C containing z of weight at most wt in G.

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

The path π_{y^1,z^2} does not have vertices from V_2 except z^2 as the vertices in V_2 are not connected by an edge and the edges connecting the vertices from V_1 to V_2 have weight at least $Q = nW_{max}$.

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} 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 $w(\pi_{z^1,z^2}) > \frac{W_{max}}{2^{k-1}} + Q$ and hence $d_{G_{i,j,k}}(y^1, z^2) \geq d_G(y, z) + w(\pi_{z^1,z^2}) > d_G(y, z) + Q + \frac{W_{max}}{2^{k-1}}$, 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.

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. □

Proof of Theorem 2.2: To compute the weight of a minimum weight cycle in G in $\tilde{O}(n^2 + T_{\text{APSP}'})$, we use the procedure MWC-to-APSP$'$ given below. By Lemmas 6.5 and 6.6, the value wt returned by this algorithm is the weight of a minimum weight cycle in G.

MWC-to-APSP′

1: $wt \leftarrow 0$
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** $d_{G_{i,j,k}}(y^1, z^2) \leq d_G(y, z) + Q + \frac{W_{max}}{2^{k-1}}$ **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** $wt \leftarrow min(wt, d_{G_{i,j,k}}(y^1, z^2) + d_G(y, z) - Q)$
7: **return** wt

Sparse Reduction to APSP: 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 Roditty and Williams [18] (see also [12]). This algorithm runs in time $\tilde{O}(n^2 \log(n\rho))$, and it can compute the last edge on each shortest path it computes. For a minimum weight cycle $C = \langle v_1, v_2, \ldots, v_l \rangle$ 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 $\tilde{O}(n^2 \log(n\rho))$ time.

In our reduction to APSP, 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 $Last_G(y, z) = q$, 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 APSP goes through without needing APSP′ to output the *Last* matrix. □

6.2 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. We also do not add a large value Q to the weights of edges connecting V_1 and V_2, so our reduction is to unweighted APSP′. Also the vertex v_p^2 which was a key element in our reduction in Section 6.1 will now be the vertex z for which we want to find the length of the shortest cycle passing through it.

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 following post-processing algorithm.

ANSC-to-APSP′

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

Correctness of the above sparse reduction follows from the following two lemmas, which are similar to Lemmas 6.5 and 6.6, and their proofs are omitted for this extended abstract.

Lemma 6.7. *Let $C = \langle z, v_2, v_3, \ldots, v_q \rangle$ be a minimum length cycle passing through vertex $z \in V$. Let (v_p, v_{p+1}) be its critical edge such that $p = \lfloor \frac{q}{2} \rfloor + 1$. Then there exists an $i \in \{1, \ldots, \lceil \log n \rceil\}$ and $j \in \{0,1\}$ such that the following conditions hold:*

(i) $d_{G_{i,j}}(v_p^1, z^2) + d_G(v_p, z) = len(C)$

(ii) $Last_{G_{i,j}}(v_p^1, z^2) \neq Last_G(v_p, z)$

(iii) $d_{G_{i,j}}(v_p^1, z^2) \leq d_G(v_p, z) + 1$

Lemma 6.8. *If there exists an $i \in \{1, \ldots, \lceil \log n \rceil\}$ and $j \in \{0,1\}$ and $y, z \in V$ such that the following conditions hold:*

(i) $d_{G_{i,j}}(y^1, z^2) + d_G(y,z) = q$ *for some q where* $d_G(y,z) = \lfloor \frac{q}{2} \rfloor$

(ii) $Last_{G_{i,j}}(y^1, z^2) \neq Last_G(y,z)$

(iii) $d_{G_{i,j}}(y^1, z^2) \leq d_G(y,z) + 1$

Then there exists a simple cycle C passing through z of length at most q in G.

Proof of Theorem 2.4: We now show that the entries in the wt array returned by the above algorithm correspond to the ANSC output for G. Let $z \in V$ be an arbitrary vertex in G and let $q = wt[z]$. Let y' be the vertex in Step 5 for which we obtain this value of q. Hence by Lemma 6.8, there exists a simple cycle C passing through z of length at most q in G. If there were a cycle through z of length $q' < q$ then by Lemma 6.7, there exists a vertex y'' such that conditions in Step 6 hold for q', and the algorithm would have returned a smaller value than $wt[z]$, which is a contradiction. This is a sparse $\tilde{O}(n^2)$ reduction since it makes $O(\log n)$ calls to APSP′, and spends $\tilde{O}(n^2)$ additional time. □

Figure 4 gives an overview of the state of sparse reductions for unweighted directed graphs.

It would be interesting to see if we can obtain a reduction from *weighted* ANSC to APSP or APSP′. The above reduction does not work for the weighted case since it exploits the fact that for any cycle C through a vertex z, an edge in C that is incident on z is a critical edge for some vertex in C. However, this property need not hold in the weighted case.

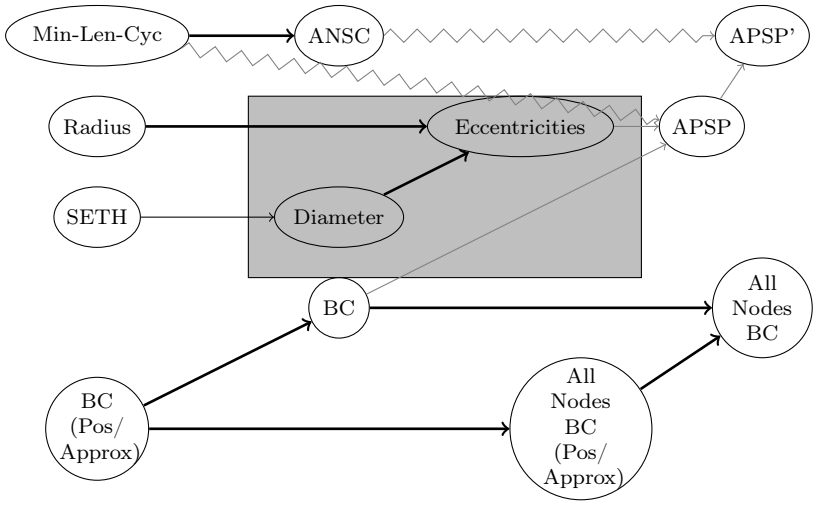


Figure 4: Chain of Reductions for Unweighted Undirected Graphs. Here BC stands for Betweenness Centrality, Pos stands for Positive and Approx stands for Approximate. The bold edges represent sparse $O(m+n)$ reductions, the bold, squiggly edges represent tilde-sparse $O(m+n)$ reductions, the gray edges represent sparse $O(n^2)$ reductions and the gray, squiggly edges represent tilde-sparse $O(n^2)$ reductions. The shaded region indicates problems hard for sub-mn computations under SETH.

