Subcubic Equivalences Between APSP, Co-Diameter, and Other Complementary Problems[*][†]

Sina Dehghani[1], Soheil Ehsani[1], MohammadTaghi Hajiaghayi[1], and Saeed Seddighin[1]

1 **Department of Computer Science, University of Maryland**
`dehghani,ehsani,hajiagha,sseddigh@umd.edu`

Abstract

Despite persistent effort, there is no known technique for obtaining super-linear lower bounds for the computational complexity of the problems in P. Vassilevska Williams and Williams [38] introduce a fruitful approach to advance a better understanding of the computational complexity of the problems in P. In particular, they consider *All Pairs Shortest Paths* (APSP) and other fundamental problems such as checking whether a matrix defines a metric, verifying the correctness of a matrix product, and detecting a negative triangle in a graph. They show if there is a truly subcubic algorithm (an $O(n^{3-\epsilon})$ time algorithm for a constant $\epsilon > 0$) for any of these problems, then there exist truly subcubic algorithms for other problems as well.

Abboud, Grandoni, and Vassilevska Williams [1] study well-known graph centrality problems such as Radius, Median, etc., and make a connection between their computational complexity to that of two fundamental problems, namely APSP and Diameter. They show any algorithm with truly subcubic running time for these centrality problems, implies a truly subcubic algorithm for either APSP or Diameter.

In this paper we define vertex versions for these centrality problems and based on that we introduce new complementary problems. The main open problem of [1] is whether or not APSP and Diameter are equivalent under subcubic reduction. One of the results of this paper is APSP and CoDiameter, which is the complementary version of Diameter, are equivalent. Moreover, for some of the problems in this set, we show that they are equivalent to their complementary versions. Considering the slight difference between a problem and its complementary, these equivalences give us the impression that every problem has such a property, and thus APSP and Diameter are equivalent. This paper is a step forward in showing a subcubic equivalence between APSP and Diameter, and we hope that the approach introduced in our paper can be helpful to make this breakthrough happen.

1998 ACM Subject Classification Complexity Measures and Classes

Keywords and phrases Complexity, Subcubic Reduction, APSP, Diameter

1 Introduction

Computational complexity focuses on classifying algorithmic problems mostly through providing lower bounds to show solving a certain problem requires at least a certain amount

[*] Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, and a DARPA/AFOSR grant FA9550-12-1-0423.

[†] A reviewer at SODA'16 mentioned the proofs in this paper are not complicated enough and thus it cannot make the SODA bar. We would like to note that the novelty of the problems and technical depth of this paper is in par with reference [1] which appeared in SODA'15 and thus the decision was unfair to us.

of time, memory/space, number of gates in a circuit, etc. However, most of the lower bounds have been proposed for the problems that are not yet proven to be in P. Recently, there has been a vast line of research (see e.g. [1, 2, 4, 27, 38]) to study the computational lower bounds of fundamental problems in P such as All Pairs Shortest Paths in graphs, matrix multiplication, 3-SUM, finding central nodes in a network, etc.

However, despite persistent effort, still there is no known technique for proving super-linear lower bounds for polynomially solvable problems. This directs this line of research to provide conditional lower bounds, i.e., leveraging common and popular assumptions to achieve lower bounds. Vassilevska Williams and Williams [38] introduce a fruitful approach to provide evidence that significantly improving the running time for solving a certain set of problems in P is unlikely. Their approach is to use combinatorial reductions to show improving upon a given upper bound for a computational problem, implies improving a breakthrough algorithm for another famous and fundamental problem. More specifically, consider a well-studied problem A for which the best known algorithm has running time $\tilde{O}(n^c)$. By providing a combinatorial reduction from another problem B to A, it can be shown that an $\tilde{O}(n^{c-\epsilon})$ time algorithm for problem B, for a constant $\epsilon > 0$, yields an $\tilde{O}(n^{c-\delta})$ time algorithm for problem A, for another constant $\delta > 0$. This means it is unlikely to obtain an $\tilde{O}(n^{c-\epsilon})$ time algorithm for problem B. For $c = 3$ a reduction of the above kind is called a *subcubic reduction.* Two problems A and B are called *subcubic equivalent*, if there is a subcubic reduction from A to B and a subcubic reduction from B to A [1, 38].

Vassilevska Williams and Williams [38] prove a subcubic equivalence between APSP and seven other fundamental problems, such as checking whether a matrix defines a metric, verifying the correctness of a matrix product over the (min, +)-semiring, and detecting if a weighted graph has a triangle of negative total edge weight. Since then many other works have used the same approach to obtain interesting hardness results for polynomially solvable problems (see e.g. [1, 3, 4, 18, 27]).

In the past few decades there has not been any significant improvement or computational lower bound for these graph centrality problems, especially for APSP. Therefore proving a subcubic equivalence between a certain problem with cubic time and APSP could be "a huge and unexpected algorithmic breakthrough"[1]. Floyd [16] and Warshall [36] proposed an $O(n^3)$ algorithm for APSP in 1962. There has been many attempts to improve this running time [8, 9, 14, 17, 21, 22, 23, 32, 33, 34, 37, 40]. Nonetheless, the best known algorithm for APSP runs in time $O(\frac{n^3}{2^{\Omega(\sqrt{\log n})}})$[1] [37]. But still "One of the Holy Grails of the graph algorithms is to determine whether this cubic complexity is basically inherent, or whether a significant improvement (say $O(n^{2.99}$ time) is possible"[38].

