Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter

Amir Abboud* Fabrizio Grandoni† Virginia Vassilevska Williams‡

Abstract

Measuring the importance of a node in a network is a major goal in the analysis of social networks, biological systems, transportation networks etc. 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 constant factor. Thus the latter two centrality problems could potentially be solved in truly subcubic time, even if APSP requires essentially cubic time.

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), etc. A variety of graph theoretic notions of node importance have been proposed, among the most relevant ones: betweenness centrality [20], graph centrality [29], closeness centrality [47], and reach centrality [28].

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 paper 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 $w : E \to \{0, \ldots, M\}$ for some $M \geq 1$[2]. 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. Let also $\sigma_{s,t}$ be the number of distinct shortest paths from s to t, and $\sigma_{s,t}(b)$ be the number of such paths that use node b as an intermediate node.

- 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 $M^* := \min_{m^* \in V} \sum_{v \in V} d(m^*, v)$.
- 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

$$BC(b) := \sum_{s,t \in V-\{b\},\, s \neq t} \frac{\sigma_{s,t}(b)}{\sigma_{s,t}}.$$

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 problem (APSP) on the underlying graph and doing a negligible amount of post-processing. The same holds for Betweenness Centrality by assuming

*Stanford University, abboud@cs.stanford.edu.

†IDSIA, University of Lugano, fabrizio@idsia.ch. Partially supported by the ERC Starting Grant NEWNET 279352.

‡Stanford University, virgi@cs.stanford.edu. Supported by NSF Grant CCF-1417238, BSF Grant BSF:2012338 and a Stanford SOE Hoover Fellowship.

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

[2] Though we focus here on non-negative weights, our results can be extended to the case of directed graphs with possibly negative weights and no negative cycles.

that shortest paths are unique: we will next make this assumption unless differently stated[3]. Using the best known algorithms for APSP [55], 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*[4] algorithm?

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 [6] and Median [30] have received more than a thousand citations each.

1.1 Approach. In this paper we address these questions with an approach which can be viewed as a refinement of NP-completeness. The approach strives to prove, via combinatorial *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 high-level, the idea is to consider a given prototypical problem A for which the fastest known algorithm has running time $\tilde{O}(n^c)$ (here n is a size parameter). Then we show that a $\tilde{O}(n^{c-\varepsilon})$-time algorithm for a second problem B, for some constant $\varepsilon > 0$, would imply a $\tilde{O}(n^{c-\delta})$-time algorithm for problem A for some other constant $\delta > 0$. This can be used as evidence that a $\tilde{O}(n^{c-\varepsilon})$ time algorithm for problem B is unlikely to exist (or at least very hard to find). For $c = 3$ a reduction of the above kind is called a *subcubic reduction* [53] 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 algorithm for one problem implies a truly subcubic algorithm for the other and vice versa.

Vassilevska Williams and Williams [53] 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 [27, 46, 51, 54]. Other examples of this approach [1, 2, 44] include the famous results on 3-SUM hardness starting with the work of Gajentaan and Overmars [21].

In this paper 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 quite different problems to the list of *APSP-hard* problems [53] and if *any* of these problems can be solved in truly subcubic time then *all* of them can.

THEOREM 1.1. *Radius, Median, and Betweenness Centrality are equivalent to APSP under subcubic reductions.*

Unfortunately, this is strong evidence that a truly subcubic algorithm for computing these centrality measures is unlikely to exist (or at least 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 [3, 5, 10, 45]. Furthermore, the exact algorithms for Diameter and Radius in graphs with small integer weights are also extremely similar [13]. Our results seem to indicate the following bizarre phenomenon. In dense graphs, Radius seems to be harder than Diameter, as it is actually equivalent to APSP, whereas a Diameter/APSP equivalence seems elusive. In sparse graphs, however, Diameter seems more difficult than Radius since a known reduction [45] from CNF-SAT seems to imply that even approximating the Diameter in subquadratic time is hard. No such reduction is known for Radius, and so far there is no evidence against a subquadratic Radius algorithm in sparse graphs.

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

THEOREM 1.2. *Diameter and Reach Centrality are equivalent under subcubic reductions.*

[3] In the case of multiple shortest paths, the fastest known algorithm for Betweenness Centrality takes $\tilde{O}(n^4)$ time; please see the discussion in the related work section.

[4] We recall that a *truly subcubic* algorithm is an algorithm with running time $\tilde{O}(n^{3-\delta})$ for some constant $\delta > 0$.

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 1.2 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 1.3. *There exists an $\tilde{O}(Mn^{\omega})$ time algorithm for Reach Centrality in directed graphs.*

Above $\omega \in [2, 2.373)$ [12, 14, 22, 52] denotes fast matrix multiplication exponent. The previous best algorithm for small integer weights, which is based on the solution of APSP, takes time $\tilde{O}(M^{0.681}n^{2.575})$ [57]. This is interesting in our opinion since Reach Centrality has been used in some very fast (in practice) shortest paths algorithms [24, 25, 28].

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 [4, 7, 23]. Furthermore, the mentioned practically fast shortest paths algorithms [24, 25, 28] can be adapted to work with approximate values of the reach centrality as well. In this paper we formally study the approximability of the mentioned problems.