7 Fine-grained Reductions for Directed Graphs

In Section 4, we gave an overview of our tilde-sparse $O(m+n\log n)$ reduction from directed 2-SiSP to the Radius problem. Here we give full details of this reduction along with the rest of our sparse reductions for directed graphs.

Both 2-SiSP and Radius are known to be subcubic equivalent to APSP [26, 1]. However in the absence of equivalence for sparse graphs, our aim is to refine the sub-mn partial order. A sparse $O(n^2)$ reduction from 2-SiSP to APSP was given in [8] and this led to a slightly faster k-SiSP algorithm using [15]. Our sparse reduction from 2-SiSP to Radius refines this result and the sub-mn partial order by plugging the Radius problem in the sparse reduction chain from 2-SiSP to APSP.

In our reduction, we construct a graph G'' where we first map every edge (v_j, v_{j+1}) lying on P to the vertices z_{j_o} and z_{j_i} such that the shortest path from z_{j_o} 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_o} and y_{j_i} in the graph and connect them to vertices z_{j_o} and z_{j_i} such that the longest shortest path from y_{j_o} is to the vertex y_{j_i}, which in turn corresponds to the shortest path from z_{j_o} to z_{j_i}. In our construction, we ensure that the center is among one of the vertices y_{j_o} and hence computing the Radius in the reduced graph gives the minimum among all the shortest paths from z_{j_o} to z_{j_i}. This corresponds to a shortest replacement path from s to t, which by definition is the second simple shortest path from s to t.

Lemma 7.1. *In weighted directed graphs, 2-SiSP* $\lesssim^{\textbf{sprs}}_{m+n}$ *Radius*

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 $(s=v_0 \to v_1 \rightsquigarrow v_{l-1} \to v_l = t)$ be the shortest path from s to t in G.

(i) *Constructing G'':* We first create the graph G', which contain G and l additional vertices z_0,z_1,...,z_{l-1}. We remove the edges lying on P from G'. For each $0 \le i \le l-1$, 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_o} and we place a directed edge of weight 0 from z_{j_i} to z_{j_o}, 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_o}) in G'.

Let M be the largest edge weight in G and let $M' = 9nM$. For each $0 \leq j \leq l-1$, we add additional vertices y_{j_i} and y_{j_o} and we place a directed edge of weight 0 from y_{j_o} to z_{j_o} 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 $y'_{j_o}s$ of weight 0 (M').

We also add edges of weight $\frac{2M'}{3}$ from y_{j_o} 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_o} to y_{k_i} (for each $k \neq j$) (similar to [1]). 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_o} 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.

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_o} to y_{k_i} (for each $k \neq j$) and the gadget does not add any new paths from y_{j_o} to y_{j_i}. The reason is that for every 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 3 depicts the full construction of G'' for $l = 3$.

(ii) *We now show that 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 nM. 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 $10nM$ as it includes the last edge (z_{j_i}, y_{j_i}) of weight $\frac{11}{9}M' = 11nM$. 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}.

(iii) *We now show that 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.

(iv) *We now show that 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 $\frac{5}{3}M' = 15nM$, 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 (ii), (iii) and (iv), 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.

The cost of this reduction is $O(m + n \log n)$. □

The above lemma also gives us a sparse reduction from the s-t replacement paths problem to the Eccentricities problem. We describe the statement formally in Lemma 7.2.

Lemma 7.2. *In weighted directed graphs, s-t replacement paths $\lesssim^{\boldsymbol{sprs}}_{m+n}$ Eccentricities*

There is a simple $O(m)$ sparse reduction from ANSC to APSP that works as follows. After computing APSP in the input graph G, for each edge (x, y) outgoing from x we compute $w(x, y) + d_G(y, x)$, and the minimum of these values gives the weight of a minimum weight cycle through x. We repeat this process for each vertex x in G to compute ANSC in $O(m)$ time after a single call to APSP. The same reduction works for reducing Min-Wt-Cycle to APSP.

We now describe a sparse reduction from directed Min-Wt-Cycle to 2-SiSP (a sub-cubic non-sparse reduction from Min-Wt-Δ to 2-SiSP is in [26].) Both Min-Wt-Cyc and 2-SiSP [8, 15] sparsely reduces to APSP under sparse $O(m + n)$ and sparse $O(n^2)$ reductions, respectively. However our reduction from Min-Wt-Cyc to 2-SiSP further refines the partial sub-mn order.

In our reduction we first create a path of length n with vertices labeled from p_0 to p_n. 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.

Lemma 7.3. *In weighted directed graphs, Min-Wt-Cycle $\leq^{\boldsymbol{sprs}}_{m+n}$ 2-SiSP*

Proof. To compute Min-Wt-Cycle in G, we first create the graph G', where we replace every vertex z by vertices z_i and z_o, and we place a directed edge of weight 0 from z_i to z_o, and we replace each incoming edge to (outgoing edge from) z with an incoming edge to z_i (outgoing edge from z_o). We also add a path P $(p_0 \to p_1 \rightsquigarrow p_{n-1} \to p_n)$ of length n and weight 0.

Let $Q = n \cdot W_{max}$, where W_{max} is the maximum weight of any edge in G. For each $1 \leq j \leq n$, we add an edge of weight $(n - j + 1)Q$ from p_{j-1} to j_o and an edge of weight jQ from j_i to p_j in G' to form G''. Figure 5 depicts the full construction of G'' for $n = 3$. This is an $(m + n)$ reduction, and it can be seen that the second simple shortest path from p_0 to p_n in G'' corresponds to a minimum weight cycle in G.

□

The problems k-SiSP and k-SiSC are shown to be equivalent to each other in [2]. We describe the statement formally in Lemma 7.4.

Lemma 7.4. *In directed graphs, k-SiSP $\equiv^{\boldsymbol{sprs}}_{m+n}$ k-SiSC*

We now establish the equivalence between ANSC and the s-t replacement paths problem under $(m+n)$-reductions by first showing 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 our reduction from Min-Wt-Cycle to 2-SiSP as described in Lemma 7.3.

