Subcubic Equivalences between Graph Centrality Problems, APSP, and Diameter

AMIR ABBOUD, IBM Almaden Research Center, USA
FABRIZIO GRANDONI, IDSIA, USI-SUPSI, Switzerland
VIRGINIA VASSILEVSKA WILLIAMS, MIT, USA

3

Measuring the importance of a node in a network is a major goal in the analysis of social networks, biological systems, transportation networks, and so forth. Different *centrality* measures have been proposed to capture the notion of node importance. For example, the *center* of a graph is a node that minimizes the maximum distance to any other node (the latter distance is the *radius* of the graph). The *median* of a graph is a node that minimizes the sum of the distances to all other nodes. Informally, the *betweenness centrality* of a node w measures the fraction of shortest paths that have w as an intermediate node. Finally, the *reach centrality* of a node w is the smallest distance r such that any s-t shortest path passing through w has either s or t in the ball of radius r around w.

The fastest known algorithms to compute the center and the median of a graph and to compute the betweenness or reach centrality even of a single node take roughly cubic time in the number n of nodes in the input graph. It is open whether these problems admit truly subcubic algorithms, i.e., algorithms with running time $\tilde{O}(n^{3-\delta})$ for some constant $\delta > 0$.[1]

We relate the complexity of the mentioned centrality problems to two classical problems for which no truly subcubic algorithm is known, namely All Pairs Shortest Paths (APSP) and Diameter. We show that Radius, Median, and Betweenness Centrality are *equivalent under subcubic reductions* to APSP, i.e., that a truly subcubic algorithm for any of these problems implies a truly subcubic algorithm for all of them. We then show that Reach Centrality is equivalent to Diameter under subcubic reductions. The same holds for the problem of approximating Betweenness Centrality within any finite factor. Thus, the latter two centrality problems could potentially be solved in truly subcubic time, even if APSP required essentially cubic time.

On the positive side, our reductions for Reach Centrality imply an improved $\tilde{O}(Mn^{\omega})$-time algorithm for this problem in case of non-negative integer weights upper bounded by M, where ω is a fast matrix multiplication exponent.

CCS Concepts: • **Theory of computation → Graph algorithms analysis;**

Additional Key Words and Phrases: Fine-grained complexity, subcubic reductions, APSP, Radius, Median, Diameter, Betweenness Centrality, reach centrality

[1] The $\tilde{O}$ notation suppresses poly-logarithmic factors in n and M.

A preliminary version of this article appeared in SODA 2015. This version corrects some subtle technical bugs.
Fabrizio Grandoni is partially supported by the SNSF Excellence Grant 200020B_182865/1 and the SNSF Grant 200021_200731/1. Virginia Vassilevska Williams is supported by an NSF CAREER Award, NSF Grants CCF-1528078 and CCF-1514339, a BSF Grant BSF:2012338, a Sloan Research Fellowship and a Google faculty fellowship.
Authors' addresses: A. Abboud, IBM Almaden Research Center; email: abboud@cs.stanford.edu; F. Grandoni, IDSIA, USI-SUPSI, Switzerland; email: fabrizio@idsia.ch; V. V. Williams, MIT; email: virgi@mit.edu.

1549-6325/2023/02-ART3 $15.00
https://doi.org/10.1145/3563393

ACM Reference format:
Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. 2023. Subcubic Equivalences between Graph Centrality Problems, APSP, and Diameter. *ACM Trans. Algorithms* 19, 1, Article 3 (February 2023), 30 pages.
https://doi.org/10.1145/3563393

1 INTRODUCTION

Identifying the importance of nodes in networks is a major goal in the analysis of social networks (e.g., citation networks, recommendation networks, or friendship circles), biological systems (e.g., protein interaction networks), computer networks (e.g., the Internet or peer-to-peer networks), transportation networks (e.g., public transportation or road networks), and so forth. A variety of graph theoretic notions of node importance have been proposed; among the most relevant ones are betweenness centrality [25], graph centrality [36], closeness centrality [54], and reach centrality [35].

The *graph centrality* of a node w is the inverse of its maximum distance to any other node. The *closeness centrality* of w is the inverse of the total distance of w to all the other nodes. The *reach centrality* of w is the maximum distance between w and the closest endpoint of any s-t shortest path passing through w. Informally, the *betweenness centrality* of w measures the fraction of shortest paths having w as an intermediate node.

In this article we study four fundamental graph centrality computational problems associated with the mentioned centrality measures. Let $G = (V, E)$ be an n-node m-edge (directed or undirected) graph, with integer edge weights $w : E \rightarrow \{0, \ldots, M\}$ for some $M \geq 1$. Though we focus here on non-negative weights, part of our results can be extended to the case of directed graphs with possibly negative weights and no negative cycles. Let $d_G(s, t)$ denote the distance from node s to node t, and let us use $d(s, t)$ instead when G is clear from the context.

- The *Radius* problem is to compute $R^* := \min_{r^* \in V} \max_{v \in V} d(r^*, v)$ (*radius* of the graph).
- The *Median* problem is to compute $Med := \min_{m^* \in V} \sum_{v \in V} d(m^*, v)$.
- The *Reach Centrality* problem (for a given node b) is to compute

$$RC(b) = \max_{\substack{s,t \in V: \\ d(s,t) = d(s,b) + d(b,t)}} \{\min\{d(s, b), d(b, t)\}\}.$$

- The *Betweenness Centrality* problem (for a given node b) is to compute the number $BC(b)$ of shortest paths that have b as an intermediate node.[2]

All of these notions are related in one way or another to shortest paths. In particular, we can solve the first three problems by running an algorithm for the classical **All-Pairs Shortest Paths (APSP)** problem on the underlying graph and doing a negligible amount of post-processing. The same holds for Betweenness Centrality by assuming that shortest paths are unique by a simple algorithm. This was recently extended to the case of (possibly) non-unique shortest paths in unweighted graphs [12]. Part of our results for Betweenness Centrality assume the uniqueness of shortest paths. Using the best known algorithms for APSP [61], this leads to a slightly subcubic (by an $n^{o(1)}$ factor) running time for the considered problems, and no faster algorithm is known.

Each of these problems, however, only asks for the computation of a single number. It is natural to ask, is solving APSP necessary? Could it be that these problems admit much more efficient solutions? In particular, do they admit a *truly subcubic*[3] algorithm?

[2]Another slightly different definition of the problem is used in the literature; this is discussed later.
[3]We recall that a *truly subcubic* algorithm is an algorithm with running time $\tilde{O}(n^{3-\delta})$ for some constant $\delta > 0$.

Besides the fundamental interest in understanding the relations between such basic computational problems (can Radius be solved truly faster than APSP?), these questions are well motivated from a practical viewpoint. As evidence to the necessity of faster algorithms for the mentioned centrality problems, we remark that some papers presenting algorithms for Betweenness Centrality [8] and Median [37] have received more than 1,000 citations each.

1.1 Approach

The techniques of this article fall within the realm of *fine-grained complexity* (see [58] for a survey on the topic). A refinement of NP-hardness, the fine-grained approach strives to prove, via "fine-grained" *reductions*, that improving on a given upper bound for a computational problem B would yield breakthrough algorithms for many other famous and well-studied problems. At a high level, the idea is to consider two problems A and B for which the fastest known algorithms have running times $O(a(n))$ and $O(b(n))$ (here n is a size parameter such as the number of nodes in a graph), respectively. Typically A is a problem that is widely believed to need $a(n)^{1-o(1)}$ time. The approach then uses special reductions to transform an instance of A to instances of B, so that if there were an algorithm for B with running time $O(b(n)^{1-\varepsilon})$ for some $\varepsilon > 0$, then composing this algorithm with the reduction would yield an algorithm for A running in time $O(a(n)^{1-\delta})$ for $\delta > 0$. Since A is widely believed to not have such an algorithm, this can be used as evidence that a $O(b(n)^{1-\varepsilon})$ time algorithm for problem B is unlikely to exist (or at least very hard to find). When $a(n) = b(n) = n^3$, a reduction of the above kind is called a *subcubic reduction* [64] from A to B. We say that two problems A and B are *equivalent under subcubic reductions* if there exists a subcubic reduction from A to B and from B to A. In other terms, a truly subcubic time algorithm for one problem implies a truly subcubic time algorithm for the other and vice versa.

In this article we will also consider randomized reductions of the above type. In more detail, there exists a Monte-Carlo subcubic reduction with success probability p from A to B if, given a truly subcubic algorithm for B, we can solve A in truly subcubic time and the answer is correct with probability at least p. If $p \geq 1 - 1/n^{O(1)}$, the above Monte-Carlo reduction is a *high-probability* one. Equivalence under such Monte-Carlo reductions is defined similarly.

Vassilevska Williams and Williams [64] introduced this approach to the realm of graph algorithms to show the subcubic equivalence between APSP and a list of seven other problems, including deciding if an edge-weighted graph has a triangle with negative total weight (*Negative Triangle*), deciding if a given matrix defines a metric, and the *Replacement Paths* problem [33, 34, 53, 59, 62]. Other examples of this approach [1, 3, 48] include the famous results on 3-SUM hardness starting with the work of Gajentaan and Overmars [26]. More recently, the fine-grained approach has gained popularity. The main prototypical hard problems used are CNF-SAT, APSP, and 3SUM, but also some others such as k-Clique and more. Many incredibly diverse problems are now known to have fine-grained reductions from these prototypical hard problems. See the survey by Vassilevska Williams [58].

In this article we exploit both APSP and Diameter as our prototypical problem and prove a collection of subcubic equivalences with the above graph centrality problems. Recall that the Diameter problem is to compute the largest distance in the graph. There is a trivial subcubic reduction from Diameter to APSP, and although no truly subcubic algorithm is known for Diameter, finding a reduction in the opposite direction is one of the big open questions in this area: can we compute the largest distance faster than we can compute all the distances?

1.2 Subcubic Equivalences with APSP

Our first main result is to show that Radius, Median, and Betweenness Centrality are *equivalent* to APSP under subcubic reductions. Therefore, we add three relevant problems to the list of

APSP-hard problems [64], and if *any* of these problems can be solved in truly subcubic time, then *all* of them can.

Theorem 1.1. *Radius is equivalent to APSP under subcubic reductions.*

Theorem 1.2. *Median is equivalent to APSP under subcubic reductions.*

Theorem 1.3. *Betweenness Centrality (with unique shortest paths) is equivalent to APSP under high-probability Monte-Carlo subcubic reductions.*

We remark that, in the proof of Theorem 1.3, randomization is used only to guarantee the uniqueness of shortest paths in the reduction from APSP to Betweenness Centrality. In particular, dropping the uniqueness requirement, the same reduction would be deterministic. However, the converse reduction would not work as we mentioned earlier since the number of alternative shortest paths could be exponentially large.

Unfortunately, this is strong evidence that a truly subcubic algorithm for computing these centrality measures is unlikely to exist (or at least is very hard to find) since it would imply a huge and unexpected algorithmic breakthrough.

We find the APSP-hardness result for Radius quite interesting since, prior to our work, there was no good reason to believe that Radius might be a truly harder problem than Diameter. Indeed, in terms of approximation algorithms, any known algorithm to approximate the diameter can be converted to also approximate the radius in undirected graphs within the same factor [4, 7, 14, 52]. Furthermore, the exact algorithms for Diameter and Radius in graphs with small integer weights are also extremely similar [17]. The same holds for the lower bounds on fast approximation algorithms for Radius and Diameter in sparse graphs [2, 52].

1.3 Subcubic Equivalence between Reach Centrality and Diameter

Our second main result is to show that Reach Centrality and Diameter are equivalent under subcubic reductions.

Theorem 1.4. *Diameter and Reach Centrality are equivalent under subcubic reductions.*

On the positive side, it is within the realm of possibility that Diameter is a truly easier problem than APSP, which would imply the same for Reach Centrality. On the negative side, Theorem 1.4 shows that finding a subcubic algorithm for Reach Centrality is as hard as finding a subcubic algorithm for Diameter—a big open problem.

As a consequence of the tightness of our reductions, namely not only the number of nodes but also the largest absolute weight is roughly preserved, we also obtain a faster algorithm for Reach Centrality in directed graphs with small integer weights.

Theorem 1.5. *There exists an $\tilde{O}(Mn^{\omega})$ time algorithm for Reach Centrality in directed graphs.*

Above $\omega \in [2, 2.373)$ [16, 19, 27, 28, 63] denotes fast matrix multiplication exponent. The previous best algorithm for small integer weights, which is based on the solution of APSP, takes time $\tilde{O}(M^{0.752}n^{2.529})$ [66].

1.4 Approximation Algorithms

An approximate value of the mentioned graph centrality measures might be sufficiently good in practice. This is indeed the topic of several empirical works on Betweenness Centrality [6, 9, 29]. Furthermore, there are practically fast shortest paths algorithms based on reach centrality [30, 31, 35]: these algorithms can be adapted to work with approximate values of the reach centrality as well. In this article we formally study the approximability of the mentioned problems.

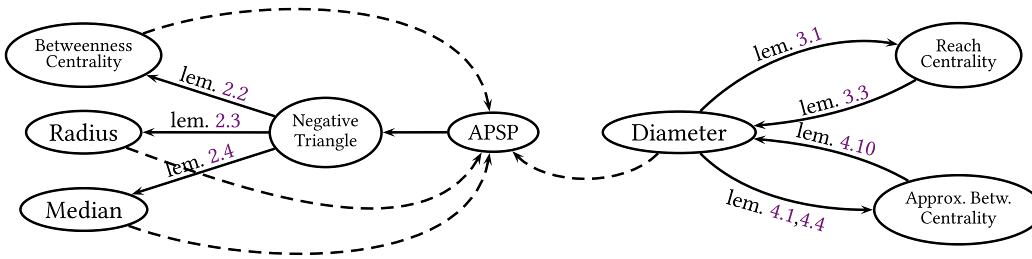

Fig. 1. The main subcubic reductions considered in this article. Dashed arrows correspond to trivial reductions. All the remaining reductions are given in this article, excluding the one from APSP to Negative Triangle, which is taken from [64].

In more detail, given a quantity X (e.g., a graph centrality measure), an α-approximation algorithm computes a quantity x such that $\frac{1}{\alpha}X \leq x \leq \alpha X$ for some $\alpha \geq 1$ (α is the approximation factor). A **polynomial-time approximation scheme (PTAS)** for a given measure X is an algorithm that, given an input parameter $\varepsilon > 0$, computes a $1 + \varepsilon$ approximate solution x in the above sense. Furthermore, the running time is polynomial for every fixed constant $\varepsilon > 0$. Our high-level goal is to design fast α-approximation algorithms with α as close to 1 as possible. It is known how to solve APSP within a multiplicative error $(1 + \varepsilon)$ in time $\tilde{O}(n^{\omega})$ for any constant ε [65]. This provides truly subcubic $(1 + \varepsilon)$ approximation algorithms for Radius and Median. However, this approach does not help with Reach/Betweenness Centrality, since in those measures *almost* shortest paths are irrelevant. Here we present some negative and (conditionally) positive results on the approximability of the latter two problems.