In more detail, given a graph centrality measure X, our goal is to compute (quickly) a quantity x such that $\frac{1}{\alpha}X \leq x \leq \alpha X$ for some $\alpha \geq 1$ as small as possible (α is the approximation factor). It is known how to solve APSP within a multiplicative error $(1+\varepsilon)$ in time $\tilde{O}(n^{\omega})$ [56]. 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.

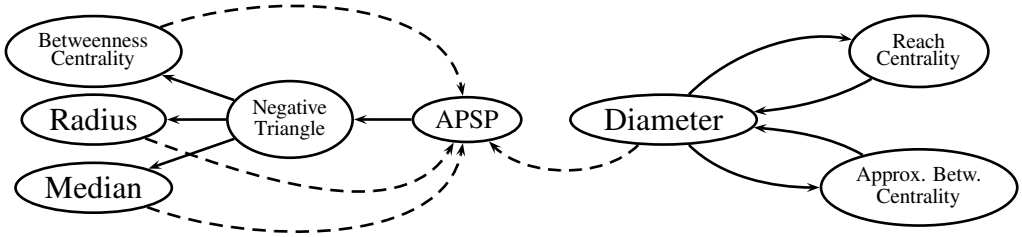
It is not hard to see that any approximation algorithm for Reach/Betweenness Centrality can be used to determine whether $BC(b) > 0$. We show that, while solving the latter problem for a single node is equivalent to Diameter, solving it for every node is at least as hard as APSP! As a consequence any approximation algorithm (for any finite α) to compute the reach/betweenness centrality of *all nodes* implies a truly subcubic algorithm for APSP.

On the positive side, we show that (single-node) Approximate Betweenness Centrality is equivalent to Diameter under subcubic reductions. This equivalence is quite strong: any truly subcubic approximation algorithm with finite approximation factor for Betweenness Centrality implies a truly subcubic Diameter algorithm, while a truly subcubic Diameter algorithm implies a truly subcubic $(1+\varepsilon)$-approximation algorithm for Betweenness Centrality, for any constant $\varepsilon > 0$. Our reductions are Monte-Carlo, i.e. the resulting algorithm might fail to provide the desired approximation with some small probability. In more detail, we provide a subcubic reduction to Diameter to compute the *exact* value of the betweenness centrality when that value is sufficiently small. For the complementary case, we use a non-trivial random sampling algorithm. Analogously to the case of Reach Centrality, this gives some more hope that a truly subcubic algorithm for Approximate Betweenness Centrality exists, however such algorithm is probably not easy to find. Part of the mentioned reductions are summarized in Figure 1.

1.5 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 [16] from every source node, and solve the problem in time $O(mn + n^2 \log n)$ (by implementing Dijkstra's algorithm with Fibonacci heaps [19]). Johnson [36] showed how to obtain the same running time in the case of negative weights also (but no negative cycles). Pettie [41] improved the running time to $O(mn + n^2 \log\log n)$ and together with Ramachandran to $O(mn \log \alpha(m,n))$ [42]. 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 [50]. 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 [18]. Following a long sequence of improvements (among others, [8, 31]), very recently Williams [55] 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 [49] and in directed graphs in $\tilde{O}(n^2(Mn)^{\frac{1}{4-\omega}})$ time [57]. The result for the directed case can be refined to $\tilde{O}(M^{0.681}n^{2.575})$ using fast rectangular matrix multiplication [32].

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 [13] and the references therein). Faster approximation algorithms are known. Aingworth et al. [3] 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 [5]. The running time for both Radius and Diameter was reduced to $\tilde{O}(m\sqrt{n})$ by Roditty and

Figure 1 The main subcubic reductions considered in this paper. Dashed arrows correspond to trivial reductions. All the remaining reductions are given in this paper, excluding the one from APSP to Negative Triangle which is taken from [53].



Vassilevska Williams [45] (see also [10] 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 Strong Exponential Time Hypothesis (SETH) of [33] fails, thus showing that improving on the $3/2$-approximation factor while still using a fast algorithm would be difficult.

The notion of betweenness centrality was introduced by Freeman in the context of social networks [20], 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 [15, 35], social networks [40, 43], sexual networks [38], and terrorist networks [11, 37]. From an algorithmic point of view, betweenness centrality was used to identify a highway-node hierarchy for routing in road networks [48]. Brandes' algorithm [6] 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 [6], similarly to other papers in the area, neglects the bit complexity of the counters which 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 $O(\log N) = O(n \log n)$. Indeed, in some applications one can even assume that shortest paths are unique (as we do in most of this paper). The uniqueness of shortest paths is either a consequence of tie breaking rules (*Canonical-Path Betweenness Centrality* problem [23]), or can be enforced by perturbing edge weights [24]. However, 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 [4, 7, 23]. 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 [4, 7, 23] for small values of the betweenness centrality.

The Reach Centrality notion was introduced by Gutman [28] 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 [24, 25].

Eppstein and Wang [17] 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 which w.h.p. computes estimates within an additive error εD^*, where D^* is the diameter of the graph. The same problem is investigated in [7] from an experimental point of view. The Median problem was also studied in a distance-oracle query model [9, 26, 34].