Abboud, Grandoni, and Vassilevska Williams [1] study a series of fundamental graph centrality problems having tons of applications such as finding influential person(s) in social networks, finding key infrastructure nodes in the Internet or urban networks, and detecting super-spreaders of disease. The problems they consider are Radius, Median, Diameter, etc., for which the fastest known algorithms are of $\tilde{O}(n^3)$ running time. Abboud *et al.* [1] make a connection between the complexity of these problems to that of two fundamental problems, namely APSP and Diameter. In Diameter we are asked to find the maximum distance between any two nodes of a graph. They prove APSP, Radius, and Median are equivalent under subcubic reductions, i.e. a truly subcubic algorithm for any of these problems implies a truly subcubic algorithm for each of the rest. They also show Diameter, reach centrality, and any constant factor approximation algorithm for betweenness centrality are equivalent

[1] This still is not $O(n^{3-\epsilon})$ for a positive constant ϵ.

under subcubic reductions.

However the main open question is whether we can obtain a similar connection between Diameter and APSP. It is straightforward to show a reduction from Diameter to APSP; Once you have all the distances between the nodes, you can find the maximum distance in time $O(n^2)$ but is there a subcubic reduction from APSP to Diameter? Or can the largest distance [2] of a graph be calculated faster [3] than the time required to calculate all pairwise distances.

In this paper we consider the complementary version of Diameter and relate its computational complexity to APSP. In particular, we define CoDiameter as the problem of finding a vertex of graph which is not an endpoint of a diameter, and show a subcubic reduction from APSP to this problem.

▶ **Theorem 1.** APSP *and* CoDiameter *are equivalent under subcubic reduction.*

Furthermore, we define complementary problems for other fundamental problems studied before. For instance, we define the CoRadiusproblem as finding a node which is not a solution of Radius, and the CoMedianproblem as finding a node which is not a solution to Median. In this paper we prove subcubic equivalences between APSP, CoMedian, and CoRadius, which lead to subcubic equivalences between Median and CoMedian, and Radius and CoRadius.

▶ **Theorem 2.** APSP*,* CoMedian*, and* CoRadius *are all equivalent under subcubic reduction.*

We also make a connection between the computational complexities of CoNegativeTriangle and CoAPSPVerification to that of the Diameter problem.

▶ **Theorem 3.** *There exists a subcubic reduction from* CoNegativeTriangle *to* Diameter*.*

▶ **Theorem 4.** *There exists a subcubic reduction from* Diameter *to* CoAPSPVerification*.*

The number of the problems considered in this paper may be high, however Figure 1 perfectly illustrates the complexity relations between the aforementioned problems. Note that in Figure 1 any path from problem a A to problem a B denotes a subcubic reduction from problem A to problem B.

2 Related Work

The most related studies to this paper are by Vassilevska Williams and Williams [38] and Abboud, Grandoni, and Vassilevska Williams [1]. Vassilevska Williams *et al.* [38] introduce the notion of subcubic reduction and prove subcubic equivalences between APSP and seven other fundamental problems. Abboud *et al.* [1] use the same approach to obtain subcubic equivalences among APSP, Diameter, and graph centrality problems such as Radius and Median to show any truly subcubic algorithm for graph centrality problems can be used as a black box to obtain a truly subcubic algorithm for APSP or Diameter.

As mentioned above, APSP is among the most well-studied problems in P, for which there has been a tremendous amount of work to improve its running time [8, 9, 14, 17, 21, 22, 23, 32, 33, 34, 37, 40]. Williams [37] proves there exists an $O(\frac{n^3}{2^{\Omega(\sqrt{\log n})}})$ time algorithm for APSP, which is the best known algorithm so far. However, there are faster algorithms for graphs with small integer weights (see [30, 39]). Also there are various studies to find fast approximations for centrality problems. Chang [10] and Goldreich and Ron [19] present

[2] Namely diameter.
[3] In a subcubic time.

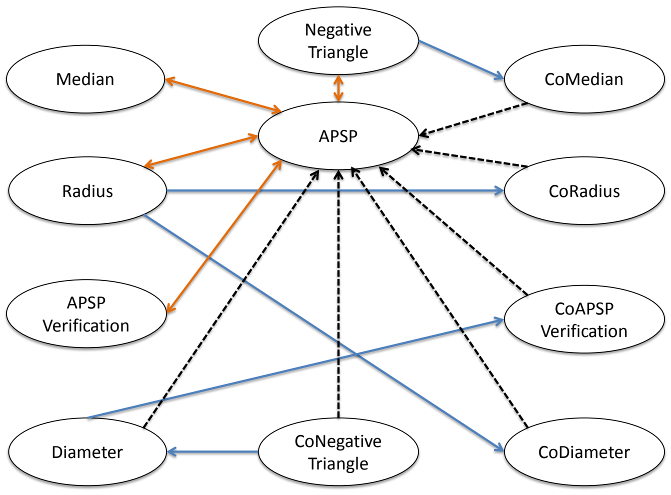

Figure 1 All reduction colored in orange have been known prior to this work, dashed arrows show trivial reductions, and blue arrows illustrate the reductions that we present in this work.

approximation algorithms for Median. Aingworth *et al.* [5] and Berman et al. [7] show there exist (roughly) 3/2 approximation algorithms for Diameter and Radius in undirected graphs that run in time $\tilde{O}(m\sqrt{n}+n^2)$. Roditty and Vassilevska Williams [29] improve the running time of these algorithms to $\tilde{O}(m\sqrt{n})$.