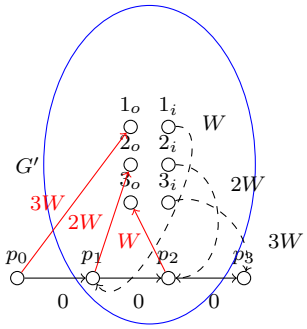


Figure 5: G'' for $n = 3$ in the reduction: directed Min-Wt-Cycle $\leq^{\boldsymbol{sprs}}_{m+n}$ 2-SiSP

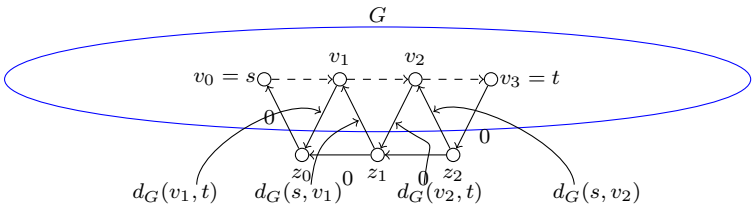


Figure 6: G' for $l = 3$ in the reduction: directed s-t replacement paths problem $\leq^{\boldsymbol{sprs}}_{m+n}$ ANSC

In the reduction from s-t replacement paths to ANSC, we map every edge (v_j, v_{j+1}) lying on the shortest path from s to t to a new vertex z_j such that the replacement path from s to t for the edge (v_j, v_{j+1}) correspond to the shortest cycle passing through z_j in the constructed graph. Thus computing ANSC in the reduced graph gives us the replacement path for every edge on the shortest path from s to t.

Our reduction from ANSC to s-t replacement paths is same as the one described in Lemma 7.3 and we observe that the replacement path for an edge (p_j, p_{j+1}) in the constructed graph correspond to the shortest cycle passing through j in the original graph. Hence computing s-t replacement paths in the constructed graph gives us the ANSC output in the original graph. This is described below in Lemma 7.5.

Lemma 7.5. *In weighted directed graphs, s-t replacement paths $\equiv^{\boldsymbol{sprs}}_{m+n}$ ANSC*

Proof. We are given an input graph $G = (V, E)$, a source vertex s and a sink vertex t and we wish to compute the replacement paths for all the edges lying on the shortest path from s to t. Let $P(s = v_0 \to v_1 \rightsquigarrow v_{l-1} \to v_l = t)$ be the shortest path from s to t in G.

(i) Constructing G': We first create the graph G', as described in the proof of Lemma 7.1. Figure 6 depicts the full construction of G' for $l = 3$.

(ii) We now show that for each $0 \leq i \leq l-1$, the replacement path from s to t for the edge (v_i, v_{i+1}) lying on P has weight equal to the shortest cycle passing through z_i. If not, assume that for some i $(0 \leq i \leq l-1)$, the weight of the replacement path from s to t for the edge (v_i, v_{i+1}) is not equal to the weight of the shortest cycle passing through z_i.

Let P_i $(s \rightsquigarrow v_j \rightsquigarrow v_k \rightsquigarrow t)$ (where v_j is the vertex where P_i 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_i, v_{i+1}) and let C_i $(z_i \rightsquigarrow z_p \to v_p \rightsquigarrow v_q \to z_q \rightsquigarrow z_i)$ be the shortest cycle passing through z_i in G'.

If $wt(P_i) < wt(C_i)$, then the cycle C'_i $(z_i \to z_{i-1} \rightsquigarrow z_j \to v_j \circ P_i(v_j, v_k) \circ v_k \to z_k \to z_{k-1} \rightsquigarrow z_i)$ (where $P_i(v_j, v_k)$ is the subpath of P_i from v_j to v_k) passing through z_i has weight equal to $wt(P_i) <$

$wt(C_i)$, resulting in a contradiction as C_i is the shortest cycle passing through z_i in G'.

Now if $wt(C_i) < wt(P_i)$, then the path P_i' ($s \rightsquigarrow v_p \circ C_i(v_p, v_q) \circ v_q \rightsquigarrow v_l$) where $C_i(v_p, v_q)$ is the subpath of C_i from v_p to v_q, is also a path from s to t avoiding the edge (v_i, v_{i+1}), and has weight equal to $wt(C_i) < wt(P_i)$, resulting in a contradiction as P_i is the shortest replacement path from s to t for the edge (v_i, v_{i+1}).

We then compute ANSC in G'. And by (ii), the shortest cycles for each of the vertices $z_0, z_1, \ldots, z_{l-1}$ gives us the replacement paths from s to t. This leads to an $(m+n)$ sparse reduction from s-t replacement paths problem to ANSC.

Now for the other direction, we are given an input graph $G = (V, E)$ and we wish to compute the ANSC in G. We first create the graph G'', as described in Lemma 7.3. We can see that the shortest path from p_0 to p_n avoiding edge (p_{j-1}, p_j) corresponds to a shortest cycle passing through j in G. This gives us an $(m+n)$-sparse reduction from ANSC to s-t replacement paths problem. □

We now give a sparse reduction from the 2-SiSP problem to Betweenness Centrality in Lemma 7.6. Our reduction is similar to the reduction from 2-SiSP to Radius, as described in Lemma 7.1.

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 7.6. *In weighted directed graphs, 2-SiSP* $\leq^{\boldsymbol{sprs}}_{m+n}$ *Betweenness Centrality*

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.

(*i*) *Constructing G'':* We first construct the graph G'', as described in the proof of Lemma 7.1, without the vertices A and B. For each $0 \leq j \leq l-1$, we change the weight of the edge from z_{j_i} to y_{j_i} to M' (where $M' = 9nM$ and M is the largest edge weight in G).

We add an additional vertex A and for each $0 \leq j \leq l-1$, we add an incoming (outgoing) edge from (to) y_{j_o} (y_{j_i}). We assign the weight of the edges from y_{j_o}'s to A as 0 and from A to y_{j_i}'s as $M' + q$ (for some q in the range 0 to nM).

Figure 7 depicts the full construction of G'' for $l = 3$.

We observe that for each $0 \leq j \leq l-1$, a shortest path from y_{j_o} to y_{j_i} with (z_{j_i}, y_{j_i}) as the last edge has weight equal to $M' + d_{G''}(z_{j_o}, z_{j_i})$.

(*ii*) *We now show that the Betweenness Centrality of A, i.e. $BC(A)$, is equal to l iff $q < d_{G''}(z_{j_o}, z_{j_i})$ for each $0 \leq j \leq l-1$.* The only paths that passes through the vertex A are from vertices y_{j_o}'s to vertices y_{j_i}'s. For $j \neq k$, as noted in the proof of Lemma 7.1, there exists some r, s such that there is a path from y_{j_o} to y_{k_i} that goes through $C_{r,s}$ and has weight equal to $\frac{2}{3}M'$. However a path from y_{j_o} to y_{k_i} has weight $M' + q$, which is strictly greater than $\frac{2}{3}M'$ and hence the pairs (y_{j_o}, y_{k_i}) does not contribute to the Betweenness Centrality of A.