1.6 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 labelled 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.

We remark that, in our subcubic reductions, it would be sufficient to preserve (modulo poly-logarithmic 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.

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.

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 lower bounded by m. More generally, however, it is sufficient for our proofs that $\tilde{O}(m+T(\tilde{O}(n),\tilde{O}(m))) = \tilde{O}(T(n,m))$ and similarly for the other cases.

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

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 subproblem: 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 [53].

LEMMA 2.1. *[53] 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 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 the final number of nodes is 2^{k+1} for some integer k.

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 2. Therefore, any algorithm with (polynomial and at least linear in m) running time $\tilde{O}(T(n,m,M))$ for the modified instance, can be used to solve the original instance in time $\tilde{O}(m+T(O(n),m+O(n),2M)) = \tilde{O}(T(n,m,M))$.

Combining the reductions below with Lemma 2.1 proves Theorem 1.1.

2.1 Betweenness Centrality.

We start with the reduction to Betweenness Centrality.

LEMMA 2.2. *Given a $\tilde{O}(T(n,m))$ time algorithm for Betweenness Centrality in directed or undirected graphs, there exists a $\tilde{O}(T(n,m))$ time algorithm for Negative Triangle.*

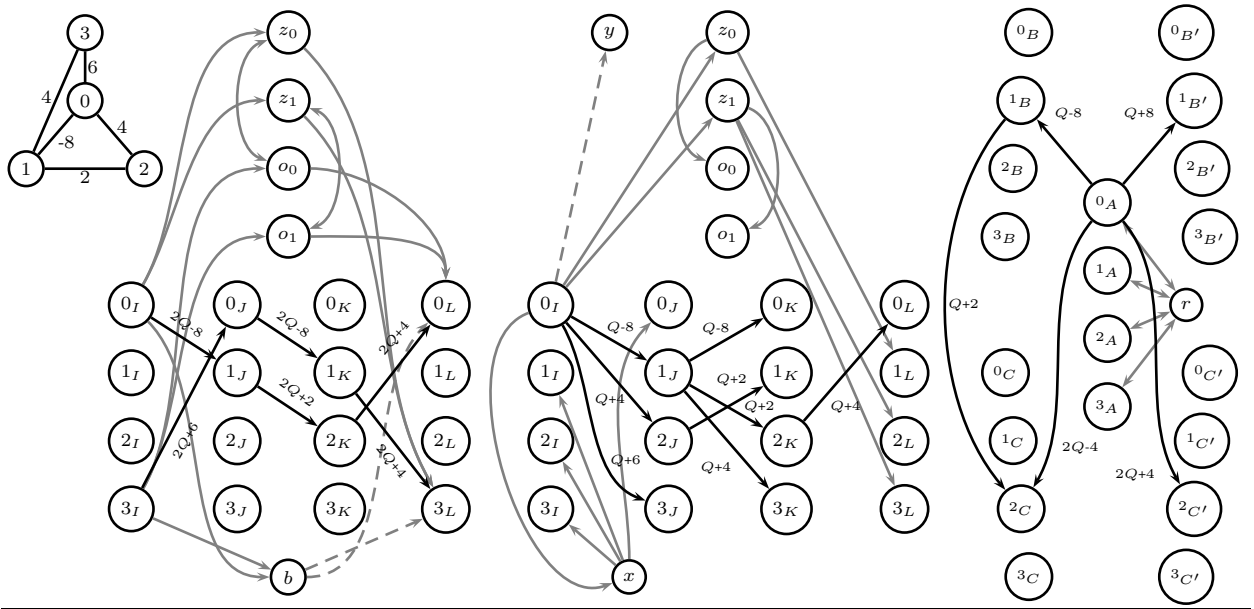
Proof. Let $(G=(V,E),w)$ be the input instance of Negative Triangle (reduced as described above). In particular, $n=2^{k+1}$ is the number of nodes of G.

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_Iv_J, u_Jv_K, and u_Kv_L, and assign to those edges weight $2Q+w(uv)$. We add to G' a dummy node b, and edges v_Ib and bv_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_Iz_j and o_jv_L if $v^j=0$, and edges v_Io_j and z_jv_L otherwise. We also add edges o_jz_j and z_jo_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 iff $BC(b)<n$. The running time of the algorithm is $\tilde{O}(m+T(O(n),O(m+n\log n))) = \tilde{O}(T(n,m))$. Let us prove its correctness. 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 algorithm is as before, and the running time trivially remains $\tilde{O}(T(n,m))$. Proving correctness requires a slightly more complicated case analysis. Consider any pair $s,t\in V-\{b\}$. Suppose $(s,t)\notin (I\times L)\cup(L\times I)$.

Figure 2 (Left) Negative Triangle instance. **(Middle-Left)** Reduction 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 it (since 3 does not belong to any negative triangle). **(Middle-Right)** Reduction 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. **(Right)** Reduction 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.



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 which 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.

For the sake of simplicity, we did not enforce uniqueness of shortest paths in our reduction but that can be done easily by perturbing the edge weights by small polynomial factors exploiting the Isolation Lemma [39].