We define the *Approximate Betweenness Centrality* problem as the problem of computing an α-approximation of $BC(b)$ for some finite $\alpha > 0$. The *Approximate Reach Centrality* problem is defined analogously. We present reductions from Approximate Reach/Betweenness Centrality to the following *Positive Betweenness Centrality* problem: determine whether there exists some shortest path using b as an intermediate node. To the best of our knowledge, the latter problem was not studied before and it might be of independent interest. We show that Positive Betweenness Centrality is equivalent to Diameter (under subcubic reductions), while the corresponding *All-Nodes* version (where we solve the problem for all possible b) is equivalent to APSP! This explains why it has been difficult to develop approximation algorithms for Betweenness Centrality and Reach Centrality that are at the same time fast and *provably* accurate.

On the positive side, we show that a truly subcubic algorithm for Diameter implies a truly subcubic Monte-Carlo PTAS for Betweenness Centrality. Analogously to the case of Reach Centrality, this gives some more hope that a truly subcubic PTAS for Betweenness Centrality exists; however, such algorithm is probably not easy to find. Part of the mentioned reductions are summarized in Figure 1.

1.5 SETH Hardness

We consider the problem of solving Approximate Reach/Betweenness Centrality in sparse graphs. Here we can prove, again passing through Positive Betweenness Centrality, that $O(m^{2-\varepsilon})$ time algorithms do not exist unless the **Strong Exponential Time Hypothesis (SETH)** fails. Our reduction can be adapted to the stronger **Orthogonal Vector Conjecture (OVC)**.

1.6 Related Work

APSP is among the best-studied problems in Computer Science. If the edge weights are non-negative, one can run Dijkstra's algorithm [21] from every source node and solve the problem in time $O(mn + n^2 \log n)$ (by implementing Dijkstra's algorithm with Fibonacci heaps [24]). Johnson [43] showed how to obtain the same running time in the case of negative weights also (but no negative cycles). Pettie [49] improved the running time to $O(mn + n^2 \log\log n)$ and together with Ramachandran to $O(mn \log \alpha(m, n))$ [50]. If the graph is undirected and the edge weights are integers fitting in a word, one can solve the problem in time $O(mn)$ in the word-RAM model [57]. In dense graphs the running time of these algorithms is $O(n^3)$. Slightly subcubic algorithms were developed as well, starting with the work of Fredman [23]. Following a long sequence of improvements (among others, [11, 38]), Williams [61] obtained an algorithm with running time $\tilde{O}(n^3/2^{\Omega(\sqrt{\log n})})$. Faster algorithms are known for small integer weights bounded in absolute value by M: in undirected graphs APSP can be solved in $\tilde{O}(Mn^{\omega})$ time [56] and in directed graphs in $\tilde{O}(n^2(Mn)^{\frac{1}{4-\omega}})$ time [66]. The result for the directed case can be refined to $\tilde{O}(M^{0.752}n^{2.529})$ using fast rectangular matrix multiplication [39].

As we already mentioned, for general edge-weights the fastest known algorithms for Diameter and Radius solve APSP (hence taking roughly cubic time). In the case of directed graphs with small integer weights bounded by M there are faster, $\tilde{O}(Mn^{\omega})$ time algorithms (see [17] and the references therein). Faster approximation algorithms are known. Aingworth et al. [4] showed how to compute a (roughly) 3/2 approximation of the diameter in time $\tilde{O}(m\sqrt{n} + n^2)$. The same approximation factor and running time can be achieved for Radius in undirected graphs [7]. The running time for both Radius and Diameter was reduced to $\tilde{O}(m\sqrt{n})$ by Roditty and Vassilevska Williams [52] (see also [14] for a refinement of the approximation factor). The authors also show that a $3/2 - \varepsilon$ approximation for Diameter running in time $O(m^{2-\varepsilon})$ (for any constant $\varepsilon > 0$) would imply that the SETH of [40] fails, thus showing that improving on the 3/2-approximation factor while still using a fast algorithm would be difficult. A similar hardness result for Radius was shown in [2] under the Hitting Set Conjecture. Under SETH, there is no better than 5/3 approximation for Diameter in time $O(m^{3/2-\varepsilon})$ [5]. See also [10] for related results on Diameter and Radius. Upper and lower bounds on fast approximation algorithms to compute the Eccentricity of all nodes are given in [2, 5, 10, 14]. Some more recent fine-grained complexity results on the fast approximability of Diameter are given in [18].

The notion of betweenness centrality was introduced by Freeman in the context of social networks [25] and since then became one of the most important graph centrality measures in the applications. For example, this notion is used in the analysis of protein networks [20, 42], social networks [47, 51], sexual networks [45], and terrorist networks [15, 44]. From an algorithmic point of view, betweenness centrality was used to identify a highway-node hierarchy for routing in road networks [55]. Brandes's algorithm [8] computes the betweenness centrality of all nodes in time $O(mn + n^2 \log n)$. This result is based on a counting variant of Dijkstra's algorithm. We remark that [8], similarly to other papers in the area, neglects the bit complexity of the counters that store the number of pairwise shortest paths. This is reasonable in practice since the maximum number N of alternative shortest paths between two nodes tends to be small in many of the applications. By considering also N, the running time grows by a factor of $O(\log N) = O(n \log n)$. Indeed, in some applications one can even assume that shortest paths are unique (as we do in some parts of this article). The uniqueness of shortest paths is either a consequence of tie-breaking rules (*Canonical-Path Betweenness Centrality* problem [29]) or can be enforced by perturbing edge weights [30]. Chan et al. [12] obtain an $\tilde{O}(n^3)$ time algorithm for the case of non-unique shortest paths in unweighted graphs. The running time to compute the exact betweenness centrality can be prohibitive in practice for very large networks even assuming the uniqueness of shortest paths. For this reason, some work was devoted to the fast approximation of the betweenness centrality of all nodes [6, 9, 29]. Those works are based on random pivot-sampling techniques. They do not provide any theoretical bound on the approximation factor: this is not surprising a posteriori, in view of our APSP-hardness results. In contrast, our results suggest a candidate way to obtain a provably fast and accurate algorithm for Approximate Betweenness Centrality (for a single node). Our approach deviates substantially from [6, 9, 29] for small values of the betweenness centrality.

The Reach Centrality notion was introduced by Gutman [35] in the framework of practically fast algorithms to solve the Single-source Shortest Paths problem. In particular, the values $RC(b)$ can be used to filter out some nodes during an execution of Dijkstra's algorithm. The notion of Reach Centrality is also used in other works on the same topic [30, 31].

Eppstein and Wang [22] consider the problem of approximating the closeness centrality of all nodes. They present a random-sampling-based $O((m + n \log n)\frac{\log n}{\varepsilon^2})$ time algorithm that w.h.p. computes estimates within an additive error εD^*, where D^* is the diameter of the graph. The same problem is investigated in [9] from an experimental point of view. The Median problem was also studied in a distance-oracle query model [13, 32, 41].

1.7 Preliminaries and Notation

W.l.o.g. we assume that the considered graph $G = (V, E)$ is connected, hence $m \geq n - 1$. We make the usual assumption that the nodes of the considered graph are labeled with integers between 0 and $n - 1$, and where needed we implicitly assume that n is lower bounded by a sufficiently large constant. For two nodes $u, v \in V$, by uv we indicate either an undirected edge between u and v or an edge directed from u to v. The interpretation will be clear from the context.

For a given node $w \in V$, we let $Rad(w) := \max_{v \in V}\{d(w, v)\}$ (*eccentricity* of w) and $Med(w) := \sum_{v \in V} d(w, v)$. A node w minimizing $Rad(w)$ and $Med(w)$ is a *center* and a *median* of the graph, respectively. By $BC_{s,t}(b)$ we denote the number of shortest s-t paths that have b as an internal node. In particular, $BC_{s,s}(b) = BC_{s,b}(b) = BC_{b,t}(b) = 0$. Furthermore, $BC_{s,t}(b) \in \{0, 1\}$ in the case of unique shortest paths. Notice that $BC(b) = \sum_{s,t \in V} BC_{s,t}(v)$.[4] In the literature the betweenness centrality is sometimes defined differently as $BC(b) = \sum_{s,t \in V - \{b\}, s \neq t} \frac{\sigma_{s,t}(b)}{\sigma_{s,t}}$, where $\sigma_{s,t}$ is the number of distinct shortest paths from s to t, and $\sigma_{s,t}(b)$ is the number of such paths that use node b as an intermediate node. Here when $\sigma_{s,t} = 0$ (hence $\sigma_{s,t}(b) = 0$), $\frac{\sigma_{s,t}(b)}{\sigma_{s,t}}$ is assumed to be 0. Notice that this is equivalent to our definition in the case of unique shortest paths.

We remark that, in our subcubic reductions, it would be sufficient to preserve (modulo polylogarithmic factors) the number n of nodes only. However, whenever possible, we will also try to preserve (in the same sense) also m and M. In many cases we obtain extremely tight reductions that even allow us to obtain new faster algorithms, as is the case with Reach Centrality via our tight reduction to Diameter. In some claims we assume that a $T(n, m)$ time, $T(n, M)$ time, or $T(n, m, M)$ time algorithm for some problem is given. In all those claims we implicitly assume that those running times are polynomial functions of the input parameters lower bounded by $\Omega(m)$. This way, one has that $O(m) + T(O(n), O(m), O(M)) = \tilde{O}(T(n, m, M))$ and similarly for $T(O(n), O(M))$ and $T(O(n), O(m))$. We will use this fact multiple times along the article. We remark that this is without loss of generality since all the considered problems admit a polynomial-time algorithm in the mentioned parameters, and a lower bound of $\Omega(m)$ is implied by the input size.

Throughout this article, ***with high probability*** **(w.h.p.)** means with probability at least $1 - 1/n^{O(1)}$.

In some reductions involving Betweenness Centrality we will need to enforce the uniqueness of shortest paths. This can be enforced w.h.p. using the Isolation Lemma from [46].[5]

[4] We remark that the s-t pairs are ordered; in particular, in undirected graphs shortest s-t paths are counted twice.

[5] In [46] the lemma is stated in a slightly less general form, but the proof extends trivially.

Lemma 1.6 (Isolation Lemma [46]). *Consider a set system* $(U, \mathcal{S})$ *over a universe* U *of* h *elements. Let us assign an integer weight* $w(i) \in \{1, \ldots, q\}$ *chosen uniformly and independently at random to each* $i \in U$ *and define the weight of each set* $S \in \mathcal{S}$ *as* $w(S) = \sum_{i \in S} w(i)$. *Then there exists a unique set of minimum weight with probability at least* $1 - h/q$.

Corollary 1.7. *Let* $G = (V, E)$ *be a directed or undirected graph with edge weights* $w : E \to \{0, \ldots, M\}$ *and let* $c \geq 5$ *be an integer. Consider the random weight function* $w' : E \to \{1, \ldots, n^c + n^{c+1}M\}$ *given by* $w'(e) = n^{c+1}w(e) + r(e)$, *where each* $r(e) \in \{1, \ldots, n^c\}$ *is chosen independently and uniformly at random (*random perturbation*). Then with probability at least* $1 - 1/n^{c-4}$ *all shortest paths induced on* G *by weights* w' *are unique. Furthermore, any such path is deterministically also a shortest path w.r.t. weights* w.

Proof. Consider the directed case, the undirected one being analogous (with slightly better bounds). We first observe that *deterministically* any shortest path for (G, w') has to be a shortest path also for (G, w). Indeed, any such shortest path of length W in (G, w) has length at most $(n-1)n^c + n^{c+1}W$ in (G, w'), while any non-shortest path would have length at least $1 + n^{c+1}(W+1)$ in (G, w').

For each pair of distinct nodes (a, b), we consider the set system $(E, \mathcal{S}_{ab})$, where $\mathcal{S}_{ab}$ is the set of shortest a-b paths in (G, w) (interpreted as subsets of edges), of (common) length W. By the previous observation, any shortest a-b path in (G, w') must belong to $\mathcal{S}_{ab}$. Define $r(S) = \sum_{e \in S} r(e)$ for each $S \in \mathcal{S}_{ab}$. The Isolation Lemma 1.6 implies that there exists exactly one $S \in \mathcal{S}_{ab}$ with minimum $r(S)$ with probability at least $1 - |E|/n^c \geq 1 - 1/n^{c-2}$. Since $w'(S) = n^{c+1}W + r(S)$ deterministically for each $S \in \mathcal{S}_{ab}$, this implies that there exists exactly one shortest path in $\mathcal{S}_{ab}$ (hence in G) according to weights w' with the same probability. The claim follows by applying the union bound over the possible pairs (a, b). □

2 SUBCUBIC EQUIVALENCE WITH APSP

In this section we prove the subcubic equivalence between APSP and the following problems: Radius, Median, and Betweenness Centrality. As mentioned in the introduction, reducing these problems to APSP is fairly straightforward and here we will focus on the opposite reductions.

We exploit *Negative Triangle* as an intermediate sub-problem: determine whether a given undirected graph $G = (V, E)$, with integer edge weights $w : E \to \{-M, \ldots, M\}$, contains a triangle whose edges sum to a negative number; such a triangle is called a *negative triangle*. The latter problem was shown to be equivalent to APSP under subcubic reductions in [64].

Lemma 2.1 ([64]). *Negative Triangle and APSP (in directed or undirected graphs) are equivalent under subcubic reductions.*

In order to simplify our proofs, we assume that the input instance of Negative Triangle satisfies the following properties:

(1) Path lengths are even. This can be achieved by multiplying the weights by a factor of 2.
(2) Any two nodes are connected by a path containing at most 2 edges. This can be achieved by adding a dummy node r, and n edges of weight $2M$ between r and any other node. Observe that no new negative triangle is created this way.
(3) By appending at most $n + 1$ leaf nodes to r with edges of cost $2M$, we can assume w.l.o.g. that n is a power of 2.

These reductions can be performed in linear time; they increase the number of nodes by $O(n)$, the number of edges by $O(n)$, and the maximum absolute weight by a factor of 2. Therefore, any algorithm with (polynomial and at least linear in m) running time $\tilde{O}(T(n, m, M))$ for the

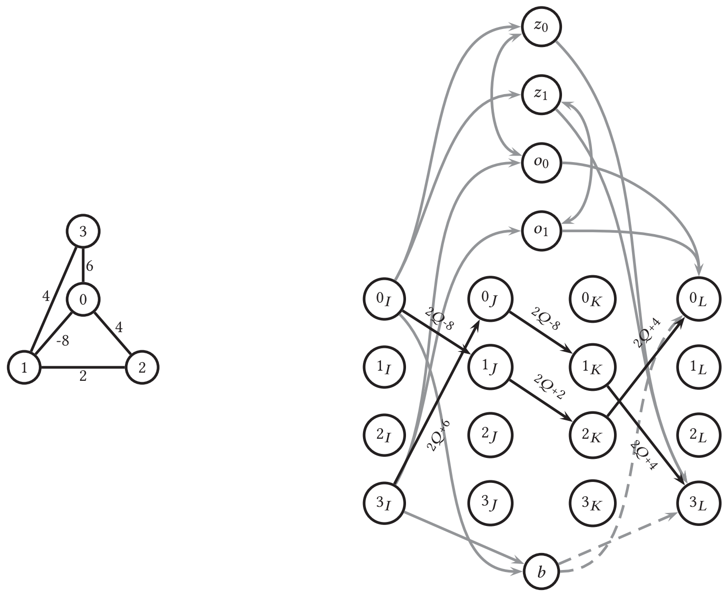

Fig. 2. Reduction from Negative Triangle to Betweenness Centrality (partially drawn). Full and dashed gray edges have weight $3Q - 1$ and $3Q$, respectively. The pair $0_I, 0_L$ does not contribute to $BC(b)$ (since 0 belongs to a negative triangle), while the pair $3_I, 3_L$ does contribute to $BC(b)$ (since 3 does not belong to any negative triangle).

modified instance can be used to solve the original instance in time $\tilde{O}(m+T(O(n), m+O(n), 2M)) = \tilde{O}(T(n, m, M))$. A similar claim holds for $T(n, m)$ and $T(n, M)$.