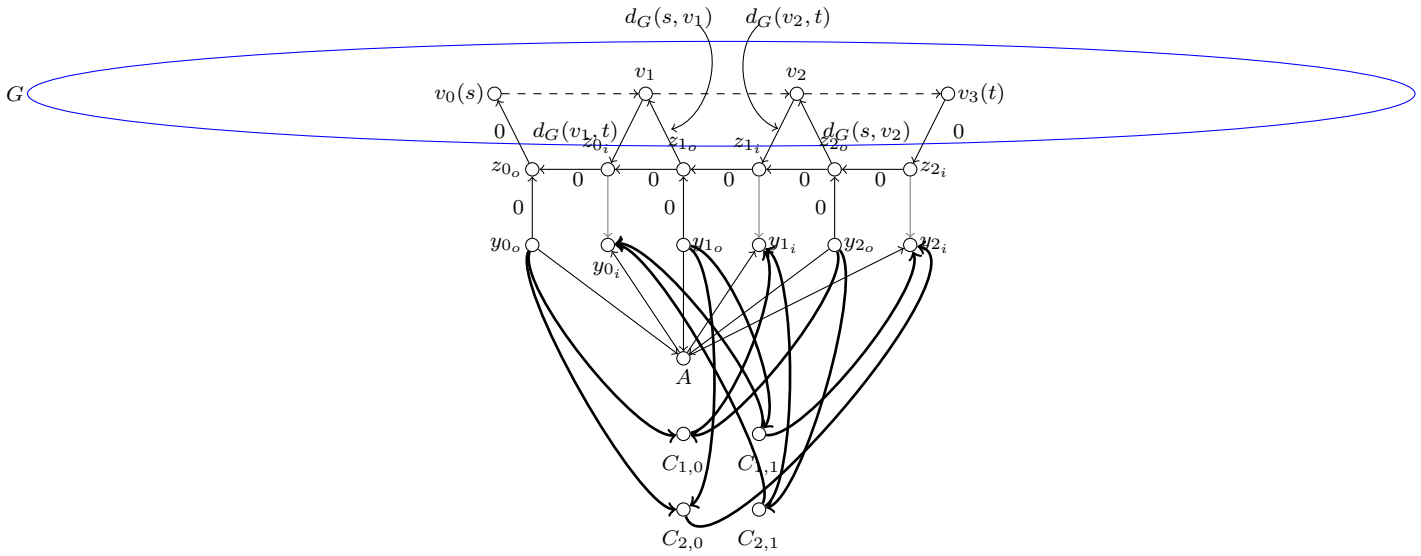


Figure 7: G'' for $l = 3$ in the reduction: directed 2-SiSP $\lesssim^{\boldsymbol{sprs}}_{m+n}$ Betweenness Centrality. The gray and the bold edges have weight M' and $\frac{1}{3}M'$ respectively. All the outgoing (incoming) edges from (to) A have weight $M' + q$ (0).

Now if $BC(A)$, is equal to l, it implies that the shortest paths for all pairs (y_{j_o}, y_{j_i}) passes through A and there is exactly one shortest path for each such pair. Hence for each $0 \leq j \leq l-1$, $M' + q < M' + d_{G''}(z_{j_o}, z_{j_i})$. Thus $q < d_{G''}(z_{j_o}, z_{j_i})$ for each $0 \leq j \leq l-1$.

On the other hand if $q < d_{G''}(z_{j_o}, z_{j_i})$ for each $0 \leq j \leq l-1$, then the path from y_{j_o} to y_{j_i} with (z_{j_i}, y_{j_i}) as the last edge has weight $M' + d_{G''}(z_{j_o}, z_{j_i})$. However the path from y_{j_o} to y_{j_i} passing through A has weight $M' + q < M' + d_{G''}(z_{j_o}, z_{j_i})$. Hence every such pair contributes 1 to the Betweenness Centrality of A and thus $BC(A) = l$.

Thus using (ii), we just need to find the minimum value of q such that $BC(A) < l$ in order to compute the value $\min_{0 \leq j \leq l-1} d_{G''}(z_{j_o}, z_{j_i})$. We can find such q by performing a binary search in the range 0 to nM and computing $BC(A)$ at every layer. Thus we make $O(\log nM)$ calls to the Betweenness Centrality algorithm.

As observed in the proof of Lemma 7.5, we know that 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. Thus by making $O(\log nM)$ calls to the Betweenness Centrality algorithm, we can compute the second simple shortest path from s to t in G. This completes the proof.

The cost of this reduction is $O((m + n \log n) \cdot \log nM)$.

□

We now describe a tilde-sparse reduction from the ANSC problem to the All Nodes Positive Betweenness Centrality problem. 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 requires the positive betweenness centrality values for n different nodes.

In our 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

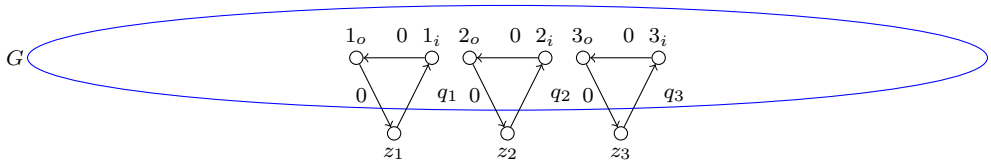


Figure 8: G' for $n = 3$ in the reduction: directed ANSC $\lesssim^{\textbf{sprs}}_{m+n}$ All Nodes Positive Betweenness Centrality.

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 calls to the Positive Betweenness Centrality problem, to compute the weight of the shortest cycle passing through x in the original graph. Our reduction is described in Lemma 7.7.

Lemma 7.7. *In weighted directed graphs, ANSC* $\lesssim^{\textbf{sprs}}_{m+n}$ *All Nodes Positive Betweenness Centrality*

Proof. We are given an input graph $G = (V, E)$ and we wish to compute the ANSC in G. Let M be the largest edge weight in G.

(i) *Constructing G':* Now we construct a graph G' from G. For each vertex $x \in V$, we replace x by vertices x_i and x_o and we place a directed edge of weight 0 from x_i to x_o, and we also replace each incoming edge to (outgoing edge from) x with an incoming edge to x_i (outgoing edge from x_o) in G'. We can observe that the shortest path from x_o to x_i in G' corresponds to the shortest cycle passing through x in G.

For each vertex $x \in V$, we add an additional vertex z_x in G' and we add an edge of weight 0 from x_o to z_x and an edge of weight q_x (where q_x lies in the range from 0 to nM) from z_x to x_i.

Figure 8 depicts the full construction of G' for $n = 3$.

We observe that the shortest path from x_o to x_i for some vertex $x \in V$ passes through z_x only if the shortest cycle passing through x in G has weight greater than q_x.

(ii) *We now show that for each vertex $x \in V$, Positive Betweenness Centrality of z_x, i.e., $BC(z_x) > 0$ iff the shortest cycle passing through x has weight greater than q_x.* It is easy to see that the only path that pass through vertex z_x is from x_o to x_i (as the only outgoing edge from x_i is to x_o and the only incoming edge to x_o is from x_i).