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 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 $\tilde{O}(T(n,m,M))$ 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 2). Let $Q = \Theta(M)$ be a sufficiently large integer. We construct a directed weighted graph (G', w') as follows. Similarly to 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 $u_I v_J$, $u_J v_K$, and $u_K v_L$ of weight $Q + w(vu)$. We also add to G' two sets of nodes $Z = \{z_0, \ldots, z_k\}$ and $O = \{o_0, \ldots, o_k\}$. 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 $v^0, v^1, \ldots, v^k$ be the binary representation of node v: we add the edges $v_I z_j$ and $o_j v_L$ if $v^j = 0$, and the edges $v_I o_j$ and $z_j v_L$ otherwise. We also

add edges $z_j o_j$ and $o_j z_j$ of weight Q for all $j = 0, \ldots, k$. Finally, we add nodes x and y, and for any $v \in V$ we add edges $v_I x$, $x v_I$, and $x v_J$ of weight Q, and edges $v_I y$ of weight $3Q - 1$.

We compute the radius R^* of (G', w'), and output YES to the input instance of Negative Triangle iff $R^* \leq 3Q - 1$. 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 trough 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 $3Q-1$ iff there exists a negative triangle in (G, w).

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.

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

LEMMA 2.4. *Given a $\tilde{O}(T(n, M))$ 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 2). 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 $r v_A$ and $v_A r$ of weight $Q/4$ for any $v \in V$.

In this graph we compute the median value M^*, and output YES to the input instance of Negative Triangle iff $M^* < Q/4+(n-1)Q/2+6nQ$. 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. Next $d(\cdot)$ denotes distances in G'. Observe that the median node has to be in $A \cup \{r\}$ since the remaining nodes cannot reach r. Note that

$$\begin{aligned} Med(r) &\geq n\frac{Q}{4} + (\frac{Q}{4} + 2Q - 2M)2n \\ &\quad + (\frac{Q}{4} + Q - M)2n \\ &> \frac{Q}{4} + (n-1)\frac{Q}{2} + 6nQ. \end{aligned}$$

On the other hand, for any node v_A,

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

Therefore the median is in A. In the last inequality we upper bounded $d(v_A, u_C)$ with $w'(v_A u_C) = 2Q-w(vu)$. Observe that a strict inequality holds if there exists a third node z_B such that $w'(v_A z_B) + w'(z_B u_C) < w'(v_A u_C)$. Note that 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 $zu \notin E$, $w'(v_A z_B) + w'(z_B u_C) \geq 2Q+M \geq w'(v_A u_C)$. Therefore we can conclude that the strict inequality holds iff 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. 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 iff v belongs to a negative triangle. For any $u_B \in B$, $Med(u_B) \geq (Q - M + Q/4) + 2n(Q-M) + n(2Q-2M) + n(3Q-2M) = (7n+5/4)Q - (6n+1)M$. Similarly $Med(u_{B'}) \geq (9n+5/4)Q-(7n+1)M$, $Med(u_C) \geq (10n+9/4)Q-(9n+2)M$ and $Med(u_{C'}) \geq (12n+9/4)Q-(8n+1)M$. Furthermore, $Med(r) \geq nQ/4 + 2n(5Q/4 - M) + n(9/4Q - 2M) + n(9/4Q - M) = (29n/4)Q - 5nM$. We can conclude that the median is in A. The correctness follows.

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 $\tilde{O}(T(n,M))$ 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. We construct the usual 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 iff $Med(v_I) \pmod 2 = 0$ 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 2.1. *Given a truly subcubic algorithm for All-Nodes Median Parity, there exists a truly subcubic algorithm for APSP.*

3 Subcubic Equivalence with 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 $\tilde{O}(T(n,m))$ time algorithm for Reach Centrality in directed (resp., undirected) graphs, there is a $\tilde{O}(T(n,m))$ time algorithm for Diameter in directed (resp., undirected) graphs.*

Proof. Let $(G = (V,E), w)$ be the input instance of Diameter. Consider first the directed case. Let D be an integer in $[1, (n-1)M]$. Construct an auxiliary weighted graph (G', w') consisting of a copy of (G, w) plus a dummy node b and dummy edges vb and bv of weight $D/2$ for any $v \in V$[5]. Observe that any pair of nodes $s, t \in V$ is connected by a path of length D using b. By performing a binary search on D and solving each time the resulting instance of Reach Centrality on b, we determine the largest value D' of D such that the answer is $RC(b) \geq D/2$. 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 $RC(b) \geq D/2$. 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 and its analysis are analogous to the directed case.

Now, we present the more tricky reduction to Diameter. The following very efficient reduction completes the equivalence between Diameter and Reach Centrality and, using the $\tilde{O}(Mn^\omega)$ [13] algorithm for Diameter in directed graphs, gives the new $\tilde{O}(Mn^\omega)$ algorithm for Reach Centrality in Theorem 1.3.

LEMMA 3.2. *Given a $\tilde{O}(T(n,m,M))$ 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. 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[6]. The value of

[5] In order to avoid fractional edge weights, it is sufficient to multiply edge weights by a factor 2.