Combining the reductions below with Lemma 2.1 proves Theorem 1.3.

2.1 Betweenness Centrality

We start with the reduction to Betweenness Centrality. We obtain slightly different results assuming that the algorithm for Betweenness Centrality works on general instances or only under the restriction that shortest paths are unique. Later when we talk about the case of *non-unique* shortest paths, we mean that the shortest paths might not be unique.

Lemma 2.2. *Given a $T(n, m)$ time algorithm for Betweenness Centrality in directed or undirected graphs in the case of non-unique (resp., unique) shortest paths, there exists a deterministic (resp., high-probability Monte-Carlo) $\tilde{O}(T(n, m))$ time algorithm for Negative Triangle.*

Proof. Let $(G = (V, E), w)$ be the input instance of Negative Triangle (reduced as described above). Let $n = 2^{k+1}$ be the number of nodes of G, for some non-negative integer k. We initially consider the case of non-unique shortest paths.

We start with the simpler directed case (see also Figure 2). We construct a weighted directed graph (G', w') as follows. Graph G' contains four sets of nodes I, J, K, and L (*layers*). Each layer contains a copy of each node $v \in V$. Let v_I be the copy of v in I, and define analogously v_J, v_K, and v_L. Let $Q = \Theta(M)$ be a sufficiently large integer. For each edge $uv \in E$, we add to G' the edges $u_I v_J$, $u_J v_K$, and $u_K v_L$ and assign to those edges weight $2Q+w(uv)$. We add to G' a dummy node b, and edges $v_I b$ and $b v_L$ for any $v \in V$, of weight $3Q-1$ and $3Q$, respectively. We also add to G' two sets of nodes $Z = \{z_0, \ldots, z_k\}$ and $O = \{o_0, \ldots, o_k\}$. For any $v \in V$, we add the following edges of weight $3Q - 1$ to G'. Let $v^0, v^1, \ldots, v^k$ be a binary representation of v (interpreted as an integer between 0 and $n-1=2^{k+1}-1$). For each $j=0,\ldots,k$, we add edges $v_I z_j$ and $o_j v_L$ if $v^j=0$, and edges $v_I o_j$ and $z_j v_L$ otherwise. We also add edges $o_j z_j$ and $z_j o_j$ of weight $3Q-1$ for $j=0,\ldots,k$. Observe that $k=O(\log n)$; hence there are $O(n\log n)$ edges of the latter type.

On (G',w') we compute $BC(b)$ and output YES to the input Negative Triangle instance if and only if $BC(b)<n$. Let us prove the correctness of this reduction. The only paths passing through b are of the form s_I, b, t_L and have weight $6Q-1$. For $s\neq t$, there must exist a node $w\in Z\cup O$ such that s_I, w, t_L is a path of cost $6Q-2$. Therefore, the only pairs of nodes that can contribute to $BC(b)$ are of the form (s_I, s_L). The shortest path of type s_I, v_J, w_K, s_L has weight at most $6Q-2$ if s belongs to a negative triangle, and at least $6Q$ otherwise. Therefore, $BC_{s_I,s_L}(b)=1$ if s does not belong to any negative triangle, and $BC_{s_I,s_L}(b)=0$ otherwise. The correctness follows.

In the undirected case, we use the same weighted graph (G',w') as before, but removing edge directions (and leaving one copy of parallel edges). The rest of the reduction is as before, with the difference that now the answer is YES if and only if $BC(v)<2n$ (the extra factor 2 here is due to the fact that there are potentially $2n$ shortest paths passing through b). Proving correctness requires a slightly more complicated case analysis. Consider any pair $s,t\in V-\{b\}$. Suppose $(s,t)\notin(I\times L)\cup(L\times I)$. Then any s-t path passing through b costs at least $2(3Q-1)+(2Q-M)$. On the other hand, any $s\in Z\cup O$ can reach any $t\in Z\cup O$ within distance $2(3Q-1)$, and any $t\in I\cup J\cup K\cup L$ within distance $3Q-1+2(2Q+M)$. If $s,t\in I\cup J\cup K\cup L$, there exists an s-t path of length at most $3(2Q+M)$. It remains to consider the case that $s=s_I\in I$ and $t=t_L\in L$. The path s_I, b, t_L has cost $6Q-1$. If $s\neq t$, analogously to the directed case there exists $w\in Z\cup O$ such that s_I, w, t_L is a path of weight $6Q-2$. We can conclude that, like in the directed case, the only pairs that can contribute to $BC(b)$ are of the form (s_I, s_L). The shortest path of the form s_I, v_J, w_k, s_L has weight at most $6Q-2$ if s belongs to a negative triangle, and at least $6Q$ otherwise. Any other path avoiding b contains at least 4 edges, and therefore costs at least $4(2Q-M)$. We can conclude that $BC_{s_I,s_L}(b)=1$ if s is not contained in a negative triangle of (G,w), and $BC_{s_I,s_L}(b)=0$ otherwise. The correctness follows.

It remains to consider the case of unique shortest paths. Observe that in the above reduction shortest paths are not necessarily unique. The latter property can, however, be enforced w.h.p. by modifying weights as in Corollary 1.7. Notice that this randomized reduction gives the right answer (at least) whenever shortest paths are unique; hence this happens w.h.p. Since weights increase by a polynomial factor in n while n and m are asymptotically preserved, the running time is $\tilde{O}(T(n,m))$, as required. □

We remark that in the reduction in Lemma 2.2 the blow-up of the weights happens only when we need to enforce the uniqueness of shortest paths. In particular, if we had a $\tilde{O}(T(n,m,M))$ time algorithm for the variant of Betweenness Centrality not requiring such uniqueness, this would imply a $\tilde{O}(T(n,m,M))$ time algorithm for Negative Triangle.

Proof of Theorem 1.3. One direction is obtained by chaining Lemmas 2.1 and 2.2. The other direction is trivial: simply solve APSP and count (in $O(n^2)$ total time) how many pairs (s,t), $s,t\in V-\{b\}$, satisfy $d(s,t)=d(s,b)+d(b,t)$. □

2.2 Radius

Our reduction from Negative Triangle to Radius is similar to the one in Lemma 2.2. Consider the same construction when we remove the node b from the graph. The key observation is that a node s_I has distance at most $6Q-2$ to *every* node t_L (including s_L) if and only if s is in a negative triangle in G. Intuitively, this allows us to show that an algorithm distinguishing between radius $6Q-2$ and radius $6Q-1$ can solve Negative Triangle. To complete the reduction we need to make sure

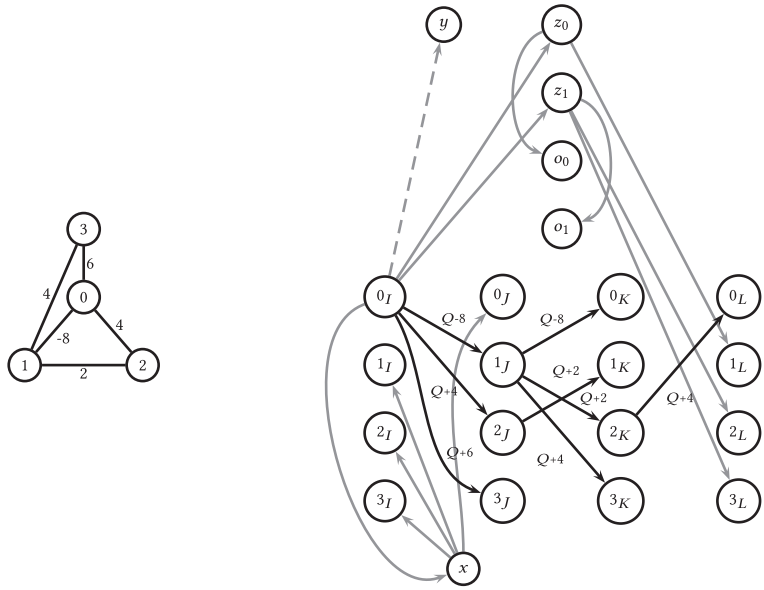

Fig. 3. Reduction from Negative Triangle to Radius. Only edges in the shortest path tree from 0_I are illustrated. The full and dashed gray edges have weight Q and $3Q-1$, respectively.

that s_I is close to every node in the graph (not only nodes in part L) and that the center can only lie in part I.

Lemma 2.3. *Given a $T(n,m,M)$ time algorithm for Radius in directed or undirected graphs, there exists a $\tilde{O}(T(n,m,M))$ time algorithm for Negative Triangle.*

Proof. Let $(G=(V,E),w)$ be the considered instance of Negative Triangle (modified as described before). We start with the directed case (see also Figure 3). Let $Q=\Theta(M)$ be a sufficiently large integer. We construct a directed weighted graph (G',w') as follows. Similarly to the proof of Lemma 2.2, graph G' contains four copies I, J, K, and L of the node set V (*layers*). Let v_X be the copy of $v\in V$ in layer X. For each edge $uv\in E$, we add to G' edges u_Iv_J, u_Jv_K, and u_Kv_L of weight $Q+w(vu)$. We also add to G' two sets of nodes $Z=\{z_0,\dots,z_k\}$ and $O=\{o_0,\dots,o_k\}$. We add edges incident to nodes $Z\cup O$ in the same way as in Lemma 2.2, using edges of cost Q. In more detail, let $v^0,v^1,\dots,v^k$ be the binary representation of node v: we add the edges v_Iz_j and o_jv_L if $v^j=0$, and the edges v_Io_j and z_jv_L otherwise. We also add edges z_jo_j and o_jz_j of weight Q for all $j=0,\dots,k$. Finally, we add nodes x and y, and for any $v\in V$ we add edges v_Ix, xv_I, and xv_J of weight Q, and edges v_Iy of weight $3Q-1$.

We compute the radius R^* of (G',w') and output YES to the input instance of Negative Triangle if and only if $R^*\leq 3Q-1$. The running time of the algorithm is $\tilde{O}(m+T(O(n),O(m+n\log n),O(M)))=\tilde{O}(T(n,m,M))$. Let us prove its correctness. We first observe that the center r^* of the graph belongs to $I\cup\{x\}$ since the other nodes cannot reach any node in I. Observe that $d(x,y)=4Q-1$. On the other hand, any node s_I is at distance at most $2Q$ to nodes in $Z\cup O\cup J\cup\{x\}\cup(L-\{s_L\})$, at most $2Q+2M$ to nodes in K (using the copy r_J of the root node r), and exactly $3Q-1$ to node y. Note also that if s belongs to a negative triangle, there exists an s_I-s_L path of the form s_I,v_J,w_K,s_L with length at most $3Q-2$. Otherwise, one shortest s_I-s_L path passes through nodes in $Z\cup O$ and has length $3Q$. We can conclude that the center of the graph belongs to I, and that the corresponding radius is upper bounded by $3Q-1$ if and only if there exists a negative triangle in (G,w).

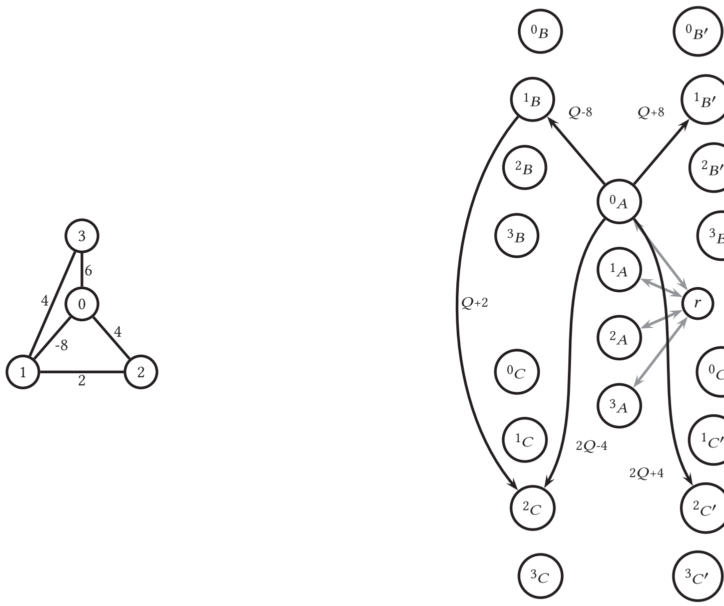

Fig. 4. Reduction from Negative Triangle to Median (partially drawn). Gray edges have weight $Q/4$. The path $0_A, 1_B, 2_C$ is shorter than the path $0_A, 2_C$: this corresponds to a negative triangle.

In the undirected case we use precisely the same construction, but removing edge directions (and leaving only one copy of parallel edges). The algorithm is analogous as well as its running time analysis. Its correctness can also be proved analogously. In more detail, similarly to the directed case, nodes in I can reach any other node within distance at most $3Q + 3M$. Since $d(y, x) = 4Q - 1$, and $d(s, y) \geq (3Q - 1) + (Q - M)$ for $s \notin I \cup \{y\}$, we can conclude that $r^* \in I$. Also in this case, for any node s_I, its maximum distance to any other node is $d(s_I, y) = 3Q - 1$ if s belongs to a negative triangle, and $d(s_I, s_L) \geq 3Q$ otherwise. □

Proof of Theorem 1.1. One direction is trivial, and the other is given by Lemmas 2.1 and 2.3. □

2.3 Median

The reduction to Median is based on a rather different approach.

Lemma 2.4. *Given a $T(n, M)$ time algorithm for Median in undirected or directed graphs, there exists a $\tilde{O}(T(n, M))$ time algorithm for Negative Triangle.*

Proof. Let $(G = (V, E), w)$ be the given instance of Negative Triangle. First, consider the directed case (see also Figure 4). We create a weighted directed graph (G', w'). Graph G' contains five copies A, B, B', C, C' of V. With the usual notation, v_A is the copy of v in A and similarly for the other sets. Let $Q = \Theta(M)$ be a large enough integer. For any pair of nodes u, v, we add the edges $u_A v_B$ of weight $Q + w(uv)$, $u_A v_{B'}$ of weight $Q - w(uv)$, $u_A v_C$ of weight $2Q - w(uv)$, $u_A v_{C'}$ of weight $2Q + w(uv)$, and $u_B v_C$ of weight $Q + w(uv)$. In this construction, when $uv \notin E$ (including the special case $u = v$), we simply assume $w(uv) = 2M$. Furthermore, we add a dummy node r and edges rv_A and $v_A r$ of weight $Q/4$ for any $v \in V$.