There are also numerous studies obtaining conditional quadratic lower bounds using the *3-SUM* problem [3, 6, 11, 12, 13, 15, 18, 26, 27, 31, 35]. In 3-SUM, given a set S of n integers, we are looking for three elements of S that sum up to zero. Gajentaan and Overmars [18], for the first time, prove many problems in computational geometry are at least as difficult as 3-SUM. Patrascu [27] and Abboud and Vassilevska Williams [3] show polynomial lower bounds for combinatorial problems in dynamic algorithms using subquadratic equivalence with 3-SUM. Subquadratic equivalence with 3-SUM is also used to obtain lower bounds for graph algorithms (see [26, 27, 35]) and lower bounds for Stringology problems (see [4, 11]).

The *Strong Exponential Time Hypothesis (SETH)* of Impagliazzo, Paturi, and Zane [24, 25], has also been an extremely popular conjecture and a powerful tool to provide surprising lower bounds on different problems. According to SETH there is no $O((2-\epsilon)^n poly(n))$ time algorithm to determine the satisfiability of an n-variable CNF formula for some positive ϵ. Interestingly, SETH can also be used to obtain lower bounds for problems in P. Abboud et al. [4] use this technique to obtain lower bounds on string matching problems. Roditty and Vassilevska Williams [29] showed a lower bounds for approximating the diameter of a sparse graph using SETH. SETH has also been used for obtaining lower bounds for dynamic algorithms for maintaining the strongly connected components [3] and 3-party communication complexity of Set-Disjointness [28].

3 Preliminaries

In all of the problems that we study, we assume the given graph has n vertices and m edges. We refer to the vertex set and edge set of a graph G by $V(G)$ and $E(G)$, respectively. For brevity, we sometimes omit the terms directed and weighted, but all of the graphs are considered to be both directed and weighted unless otherwise is stated. Also, the weights of the edges are integer numbers between $-M$ and M where M is a large enough integer number that is polynomially bounded by n. We assume all of the basic operations (addition, subtraction, multiplication, etc.) take time $O(1)$. Whenever we use ∞, it represents a number larger than any other integer number including M. Similarly $-\infty$ is always strictly less than any integer number including $-M$. If there is no edge between a pair of vertices, we assume an edge with weight ∞ for that pair. Addition and multiplication of positive numbers to ∞ and $-\infty$ result in ∞ and $-\infty$ respectively, with an exception of multiplication by zero which is zero.

We say a problem A is *subcubically not harder* than a problem B or there is a subcubic reduction from A to B , if every algorithm that solves problem B in truly subcubic time can be used as a black box to solve problem A in truly subcubic time. We denote this reduction with $A \leq_{n^3} B$. Similarly, two problems A and B are *subcubically equivalent* if both $A \leq_{n^3} B$ and $B \leq_{n^3} A$ hold. This relation is referred to by $A =_{n^3} B$.

In the following, we define all of the problems in detail and explain the relation between them. We divide the problems in three different categories. The first category contains the problems in which the objective function is to measure a quantity of a given graph. The output of these problems are either YES/NO, an integer number, or a matrix of integer numbers. In all of these problems we are given a graph, and the goal is to determine a measure or verify a given property of this graph. In the following we provide a formal definitions for the problems of these three categories. The definitions of the problems may seem repetitive, but as we show later in the paper, this does not necessarily mean the problems are equivalent, and thus such claims require proofs.

3.1 The First Category: Original Version

The problems of this category are in fact some of the well-studied cubic-time problems in their common definition. In the following we shortly bring a definition of each problem so that the reader has a reference to compare these problems with the problems of the next categories.

▶ **Definition 5.** Given a graph G, APSP asks for an $n \times n$ matrix D such that $D_{i,j}$ specifies the distance of the j'th vertex from i'th vertex of G.

We also study another variant of the APSP problem in which we are not required to compute the whole matrix of distances but we only need to verify if a given matrix is the correct distance matrix of the graph.

▶ **Definition 6.** Given a graph G and a matrix D, the objective of APSPVerification is to determine whether D is the correct distance matrix of G.

One of the important problems that has been studied in the literature of subcubic equivalences is the NegativeTriangle problem. In this problem the goal is to determine whether a given digraph has a triangle with a negative weight. Although the solution of every instance of this problem is either YES or NO, it has been shown that this problem is as hard as APSP with regard to having a subcubic algorithm [38].