[6] Observe that the reach of a node is upper bounded by one half of the diameter.

$RC(b)$ can then be determined via binary search with an extra factor $O(\log(nM)) = \tilde{O}(1)$ 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 $RC(b) \geq K$ iff $D^* \geq 2K + 2M$. 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 iff b is along some shortest s-t path. Therefore $D^* \leq 2K + 2M$ and the equality holds iff 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. iff $RC(b) \geq K$. The correctness follows.

Our subcubic reduction in the undirected case is slightly less efficient in terms of the edge weights and will follow from Lemmas 4.2 and 4.3 of the next section.

4 Fast 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 $BC(b) > 0$ for a given node b (i.e., whether some shortest path uses b as an intermediate node). In this case it is convenient to consider the standard definition of $BC(b)$, where multiple shortest paths are allowed. Our results can be easily extended to the case of unique shortest paths by perturbing weights by small polynomial factors.

Trivially, any α-approximation algorithm for Betweenness Centrality, for any finite α, also solves Positive Betweenness Centrality since the answer is 0 iff $BC(b) = 0$. A similar reduction also works for Reach Centrality. In more detail, by definition $RC(b) \geq \min\{d(b, b), d(b, b)\} = 0$ and $RC(b) > 0$ implies $BC(b) > 0$. However, it might still be that $RC(b) = 0$ and $BC(b) > 0$. We can solve this issue by initially perturbing edge weights by small polynomial factors, so as to obtain an *equivalent* instance of Positive Betweenness Centrality where all weights are strictly positive. In the reduced instance $RC(b) > 0$ iff $BC(b) > 0$.

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 4.1. *Diameter and Positive Betweenness Centrality are equivalent under subcubic reductions.*

Theorem 4.1 follows from the following two lemmas.

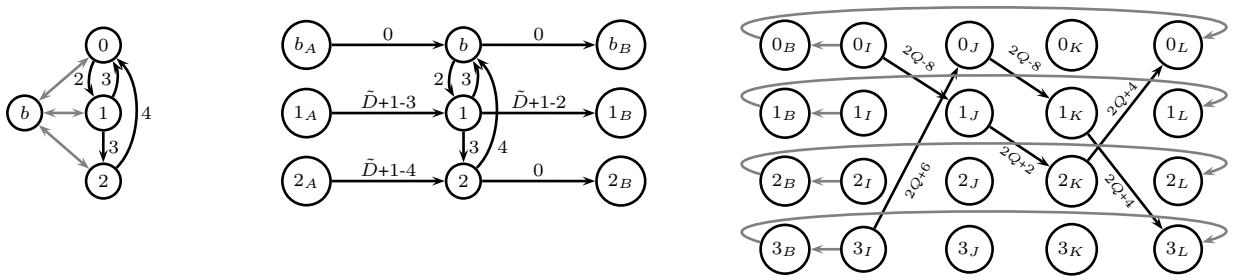
LEMMA 4.1. *Given a $\tilde{O}(T(n, m))$ time algorithm for Positive Betweenness Centrality in directed (resp., undirected) graphs, there is a $\tilde{O}(T(n, m))$ time algorithm for Diameter in directed (resp., undirected) graphs.*

Proof. Let $(G = (V, E), w)$ be the input instance of Diameter. Consider first the directed case (see also Figure 3). Let D be an integer in $[1, (n - 1)M]$. Construct an auxiliary weighted graph (G', w') consisting of a copy of (G, w) plus a dummy node b and dummy edges vb and bv of weight $D/2$ for any $v \in V$[7]. Observe that any pair of nodes $s, t \in V$ is connected by a path of length D using b. By performing a binary search on D and solving each time the resulting instance of Positive Betweenness Centrality on b, 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

[7] In order to avoid fractional edge weights, it is sufficient to multiply edge weights by a factor 2.

Figure 3 (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. **(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 it.



(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 4.2. *Given a $\tilde{O}(T(n,m,M))$ time algorithm for Diameter in directed (resp., undirected) graphs, there is a $\tilde{O}(T(n,m,M))$ time algorithm for Positive Betweenness Centrality in directed (resp., undirected) graphs.*

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

Let us consider the directed case first (see also Figure 3). 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 $D = 3M$ (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. Initially $G' = G$. We add two copies A and B of V. Let v_A be the copy of $v \in V$ and define v_B analogously for B. For every $v \in V$, we add edges v_Av and vv_B 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_Av and vv_B, 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 iff $D^* \geq 2\tilde{D}+2$. 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_Av and vv_B have positive weight, we can assume w.l.o.g. that $s^* = s_A \in A$ and $t^* = t_B \in B$. 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 iff 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 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).

LEMMA 4.3. *Given a $\tilde{O}(T(n,m))$ time algorithm for Positive Betweenness Centrality in directed (resp., undirected) graphs, there is a $\tilde{O}(T(n,m))$ time algorithm for Reach Centrality in directed (resp., undirected) graphs.*

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 $O(\log(nM)) = \tilde{O}(1)$ 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 $d(v,b)$ iff $d(v,b) \geq K$.

Symmetrically, we add an edge bv of weight $d(b,v)$ iff $d(b,v) \geq K$.