In this graph we compute the median value *Med* and output YES to the input instance of Negative Triangle if and only if $Med < Q/4 + (n-1)Q/2 + 6nQ$. The running time of the algorithm is

$\tilde{O}(m + T(O(n), O(M))) = \tilde{O}(T(n, M))$. Let us show its correctness. Let $d(\cdot)$ denote distances in G'. The median node has to be in $A \cup \{r\}$ since the remaining nodes cannot reach r. Recall that, for a node w, $Med(w) := \sum_{v\in V} d(w, v)$. Note that

$$\begin{aligned} Med(r) &\geq n\left(\frac{Q}{4} + \left(\frac{Q}{4} + Q\right) + \left(\frac{Q}{4} + Q - M\right) + \left(\frac{Q}{4} + 2Q - M\right) + \left(\frac{Q}{4} + 2Q\right)\right) \\ &= \frac{29}{4}Qn - 2Mn > \frac{Q}{4} + (n-1)\frac{Q}{2} + 6nQ. \end{aligned}$$

In the first inequality above we lower bounded the distances to nodes in A, B, B', C, and C' with $Q/4$, $Q/4 + Q$, $Q/4 + Q - M$, $Q/4 + 2Q - M$, and $Q/4 + 2Q$, respectively. In the second inequality above we used the assumption that Q is sufficiently larger than M. On the other hand, for any node v_A,

$$\begin{aligned} &Med(v_A) = \\ &= \ d(v_A, r) + \sum_{u\in V} d(v_A, u_A) + \sum_{u\in V}(d(v_A, u_B) + d(v_A, u_{B'})) + \sum_{u\in V}(d(v_A, u_C) + d(v_A, u_{C'})) \\ &= \ \frac{Q}{4} + (n-1)\frac{Q}{2} + \sum_{u\in V}(Q + w(vu) + Q - w(vu)) + \sum_{u\in V}(d(v_A, u_C) + 2Q + w(vu))) \\ &= \ \frac{Q}{4} + (n-1)\frac{Q}{2} + 2nQ + \sum_{u\in V}(d(v_A, u_C) + 2Q + w(vu))) \\ &\leq \ \frac{Q}{4} + (n-1)\frac{Q}{2} + 6nQ. \end{aligned}$$

Therefore, the median is in A. In the last inequality we upper bounded $d(v_A, u_C)$ with $w'(v_A u_C) = 2Q - w(vu)$. Here a strict inequality holds if there exists a third node z_B such that $w'(v_A z_B) + w'(z_B u_C) < w'(v_A u_C)$. However, this can happen only if $vu \in E$, since otherwise $w'(v_A u_C) = 2Q{-}2M \leq w'(v_A z_B) + w'(z_B u_C)$. Note also that, if either $vz \notin E$ or $zu \notin E$, then $w'(v_A z_B) + w'(z_B u_C) \geq 2Q{+}M \geq w'(v_A u_C)$. Therefore, we can conclude that the strict inequality holds if and only if there exists a triangle $\{v, z, u\}$ in G such that $Q{+}w(vz) + Q{+}w(zu) < 2Q - w(vu)$, i.e., a negative triangle. The claim follows.

Consider next the undirected case. We construct the same weighted graph (G', w') as in the directed case, but removing edge directions (and leaving one copy of parallel edges). The rest of the algorithm is as in the directed case, and the running time remains $\tilde{O}(T(n, M))$. In order to prove correctness, we need a slightly more complicated case analysis. Like in the directed case, $Med(v_A) \leq Q/4 + (n-1)Q/2 + 6nQ$, where a strict inequality holds if and only if v belongs to a negative triangle. For any $u_B \in B$,

$$\begin{aligned} Med(u_B) &\geq (Q - M + Q/4) + 2n(Q - M) + n(2Q - 2M) + n(3Q - 2M) \\ &= (7n + 5/4)Q - (6n + 1)M. \end{aligned}$$

Similarly,

$$\begin{aligned} Med(u_{B'}) &\geq (9n + 5/4)Q - (7n + 1)M, \\ Med(u_C) &\geq (10n + 9/4)Q - (9n + 2)M, \end{aligned}$$

and

$$Med(u_{C'}) \geq (12n + 9/4)Q - (8n + 1)M.$$

Furthermore,

$$\begin{aligned} Med(r) &\geq nQ/4 + 2n(5Q/4 - M) + n(9/4Q - 2M) + n(9/4Q - M) \\ &= (29n/4)Q - 5nM. \end{aligned}$$

We can conclude that the median is in A. The correctness follows. □

Proof of Theorem 1.2. One direction is trivial, and the other is given by Lemmas 2.1 and 2.4. □

Finally, we also prove a similar reduction for the following *All-Nodes Median Parity* problem: compute $Med(v) \pmod 2$ for all nodes v.

Lemma 2.5. *Given a $T(n, M)$ time algorithm for the All-Nodes Median Parity problem in a directed or undirected graph, there exists a $\tilde{O}(T(n, M))$ time algorithm for Negative Triangle.*

Proof. Let $(G = (V, E), w)$ be the considered instance of Negative Triangle. Let us start with the directed case. Let $Q = \Theta(M)$ be a sufficiently large even integer. Similarly to the proofs of Lemmas 2.2 and 2.3 and with a similar notation, we construct a four-layer weighted directed graph (G', w') with layers I, J, K, and L, and edges $v_I u_J$, $v_J u_K$, and $v_K u_L$ of weight $2Q + w(vu)$ for any $uv \in E$. We also introduce a fifth copy B of V, and for any $v_B \in B$ we add edges $v_I v_B$ and $v_B v_L$ of weight $3Q$ and $3Q - 1$, respectively. We also add edges $v_I u_B$ of weight $3Q + 3M + 2$ for any $u \neq v$. Finally, we add a node r, and edges $v_I r$ and $r v_I$ of weight Q for all $v \in V$. Observe that the edges of type $v_B v_L$ are the only edges of odd weight (by the preprocessing of the Negative Triangle instance).

In this graph we compute $Med(v) \pmod 2$ for all $v \in V(G')$ and we output YES to the input Negative Triangle instance if and only if $Med(v_I) \pmod 2 = 0$ for some $v_I \in I$ (i.e., some $Med(v_I)$ is even). The running time is $\tilde{O}(T(O(n), O(M))) = \tilde{O}(T(n, M))$. Let us prove correctness. Consider any $v_I \in I$. Any node is reachable from v_I; hence $Med(v_I)$ is finite. Any path of type v_I, u', u_L, $u \neq v$, cannot be a shortest path since it has length $6Q + 3M + 2 - 1$ while there exists a v_I-u_L path of length at most $6Q + 3M$ avoiding B. Therefore, the unique candidate shortest path of odd weight is v_I, v', v_L of length $6Q - 1$. However, by the usual argument, this is not a shortest path if v is contained in some negative triangle. The claim follows.

In the undirected case we can use the same graph (G', w'), but removing edge directions (and leaving one copy of parallel edges). The rest of the algorithm is the same and its analysis is analogous to the directed case. □

Corollary 2.6. *Given a truly subcubic algorithm for All-Nodes Median Parity, there exists a truly subcubic algorithm for APSP.*

3 SUBCUBIC EQUIVALENCE BETWEEN REACH CENTRALITY AND DIAMETER

In this section we show that Diameter is equivalent to Reach Centrality under subcubic reductions. We start with the simple reductions from Diameter.

Lemma 3.1. *Given a $T(n, m)$ time algorithm for Reach Centrality in directed or undirected graphs, there is a $\tilde{O}(T(n, m))$ time algorithm for Diameter in the same graph class.*

Proof. Let $(G = (V, E), w)$ be the input instance of Diameter, and let M be the largest integer weight. Consider first the directed case. Let G' be an auxiliary graph consisting of a copy of G plus a dummy node b and edges vb and bv for all $v \in V$. For each integer $D \in [1, (n-1)M]$, we define an auxiliary weight function $w'(D)$ on the edges of G', which is $D/2$ for the edges incident on b and identical to w on the remaining edges. Observe that in $(G', w'(D))$ any pair of nodes $s, t \in V$ is connected by a path of length D using b. We identify the largest value D' of D such that $RC(b) \geq D/2$ for the Reach Centrality instance induced by $(G', w'(D))$: this is done via a binary search over $D \in [1, (n-1)M]$, and using the Reach Centrality algorithm given in the claim. The output value of the diameter is D'. For the sake of presentation, in the above reduction we tolerate fractional weights for odd D: this can be trivially avoided by initially multiplying all weights w by a factor of 2, considering even values of D only, and finally outputting $D'/2$.

The running time of the algorithm is $\tilde{O}((m+T(n+1, 2n+m))\log(nM)) = \tilde{O}(T(n, m))$. Let (s^*, t^*) be a witness pair for the diameter D^*. In any execution where $D^* \geq D$, there exists a shortest s^*-t^* path using node b and hence the answer is $RC(b) \geq D/2$. In any other execution (where $D^* < D$), any shortest s-t path avoiding b has length at most $D^* \leq D - 1$, while passing through b would cost at least D (thus the answer is $RC(b) = 0$). The correctness of the algorithm follows.

For the undirected case, we use the same auxiliary weighted graph, but without edge directions (and leaving one copy of parallel edges). The algorithm is the same. The running time is $\tilde{O}((m + T(n + 1, n + m))\log(nM)) = \tilde{O}(T(n, m))$. Similarly to the directed case, in any execution where D is upper bounded by the diameter D^*, there exists a shortest s^*-t^* path using node b; hence $RC(b) \geq D/2$. In the remaining executions no shortest path uses b; hence $RC(b) = 0$. □

Now, we present the more tricky reduction to Diameter. The following very efficient reduction completes the equivalence between Diameter and Reach Centrality in directed graphs and implies directly Theorem 1.5.

Lemma 3.2. *Given a $T(n, m, M)$ time algorithm for Diameter in directed graphs, there is a $\tilde{O}(T(n, m, M))$ time algorithm for Reach Centrality in directed graphs.*

Proof. Let $(G = (V, E), w, b)$ be the input instance of Reach Centrality. Observe that $RC(b)$ is upper bounded by one-half of the diameter of G; hence in particular $RC(b) \leq (n - 1)M/2$. We show how to determine whether $RC(b) \geq K$ for a given integer parameter $0 \leq K \leq (n - 1)M/2$ in $\tilde{O}(T(n, m, M))$ time. The value of $RC(b)$ can then be determined via binary search with an extra factor of $O(\log(nM)) = \tilde{O}(1)$ in the running time.

Observe that, if the answer is YES, there must be two nodes $s, t \in V - \{b\}$ such that some shortest s-t path passes through b, $K + M > d(s, b) \geq K$, and $K + M > d(b, t) \geq K$. We construct an instance (G', w') of Diameter as follows. We add to G' a copy of G. Furthermore, we add a set of nodes A that contains a node v_A for each node $v \in V$ such that $K + M > d(v, b) \geq K$. Symmetrically, we add a set of nodes B that contains a node v_B for each node $v \in V$ such that $K + M > d(b, v) \geq K$. We also add edges $v_A v$ and $v v_B$ of weight $K + M - d(v, b)$ and $K + M - d(b, v)$, respectively. Note that the weight of the latter edges is in $[1, M]$ by construction. Finally, we add a directed path $P = v_0, \ldots, v_q$, $q = \lceil (2K + 2M - 2)/M \rceil$, whose edge weights are chosen arbitrarily in $[1, M]$ so that the length of P is exactly $2K + 2M - 2$. For every $v \in V$, we add edges $v v_0$ and $v_q v$ of weight zero. We also add edges $a v_0$ of weight 1 and $v_q a$ of weight 0 for any $a \in A$. Symmetrically, we add edges $v_q b$ of weight 1 and $b v_0$ of weight 0 for any $b \in B$.

We compute the diameter D^* of (G', w') and output that $RC(b) \geq K$ if and only if $D^* \geq 2K + 2M$. The running time of the algorithm is $\tilde{O}(m + T(O(n), O(m + n), M)) = \tilde{O}(T(n, m, M))$. Consider its correctness. The distance between any two nodes in $G \cup P$ is at most $2K + 2M - 2$. The distance between any node in $G \cup P$ and any other node is at most $2K + 2M - 1$. The distance between any node in B and any other node is at most $2K + 2M - 1$. The distance between any node in A and any node in $G \cup P \cup A$ is at most $2K + 2M - 1$.

Consider next any pair $s_A \in A$ and $t_B \in B$. An s_A-t_B path using P would cost at least $2K + 2M$. A shortest s_A-t_B path avoiding P costs $2K + 2M - d(s, b) - d(b, t) + d(s, t) \leq 2K + 2M$, where the equality holds if and only if b is along some shortest s-t path. Therefore, $D^* \leq 2K + 2M$ and the equality holds if and only if there exists a pair $(s_A, t_B) \in A \times B$ such that $d(s, t) = d(s, b) + d(b, t)$, i.e., if and only if $RC(b) \geq K$. The correctness follows. □

Proof of Theorem 1.5. It follows from Lemma 3.2 by exploiting the $\tilde{O}(Mn^{\omega})$ time algorithm for Diameter in directed graphs in [17]. □

Notice that Lemma 3.2 works only for directed graphs. In the next section we will prove the following reduction, which works also for undirected graphs at a cost of not preserving asymptotically the edge weights.

Lemma 3.3. *Given a $T(n, m)$ time algorithm for Diameter in directed or undirected graphs, there is a $\tilde{O}(T(n, m))$ time algorithm for Reach Centrality in the same graph class.*

Theorem 1.4 directly follows.

Proof of Theorem 1.4. One direction is implied by Lemma 3.1 and the other by Lemma 3.3. □

4 APPROXIMATION OF REACH AND BETWEENNESS CENTRALITY

In this section we present our results about the approximability of Reach and Betweenness Centrality. A key idea in our approach is to consider the following *Positive Betweenness Centrality* problem, which might be of independent interest: determine whether, for a given node b, there exists some shortest path using b as an intermediate node. We let $PosBC(b)$ denote the answer to this problem (YES or NO).

The following two lemmas show that Approximate Betweenness and Reach Centrality are at least as hard as Positive Betweenness Centrality under subcubic reductions.