Now if $BC(z_x) > 0$, it implies that the shortest path from x_o to x_i passes through z_x and hence the path from x_o to x_i corresponding to the shortest cycle passing through x has weight greater than q_x.

On the other hand, if the shortest cycle passing through x has weight greater than q_x, then the shortest path from x_o to x_i passes through z_x. And hence $BC(z_x) > 0$.

Then using (ii), we just need to find the maximum value of q_x such that $BC(z_x) > 0$ in order to compute the weight of the shortest cycle passing through x in the original graph. We can find such q_x by performing a binary search in the range 0 to nM and computing Positive Betweenness Centrality for all nodes at every layer. Thus we make $O(\log nM)$ calls to the All-Nodes Positive Betweenness Centrality algorithm. This completes the proof.

The cost of this reduction is $O((m + n) \cdot \log nM)$. □

We also have tilde-sparse $O(m + n)$ equivalences between the Diameter problem, Betweenness Centrality (Positive/Approximate) and Reach Centrality [1]. We state this formally in Lemma 7.8.

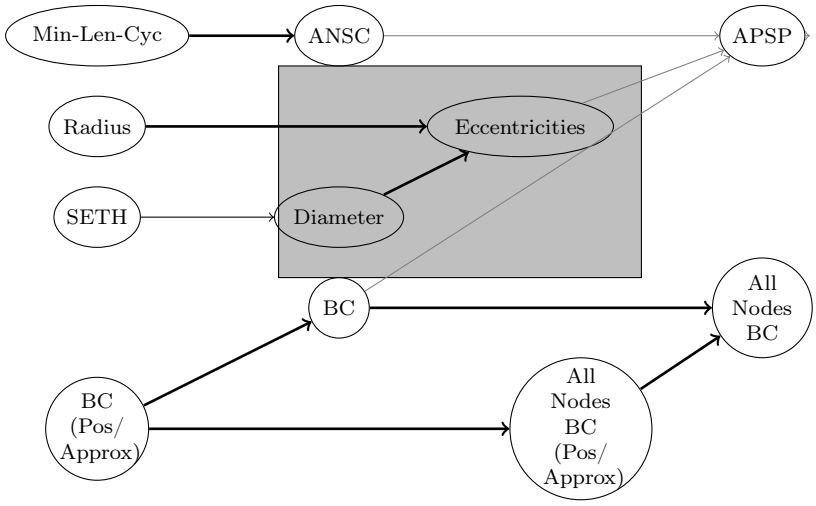


Figure 9: Chain of Reductions for Unweighted Directed Graphs. Here BC stands for Betweenness Centrality, Pos stands for Positive and Approx stands for Approximate. The bold edges represent sparse $O(m+n)$ reductions and the gray edges represent sparse $O(n^2)$ reductions. The shaded region indicates problems hard for sub-mn computations under SETH.

Lemma 7.8. *([1]) In weighted directed graphs, the following problems are tilde-sparse $O(m+n)$ equivalent: Diameter, Betweenness Centrality (Positive/Approximate) , and Reach Centrality.*

Unweighted Directed Graphs.

Our sparse reduction from ANSC to APSP and the sparse equivalence between k-SiSP and k-SiSC holds for the unweighted case as well. But our other sparse reductions from Min-Wt-Cycle to 2-SiSP, from 2-SiSP to Radius and Betweenness Centrality and from ANSC to All-Nodes Positive Betweenness Centrality does not work for unweighted graphs since they use very large weights on certain edges. Here, in fact, there is a randomized $\tilde{O}(k \cdot m\sqrt{n})$ time algorithm for unweighted k-SiSP [19], hence such a reduction from Min-Wt-Cycle to 2-SiSP would imply a significantly faster algorithm for a shortest cycle in a graph, a long-standing open question. Figure 9 gives an overview of the state of sparse reductions for unweighted directed graphs.

8 Algorithms for Shortest Cycles in Undirected Graphs

1. **ANSC in Unweighted Undirected Graphs.** In Section 6.2, we gave a tilde-sparse $O(n^2 \log n)$ reduction from ANSC to APSP$'$ in unweighted undirected graphs. To compute ANSC, our reduction constructs $2\lceil \log n \rceil$ graphs $G_{i,j}$ and makes one call to APSP$'$ for each of these graphs, with an additional $O(n^2 \log n)$ post processing time. Using the $O(n^\omega \log n)$ time algorithm for computing APSP$'$ in unweighted undirected graphs [21, 3], we obtain an $O(n^\omega \log^2 n)$ time algorithm for computing ANSC in unweighted undirected graphs, giving bound in Theorem 2.4(a). This improves on the best previous bound in Yuster [27]. That algorithm is randomized and runs in $\tilde{O}(n^{\frac{\omega+3}{2}})$ time.

Recently there has been some progress in the unweighted directed case as well [20]. Their algorithm runs in $\tilde{O}(n^\omega)$ time and is faster than the previous best known algorithm given by Yuster [27].

2. **k-SiSC in Undirected Graphs.** We obtain an $\tilde{O}(m+n)$ time algorithm for k-SiSC in weighted undirected graphs by giving a tilde-sparse $\tilde{O}(m+n)$ time reduction from k-SiSC to k-SiSP. This

reduction uses our bit-fixing method with $\lceil \log n \rceil$ different graphs. The reverse reduction from k-SiSP to k-SiSC is also sparse and is simpler, so we can show the tilde-equivalence of k-SiSP and k-SiSC in undirected graphs. We did not include this result in our partial order because both problems have near linear-time algorithms and are not in the mn partial order.

Lemma 8.1. *In undirected graphs, k-SiSC $\lesssim^{\boldsymbol{sprs}}_{(m+n)}$ k-SiSP and k-SiSP $\leq^{\boldsymbol{sprs}}_{(m+n)}$ k-SiSC*

Proof. Let the input be $G = (V, E)$ and let $x \in V$ be the vertex for which we need to compute k-SiSC. We assume that the vertices are labeled from 1 to n. We first show that k-SiSC in G can be computed with $\lceil \log n \rceil$ calls to k-SiSP. Let $\mathcal{N}(x)$ be the neighbor-set of x. We create $\lceil \log n \rceil$ graphs $G_i = (V_i, E_i)$ such that $\forall 1 \leq i \leq \lceil \log n \rceil$, G_i contains two additional vertices $x_{0,i}$ and $x_{1,i}$ (instead of the vertex x) and $\forall y \in \mathcal{N}(x)$, the edge $(y, x_{0,i}) \in E_i$ if y's i-th bit is 0, otherwise the edge $(y, x_{1,i}) \in E_i$. This takes $O((m + n) \cdot \log n)$ time and we observe that every cycle through x will appear as a path from $x_{0,i}$ to $x_{1,i}$ in at least one of the G_i. Hence, the k-th shortest path in the collection of k-SiSPs from $x_{0,i}$ to $x_{1,i}$ in $\log n$ G_i, $1 \leq i \leq \lceil \log n \rceil$ (after removing duplicates), corresponds to the k-th SiSC passing through x.