▶ **Definition 7.** Given a graph G, NegativeTriangle asks whether the graph has a triangle with a negative weight.

We also study the Median, Radius, and Diameter problems for weighted digraphs with non-negative weights. All these problems have been vastly studied in the literature. Many algorithms have been proposed for each of these problems but none of them has a truly subcubic runtime [20, 5]. In a recent work of Abboud et al.[1] it has been shown that a subcubic algorithm for either of these problems leads to a subcubic algorithm for the APSP problem. It is trivial to show that any truly subcubic algorithm for APSP solves any of these problems in truly subcubic time.

▶ **Definition 8.** Given a graph G with non-negative edge weights, the goal of Radius is to find the smallest number R^*, such that there exists a $v \in G$ that can reach every other vertex within a distance of R^*.

▶ **Definition 9.** Given a graph G with non-negative edge weights, the goal of Median is to find a vertex whose total sum of distances to all other vertices is minimal and report this total sum.

▶ **Definition 10.** Given a graph with non-negative edge weights, Diameter asks to compute the longest distance between any pair of vertices in G.

3.2 The Second Category: Vertex Version

In the second category we introduce the vertex versions of the problems in the first category. In Lemma 16 we show equivalences between the original version and the vertex version for some of the problem [4].

▶ **Definition 11.** Given a graph G and a matrix D, the goal of the APSPVerificationIndex problem is to either report that D is the correct distance matrix of G or return an index (i, j) such that $D_{i,j}$ is **not** equal to the distance of vertex j from vertex i.

▶ **Definition 12.** Given a graph G, the goal of the NegativeTriangleVertex problem is to either report the graph has no triangle with a negative weight or report a vertex i which forms such a triangle with two other vertices.

▶ **Definition 13.** Given a graph G with non-negative edge weights, the goal of the RadiusVertex problem is to find a vertex which has the minimum maximum distance to all other vertices. Note that this problem is equivalent to finding a center of G.

▶ **Definition 14.** Given a graph G with non-negative edge weights, the goal of the MedianVertex problem is to find a vertex which has the minimum total sum of distances to all other vertices.

▶ **Definition 15.** Given a graph G with non-negative edge weights, the goal of the DiameterVertex problem is to find a vertex i such that there exists a vertex j that has a distance from i equal to the diameter of the graph.

The reason we define different versions of a problems is because this helps convey a better understanding of the idea behind our reductions. It is important to mention that these different definitions of a problem do not change its hardness under subcubic reductions. To prove this we use binary search as the main tool to solve one problem from another. In other words, it can simply be shown that each problem in the first category is equivalent to its corresponding problem of the second category.

▶ **Lemma 16.** *Given a graph $G = (V, E)$, the following pairs of problems are equivalent under subcubic reduction.*

[4] A similar idea can be used to prove the same claim for the rest of the problems.

- Radius *and* RadiusVertex*(*Center*)*.
- Median *and* MedianVertex.
- Diameter *and* DiameterVertex.

The proof of this lemma is in the Appendix A.

3.3 The Third Category: Complementary Version

The problems of this category are defined in the same way as the problems of the second category, however, the objective here is exactly the opposite. For instance, in DiameterVertex the goal is to find an endpoint of a diameter, where the goal of CoDiameter is to find a vertex that is not an endpoint of a diameter. Although the definitions of two problems seem very similar, we point out a wide gap between them. This is interesting since as shown in this paper, some of the other similar problems such as Median and Radius, are equivalent to their complementary versions. As a contribution of this paper, we simplify the gap between Diameter and APSP to the gap between Diameter and CoDiameter.

▶ **Definition 17.** Given a graph G and a matrix D, the goal of CoAPSPVerification is to either report that none of the entries of D is correct or report a pair (i, j) such that $D_{i,j}$ is equal to the distance of vertex j from vertex i of G.

▶ **Definition 18.** Given a graph G, the goal of CoNegativeTriangle is to either report that every vertex of G is in a negative triangle or return a vertex which is **not** in any negative triangle.

▶ **Definition 19.** Given a graph G, The goal of CoRadius is to report a vertex which is **not** a solution to the RadiusVertex problem for the same input, if exists one. Otherwise reports that every vertex is a solution to Radius, i.e. all vertices are centers of G.

▶ **Definition 20.** Given a graph G, the goal of CoMedian is to report a vertex which is **not** a solution to Median, if exists one.

▶ **Definition 21.** Given a graph G, the goal of CoDiameter is to report a vertex which is **not** a solution to DiameterVertex, if exists one.

4 Reductions

In this section we explain all of our reductions in detail. In Section 4.1 we provide a subcubic reduction from Radius to CoRadius and CoDiameter. In Section 4.2 we show a subcubic reduction from NegativeTriangle to CoMedian. Next, in Sections 4.3 and 4.4 we demonstrate subcubic reductions from Diameter to CoAPSPVerification and from CoNegativeTriangle to Diameter, respectively.

4.1 Radius to CoRadius and Radius to CoDiameter

The main idea behind our proof is constructing a new graph instance and provide a subcubic reduction via a binery search. In contrast to Theorem 25 which directly follows from trivial observations, the proof of Theorem 24 involves more a complicated combinatorial analysis.