Lemma 4.1. *Given a $T(n, m)$ time algorithm for Approximate Betweenness Centrality in the case of non-unique (resp., unique) shortest paths, there exists a deterministic (resp., high-probability Monte-Carlo) $\tilde{O}(T(n, m))$ time algorithm for Positive Betweenness Centrality with non-unique (hence unique) shortest paths.*

Proof. Let us initially modify the edge weights of the input Positive Betweenness Centrality instance as follows. We first multiply edge weights by $3n$. Then we add 1 to the weights of edges incident to b (considering both ingoing and outgoing edges for directed graphs), and we add 3 to all other edges. Let w' be the new edge weights. Observe that any shortest path w.r.t. w' is also a shortest path w.r.t. w by an argument similar to Corollary 1.7. In more detail, let W be the length of an a-c shortest path for some pair of distinct nodes a and c w.r.t. w. The same path w.r.t. w' has length at most $3(n-1) + 3nW$, while any non-shortest a-c path w.r.t. w would have length at least $1 + 3n(W + 1)$ w.r.t. w'.

Let $PosBC'(b)$ be the answer to the Positive Betweenness Centrality instance induced by the weights w'. We claim that $PosBC'(b) = PosBC(b)$ (i.e., the two instances are equivalent). Indeed, if $PosBC(b) = NO$, it must be $PosBC'(b) = NO$ since, as said before, we are not creating alternative shortest paths using b with weights w'. Suppose instead $PosBC(b) = YES$. This implies that w.r.t. weights w there exists a shortest path P, say from u to v, that goes through b, where u, v, b are all distinct. Consider the nodes right before and after b on P; call them a and c. Here again, a, b, c are all distinct. Let W be the length of the abc path. With weights w' any a-c path avoiding b would cost at least $3nW + 3$, while abc costs $3nW + 2$ only. Thus, all shortest a-c paths w.r.t. w' pass through b. In particular, $PosBC'(b) = YES$.

If the given algorithm for Approximate Betweenness Centrality works in the case of non-unique shortest paths or the input instance of Positive Betweenness Centrality has unique shortest paths, we simply apply that algorithm with weights w' and return NO if and only if the approximate value is 0. The claim on the running time holds trivially. Let $BC'(b)$ be the value of $BC(b)$ w.r.t. weights w'. If $PosBC'(b) = NO$, then $BC'(b) = 0$ since the initial modification of the weights does not create new shortest paths. Hence the approximate solution must be 0. Otherwise, by construction necessarily $BC'(b) > 0$; hence the approximate value must be positive. The correctness follows.

Otherwise, we first randomly perturb the weights w' of the input Positive Betweenness Centrality instance as in Corollary 1.7. Let w'' be the perturbed weights. Next assume that shortest paths are unique w.r.t. weights w'', which happens w.h.p., and let $BC''(b)$ be the value of $BC(b)$ w.r.t. weights w''. Then we apply the approximation algorithm for Betweenness Centrality and declare $PosBC'(b) = NO$ if and only if the approximate value is 0. Clearly the running time is as in the claim since m and n are preserved, while the largest edge weight is increased by a polynomial factor in n. By the above arguments, if $PosBC'(b) = NO$, it must be the case that $BC''(b) = 0$ since the perturbation from Corollary 1.7 does not create alternative shortest paths using b. Hence the approximate algorithm would return 0. Otherwise, there will be some pair (a, c) such that all shortest a-c paths w.r.t. weights w' use node b; hence one such path will cause $BC''(b) > 0$. Therefore, the approximation algorithm has to return a positive value. □

Lemma 4.2. *Given a $T(n, m)$ time algorithm for Approximate Reach Centrality, there is a $\tilde{O}(T(n, m))$ time algorithm for Positive Betweenness Centrality with non-unique shortest paths.*

Proof. By definition, $RC(b) \geq \min\{d(b, b), d(b, b)\} = 0$ and $RC(b) > 0$ implies $PosBC(b) = YES$. However, due to 0 weights, it might still be that $RC(b) = 0$ and $PosBC(b) = YES$. To avoid this issue we build weights w' exactly as in the proof of Lemma 4.1. Recall that, with the same notation, $PosBC'(b) = PosBC(b)$. Furthermore, $PosBC'(b) = YES$ if and only if there exists some pair of nodes (a, c), with a, b, c all distinct, such that all shortest a-c paths use node b. Let $RC'(b)$ denote the value of $RC(b)$ w.r.t. weights w'.

We apply the approximation algorithm for Reach Centrality to the resulting instance and return $PosBC(b) = NO$ if and only if the answer is 0. The running time satisfies the claim since m and n are preserved, while the largest edge weight is increased by a polynomial factor in n. For the correctness, observe that $PosBC'(b) = PosBC(b) = NO$ implies that $RC'(b) = 0$; hence the approximation algorithm has to return 0. Otherwise, since all weights are at least 1, the mentioned pair (a, c) guarantees that $RC'(b) \geq 1$; hence the approximation algorithm has to return a positive value. □

4.1 Some Results on Positive Betweenness Centrality

A simple observation is that on unweighted graphs, Positive Betweenness Centrality is asking whether there is an in-neighbor x of b and an out-neighbor y of b such that $xy \notin E$, and therefore can be solved in $O(m)$ time. We next show that, on weighted graphs, Positive Betweenness Centrality and Diameter are equivalent under subcubic reductions.

Theorem 4.3. *Diameter and Positive Betweenness Centrality with non-unique shortest paths are equivalent under subcubic reductions.*

Theorem 4.3 follows directly from the next two lemmas.

Lemma 4.4. *Given a $T(n, m)$ time deterministic (resp., high-probability Monte-Carlo) algorithm for Positive Betweenness Centrality with non-unique shortest paths in directed or undirected graphs, there is a deterministic (resp., high-probability Monte-Carlo) $\tilde{O}(T(n, m))$ time algorithm for Diameter in the same graph class.*

Proof. Let us focus on the deterministic case, the other case being analogous. This proof is similar in spirit to the proof of Lemma 3.1. Let $(G = (V, E), w)$ be the input instance of Diameter, where M is the largest integer weight. Consider first the directed case (see also Figure 5). Let D be an integer in $[1, (n-1)M]$. Let $(G', w'(D))$ denote the auxiliary weighted graph consisting of a copy of (G, w) plus a dummy node b and dummy edges vb and bv of weight[6] $D/2$ for any $v \in V$. Observe that any pair of nodes $s, t \in V$ is connected by a path of length D using b. By performing a binary

[6]Fractional weights can be avoided similarly to the proof of Lemma 3.1.

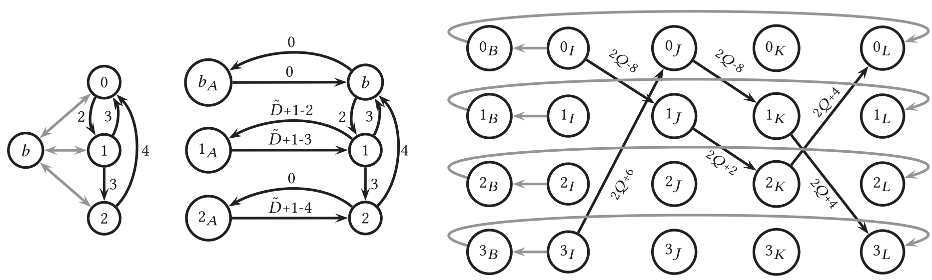

Fig. 5. **(Left)** Reduction from Diameter to Positive Betweenness Centrality in directed graphs. Gray edges have weight $D/2$, where D is a *guess* of the diameter. **(Middle)** Reduction from Positive Betweenness Centrality to Diameter in directed graphs. Here $\tilde{D}$ is a proper upper bound on the diameter. Notice that the preprocessing involving the dummy node r is not illustrated in the figure. **(Right)** Reduction from the Negative Triangle instance of Figure 2 to All-Nodes Positive Betweenness Centrality in directed graphs (partially drawn). Gray edges have weight $3Q$. One has $BC(3_B) > 0$ and $BC(0_B) = 0$ since node 3 does not belong to a negative triangle, while node 0 does.

search on D and solving each time the resulting instance $(G', w'(D), b)$ of Positive Betweenness Centrality, we determine the largest value D' of D such that the answer is YES (i.e., $BC(b) > 0$). The output value of the diameter is D'.

The running time of the algorithm is $\tilde{O}((m+T(n+1, 2n+m))\log(nM)) = \tilde{O}(T(n,m))$. Let (s^*, t^*) be a witness pair for the diameter D^*. In any execution where $D^* \geq D$, there exists a shortest s^*-t^* path using node b and hence the answer is YES. In any other execution (where $D^* < D$), any shortest s-t path avoiding b has length at most $D^* \leq D-1$, while passing through b would cost at least D (thus the answer is NO). The correctness of the algorithm follows.

For the undirected case, we use the same auxiliary weighted graph, but without edge directions (and leaving one copy of parallel edges). The algorithm and its analysis are analogous to the directed case. □

Lemma 4.5. *Given a $T(n, m, M)$ time algorithm for Diameter in directed or undirected graphs, there is a $\tilde{O}(T(n, m, M))$ time algorithm for Positive Betweenness Centrality with non-unique (hence unique) shortest paths in the same graph class.*

Proof. Let (G, w, b) be the input instance of Positive Betweenness Centrality. Observe that the answer is YES if and only if there exists a shortest path of the form s, b, t.

Let us consider the directed case first. By adding a dummy node r and dummy edges vr and rv of weight M for any $v \in V - \{b\}$, we can assume that the diameter of G is at most $\tilde{D} = 3M$ (w.l.o.g., b has at least one in-neighbor and one out-neighbor). Note that we did not introduce new paths of the form s, b, t. Furthermore, the new graph has $n+1$ nodes, $m+2n$ edges, and maximum weight M. Hence a $\tilde{O}(T(n, m, M))$ time algorithm for the modified instance implies the same running time for the original one.

We construct an instance (G', w') of Diameter as follows (see also Figure 5). Initially $G' = G$. We add a copy A of V. Let v_A be the copy of $v \in V$. For every $v \in V$, we add edges $v_A v$ and $v v_A$ of weight $\tilde{D} + 1 - w(vb)$ and $\tilde{D} + 1 - w(bv)$, respectively. If edges vb or bv are missing (including the case $v = b$), we set the weight of the corresponding edges $v_A v$ and $v v_A$, respectively, to 0. Observe that edge weights are $O(M)$.

In this graph we compute the diameter D^* and output YES to the input Positive Betweenness Centrality instance if and only if $D^* \geq 2\tilde{D} + 2$. The running time of the algorithm is $\tilde{O}(m + T(O(n), O(m), O(M))) = \tilde{O}(T(n, m, M))$. Consider a witness pair s^*, t^* for the value of the diameter.

Since edges of type $v_A v$ and vv_A have non-negative weight, we can assume w.l.o.g. that $s^* = s_A \in A$ and $t^* = t_A \in A$. If both edges sb and bt are missing, one has $D^* = d_G(s,t) \leq \tilde{D}$. If exactly one of the mentioned edges is missing, say bt, one has $D^* = \tilde{D} + 1 - w(sb) + d_G(s,t) \leq 2\tilde{D} + 1$. Finally, if both edges are present, one has $D^* = 2(\tilde{D}+1) - w(sb) - w(bt) + d_G(s,t) \leq 2\tilde{D} + 2$, where equality holds if and only if s, b, t is a shortest path. In particular, if there exists a shortest path of the mentioned type, $D^* = 2\tilde{D} + 2$, and otherwise $D^* \leq 2\tilde{D} + 1$. The correctness follows.

By simply removing edge directions (and leaving one copy of parallel edges) in the above construction, one obtains the claim in the undirected case. □

We can exploit the above equivalence to derive (indirectly) the equivalence between Diameter and Reach Centrality in both directed and undirected graphs (recall that we showed this equivalence only in directed graphs; see Lemma 3.2).

Lemma 4.6. *Given a $T(n,m)$ time algorithm for Positive Betweenness Centrality with non-unique shortest paths in directed or undirected graphs, there is a $\tilde{O}(T(n,m))$ time algorithm for Reach Centrality in the same graph class.*

Proof. Let (G, w, b) be the input instance of Reach Centrality. We show how to determine whether $RC(b) \geq K$ for a given parameter K in $\tilde{O}(T(n,m))$ time. The value of $RC(b)$ can then be determined via binary search with an extra factor of $O(\log(nM)) = \tilde{O}(1)$ in the running time.

Let us consider the directed case first. We compute the shortest path distances from and to b in G. Next we construct an auxiliary weighted graph (G', w') as follows. We let G' initially contain a copy of $G - \{b\} = G[V - \{b\}]$, plus an isolated node b. Next, for any $v \in V - \{b\}$, we add an edge vb of weight $d(v,b)$ if and only if $d(v,b) \geq K$. Symmetrically, we add an edge bv of weight $d(b,v)$ if and only if $d(b,v) \geq K$.

We solve the Positive Betweenness Centrality instance (G', w', b) and output that $RC(b) \geq K$ if and only if the answer is YES. The running time of the algorithm is $\tilde{O}(m + T(n, m+2n)) = \tilde{O}(T(n,m))$. Let us prove its correctness. Suppose that $RC(b) \geq K$ and let (s,t) be a witness pair of that. Then s, b, t is a shortest s-t path in G' and therefore the answer to the Positive Betweenness Centrality instance is YES. Vice versa, suppose that the answer to the Positive Betweenness Centrality instance is YES; i.e., there exists a shortest s-t path passing through b. This implies that there exists a shortest path of the form s', b, t'. Observe that the shortest paths not involving node b are the same in G and G'. Therefore, there exists a shortest s'-t' path in G' passing through b. Since by construction $d_G(s', b), d_G(b, t') \geq K$, the pair (s', t') witnesses that $RC(b) \geq K$.

The claim in the undirected case follows from the same reduction, but removing edge directions (and leaving only one copy of parallel edges). □

Lemma 3.3 directly follows.

Proof of Lemma 3.3. It follows by chaining Lemmas 4.5 and 4.6. □

Another interesting observation about Positive Betweenness Centrality is that although solving it for a single node b is equivalent to Diameter under subcubic reductions, the *all-nodes* version of the problem (where one wants to determine whether $BC(b) > 0$ for all nodes b) is actually at least as hard as APSP.

Lemma 4.7. *Given a $T(n,m,M)$ time algorithm for All-Nodes Positive Betweenness Centrality with non-unique shortest paths in directed or undirected graphs, there is a $\tilde{O}(T(n,m,M))$ time algorithm for Negative Triangle.*

Proof. Let (G, w) be the input instance of Negative Triangle. Consider first the directed case (see also Figure 5). We create a directed weighted graph (G', w') as follows. Graph G' contains five copies I, J, K, L, and B of the node set V. With the usual notation v_X is the copy of node $v \in V$ in set X. Let $Q = \Theta(M)$ be a sufficiently large integer. For every edge $uv \in E$ we add the edges $u_I v_J, u_J v_K, u_K v_L$ to G' and set their weight to $2Q + w(uv)$. We also add edges $u_I u_B$ and $u_B u_L$ for every node u in G and set the weight of these edges to $3Q$.

The algorithm solves the All-Nodes Positive Betweenness Centrality problem on (G', w') in time $\tilde{O}(T(n, m, M))$ and outputs YES to the input Negative Triangle instance if and only if $BC(u_B) > 0$ for some $u_B \in B$. To show correctness, observe that the only path through u_B is from u_I to u_L and it has weight $6Q$, while every path of type u_I, v_J, w_K, u_L corresponds to a triangle $\{u, v, w\}$ in G and the weight of the path equals the weight of the triangle plus $6Q$. The claim follows.