We solve the Positive Betweenness Centrality instance (G', w', b) and output that $RC(b) \geq K$ iff 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).

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 4.4. *Given a $\tilde{O}(T(n,m,M))$ time algorithm for All-Nodes Positive Betweenness Centrality 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 3). 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 iff $BC(u_B) > 0$ 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 4.1. *Given a truly subcubic approximation algorithm for All-Nodes Reach/Betweennees Centrality, there exists a truly subcubic algorithm for APSP.*

4.2 A PTAS for Betweenness Centrality. In this section we prove the subcubic equivalence between Approximate Betweenness Centrality (for any constant approximation factor $\alpha > 1$) and Diameter.

THEOREM 4.2. *Diameter and Approximate Betweenness Centrality are equivalent under subcubic Monte-Carlo reductions.*

Similarly to the case of Reach Centrality, it is not hard to see that a truly subcubic α-approximation algorithm for Betweenness Centrality provides a truly subcubic algorithm for Positive Betweenness Centrality (hence for Diameter via Theorem 4.1).

We next show that a truly subcubic algorithm for Diameter implies a truly subcubic PTAS for Betweenness Centrality, i.e. an algorithm that computes a $(1+\varepsilon)$ approximation of the betweenness centrality of a given node for any given constant $\varepsilon > 0$. Our PTAS is Monte-Carlo: it provides the desired approximation w.h.p.

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 $B_{med} \in [0, (n-1)(n-2)]$ be a integer parameter to be fixed later. Our PTAS is based on two different algorithms: one for $B^* \leq B_{med}$ and the other for $B^* > B_{med}$.

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 that problem to Diameter.

LEMMA 4.5. *Given a $\tilde{O}(T(n,m))$ time algorithm for Diameter in directed (resp., undirected) graphs, there exists a $\tilde{O}(T(n,m))$ time algorithm for Positive (S,T)-Betweenness Centrality in directed (resp., undirected) graphs.*

Proof. We use a construction similar to the one in the proof of Lemma 4.2. Let (G, w, b, S, T) be the considered instance of Positive (S,T)-Betweenness Centrality. We start with the directed case, the undirected one being analogous. Let us construct a directed weighted graph (G', w'). Graph G' contains a copy of G. Furthermore, it contains a copy of S and a copy 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 $d(s,b)$ and $d(b,t)$, 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 $K - d(s,b)$ and $K - d(b,t)$, respectively. Observe that the latter weights are lower bounded by 2. We also add two nodes r' and r'', with an edge $r'r''$ of weight 2. We add edges vr' and $r'v$ for every $v \in V \cup S$ of weight $K-1$. Symmetrically, we add edges vr'' and $r''v$ for every $v \in V \cup T$ of weight $K-1$. We also add edges vr' of weight $K-1$ for every $v \in T$. We compute the diameter D^* of (G', w'), and output YES iff $D^* < 2K$.

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. The distance from any node $v \in V \cup T \cup \{r', r''\}$ to any node $w \in V \cup S \cup T \cup \{r', r''\}$ is at most $2(K-2)$. Consider next any node $s \in S$. Its distance to any node in $G \cup \{r', r''\}$ is also at most $2(K-2)$. It remains to consider the distance from s to any $t \in T$. Any path using r' or r'' (or both) has length at least $2K$. The shortest path avoiding r' and r'' has length precisely $2K - d(s,b) - d(b,t) + d(s,t) \leq 2K$, where the equality holds iff b belongs to some shortest s-t path. We can conclude that the diameter is smaller than $2K$ iff b is along some shortest s-t path with $s \in S$ and $t \in T$, i.e., iff the answer to the input instance of Positive (S,T)-Betweenness Centrality is YES.

The construction for the undirected case is similar, where we remove edge directions and also remove the edges of type vr' with $v \in T$ (those edges were needed only to guarantee that distances from T to S are smaller than $2K$, while this property should not always hold in the undirected case). The running time remains $\tilde{O}(T(n,m))$. Analogously to the directed case, it is not hard to prove that pairwise distances are always smaller than $2K$ excluding possibly the distance between some $s \in S$ and $t \in T$ which is equal to $2K$ iff b belongs to some shortest s-t path. The correctness follows.

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

LEMMA 4.6. *Given a $\tilde{O}(T(n,m))$ 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, 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 4.5.

The recursive algorithm works as follows. It initially solve the corresponding Positive (S,T)-Betweenness instance. If the answer is NO, the algorithm outputs 0. Otherwise, if $|S| = |T| = 1$, the algorithm outputs 1. Otherwise, the algorithm partitions arbitrarily S into two subsets S_1 and S_2 of roughly the same cardinality, and it splits similarly T into T_1 and T_2. Then the algorithm solves recursively the sub-problems 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 4 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 4.7. *Given an instance (G, w, b) of Betweenness Centrality, a parameter B_{med}, and an algorithm for Diameter of running time $\tilde{O}(T(n,m))$. There is an $\tilde{O}(B_{med} T(n,m))$ time algorithm which either outputs $B^* = BC(b)$ or answers NO in which case $B^* > B_{med}$.*

Proof. Consider the recursive algorithm from Lemma 4.6. We run that algorithm with $S = T = V - \{b\}$. Furthermore, 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 4.6 shows that the running time is as in the claim.