To compute k-SiSP from x to y in G, we reduce to k-SiSC by forming G', where we add a new vertex z and place edges of weight 1 from z to x and y. Then the k-th simple shortest cycle through z in G' corresponds to the k-th simple shortest path from x to y in G. □

Using the undirected k-SiSP algorithm in [10] that runs in $O(k \cdot (m + n \log n))$, we obtain an $O(k \log n \cdot (m + n \log n))$ time algorithm for k-SiSC in undirected graphs.

3. **k-ANSiSC in Undirected Graphs.** We can compute k-SiSC for every vertex $x \in V$ using the above algorithm to obtain an $O(kn \log n \cdot (m+n \log n))$ (i.e., $\tilde{O}(kmn)$) time algorithm for k-ANSiSC.

4. **k-All-SiSC in Undirected Graphs.**. Recently an algorithm for directed k-All-SiSC was given in [2] that runs in $\tilde{O}(kmn)$ time. Our k-All-SiSC algorithm for undirected graphs, which is adapted from the directed version in [2], is faster and works as follows. During an initialization phase, we find a shortest cycle passing through a vertex j, for each $j \in V$, in the graph G_j induced on the vertex set $\{v \geq j\}$. We store these cycles in an array $sc[1 \ldots n]$. (We use the graphs G_j to avoid generating duplicate cycles.) The shortest cycle among these cycles gives us a minimum weight cycle in G. To find the second shortest cycle in G, we generate the second SiSC passing through the vertex i in graph G_i, where i is the vertex with the minimum id in the previously generated cycle. We store this cycle in the i-th entry of the array sc.

Now assume that we have already generated $k - 1$ shortest cycles in the graph and we want to generate the k-th shortest cycle. Let p be the vertex with the minimum id on the $(k - 1)$-th cycle and let k_p be the number of cycles among these $k-1$ cycles that have p as the vertex with minimum id. We then generate the $(k_p + 1)$-th SiSC passing through p in G_p and we store it in the array sc at index p. The k-th shortest cycle is then the shortest cycle in this array.

The initialization cost is $O((m + n \log n) \cdot n)$ for n calls to Suurballe and Tarjan's algorithm [23, 6] for finding a shortest cycle passing through a specified vertex. Thereafter, we can generate each successive cycle by using our k-SiSC algorithm that runs in $O((m + n \log n) \cdot \log n)$ time, if we store the cycles that we have already generated in the array sc, and the data structures needed for computing successive simple shortest cycles through each vertex. This results in an $\tilde{O}(m \cdot (n + k))$ time algorithm for k-All-SiSC in undirected graphs.

9 Proofs for Section 5

We give details of the proofs of Lemma 5.1 and Lemma 5.2 in Section 5.

Proof of Lemma 5.1. Our reduction is similar to a result in [17]. Consider the undirected case first, so we are given an input graph $G = (V, E)$, and our goal is to determine if G has a k-dominating set.

Consider first the case when k is odd, so $k = 2r + 1$. To find a k-dominating set in G, we make n calls to graphs derived from the following instance $G' = (V', E')$ of the Diameter problem (G' is used in [17] where k is assumed to be even).

In G' the vertex set $V' = V_1 \cup V_2$, where V_1 contains a vertex for each subset of V of size r and $V_2 = V$. We add an edge from a vertex $v \in V_1$ to a vertex $x \in V_2$ if the subset corresponding to v does not dominate x. We induce a clique in the vertex partition V_2. As shown in [17] G' has diameter 3 if G has a dominating set of size $2r$ and has diameter 2 otherwise.

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 $V - V_x$. If G has a dominating set D of size k that includes vertex x then consider any partition of the remaining $2r$ vertices in D into two subsets of size r each, and let u and v be the vertices corresponding to these two sets in V_1. Since all paths from u to v in G_x pass through $V_2 - V_x$, there is no path of length 2 from u to v since every vertex in $V_2 - V_x$ is covered by either u or v. Hence the diameter of G_x is greater than 2 in this case. But if there is no dominating set of size k that includes x in G, then for any $u, v \in V_1$, at least one vertex in $V_2 - V_x$ is not covered by both u and v and hence there is a path of length 2 from u to v. If we now compute the diameter in each of graphs G_x, $x \in V$, we will detect a graph with diameter greater than 2 if and only if G has a dominating set of size k.

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 $O(n \cdot M^\alpha \cdot N^{2-\alpha-\epsilon}) = O(n^{2r+1-\epsilon r+\alpha})$. Then for $r \geq 1 + \frac{\alpha}{\epsilon}$ (i.e., $k \geq 3 + \frac{2\alpha}{\epsilon}$) the above algorithm runs in $O(n^{k-\epsilon})$ time. The case when k is even is simpler since we can work with just one graph G' (as in [17]). In the directed case, we get the same result by replacing every edge in G' with two directed edges in opposite directions. □

Proof of Lemma 5.2. (*a*) *If* $\alpha_2 + \beta_2 > \alpha_1 + \beta_1$, *then* we show that for any $\gamma < \frac{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)}{(\alpha_1-\alpha_2)}$ (if $\alpha_1 > \alpha_2$) and if $\alpha_1 \leq \alpha_2$, then for any $\gamma > 0$, $T_1(m, n)$ is a smaller sparse time bound than $T_2(m, n)$.

Let $m = n^{1+\gamma'}$ (where $\gamma' \leq \gamma$) and $\epsilon = \frac{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)+(\alpha_2-\alpha_1)\gamma}{1+\gamma}$. Then,

$$
\begin{aligned}
T_1(m,n) &= n^{\alpha_1+\beta_1+\alpha_1\gamma'} \\
&= \frac{n^{\alpha_2+\beta_2+\alpha_2\gamma'}}{n^{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)+(\alpha_2-\alpha_1)\gamma'}} \\
&= \frac{T_2(m,n)}{n^{\frac{((\alpha_2+\beta_2)-(\alpha_1+\beta_1)+(\alpha_2-\alpha_1)\gamma)(1+\gamma)}{(1+\gamma)}} \cdot n^{(\alpha_2-\alpha_1)(\gamma'-\gamma)}} \\
&\leq \frac{T_2(m,n)}{n^{\epsilon(1+\gamma)}} \quad (\text{Since } (\alpha_2-\alpha_1)(\gamma'-\gamma) > 0)
\end{aligned}
$$

$$\leq \frac{T_2(m,n)}{n^{\epsilon(1+\gamma')}} \quad (\text{Since } \gamma' < \gamma)$$
$$= O\left(\frac{T_2(m,n)}{m^{\epsilon}}\right)$$

And if $\alpha_1 \leq \alpha_2$, then for any $\gamma > 0$ and $\epsilon = \frac{(\alpha_2+\beta_2)-(\alpha_1+\beta_1)}{1+\gamma}$. Now it is easy to see that $T_1(m,n) = O(\frac{T_2(m,n)}{m^{\epsilon}})$.