▶ **Lemma 22.** *Given an $\tilde{O}(T(n))$ time algorithm for* CoRadius*, where $T(n)$ is polynomial in n, there exists an $\tilde{O}(T(n) + n^2)$ time algorithm for* Radius*.*

Proof. First, without loss of generality we assume every edge in G has an even weight since otherwise we can double the weight of each edge without loss of generality. Let $\mathcal{A}$ be an $\tilde{O}(T(n))$ time algorithm for CoRadius. Given graph G, we construct graph G' as follows. Put all vertices and edges of G in G', plus two new vertices x and y. For each vertex $v \in V(G') \setminus \{x, y\}$ add two bidirectional edges from v to x and y with weight q. Now we claim that the radius of G is less than $2q$ if and only if there is a vertex in G' which is not a center. For simplicity we call such a vertex a CoCenter.

Given the claimed proposition we can use algorithm $\mathcal{A}$ to determine whether there exists a CoCenterin G', in time $\tilde{O}(T(n))$. Hence, a binary search on q can find the minimum value of q such that the radius of G is no less than $2q$, i.e. every vertex in G' is a center. The number of times we need to use $\mathcal{A}$ is $O(\log nM) \in \tilde{O}(\log n)$, therefore there exists an $\tilde{O}(T(n))$ time algorithm for Radius.

In order to prove the claim, we first show that if there exists a CoCenterin G', then the radius of G is less than $2q$. Let u be such a vertex. Note that for each vertex $v \in G'$, there exists a path from u to v of length at most $2q$. Since u is not a center, there exists a vertex $t \in V(G')$ such that for each vertex $v \in V(G')$, there is a path in G' from t to v of length less than $2q$. This implies there exists a vertex $t \in V(G)$ such that for each vertex $v \in V(G)$, there is a path in G from t to v of length less than $2q$. Thus the radius of G is less than $2q$.

Similarly, if the radius of G is less than $2q$, then there exists a CoCenterin G'. The shortest path between x and y is of length $2q$, which implies x is not a center.

The claim is proved and thus there exists an $\tilde{O}(T(n) + n^2)$ time algorithm for Radius. ◀

Interestingly, in proof of Lemma 22, we do not need to know which vertex is not a center. More precisely, it is only sufficient to know whether all vertices of the graph are centers or not. Via the following observation, we conclude the same proof can be used to reduce Radius to CoDiameter.

▶ Observation 4.1. Every graph G has a vertex u which is not a center if and only if it has a vertex v which is not a diameter endpoint of the graph.

▶ **Corollary 23.** *Given an $\tilde{O}(T(n))$ time algorithm for* CoDiameter*, there exists an $\tilde{O}(T(n) + n^2)$ time algorithm for* Radius*.*

The following theorems follow directly from Lemma 22 and Corollary 23.

▶ **Theorem 24.** Radius$\leq_{n^3}$ CoCenter.

▶ **Theorem 25.** Radius$\leq_{n^3}$ CoDiameter.

Note that due to Abboud et al. [1], APSP and Radius are equivalent under subcubic reduction. Thus by Theorems 24 and 25 CoRadius and CoDiameter are also equivalent to APSP under subcubic reduction.

▶ **Corollary 26.** APSP $=_{n^3}$CoRadius $=_{n^3}$CoDiameter.

4.2 NegativeTriangle **to** CoMedian

In this section we provide a subcubic reduction from NegativeTriangle to CoMedian. The reduction uses a tricky graph construction to create a symmetric instance graph that helps to make a connection from NegativeTriangle, which is subcubically equivalent to APSP, to CoMedian.

▶ **Lemma 27.** *Given an $\tilde{O}(T(n))$ time algorithm for* CoMedian*, where $T(n)$ is polynomial in n, there exists an $\tilde{O}(T(n) + n^2)$ time algorithm for* NegativeTriangle*.*

Proof. Given a graph $G(V, E, w)$, we construct a directed graph $G'(V', E', w')$. The approach to solve the problem is to see whether G' has a vertex which is not median (*co-median*). If not, we show finding out whether G has a negative triangle can be done simply by a *Dijkstra* from an arbitrary vertex of G'. To this end, we construct G' with 3 times as many vertices as G has. The limited size of G' ensures that the subcubic reduction is preserved here.

Without loss of generality we assume there exists an edge between every two vertices of G, because otherwise we can put an edge with a big enough weight H and be sure that it does not contribute to any negative triangle. The vertex set of G' contains three copies of $V(G)$, namely A, B and C. Let v_X denote a copy of a vertex $v \in V(G)$ in part $X \in \{A, B, C\}$ of G'. We draw an edge of weight $H/2$ from every v_X to every other u_X in the same part. Moreover, for every two vertices v_A and u_B we draw an edge of weight $H + w_{v,u}$ from v_A to u_B and an edge of weight $2H - w_{v,u}$ from u_B to v_A. We do the same for edges between parts B and C and parts C and A. Figure 2 shows graph G' and the symmetry between its three parts.