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, we next drop b (which is clear from the context). Recall 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. 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} iff $BC_{s,T} \geq C \log n$. 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 $\tilde{O}(\frac{nm}{\sqrt{B_{med}}})$. 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 4.8. *W.h.p.* $\tilde{B} \in [(1-2\varepsilon)B^*, (1+2\varepsilon)B^*]$, *where* $\varepsilon > 0$ *tends to zero as* C *tends to* $+\infty$.

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 on $B < \frac{\sqrt{B_{med}}}{1+\varepsilon}$, one obtains

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

Above we used the fact that 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 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

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

The claim follows from the union bound for C large enough.

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,

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

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,

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

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

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

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

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

Similarly

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

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

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

The following lemma summarizes the above discussion.

LEMMA 4.9. *Given an instance (G, w, b) of Betweenness Centrality with $BC(b) = B^* \geq B_{med}$, there is an $\tilde{O}(\frac{nm}{\sqrt{B_{med}}})$ time algorithm that returns a $(1+\varepsilon)$ approximation of B^* w.h.p.*

Combining the algorithms for small and large B^*, we obtain the following result.

LEMMA 4.10. *Given a truly subcubic algorithm for Diameter, there exists a truly subcubic Monte-Carlo PTAS for Betweenness Centrality.*

Proof. Let $\tilde{O}(n^{3-\delta})$ be the running time of the given Diameter algorithm, for some constant $\delta > 0$. From Lemmas 4.7 and 4.9, we can use it to compute w.h.p. a $(1+\varepsilon)$ approximation of the betweenness centrality of a given node in time $\tilde{O}(B_{med} n^{3-\delta} + \frac{n^3}{\sqrt{B_{med}}})$. Choosing $B_{med} = n^{2\delta/3}$ gives a truly subcubic running time in $\tilde{O}(n^{3-\delta/3})$.

Theorem 4.2 follows directly from Lemma 4.10.

4.3 Reductions based on SETH. We are able to show that, assuming the *Strong Exponential Time Hypothesis* (SETH) [33], 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 $\delta > 0$. By the same observation as before, one obtains as a corollary a lower bound on the running time of any approximation algorithm for Betweenness/Reach Centrality.

THEOREM 4.3. *Suppose that there is an $O(m^{2-\varepsilon})$ time algorithm, for any constant $\varepsilon > 0$, that solves Positive Betweenness Centrality 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 for some constant $\delta > 0$[8]. Using the sparsification lemma of [33] (as, e.g., in [10]), we can assume w.l.o.g. that F contains $O(n)$ clauses.

Let us consider the undirected case first (see also Figure 4). We partition the variables into two sets A and B of (roughly) $n/2$ variables each, and create a node for each partial assignment of the variables in A and B, respectively. 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 any assignment of A and B.

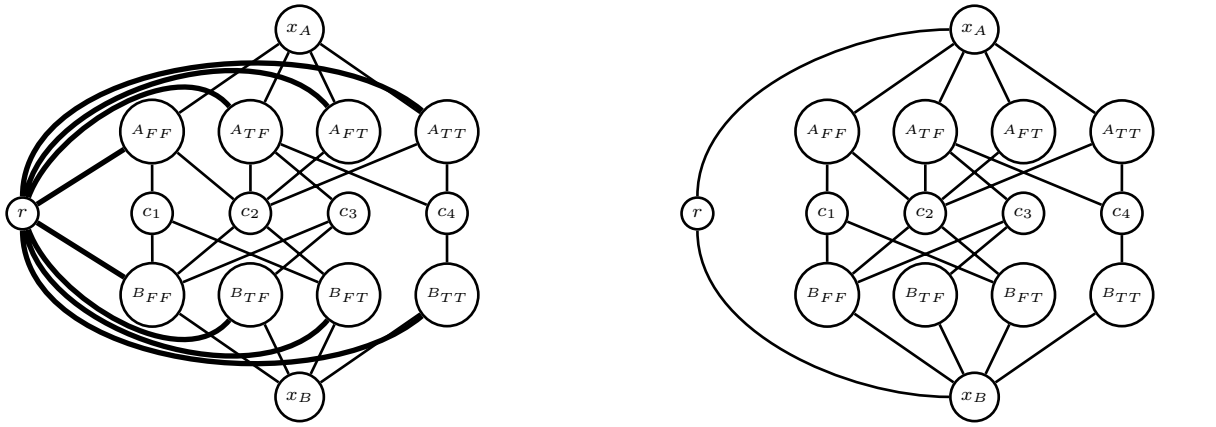
We claim that F is satisfiable iff $BC(b) > 0$. Observe that the distance between any clause node c and any other node is at most 4, while any path passing through b would cost at least 5. Similarly, the distance between any two assignment of A or of B is at most 2, so the corresponding shortest paths do not use b. Given an assignment ϕ_A of A and an assignment ϕ_B of B, there exists a ϕ_A-ϕ_B path of length 2 (hence $BC_{\phi_A,\phi_B}(b) = 0$) iff there exists a clause c that is not satisfied by ϕ_A nor by ϕ_B. Otherwise (i.e., ϕ_A and ϕ_B together satisfy F), ϕ_A, b, ϕ_B is a a shortest such path (hence $BC(b) > 0$). The graph has $O(2^{n/2}n)$ edges, and the conclusion of the lemma follows.