The same construction, without edge directions, proves the claim for undirected graphs. □

Corollary 4.8. *Given a truly subcubic algorithm for All-Nodes Approximate Reach Centrality or for All-Nodes Approximate Betweenness Centrality with non-unique shortest paths, there exists a truly subcubic algorithm for APSP.*

Proof. In case of strictly positive weights, a truly subcubic algorithm for All-Nodes Approximate Reach Centrality or for All-Nodes Approximate Betweenness Centrality with non-unique shortest paths directly implies a truly subcubic algorithm for All-Nodes Positive Betweenness Centrality with non-unique shortest paths (the answer for a node b is YES if and only if the associate approximate value is strictly positive). Notice that in the reduction of Lemma 4.7 all weights are positive; hence this implies a truly subcubic algorithm for Negative Triangle. The claim follows by the subcubic equivalence between Negative Triangle and APSP [64]. □

4.2 A PTAS for Betweenness Centrality

In this section we prove the subcubic equivalence between Approximate Betweenness Centrality and Diameter.

Theorem 4.9. *Diameter and Approximate Betweenness Centrality with unique shortest paths are equivalent under subcubic high-probability Monte-Carlo reductions.*

The main result in this section is the proof of the following lemma.

Lemma 4.10. *Given a truly subcubic algorithm for Diameter, there exists a truly subcubic high-probability Monte-Carlo PTAS for Betweenness Centrality with unique shortest paths.*

We recall that a PTAS for the problem of estimating a value X is an algorithm that takes in input an instance of the problem and a parameter $\varepsilon > 0$ and outputs a $(1+\varepsilon)$ approximation x or X, i.e., $\frac{1}{1+\varepsilon} X \le x \le (1+\varepsilon) X$. Furthermore, the running time of the algorithm is polynomial whenever ε is lower bounded by some constant. The proof of Theorem 4.9 follows easily.

Proof of Theorem 4.9. Lemma 4.10 gives one direction. The other direction is obtained by chaining Lemmas 4.1 and 4.4. □

It remains to prove Lemma 4.10. Let (G, w, b) be the considered instance of Betweenness Centrality, and define $B^* = BC(b)$. Observe that, under the assumption that shortest paths are unique, $BC_{s,t}(b) \in \{0, 1\}$ and therefore $B^* \in \{0, \ldots, (n-1)(n-2)\}$. Given $s, t \in V - \{b\}$ such that $BC_{s,t}(b) = 1$, we call (s, t) a *witness pair*, s a *witness source*, and t a *witness target* (of $BC(b)$).

Let also $B_{med} \in \{0, \ldots, (n-1)(n-2)\}$ be an integer parameter to be fixed later. Our PTAS is based on two different algorithms: one for $B^* \le B_{med}$ (*small* B^*) and the other for $B^* > B_{med}$ (*large* B^*).

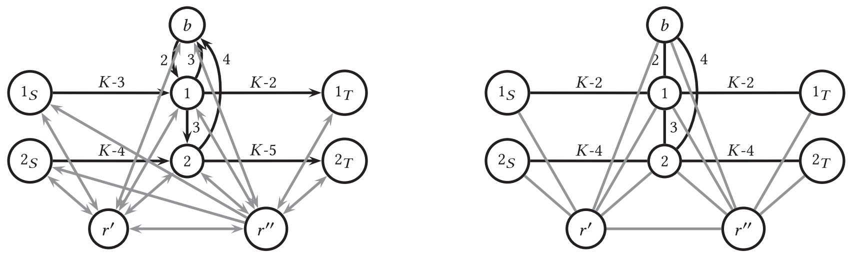

Fig. 6. Reduction from Positive (S, T)-Betweenness Centrality to Diameter with $S = T = \{1, 2\}$. Gray edges have weight $K - 1$. On the left and right are the reduction for the directed and undirected case, respectively.

4.2.1 An Exact Algorithm for Small B^*. Let us start with the algorithm for small B^*. Recall that a witness pair (s, t) satisfies $BC_{s,t}(b) = 1$. A crucial observation is that the number of witness pairs is equal to B^* in case of unique shortest paths.

It is convenient to define a generalization of Betweenness Centrality, where we consider only some pairs (s, t). For $S, T \subseteq V - \{b\}$, we define $BC_{S,T}(b) := \sum_{(s,t) \in S \times T} BC_{s,t}(b)$. The (S, T)-*Betweenness Centrality* problem is to compute $BC_{S,T}(b)$. The *Positive (S, T)-Betweenness Centrality* problem is to determine whether $BC_{S,T}(b) > 0$. We use the shortcuts $BC_{s,T}(b) = BC_{\{s\},T}(b)$ and $BC_{S,t}(b) = BC_{S,\{t\}}(b)$. Our first ingredient is a reduction of Positive (S, T)-Betweenness Centrality to Diameter.

Lemma 4.11. *Given a $T(n, m)$ time algorithm for Diameter in directed or undirected graphs, there exists a $\tilde{O}(T(n, m))$ time algorithm for Positive (S, T)-Betweenness Centrality with non-unique (hence unique) shortest paths in the same graph class.*

Proof. We use a construction similar to the one in the proof of Lemma 4.5 (see also Figure 6). Let (G, w, b, S, T) be the considered instance of Positive (S, T)-Betweenness Centrality.

We start with the directed case. Let us construct a directed weighted graph (G', w'). Graph G' contains a copy of G. Furthermore, it contains a copy S' of S and a copy T' of T. Let v_S be the copy of node v in S, and define v_T analogously. Let $K := 2 + A$, where A is the maximum distance of type $d_G(s, b)$ and $d_G(b, t)$, with $s \in S$ and $t \in T$. For each $s \in S$ and $t \in T$, we add edges $s_S s$ and $t t_T$ of weight $K - d_G(s, b)$ and $K - d_G(b, t)$, respectively. We add one dummy node r' (resp., r'') and bidirected[7] edges $r'v$ for all $v \in S' \cup V$ (resp., $r''v$ for all $v \in T' \cup V$). We also add edges $r''v$ for each $v \in S'$ (in particular these edges are not bidirected). Finally, we add bidirected edges $r'r''$. All edges incident on r' and r'' have weight $K - 1$ (dummy edges). We compute the diameter D^* of (G', w') and output YES if and only if $D^* = 2K$.

The running time of the algorithm is $\tilde{O}(m + T(O(n), O(m))) = \tilde{O}(T(n, m))$. Let us prove its correctness. Let s^*, t^* be a witness pair for the diameter. If $s^* \in V \cup T' \cup \{r', r''\}$, then $D^* \leq 2(K-1)$. Hence we can assume $s^* = s_S \in S'$ for some $s \in S$. If $t^* \in S' \cup V \cup \{r', r''\}$, then $D^* \leq 2(K - 1)$. So we can also assume $t^* = t_T \in T'$.

Any s_S-t_T path using dummy edges has to use at least two such edges. If it uses three such edges, it costs at least $3(K-1) > 2K$. Otherwise, it costs at least $K - d_G(s, b) + 2(K-1) \geq K - A + 2(K-1) = 2K$ or $2(K - 1) + K - d_G(b, t) \geq K - A + 2(K - 1) = 2K$. Any shortest s_S-t_T avoiding dummy edges has cost $2K - d_G(s, b) - d_G(b, t) + d_G(s, t) \leq 2K$, where the equality holds if and only if b belongs to some shortest s-t path in G. Summarizing, if there exists a shortest s-t path passing through b (in which case the answer is YES), then the diameter is $2K$. Otherwise, the diameter is at most $2K - 1$.

[7]By a bidirected edge uv of weight w, we mean a directed edge uv and a directed edge vu, both of weight w.

The construction for the undirected case is similar, where we remove edge directions (leaving one copy of parallel edges) and the edges of type $r''v$ with $v \in S'$. By the same argument as before, we can assume that $s^*, t^* \in S' \cup T'$ and furthermore they do not belong simultaneously to S' or to T' (otherwise, $D^* \leq 2(K-1)$). Thus, modulo switching the endpoints (which is w.l.o.g. in the undirected case), we can assume $s^* = s_S \in S'$ and $t^* = t_T \in T'$. Then by the same argument as before, one has that the diameter is $2K$ if there exists a shortest s-t path passing through b (in which case the answer is YES), and otherwise the diameter is at most $2K-1$. □

We will exploit the following recursive algorithm for (S,T)-Betweenness Centrality.

Lemma 4.12. *Given a $T(n,m)$ time algorithm for Diameter in directed (resp., undirected) graphs, there is a $\tilde{O}(W \cdot T(n,m))$ time algorithm for (S,T)-Betweenness Centrality with unique shortest paths, where W is the number of pairs $(s,t) \in S \times T$ such that $BC_{s,t}(b) = 1$.*

Proof. We describe a recursive algorithm with the claimed running time, given a $\tilde{O}(T(n,m))$ time algorithm for Positive (S,T)-Betweenness Centrality. The claim follows from Lemma 4.11.

The recursive algorithm works as follows. It initially solves the corresponding Positive (S,T)-Betweenness instance. If the answer is NO, the algorithm outputs 0. If the answer is YES, we distinguish two subcases. If $|S| = |T| = 1$, the algorithm outputs 1. Otherwise, the algorithm partitions arbitrarily S into two subsets S_1 and S_2, which differ by at most 1 in cardinality, and it splits similarly T into T_1 and T_2. Then the algorithm solves recursively the sub-problem induces by the pairs (S_i, T_j), $i,j \in \{1,2\}$ and outputs the sum of the four obtained values.

The correctness of the algorithm is obvious. Concerning its running time, consider the recursion tree. Let us call a subproblem whose corresponding Positive (S,T)-Betweenness Centrality instance is a YES/NO instance a YES/NO subproblem. Observe that, excluding the root problem, any NO subproblem must have at least one sibling YES subproblem in the recursion tree. Furthermore, each sub-problem has at most four children in the recursion tree. Therefore, if the root subproblem is a YES subproblem, the total number of subproblems is at most 4 times the number of YES subproblems. Note also that the number of leaf YES subproblems is equal to W, and that each YES subproblem must have at least one leaf YES subproblem among its descendants. Finally, the depth of the recursion tree is $O(\log(|S|+|T|)) = O(\log n)$. Thus the number of subproblems is $\tilde{O}(W)$. The claim on the running time follows. □

We are now ready to present our algorithm for small B^*.

Lemma 4.13. *Given an instance (G,w,b) of Betweenness Centrality with unique shortest paths, a parameter B_{med}, and an algorithm for Diameter of running time $T(n,m)$, there is an $\tilde{O}(B_{med}T(n,m))$ time algorithm that either outputs $B^* = BC(b)$ or answers NO, in which case $B^* > B_{med}$.*

Proof. Consider the recursive algorithm from Lemma 4.12. We run that algorithm with $S = T = V$, however with the following modifications. We keep track of the number W of leaf YES sub-problems found so far. If $W > B_{med}$ at any point, we halt the recursive algorithm and output NO. Otherwise, we output the value W returned by the root call of the recursive algorithm.

The correctness of the algorithm follows immediately since the number of leaf YES subproblems in the original (non-truncated) algorithm equals B^*. An easy adaptation of the running time analysis in Lemma 4.12 shows that the running time is as in the claim (in particular, the number of recursive calls is $O(B_{med})$). □

4.2.2 A Monte-Carlo PTAS for Large B^*. We next assume that $B^* > B_{med}$, and we present an algorithm for this case. In order to lighten the notation, since b is clear from the context, we next use $BC_{S,T}$ instead of $BC_{S,T}(b)$ and similarly for related notation. Observe that a node w is a witness source (resp., witness target) if $BC_{w,V} > 0$ (resp., $BC_{V,w} > 0$). At high level, our algorithm is based on the computation of the contribution $BC_{s,V}$ to BC of a random sample of candidate witness sources s. Then we exploit Chernoff's bound to prove that the approximation factor is small w.h.p. One technical difficulty here is that some witness sources might give a very large contribution to BC, which is problematic since we need concentrated results. In order to circumvent this problem, we first sample a random subset of candidate witness targets to identify the problematic witness sources (which are considered separately).

In more detail, we sample a random subset T of $p_{med} \cdot n$ nodes, where $p_{med} = \frac{C \log n}{\sqrt{B_{med}}}$ and C is a sufficiently large constant (more precisely $C = O(1/\varepsilon^2)$ is sufficient). We compute all the shortest paths ending in T and use them to derive $BC_{s,T}$ for all $s \in V$. We partition V into sets S_{large} and S_{small}, where $s \in V$ belongs to S_{large} if and only if $BC_{s,T} \geq C \log n$. Then we sample a random subset R_{small} of $p_{med}|S_{small}|$ nodes in S_{small} and compute $BC_{s,V}$ for all $s \in R_{small}$. Finally, we output the estimate

$$\tilde{B} = \frac{1}{p_{med}} \left(\sum_{s \in S_{large}} BC_{s,T} + \sum_{s \in R_{small}} BC_{s,V} \right).$$

It is easy to see that the running time of the algorithm is $\tilde{O}(\frac{Cnm}{\sqrt{B_{med}}})$. It is also not hard to see that $E[\frac{1}{p_{med}} \sum_{s \in S_{large}} BC_{s,T}] = \sum_{s \in S_{large}} BC_{s,V}$ and $E[\frac{1}{p_{med}} \sum_{s \in R_{small}} BC_{s,V}] = \sum_{s \in S_{small}} BC_{s,V}$. Therefore, $E[\tilde{B}] = B^*$. The following lemma shows that $\tilde{B}$ is concentrated around its mean.