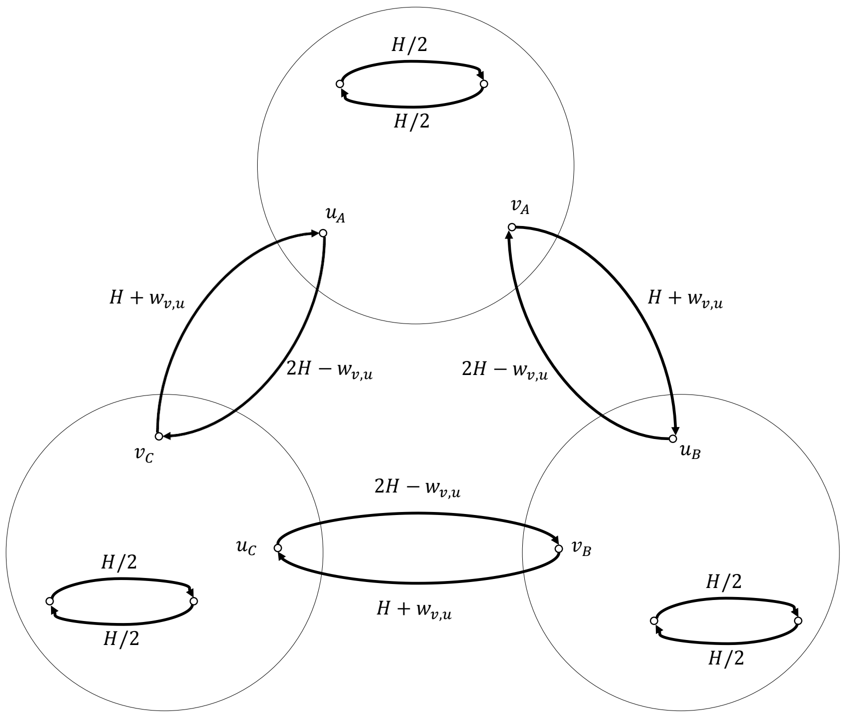

Figure 2 Constructing a symmetric graph G' from G in the reduction from NegativeTriangle to CoMedian.

Since G' is symmetric, we can assume that it has a median in every part. Let r_A be a median in part A. The shortest path from r_A to v_A is a direct edge of weight $H/2$. The shortest path from r_A to every v_B is also a direct edge with weight $H + w'_{r_A,v_B}$. For every v_C, the shortest path from r_A is either a direct edge of weight $2H - w'_{r_A,v_C}$ or a path through an intermediate vertex u_B with total length of $H + w'_{r_A,u_B} + H + w'_{u_B,v_C}$. If the latter is smaller than the former, we can imply that r, u and v form a NegativeTriangle in G:

$$2H + w'_{r_A,u_B} + w'_{u_B,v_C} < 2H - w'_{r_A,v_C} \Rightarrow w_{r,u} + w_{u,v} < -w_{v,r} \Rightarrow w_{r,u} + w_{u,v} + w_{v,r} < 0$$

On the other hand, if the shortest path from r_A to every vertex v_C is the direct edge (r_A, v_C), then using similar inequalities to above, it can be shown that r_A does not contribute to any negative triangle in G. In this case, all vertices of G' have a fixed summation of distances from all other vertices. Let $sum(v)$ denote this summation for a vertex v. Below, we formulate this value only for vertices in part A, because based on the symmetricity of G', the value of $sum(v)$ can be determined via the same formulas for the vertices in parts B and C.

$$\begin{aligned}\forall v_A \in G' : sum(v_A) &= \sum_{x_A \in A \setminus \{v_A\}} (H/2) + \sum_{x_B \in B} (H + w'_{v_A,x_B}) + \sum_{x_C \in C} (2H - w'_{v_A,x_C}) \\ &= (n-1)H/2 + \sum_{x \in V(G)} (H + w_{v,x}) + \sum_{x \in V(G)} (2H - w_{v,x}) \\ &= (n-1)H/2 + 3nH \qquad (1)\end{aligned}$$

According to 1, we only need to construct G' as above and see if all vertices are medians, and if $sum(v) = (n-1)H/2 + 3nH$ for every $v \in V(G')$ [5]. If these two hold, then G is free of negative triangles. Otherwise, there exists a median r_A in G' with $sum(r_A) < (n-1)H/2 + 3nH$ indicating the existence of a negative triangle in G.

◀

▶ **Theorem 28.** NegativeTriangle $\leq_{n^3}$ CoMedian.

4.3 Diameter **to** CoAPSPVerification

In this section we provide a subcubic reduction from Diameter to the CoAPSPVerification problem.

▶ **Lemma 29.** *Given an $\tilde{O}(T(n))$ time algorithm for* CoAPSPVerification*, where $T(n)$ is polynomial in n, there exists an $\tilde{O}(T(n) + n^2)$ time algorithm for* Diameter*.*

Proof. The outline of the proof is as follows. First, we show an algorithm for finding the solution of the CoAPSPVerification problem can be used as a black box for determining whether the diameter of a graph is greater than or equal to an integer number d. Then we run a binary search on d to find the exact diameter of the graph. Assuming the algorithm for CoAPSPVerification runs in subcubic time, the total running time of the algorithm remains subcubic. In the rest we show how we can determine if the diameter of G is at least some given value d.