In the directed case we can use a similar construction, 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 and its analysis are analogous to the undirected case.

COROLLARY 4.2. *Suppose that there is an $O(m^{2-\varepsilon})$ time α-approximation algorithm for Betweenness Centrality or Reach Centrality, for any constant $\varepsilon > 0$ and any finite α (possibly depending on m). Then SETH is false.*

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

Figure 4 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}, r, B_{TT} on the left and $A_{FF}, x_A, r, x_B, B_{TT}$ on the right witness that $(X, Y, Z, Q) = (F, F, T, T)$ is a satisfying assignment.



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

THEOREM 4.4. *Suppose there is a $O(m^{2-\varepsilon})$-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 4.3, we can start with a CNF-SAT formula F containing n variables and $m = O(n)$ clauses [33]. 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).

Similarly to the proof of Theorem 4.3, we partition the variables into two subsets A and B of (roughly) $n/2$ variables each, 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 4.3).

To show correctness, note that b is on the shortest path between x_A and x_B and therefore $RC(b) \geq 1$. Second, note that 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, and by definition of these edges it 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.

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 [13]. To the best of our knowledge, it is open whether a $\tilde{O}(Mn^\omega)$ 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 [45] an analogous result is proved for Diameter. It would be interesting to show similar negative results for Radius, Betweenness Centrality and Median (or find faster approximation algorithms in sparse graphs for those problems).

References

[1] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. *FOCS*, 2014.

[2] A. Abboud, V. V. Williams, and O. Weimann. Consequences of faster alignment of sequences. In *ICALP (1)*, pages 39–51, 2014.

[3] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.

[4] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *WAW*, pages 124–137, 2007.

[5] P. Berman and S. P. Kasiviswanathan. Faster approximation of distances in graphs. In *WADS*, pages 541–552, 2007.

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

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

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

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

[10] S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. V. Williams. Better approximation algorithms for the graph diameter. In *SODA*, pages 1041–1052, 2014.

[11] T. Coffman, S. Greenblatt, and S. Marcus. Graph-based technologies for intelligence analysis. *Communications of the ACM*, 47(3):45–47, 2004.

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

[13] M. Cygan, H. N. Gabow, and P. Sankowski. Algorithmic applications of Baur-Strassen's theorem: Shortest cycles, diameter and matchings. In *FOCS*, pages 531–540, 2012.

[14] A. Davie and A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh, Section: A Mathematics*, 143:351–369, 4 2013.

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

[16] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[17] D. Eppstein and J. Wang. Fast approximation of centrality. *J. Graph Algorithms Appl.*, 8:39–45, 2004.

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

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

[20] L. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.

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

[22] F. L. Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

[23] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *ALENEX*, pages 90–100, 2008.

[24] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A*: Efficient point-to-point shortest path algorithms. In *ALENEX*, pages 129–143, 2006.

[25] A. V. Goldberg, H. Kaplan, and R. F. F. Werneck. Better landmarks within reach. In *WEA*, pages 38–51, 2007.

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

[27] F. Grandoni and V. Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *FOCS*, pages 748–757, 2012.

[28] R. J. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *ALENEX/ANALC*, pages 100–111, 2004.

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

[30] S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, 12(3):450–459, 1964.

[31] Y. Han and T. Takaoka. An $O(n^3 \log\log n/\log^2 n)$ time algorithm for all pairs shortest paths. In *SWAT*, pages 131–141, 2012.

[32] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998.

[33] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[34] P. Indyk. Sublinear time algorithms for metric space problems. In *STOC*, pages 428–434, 1999.

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

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

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

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

[39] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

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

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

[42] S. Pettie and V. Ramachandran. A shortest path algorithm

for real-weighted undirected graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

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

[44] M. Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610, 2010.

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

[46] L. Roditty and U. Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Transactions on Algorithms*, 8(4):33, 2012.

[47] G. Sabidussi. The centrality index of a graph. *Psychometirka*, 31:581–606, 1966.

[48] D. Schultes and P. Sanders. Dynamic highway-node routing. In *WEA*, pages 66–79, 2007.

[49] A. Shoshan and U. Zwick. All pairs shortest paths in undirected graphs with integer weights. In *FOCS*, pages 605–615, 1999.

[50] M. Thorup. Undirected single source shortest path in linear time. In *FOCS*, pages 12–21, 1997.

[51] V. Vassilevska Williams. Faster replacement paths. In *SODA*, pages 1337–1346, 2011.

[52] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *STOC*, pages 887–898, 2012.

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

[54] O. Weimann and R. Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Transactions on Algorithms*, 9(2):14, 2013.

[55] R. Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, 2014.

[56] U. Zwick. All pairs shortest paths in weighted directed graphs − exact and almost exact algorithms. In *FOCS*, pages 310–319, 1998.

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