Lemma 4.14. *For $C = O(1/\varepsilon^2)$ large enough, w.h.p. $\tilde{B} \in [(1-2\varepsilon)B^*, (1+2\varepsilon)B^*]$.*

Proof. We start by showing that w.h.p., for any $s \in V$, if $s \in S_{large}$, then $BC_{s,V} \geq \sqrt{B_{med}}/(1+\varepsilon)$, and otherwise $BC_{s,V} \leq \sqrt{B_{med}}/(1-\varepsilon)$. Define $B' = BC_{s,T}$ and $B = BC_{s,V}$. Note that $E[B'] = \frac{C \log n}{\sqrt{B_{med}}} B$. Note also that $B' = BC_{s,T} = \sum_{t \in V} X_{s,t}$, where $X_{s,t} = 0$ if $t \notin T$ and $X_{s,t} = BC_{s,t}$ otherwise. Since the variables $X_{s,t}$ are negatively correlated, we can apply Chernoff's bound to $BC_{s,T}$. In particular, conditioning implicitly on $B < \frac{\sqrt{B_{med}}}{1+\varepsilon}$, one obtains

$$Pr[B' \geq C \log n] = Pr\left[B' \geq \frac{\sqrt{B_{med}}}{B} E[B']\right] \leq \left(\frac{e^{(\sqrt{B_{med}}/B)-1}}{(\sqrt{B_{med}}/B)^{\sqrt{B_{med}}/B}}\right)^{\frac{C \log n}{\sqrt{B_{med}}} B}$$
$$\leq \left(\frac{e^{\varepsilon/(1+\varepsilon)}}{1+\varepsilon}\right)^{C \log n}.$$

Above we used the fact that the function xe^{1-x} is increasing for $x \in [0, \frac{1}{1+\varepsilon}]$ (and strictly smaller than 1 in the same range). Similarly, conditioning implicitly on the event that $B > \frac{\sqrt{B_{med}}}{1-\varepsilon}$, one obtains $E[B'] = \frac{C \log n}{\sqrt{B_{med}}} B \geq \frac{C \log n}{1-\varepsilon}$ and

$$Pr[B' < C \log n] = Pr\left[B' < \frac{\sqrt{B_{med}}}{B} E[B']\right] \leq Pr[B' \leq (1-\varepsilon)E[B']]$$
$$\leq e^{-\frac{\varepsilon^2 E[B']}{2}} \leq e^{-\frac{\varepsilon^2 C \log n}{2(1-\varepsilon)}}.$$

Thus summarizing, for a fixed s,

$$Pr\left[s \in S_{large} | B_{s,V} < \frac{\sqrt{B_{med}}}{1+\varepsilon}\right] \leq \left(\frac{e^{\varepsilon/(1+\varepsilon)}}{1+\varepsilon}\right)^{C \log n}, \text{ and}$$
$$Pr\left[s \in S_{small} | B_{s,V} > \frac{\sqrt{B_{med}}}{1-\varepsilon}\right] \leq e^{-\frac{\varepsilon^2 C \log n}{2(1-\varepsilon)}}.$$

From the union bound and assuming that the constant $C = O(1/\varepsilon^2)$ is large enough, we conclude that w.h.p. for all $s \in S_{large}$ one has $BC_{s,V} \geq \sqrt{B_{med}}/(1+\varepsilon)$ and for all $s \in S_{small}$ one has $BC_{s,V} \leq \sqrt{B_{med}}/(1-\varepsilon)$.

Next assume that the mentioned high-probability event happens for all $s \in V$. Define $B^*_{large} = \sum_{s \in S_{large}} BC_{s,V}$ and $B^*_{small} = \sum_{s \in S_{small}} BC_{s,V}$. Clearly $B^* = B^*_{large} + B^*_{small}$. Define also $\tilde{B}_{large} := \frac{1}{p_{med}} \sum_{s \in S_{large}} BC_{s,T}$ and $\tilde{B}_{small} := \frac{1}{p_{med}} \sum_{s \in R_{small}} BC_{s,V}$, so that $\tilde{B} = \tilde{B}_{large} + \tilde{B}_{small}$.

Consider any $s \in S_{large}$, and define $B' = BC_{s,T}$ and $B = BC_{s,V}$. Recall that by assumption $B \geq \frac{\sqrt{B_{med}}}{1+\varepsilon}$ and observe that $E[B'] = p_{med} B \geq \frac{C \log n}{1+\varepsilon}$. Then, by Chernoff's bound,

$$Pr[|B' - E[B']| \geq \varepsilon E[B']] \leq 2e^{-\frac{\varepsilon^2}{3} E[B']} \leq 2e^{-\frac{\varepsilon^2}{3(1+\varepsilon)} C \log n}.$$

Since $E[\tilde{B}_{large}] = \frac{1}{p_{med}}[\sum_{s \in S_{large}} BC_{s,T}] = B^*_{large}$, we can conclude that w.h.p. $\tilde{B}_{large} \in [(1-\varepsilon)B^*_{large}, (1+\varepsilon)B^*_{large}]$.

Consider next $\tilde{B}_{small}$. Define $B' = p_{med}\tilde{B}_{small} = \sum_{s \in R_{small}} BC_{s,V}$. Observe that $E[B'] = p_{med} B^*_{small}$. Furthermore, B' is the sum of independent random variables each one of value at most $\frac{\sqrt{B_{med}}}{1-\varepsilon}$ by the assumption on S_{small}. Therefore, by Chernoff's bound,

$$Pr[B' \geq E[B'] + \varepsilon p_{med} B^*] \leq \left(\frac{e^{\frac{\varepsilon B^*}{B^*_{small}}}}{\left(\frac{\varepsilon B^*}{B^*_{small}} + 1\right)^{\frac{\varepsilon B^*}{B^*_{small}} + 1}} \right)^{\frac{(1-\varepsilon) C \log n B^*_{small}}{B_{med}}}.$$

Assuming $B^*_{small} \geq \varepsilon B_{med}/2$ and observing that $B^* \geq B^*_{small}$, one obtains

$$Pr[B' \geq E[B'] + \varepsilon p_{med} B^*] \leq \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}} \right)^{\frac{(1-\varepsilon)\varepsilon C \log n}{2}}.$$

Otherwise, $B^*_{small} < \varepsilon B_{med}/2 \leq \varepsilon B^*/2$ and thus

$$Pr[B' \geq E[B'] + \varepsilon p_{med} B^*] \leq \left(\frac{e^{\varepsilon}}{\left(1 + \frac{\varepsilon B^*}{B^*_{small}}\right)^{\frac{B^*_{small}}{B^*} + \varepsilon}} \right)^{\frac{(1-\varepsilon) C \log n B^*}{B_{med}}} \leq \left(\frac{e}{3}\right)^{\varepsilon(1-\varepsilon) C \log n}.$$

Similarly,

$$\begin{aligned} Pr[B' \leq E[B'] - \varepsilon p_{med} B^*] &\leq e^{-\frac{1}{2}\left(\frac{\varepsilon B^*}{B^*_{small}}\right)^2 \frac{p_{med} B^*_{small}}{\sqrt{B_{med}}/(1-\varepsilon)}} \\ &= e^{-\frac{(1-\varepsilon)\varepsilon^2}{2} \frac{(B^*)^2}{B^*_{small}} \frac{C \log n}{B_{med}}} \leq e^{-\frac{(1-\varepsilon)\varepsilon^2}{2} C \log n}. \end{aligned}$$

Therefore, w.h.p. $\tilde{B}_{small} \in [B^*_{small} - \varepsilon B^*, B^*_{small} + \varepsilon B^*]$. Altogether, w.h.p. one has

$$\begin{aligned} (1-2\varepsilon)B^* &\leq (1-\varepsilon)B^*_{large} + B^*_{small} - \varepsilon B^* \leq \tilde{B} \\ &\leq (1+\varepsilon)B^*_{large} + B^*_{small} + \varepsilon B^* \leq (1+2\varepsilon)B^*. \end{aligned}$$ □

The following lemma summarizes the above discussion.

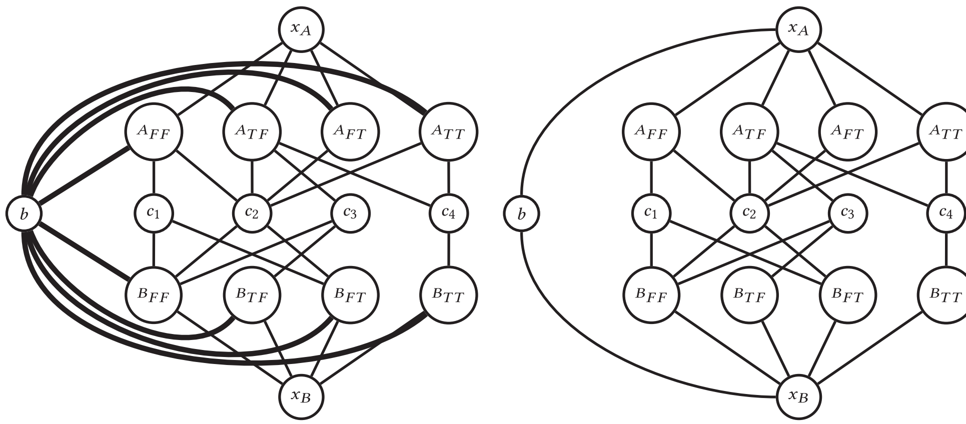

Fig. 7. Reduction from CNF-SAT to Positive Betweenness Centrality (left) and Reach Centrality (right) in undirected graphs for the CNF-SAT formula $c_1 \wedge c_2 \wedge c_3 \wedge c_4 = (X \vee Y \vee Z) \wedge (Z \vee \overline{Y}) \wedge (\overline{X} \vee Y \vee Q) \wedge (\overline{X} \vee \overline{Z} \vee \overline{Q})$. The set of variables are $A = \{X, Y\}$ and $B = \{Z, Q\}$. Node A_{FF} corresponds to the partial assignment $(X, Y) = (F, F)$ and similarly for the other nodes. Bold edges have weight 2; all other edges have weight 1. The shortest paths A_{FF}, b, B_{TT} on the left and $A_{FF}, x_A, b, x_B, B_{TT}$ on the right witness that $(X, Y, Z, Q) = (F, F, T, T)$ is a satisfying assignment.

Lemma 4.15. *Given an instance (G, w, b) of Betweenness Centrality with unique shortest paths and $BC(b) = B^* \geq B_{med}$, there is an $\tilde{O}(\frac{nm}{\varepsilon^2\sqrt{B_{med}}})$ time algorithm that returns a $(1+\varepsilon)$ approximation of B^* w.h.p.*

Proof. Consider the above algorithm. Its running time is $\tilde{O}(\frac{nm}{\varepsilon^2\sqrt{B_{med}}})$ since $C = O(\frac{1}{\varepsilon^2})$. By Lemma 4.14, the estimate $\tilde{B}$ of B^* that it outputs satisfies the claim (modulo scaling ε by a constant factor). □

Combining the algorithms for small and large B^*, we obtain Lemma 4.10.

Proof of Lemma 4.10. Let $\tilde{O}(n^{3-\delta})$ be the running time of the given Diameter algorithm, for some constant $\delta > 0$. From Lemmas 4.13 and 4.15, we can use it to compute w.h.p. a $(1+\varepsilon)$ approximation of the betweenness centrality of a given node in time $\tilde{O}(B_{med}n^{3-\delta} + \frac{n^3}{\varepsilon^2\sqrt{B_{med}}})$. Choosing $B_{med} = \frac{n^{2\delta/3}}{\varepsilon^{4/3}}$ gives a truly subcubic running time in $\tilde{O}(\frac{n^{3-\delta/3}}{\varepsilon^{4/3}})$. □

4.3 Reductions Based on SETH

We are able to show that, assuming the SETH [40], a subquadratic algorithm for Positive Betweenness Centrality does not exist even in sparse graphs. We recall that SETH claims that CNF-SAT on n variables cannot be solved in time $O((2-\delta)^n)$ for any constant $\delta > 0$. One obtains as a corollary a lower bound on the running time of any approximation algorithm for Betweenness/Reach Centrality by the reductions in Lemmas 4.1 and 4.2.

Theorem 4.16. *Suppose that there is an $O(m^{2-\varepsilon})$ time algorithm, for some constant $\varepsilon > 0$, that solves Positive Betweenness Centrality with non-unique shortest paths in directed or undirected graphs with edge weights in $\{1, 2\}$. Then SETH is false.*

Proof. Let F be a CNF-SAT formula on n variables. Our goal is to show that we can determine whether F is satisfiable in $O^*(2^{(1-\delta)n})$ time[8] for some constant $\delta > 0$. Using the sparsification lemma of [40] (as, e.g., in [14]), we can assume w.l.o.g. that F contains $O(n)$ clauses.

Let us consider the undirected case first (see also Figure 7). We partition the variables into two sets A and B, which differ by at most 1 in cardinality, and create a node ϕ_A (resp., ϕ_B) for each partial assignment ϕ_A of the variables in A (resp., ϕ_B of the variables in B). We also add a node for each clause c and add one edge of weight 1 between each clause c and any partial assignment ϕ of A or B that does *not* satisfy any literal of c (including the special case that c does not contain any variable in A or B). We also add two nodes x_A and x_B and add one edge of weight 1 between them and any node in A and B, respectively. Finally, we add a node b and add one edge of weight 2 between b and each assignment of A and B. The algorithm returns YES (i.e., F is satisfiable) if and only if $BC(b) > 0$.

Let us prove correctness. The distance between any clause node c and any other node is at most 4, while any path passing through b would cost at least 5. Hence the corresponding shortest paths do not use b. The same claim holds for x_A and x_B. The distance between any two assignments of A or of B is at most 2, while passing through b would cost at least 4. Hence also the corresponding shortest paths do not use b. It remains to consider shortest paths from some node of type ϕ_A to some node of type ϕ_B. Observe that there exists one such path of length 2 (hence $BC_{\phi_A,\phi_B}(b) = 0$) if and only if there exists a clause c that is not satisfied by ϕ_A or by ϕ_B. Otherwise (i.e., ϕ_A and ϕ_B together satisfy F), ϕ_A, b, ϕ_B is a shortest such path (hence $BC(b) > 0$). The graph has $O(2^{n/2}n)$ edges, leading to a running time of the form $O^*(2^{(1-\varepsilon/2)n})$. The claim follows.

In the directed case we use a similar construction (with a similar notation), without nodes x_A and x_B, and orienting the edges from the assignments of A to the clause nodes and to b, and from the latter nodes to the assignments of B. The algorithm is the same. The proof of correctness is simpler: the only shortest paths that can use b are from a node of type ϕ_A to a node of type ϕ_B. Similarly to the undirected case, ϕ_A and ϕ_B together satisfy F if and only if ϕ_A, b, ϕ_B is a shortest path (hence $BC(b) > 0$). Also in this case the running time is $O^*(2^{(1-\varepsilon/2)n})$, implying the claim. □

Corollary 4.17. *Suppose that there is an $O(m^{2-\varepsilon})$ time algorithm for Approximate Betweenness Centrality with non-unique shortest paths or for Approximate Reach Centrality, for some constant $\varepsilon > 0$. Then SETH is false.*

Proof. It follows by chaining Theorem 4.16 with Lemmas 4.1 and 4.2. □

For Reach Centrality we can also show an approximation lower bound for unweighted undirected graphs.

Theorem 4.18. *Suppose there is an $O(m^{2-\varepsilon})$ time $(2-\varepsilon)$-approximation algorithm for Reach Centrality in undirected unweighted graphs, for some constant $\varepsilon > 0$. Then SETH is false.*

Proof. Similarly to the proof of Theorem 4.16, we can start with a CNF-SAT formula F containing n variables and $m = O(n)$ clauses [40]. We will show how to construct an instance (G, b) of Reach Centrality on an unweighted undirected graph $G = (V, E)$ with $|V| = O(2^{n/2} + m)$ nodes and $|E| = O(2^{n/2}m)$ edges, such that $RC(b) = 2$ if F is satisfiable and $RC(b) = 1$ otherwise. The generation of the graph from the formula takes $O(2^{n/2}m)$ time and therefore if we could compute a $(2-\varepsilon)$ approximation of $RC(b)$ in $O^*(|E|^{2-\varepsilon})$ time, for some $\varepsilon > 0$, we would be able to solve CNF-SAT in $O^*(2^{(1-\varepsilon/2)n})$ time (which would refute SETH).