(b) If $\alpha_2 + \beta_2 = \alpha_1 + \beta_1$, and $\alpha_2 > \alpha_1$, then we show that for any $\gamma > 0$ and for any constant $0 < \delta < \gamma$ and $\epsilon = \frac{(\alpha_2-\alpha_1)\delta}{1+\delta}$ (where $m = n^{1+\delta}$), $T_1(m,n)$ is a weakly smaller sparse time bound than $T_2(m,n)$.

Now,

$$\begin{aligned} T_1(m,n) &= n^{\alpha_1+\beta_1+\alpha_1\delta} \\ &= \frac{n^{\alpha_2+\beta_2+\alpha_2\delta}}{n^{(\alpha_2-\alpha_1)\delta}} \\ &= \frac{T_2(m,n)}{m^{\epsilon}} \end{aligned}$$

This establishes the proof. □

10 Definitions of Graph Problems

All Pairs Shortest Paths (APSP). Given a graph $G = (V, E)$, the APSP problem is to compute the shortest path distances for every pair of vertices in G.

The problem can be solved in $O(mn + n^2 \log\log n)$ time for weighted directed graphs [15], and in $O(mn)$ time using breadth first search for unweighted directed and undirected graphs. A slightly faster algorithm exists for weighted undirected graphs that runs in $O(mn \log \alpha(m,n))$ time [16].

APSP′. This is the problem of finding both the weights of the shortest paths as well as an $n \times n$ matrix $Last$, where $Last(x,y)$ contains the predecessor vertex of y on a shortest path from x to y. For each x, we assume that these predecessors represent a shortest path tree for x. All currently known APSP algorithms [21, 22, 28] can compute APSP′ as well.

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

Given the APSP output for the graph G, Min-Wt-Cycle can be computed in additional $O(m)$ time in a directed graph by computing, for each edge (x,y), the quantity $w(x,y) + d(y,x)$, which gives the weight of a minimum weight cycle through edge (x,y), and then choosing the minimum of the weights computed. By Definition 2.1, this is a sparse $O(m+n)$ reduction from Min-Wt-Cycle to APSP for directed graphs, but it does not work for undirected graphs since an edge (x,y) may itself be a shortest path from x to y. The challenges that arise in trying to reduce Min-Wt-Cycle to APSP in undirected graphs are discussed at length in [18, 27]. For sparse graphs, the fastest known algorithm for computing Min-Wt-Cycle in weighted undirected graphs uses n Dijkstra computations

and runs in $O((m+n\log n)\cdot n)$ time, and the algorithm for unweighted graphs runs in $O(mn)$ time with n BFS computations.

Roditty and Williams [18] gave an $\tilde{O}(M\cdot n^{\omega})$ time algorithm for computing Min-Wt-Cycle in directed graphs with bounded integer weights up to M using fast matrix multiplication. For undirected graphs with bounded positive integer weights up to M, they gave a $\tilde{O}(n^2)$ reduction from Min-Wt-Cycle to Min-Wt-Triangle, and since the latter problem can be computed in $\tilde{O}(M\cdot n^{\omega})$, the same bound as APSP, for graphs with bounded integer weights, they match the current fast matrix multiplication bound for APSP on undirected graphs. However, this reduction from Min-Wt-Cycle to APSP in [18] is not a sparse reduction. The graph in which Min-Wt-Triangle is computed has $O(n)$ vertices, but it has $\Theta(n^2)$ edges, regardless of the sparsity of the original graph.

In Section 3, we give an $\tilde{O}(n^2)$ sparse reduction from Min-Wt-Cycle to APSP for undirected graphs.

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.

In a directed graph the sparse $O(m+n)$ reduction from Min-Wt-Cycle to APSP gives a sparse reduction from ANSC to APSP, and ANSC can be solved in $O(APSP)$ time. For sparse directed graphs, the ANSC problem can be solved in $O((m+n\log n)\cdot n)$ time by making n calls to Suurballe and Tarjan's algorithm [23, 6] for finding a shortest cycle passing through a specified vertex.

A randomized subcubic reduction from ANSC to APSP for undirected graphs can be obtained through Yuster's algorithm [27, 25]. In Section 6.2 we present a deterministic sparse $\tilde{O}(n^2)$ reduction from the ANSC to APSP$'$ for undirected unweighted graphs (this is clearly a sub-cubic reduction as well). This also leads to a faster algorithm for this problem for dense graphs. The previous best algorithm of Yuster [27] is randomized. It uses fast matrix multiplication and runs in $\tilde{O}(n^{\frac{\omega+3}{2}})$ time, and it left the possibility of an $\tilde{O}(n^{\omega})$ algorithm or even a deterministic $\tilde{O}(n^{\frac{\omega+3}{2}})$ time algorithm as an open question.

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 lying on the shortest path from s to t, a shortest path from s to t avoiding the edge e.

For weighted directed graphs, the replacement paths problem can be solved in $O(mn+n^2\log\log n)$ time [8, 15]. A faster algorithm exists for unweighted directed graphs that takes $\tilde{O}(m\sqrt{n})$ time [19]. This problem can be solved in $\tilde{O}(m)$ time for undirected graphs [13].

A subcubic reduction from the Min-Wt-Δ problem is known for this problem [26] for weighted directed graphs. For sparse weighted directed graphs, we present a sparse $O(m+n)$ reduction from the Min-Wt-Cycle problem in Section 7.

k-SiSP and k-SiSC. Given a graph $G=(V,E)$ and a pair of vertices s,t, the k-SiSP problem is to find, the k shortest simple paths from s to t, such that the i-th path generated is different from all the previously generated $(i-1)$ paths and has weight greater than or equal to the weight of any of these $(i-1)$ paths.

The k-SiSP problem can be solved in $O(k\cdot(mn+n^2\log\log n))$ time using the 2-SiSP algorithm in [8, 15] for weighted directed graphs, and in $\tilde{O}(km\sqrt{n})$ time [19] for unweighted directed graphs. The problem can be solved more efficiently in undirected graphs and runs in $O(k(m+n))$ time [10].

In weighted directed graphs there is a sub-cubic reduction from Min-Wt-Δ to k-SiSP [26] but the resulting graph is dense. In Section 7, we describe a sparse $O(m+n)$ reduction from Min-Wt-Cycle to this problem that preserves sparsity.