We construct a graph G' from G by taking all the vertices and edges of G and adding an additional edge from every vertex of G to every other vertex with weight d. By taking the minimum, multiple edges of G' can become simple edges. With this construction the diameter of G' is at most d since there exists a shortcut of weight d between every two vertices. Moreover, if the diameter of G' is exactly d, it means there are two vertices x and y in G such that the distance of y from x is at least d. Otherwise the distance of every vertex of G from every other vertex is at most $d-1$. Thus, the diameter of G is more than or equal to d if and only if there exists a pair (x, y) of vertices in G' such that distance of y from x is exactly d. Let D be an $n \times n$ matrix such that all of its entries are equal to d. If we give G' and D as inputs to the algorithm for CoAPSPVerification, it will report if any

[5] It suffices to check this value just for one vertex, because now we know all vertices are median.

index of D represents the true distance of the corresponding vertices in G', and hence we can determine if the distance of any two vertices of G' is exactly d which is equivalent to G having a diameter of no less than d. ◀

The following theorem follows directly from Lemma 29.

▶ **Theorem 30.** Diameter $\leq_{n^3}$ CoAPSPVerification.

4.4 CoNegativeTriangle **to** Diameter

In this section we provide a subcubic reduction from CoNegativeTriangle to the Diameter problem.

▶ **Lemma 31.** *Given an $\tilde{O}(T(n))$ time algorithm for* Diameter*, where $T(n)$ is polynomial in n, there exists an $\tilde{O}(T(n) + n^2)$ time algorithm for* CoNegativeTriangle.

Proof. For every graph G, we create a graph G' with six times as many vertices. More precisely, $V(G')$ is consisted of six parts A, B, C, D, X, and Y. For every vertex $v \in V(G)$, we put vertices v_A, v_B, v_C, v_D, v_X, and v_Y in parts A, B, C, D, X, and Y, respectively. Moreover, for every edge from a vertex u to a vertex v with weight w in $E(G)$ we draw an edge with from u_A to v_B, u_B to v_C, and from u_C to v_D with weight $w + H$ where $H = 10M$. Furthermore, we add an edge from every v_X to v_A with H. Similarly we draw an edge from every vertex v_D to v_Y with weight H. Finally for every $u \neq v$ we add an edge from u_A to v_D with weight 0.

In the following we show G has a vertex u which does not take part in any negative triangle if and only if the diameter of G' is at least $5H$. Note that due to the construction of G', the diameter of the graph is always the distance of a vertex of part X to a vertex of part Y. Since we put an edge of weight 0 from u_A to v_D for every $u \neq v$, the distance from every vertex u_X to every vertex v_Y is at most $3H$ for $u \neq v$. However, the distance of every vertex v_X to v_Y is more than $3H$. Therefore the diameter of the graph is always from a vertex v_X to a vertex v_Y. Note that, the distance of a vertex v_A to v_D is equal to weight of the minimum weight triangle in G that contains v plus $3H$. Thus, the diameter of the graph is at least $5H$ if and only if there exists a vertex v in G which lies in no negative triangle.

All that remains is to find a vertex which does not contribute to any negative triangle if there exists one. Lemma 16 shows given a truly subcubic algorithm for Diameter that only finds the length of the diameter, we can obtain a truly subcubic algorithm that also find the endpoints of the diameter. Therefore, we can find two vertices v_X and v_Y such that their distance is equal to the diameter of the graph in time $\tilde{O}(n^{3-\delta})$ for some constant δ. If the distance of v_A and v_D is at least $3H$, we can report v as a vertex which does not contribute to any negative triangle, otherwise every vertex of G contributes to a negative triangle. ◀

Theorem 32 follows directly from Lemma 31.

▶ **Theorem 32.** CoNegativeTriangle$\leq_{n^3}$ Diameter.

5 Conclusion and Open Problems

This paper has been an effort to close the gap between sucubical reducibility between APSP and Diameter. We introduce complementary problems in order to draw new reductions between the problems that are subcubically equivalent to either of APSP or Diameter. There are some interesting problems that are left open in this paper. One is whether Diameter or NegativeTriangle are equivalent to their complementary versions. In this paper, we show for

Median and Radius that they are subcubically equivalent to their complementary versions. This gives us the impression that the same claim also holds for the other complementary problems. If so, a groundbreaking subcubic equivalence between APSP and Diameter would be reinforced.

References

1 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1681–1697. SIAM, 2015.

2 Amir Abboud, Virginia Vassilevska, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. 2015.

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

4 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

5 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.

6 Gill Barequet and Sariel Har-Peled. Some variants of polygon containment and minimum hausdorff distance under translation are 3sum-hard. 1998.

7 Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Algorithms and Data Structures*, pages 541–552. Springer, 2007.

8 Timothy M Chan. All-pairs shortest paths with real weights in o (n 3/log n) time. In *Algorithms and Data Structures*, pages 318–324. Springer, 2005.

9 Timothy M Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM Journal on Computing*, 39(5):2075–2089, 2010.

10 Ching-Lueh Chang. Deterministic sublinear-time approximations for metric 1-median selection. *Information Processing Letters*, 113(8):288–292, 2013.