[8]The O^* notation suppresses polynomial factors.

Similarly to the proof of Theorem 4.16, we partition the variables into two subsets A and B that differ by at most 1 in cardinality and create a node for each partial assignment of the variables in A and B. We also create a node c for each clause c and connect c to each partial assignment that does not satisfy any literal in c. We also add nodes x_A and x_B and add edges between them and any node in A and B, respectively. Finally, we add a node b and connect it to x_A and x_B (note that the final part of the construction deviates from Theorem 4.16).

To show correctness, note that b is on the shortest path between x_A and x_B and therefore $RC(b) \geq 1$. Furthermore, b cannot be on the shortest path between a clause node c and another node in G, and therefore $RC(b) = 2$ if and only if b is on the shortest path between an assignment ϕ_A of A and an assignment ϕ_B of B. But a shortest path between ϕ_A and ϕ_B goes through b if and only if for every clause node c either $\phi_A c$ is not an edge or $\phi_B c$ is not an edge. By definition of these edges, this implies that for every clause c, either ϕ_A or ϕ_B satisfies c (i.e., ϕ_A and ϕ_B induce a satisfying assignment of F). The claim follows. □

As observed by one careful reviewer, the above reductions can be adapted to the **Orthogonal Vector Conjecture (OVC)**. In the **Orthogonal Vector (OV)** problems we are given a set on n binary vectors of dimension $D = O(\log n)$. The goal is to determine whether there exists a pair of orthogonal vectors in the set. OVC states that there is no $O(n^{2-\delta})$ time algorithm for OV where $\delta > 0$ is a fixed constant. We remark that SETH implies OVC; i.e., OVC is a stronger conjecture [60]. Our reductions can be adapted as follows. For each vector v we create a node v_A in the set A (resp., v_B in the set B). The set C contains one node w for each dimension/entry w. We connect each vector node $v_A \in A$ (resp., $v_B \in B$) to each dimension node w such that the wth entry of v is 1. Now a length 2 path between $v_A \in A$ and $u_B \in B$ through a node in C means that the vectors v and u are *not* orthogonal. The rest of the construction is similar. The simple details are left to the reader.

5 CONCLUSIONS AND OPEN PROBLEMS

There are many interesting problems that we left open. The main one is probably whether Diameter and APSP are equivalent under subcubic reductions. By our reductions, on one hand a positive answer would indicate that truly subcubic algorithms for Reach Centrality and for Approximate Betweenness Centrality are unlikely to exist. On the other hand, a negative answer would give truly subcubic algorithms for the latter problems as well.

We have shown that Reach Centrality can be solved in $\tilde{O}(Mn^{\omega})$ time in directed graphs, improving on the previous best algorithm based on APSP. Similar running times are known for Diameter and Radius [17]. To the best of our knowledge, it is open whether an $\tilde{O}(Mn^{\omega})$ time algorithm exists also for Median and Betweenness Centrality in directed graphs.

We proved that a subquadratic $2 - \varepsilon$ approximation algorithm for Reach Centrality in sparse graphs is unlikely to exist. In [2, 52] analogous results are proved for Diameter and Radius. It would be interesting to show similar negative results for Betweenness Centrality and Median (or find faster approximation algorithms in sparse graphs for those problems).

ACKNOWLEDGMENTS

We thank the anonymous reviewers for helpful suggestions.

REFERENCES

[1] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science (FOCS'14)*. IEEE Computer Society, 434–443. https://doi.org/10.1109/FOCS.2014.53

[2] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. 2016. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, Robert Krauthgamer (Ed.). SIAM, 377–391. https://doi.org/10.1137/1.9781611974331.ch28

[3] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. 2014. Consequences of faster alignment of sequences. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP'14), Part I (Lecture Notes in Computer Science)*, Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias (Eds.), Vol. 8572. Springer, 39–51. https://doi.org/10.1007/978-3-662-43948-7_4

[4] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. 1999. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.* 28, 4 (1999), 1167–1181.

[5] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. 2021. Toward tight approximation bounds for graph diameter and eccentricities. *SIAM J. Comput.* 50, 4 (2021), 1155–1199. https://doi.org/10.1137/18M1226737

[6] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. 2007. Approximating betweenness centrality. In *Proceedings of the 5th International Workshop on Algorithms and Models for the Web-Graph (WAW'07) (Lecture Notes in Computer Science)*, Anthony Bonato and Fan R. K. Chung (Eds.), Vol. 4863. Springer, 124–137. https://doi.org/10.1007/978-3-540-77004-6_10

[7] Piotr Berman and Shiva Prasad Kasiviswanathan. 2007. Faster approximation of distances in graphs. In *Proceedings on the 10th International Workshop on Algorithms and Data Structures, (WADS'07) (Lecture Notes in Computer Science)*, Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh (Eds.), Vol. 4619. Springer, 541–552. https://doi.org/10.1007/978-3-540-73951-7_47

[8] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* 25, 2 (2001), 163–177.

[9] U. Brandes and C. Pich. 2007. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos* 17, 7 (2007), 2303–2318.

[10] Massimo Cairo, Roberto Grossi, and Romeo Rizzi. 2016. New bounds for approximating extremal distances in undirected graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, Robert Krauthgamer (Ed.). SIAM, 363–376. https://doi.org/10.1137/1.9781611974331.ch27

[11] Timothy M. Chan. 2010. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.* 39, 5 (2010), 2075–2089.

[12] Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. 2021. Algorithms, reductions and equivalences for small weight variants of all-pairs shortest paths. In *48th International Colloquium on Automata, Languages, and Programming (ICALP'21) (Leibniz International Proceedings in Informatics (LIPIcs))*, Nikhil Bansal, Emanuela Merelli, and James Worrell (Eds.), Vol. 198. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 47:1–47:21. https://doi.org/10.4230/LIPIcs.ICALP.2021.47

[13] Ching-Lueh Chang. 2013. Deterministic sublinear-time approximations for metric 1-median selection. *Inf. Process. Lett.* 113, 8 (2013).

[14] Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. 2014. Better approximation algorithms for the graph diameter. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, Chandra Chekuri (Ed.). SIAM, 1041–1052. https://doi.org/10.1137/1.9781611973402.78

[15] T. Coffman, S. Greenblatt, and S. Marcus. 2004. Graph-based technologies for intelligence analysis. *Commun. ACM* 47, 3 (2004), 45–47.

[16] D. Coppersmith and S. Winograd. 1990. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation* 9, 3 (1990), 251–280.

[17] Marek Cygan, Harold N. Gabow, and Piotr Sankowski. 2012. Algorithmic applications of Baur-Strassen's theorem: Shortest cycles, diameter and matchings. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*. IEEE Computer Society, 531–540. https://doi.org/10.1109/FOCS.2012.72

[18] Mina Dalirrooyfard, Ray Li, and Virginia Vassilevska Williams. 2021. Hardness of approximate diameter: Now for undirected graphs. In *62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS'21)*. IEEE, 1021–1032. https://doi.org/10.1109/FOCS52979.2021.00102

[19] A. M. Davie and A. J. Stothers. 2013. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh, Section: A Mathematics* 143, 2 (4 2013), 351–369. https://doi.org/10.1017/S0308210511001648

[20] A. Del Sol, H. Fujihashi, and P. O'Meara. 2005. Topology of small-world networks of protein- protein complex structures. *Bioinformatics* 21, 8 (2005), 1311–1315.

[21] E. W. Dijkstra. 1959. A note on two problems in Connexion with graphs. *Numer. Math.* 1 (1959), 269–271.

[22] David Eppstein and Joseph Wang. 2004. Fast approximation of centrality. *J. Graph Algorithms Appl.* 8 (2004), 39–45.

[23] Michael L. Fredman. 1976. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.* 5, 1 (1976), 83–89.

[24] Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34, 3 (1987), 596–615.

[25] Linton Freeman. 1977. A set of measures of centrality based upon betweenness. *Sociometry* 40 (1977), 35–41.

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

[27] François Le Gall. 2014. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation (ISSAC'14)*, Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó (Eds.). ACM, 296–303. https://doi.org/10.1145/2608628.2608664

[28] Francois Le Gall and Florent Urrutia. 2018. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, Artur Czumaj (Ed.). SIAM, 1029–1046. https://doi.org/10.1137/1.9781611975031.67

[29] Robert Geisberger, Peter Sanders, and Dominik Schultes. 2008. Better approximation of betweenness centrality. In *Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX'08)*, J. Ian Munro and Dorothea Wagner (Eds.). SIAM, 90–100. https://doi.org/10.1137/1.9781611972887.9

[30] Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. 2006. Reach for A*: Efficient point-to-point shortest path algorithms. In *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX'06)*, Rajeev Raman and Matthias F. Stallmann (Eds.). SIAM, 129–143. https://doi.org/10.1137/1.9781611972863.13

[31] Andrew V. Goldberg, Haim Kaplan, and Renato Fonseca F. Werneck. 2007. Better landmarks within reach. In *Proceedings of the 6th International Workshop on Experimental Algorithms (WEA'07) (Lecture Notes in Computer Science)*, Camil Demetrescu (Ed.), Vol. 4525. Springer, 38–51. https://doi.org/10.1007/978-3-540-72845-0_4

[32] Oded Goldreich and Dana Ron. 2008. Approximating average parameters of graphs. *Random Struct. Algorithms* 32, 4 (2008), 473–493.

[33] Fabrizio Grandoni and Virginia Vassilevska Williams. 2020. Faster replacement paths and distance sensitivity oracles. *ACM Transactions on Algorithms* 16, 1 (2020), 15:1–15:25.

[34] Fabrizio Grandoni and Virginia Vassilevska Williams. 2012. Improved distance sensitivity oracles via fast single-source replacement paths. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*. IEEE Computer Society, 748–757. https://doi.org/10.1109/FOCS.2012.17

[35] Ronald J. Gutman. 2004. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments and the 1st Workshop on Analytic Algorithmics and Combinatorics*, Lars Arge, Giuseppe F. Italiano, and Robert Sedgewick (Eds.). SIAM, 100–111.

[36] P. Hage and F. Harary. 1995. Eccentricity and centrality in networks. *Social Networks* 17 (1995), 57–63.

[37] S. Louis Hakimi. 1964. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research* 12, 3 (1964), 450–459.

[38] Yijie Han and Tadao Takaoka. 2016. An O($n^3 \log\log n / \log^2 n$) time algorithm for all pairs shortest paths. *J. Discrete Algorithms* 38–41 (2016), 9–19. https://doi.org/10.1016/j.jda.2016.09.001

[39] Xiaohan Huang and Victor Y. Pan. 1998. Fast rectangular matrix multiplication and applications. *J. Complexity* 14, 2 (1998), 257–299.

[40] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* 63, 4 (2001), 512–530.

[41] Piotr Indyk. 1999. Sublinear time algorithms for metric space problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton (Eds.). ACM, 428–434. https://doi.org/10.1145/301250.301366

[42] H. Jeong, S. Mason, A. L. Barabási, and Z. Oltvai. 2001. Lethality and centrality in protein networks. *Nature* 411 (2001), 41–42.

[43] Donald B. Johnson. 1977. Efficient algorithms for shortest paths in sparse networks. *J. ACM* 24, 1 (1977), 1–13.

[44] V. Krebs. 2002. Mapping networks of terrorist cells. *Connections* 24, 3 (2002), 43–52.

[45] F. Liljeros, C. Edling, L. Amaral, H. Stanley, and Y. Aberg. 2001. The web of human sexual contacts. *Nature* 411 (2001), 907–908.

[46] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. 1987. Matching is as easy as matrix inversion. *Combinatorica* 7, 1 (1987), 105–113.

[47] M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2 (2004), 26–113.

[48] Mihai Patrascu. 2010. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC'10)*, Leonard J. Schulman (Ed.). ACM, 603–610. https://doi.org/10.1145/1806689.1806772

[49] Seth Pettie. 2004. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.* 312, 1 (2004), 47–74.

[50] Seth Pettie and Vijaya Ramachandran. 2005. A shortest path algorithm for real-weighted undirected graphs. *SIAM J. Comput.* 34, 6 (2005), 1398–1431.

[51] J. W. Pinney and D. R. Westhead. 2006. Betweenness-based decomposition methods for social and biological networks. In *Interdisciplinary Statistics and Bioinformatics.* 87–90.

[52] Liam Roditty and Virginia Vassilevska Williams. 2013. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference (STOC'13)*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM, 515–524. https://doi.org/10.1145/2488608.2488673

[53] Liam Roditty and Uri Zwick. 2012. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Transactions on Algorithms* 8, 4 (2012), 33.

[54] Gert Sabidussi. 1966. The centrality index of a graph. *Psychometirka* 31 (1966), 581–606.

[55] Dominik Schultes and Peter Sanders. 2007. Dynamic highway-node routing. In *Proceedings of the 6th International Workshop on Experimental Algorithms (WEA'07) (Lecture Notes in Computer Science)*, Camil Demetrescu (Ed.), Vol. 4525. Springer, 66–79. https://doi.org/10.1007/978-3-540-72845-0_6

[56] Avi Shoshan and Uri Zwick. 1999. All pairs shortest paths in undirected graphs with integer weights. In *40th Annual Symposium on Foundations of Computer Science (FOCS'99).* IEEE Computer Society, 605–615. https://doi.org/10.1109/SFFCS.1999.814635

[57] Mikkel Thorup. 1997. Undirected single source shortest path in linear time. In *38th Annual Symposium on Foundations of Computer Science (FOCS'97).* IEEE Computer Society, 12–21. https://doi.org/10.1109/SFCS.1997.646088

[58] Virginia Vassilevska Williams. 2018. On some fine-grained questions in algorithms and complexity. *Proc. of the ICM* (2018).

[59] Oren Weimann and Raphael Yuster. 2013. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Transactions on Algorithms* 9, 2 (2013), 14.

[60] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2–3 (2005), 357–365. https://doi.org/10.1016/j.tcs.2005.09.023

[61] Ryan Williams. 2018. Faster all-pairs shortest paths via circuit complexity. 47, 5 (2018), 1965–1985.

[62] Virginia Vassilevska Williams. 2011. Faster replacement paths. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'11)*, Dana Randall (Ed.). SIAM, 1337–1346. https://doi.org/10.1137/1.9781611973082.102

[63] Virginia Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC'12)*, Howard J. Karloff and Toniann Pitassi (Eds.). ACM, 887–898. https://doi.org/10.1145/2213977.2214056

[64] Virginia Vassilevska Williams and R. Ryan Williams. 2018. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM* 65, 5 (2018), 27:1–27:38. https://doi.org/10.1145/3186893

[65] Uri Zwick. 1998. All pairs shortest paths in weighted directed graphs - Exact and almost exact algorithms. In *39th Annual Symposium on Foundations of Computer Science (FOCS'98).* IEEE Computer Society, 310–319. https://doi.org/10.1109/SFCS.1998.743464

[66] Uri Zwick. 2002. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM* 49, 3 (2002), 289–317.

Received 9 September 2021; revised 11 August 2022; accepted 19 August 2022