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. For weighted directed graphs sparse equivalence between k-SiSP and k-SiSC is shown in [2].

k-ANSiSC. Given a graph $G=(V,E)$, the k-ANSiSC problem is to find the k shortest simple cycles through each vertex in G.

In directed graphs, the k-ANSiSC problem can be solved in $O(k \cdot n \cdot (mn + n^2 \log\log n))$ time [2].

k-All-SiSC. Given a graph $G=(V,E)$, the k-All-SiSC problem is to find the k shortest simple cycles in G.

For directed graphs, the k-All-SiSC problem can be solved in $O(k \cdot (mn + n^2 \log\log n))$ time with an additional $O(mn + n^2 \log n)$ startup cost [2].

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.

The only known algorithm for computing Radius involves computing APSP and then finding the radius in additional $O(n^2)$ time. There is a subcubic reduction from Min-Wt-Δ problem to this problem for both weighted directed and undirected graphs as shown in [1]. For weighted directed graphs, we give a sparse $O(m+n\log n)$ reduction from the 2-SiSP problem to Radius in Section 4.

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

The current best algorithm for computing Diameter involves computing APSP and then computing the diameter in additional $O(n^2)$ time. Diameter is not known to be sub-cubic equivalent to APSP. In [17], it was shown that an $O(m^{2-\epsilon})$ (for any constant $\epsilon > 0$) time algorithm for computing Diameter will refute SETH. In Section 5, we further analyze that any sub-mn time algorithm for computing Diameter will also refute SETH.

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 $x \in V$. Thus the Radius problem is equivalent to finding minimum of all the eccentricities and the Diameter is equivalent to finding maximum of all the eccentricities.

The problem can be solved by computing APSP with additional $O(n^2)$ post-processing time. A subcubic reduction from Min-Wt-Δ problem for both weighted directed and undirected graphs follows from the subcubic reduction from Min-Wt-Δ to Radius as shown in [1]. For weighted directed graphs, we describe a tilde-sparse $O(m+n)$ reduction from the ANSC problem in Section 4.

Betweenness Centrality. Betweenness Centrality of a node is an indicator of its centrality in the network. For a given graph $G = (V, E)$ and a node $v \in V$, the Betweenness Centrality of 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. In other words, Betweenness Centrality of v is the fraction of shortest paths passing through v.

As in [1] we assume that the graph has unique shortest paths and hence the Betweenness Centrality of a node is simply the number of s, t pairs such that the shortest path from s to t passes through v. This can be solved by first computing APSP and then taking additional $O(n^2)$ post processing time for checking al s, t pairs.

For both weighted directed and undirected graphs, a subcubic reduction from Min-Wt-Δ problem is known for this version of the problem [1]. In Section 7 we however describe a tilde-sparse $O(m+n)$ reduction from 2-SiSP to Between Centrality in weighted directed graphs. In Section 5, we also observe that any sub-mn time algorithm for computing Betweenness Centrality in both weighted directed and undirected graphs will refute SETH.

The all-nodes version of this problem is called as *All Nodes Betweenness Centrality*. *Positive Betweenness Centrality* of a node v is the problem of checking whether $BC(v) > 0$ and its corresponding all-nodes version is known as *All-Nodes Positive Betweenness Centrality*. The problem of computing an α-approximation of the Betweenness Centrality is called as *α-Approximate Betweenness Centrality* and *All-Nodes α-Approximate Betweenness Centrality* is its all-nodes version.

For weighted graphs, both directed and undirected, the Positive and Approximate Betweenness Centrality problems are known to be tilde-sparse $O(m+n)$ equivalent to Diameter [1]. For weighted directed graphs, we give a tilde-sparse $O(m + n)$ reduction from ANSC to All Nodes Positive Betweenness Centrality problem (Section 7).

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

The current best known approach for solving Reach Centrality is via computing APSP with $O(n^2)$ post-processing time. For both weighted directed and undirected graphs, the Reach Centrality problem is tilde-sparse $O(m + n)$ equivalent to Diameter [1].

Acknowledgements. We thank Virginia Vassilevska Williams for helpful discussions.

References

[1] A. Abboud, F. Grandoni, and V. V. Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proc. SODA*, pages 1681–1697, 2015.

[2] U. Agarwal and V. Ramachandran. Finding k simple shortest paths and cycles. In *Proc. ISAAC*, 2016. To appear.

[3] N. Alon, Z. Galil, O. Margalit, and M. Naor. Witnesses for boolean matrix multiplication and for shortest paths. In *Proc. IEEE FOCS*, pages 417–426. IEEE, 1992.

[4] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proc. STOC*, pages 51–58. ACM, 2015.

[5] M. L. Carmosino, J. Gao, R. Impagliazzo, I. Mihajlin, R. Paturi, and S. Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. ITCS*, pages 261–270. ACM, 2016.

[6] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

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

[8] Z. Gotthilf and M. Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Inf. Proc. Lett.*, 109(7):352–355, 2009.

[9] A. Gronlund and S. Pettie. Threesomes, degenerates, and love triangles. In *Proc. IEEE FOCS*, pages 621–630. IEEE, 2014.

[10] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12(4):411–427, 1982.

[11] T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3sum conjecture. In *Proc. SODA*, pages 1272–1287. SIAM, 2016.

[12] A. Lingas and E.-M. Lundell. Efficient approximation algorithms for shortest cycles in undirected graphs. *Inf. Proc. Lett.*, 109(10):493–498, 2009.

[13] K. Malik, A. K. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8(4):223–227, 1989.

[14] M. Pătraşcu and R. Williams. On the possibility of faster sat algorithms. In *Proc. SODA*, pages 1065–1075. SIAM, 2010.

[15] S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312(1):47–74, 2004.

[16] S. Pettie and V. Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM Jour. Comput.*, 34(6):1398–1431, 2005.

[17] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. STOC*, pages 515–524. ACM, 2013.

[18] L. Roditty and V. V. Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *Proc. IEEE FOCS*, pages 180–189. IEEE, 2011.

[19] L. Roditty and U. Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Trans. Alg. (TALG)*, 8(4):33, 2012.

[20] P. Sankowski and K. Węgrzycki. Improved distance queries and cycle counting by frobenius normal form. *arXiv preprint arXiv:1611.03789*, 2016.

[21] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Jour. Comput. Sys. Sci.*, 51(3):400–403, 1995.

[22] A. Shoshan and U. Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Proc. IEEE FOCS*, pages 605–614. IEEE, 1999.

[23] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

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

[25] V. V. Williams. Personal communication, November 2016.

[26] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. IEEE FOCS*, pages 645–654. IEEE, 2010.

[27] R. Yuster. A shortest cycle for each vertex of a graph. *Inf. Proc. Lett.*, 111(21):1057–1061, 2011.

[28] U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *JACM*, 49(3):289–317, 2002.