11 Kuan-Yu Chen, Ping-Hui Hsu, and Kun-Mao Chao. Approximate matching for run-length encoded strings is 3sum-hard. In *Combinatorial Pattern Matching*, pages 168–179. Springer, 2009.

12 Otfried Cheong, Alon Efrat, and Sariel Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete & Computational Geometry*, 37(4):545–563, 2007.

13 Mark De Berg, Marko M de Groot, and Mark H Overmars. Perfect binary space partitions. *Computational Geometry*, 7(1):81–91, 1997.

14 Wlodzimierz Dobosiewicz. A more efficient algorithm for the min-plus multiplication. *International journal of computer mathematics*, 32(1-2):49–60, 1990.

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

16 Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

17 Michael L Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5(1):83–89, 1976.

18 Anka Gajentaan and Mark H Overmars. On a class of o (n 2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

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

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

21 Yijie Han. Improved algorithm for all pairs shortest paths. *Information Processing Letters*, 91(5):245–250, 2004.

22 Yijie Han. An o (n 3 (log log n/log n) 5/4) time algorithm for all pairs shortest path. *Algorithmica*, 51(4):428–434, 2008.

23 Yijie Han and Tadao Takaoka. An o (n 3 log log n/log 2 n) time algorithm for all pairs shortest paths. In *Algorithm Theory–SWAT 2012*, pages 131–141. Springer, 2012.

24 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

25 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 653–662. IEEE, 1998.

26 Zahra Jafargholi and Emanuele Viola. 3sum, 3xor, triangles. *arXiv preprint arXiv:1305.3827*, 2013.

27 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 603–610. ACM, 2010.

28 Mihai Patrascu and Ryan Williams. On the possibility of faster sat algorithms. In *SODA*, volume 10, pages 1065–1075. SIAM, 2010.

29 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524. ACM, 2013.

30 Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 605–614. IEEE, 1999.

31 Michael Soss, Jeff Erickson, and Mark Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Computational Geometry*, 26(3):235–246, 2003.

32 Tadao Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. In *Graph-Theoretic Concepts in Computer Science*, pages 209–213. Springer, 1992.

33 Tadao Takaoka. A faster algorithm for the all-pairs shortest path problem and its application. In *Computing and Combinatorics*, pages 278–289. Springer, 2004.

34 Tadao Takaoka. An o (n3loglogn/logn) time algorithm for the all-pairs shortest path problem. *Information Processing Letters*, 96(5):155–161, 2005.

35 Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 455–464. ACM, 2009.

36 Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.

37 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 664–673. ACM, 2014.

38 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.

39 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM (JACM)*, 49(3):289–317, 2002.

40 Uri Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. In *Algorithms and Computation*, pages 921–932. Springer, 2005.

A Equivalence of vertex version and numerical version

Proof of Lemma 16: For each of these three problems, the reduction from the numerical version to the variant in which the output of the problem is a vertex is trivial. This holds since having an optimal solution of the vertex version, we can simply run a single-source-shortest-path algorithm and find the numerical solution of the problem in time $O(n^2)$.

To reduce the vertex version to the numerical version, we assume there is an algorithm that solves the numerical version and will access the algorithm as a black box $\tilde{O}(\log(n))$ times. Note that this keeps the reduction subcubic since the number of accesses is less than n^ϵ for any $\epsilon > 0$. The overall idea is the same for all of the three problems. We first show how to use the solver of the numerical version to see whether or not a set $S \subseteq V$ contains a solution of the vertex version. Then everything boils down to a binary search: Beginning from a set of vertices $S = V(G)$, at each step we divide S into two subsets of size fairly equal S_1 and S_2 and search for the solution of the vertex version in either S_1 or S_2. This cuts the size of the search space into half at every step and finally finds the desired vertex in at most $\lceil \log(n) \rceil$ steps.

In the following, for each of the three problems we show how to use a solver of the numerical version to see whether or not there exists a solution of the vertex version in S.

- Diameter: Suppose the diameter of G is equal to d. We construct G' from G by adding a dummy vertex x' for each $x \in S$ and connecting them with an edge of weight ϵ. Let d' denote the diameter of G'. If there is no vertex in S that is an endpoint of a diameter of G then d' will be equal to d. Otherwise, d' will be at least $d + \epsilon$.
- Radius: Suppose the radius of G is r. We construct G' from G by adding a dummy node x and connecting it to all vertices of S with edges of weight r. Let r' be the radius of G'. If $r' > r$ then all centers of G are in $V \backslash S$, since they need to reach x through S in G'. Otherwise, there exists a center of G in S and $r' = r$.
- Median: Suppose m is the value of the median of G. We construct G' from G by adding a dummy vertex x. Let's Q be a very big number. We connect x to vertices of S with edges of weight Q and to the rest of graph with edges of weight $Q + \epsilon$. Let us use m' to denote the value of median in G'. If there exists a median vertex of G in S, then that vertex can be a median vertex of G' too. In this case, $m' = m + Q$. If no such a vertex exists, then all medians of G' are out of S and $m' = m + Q + \epsilon$.

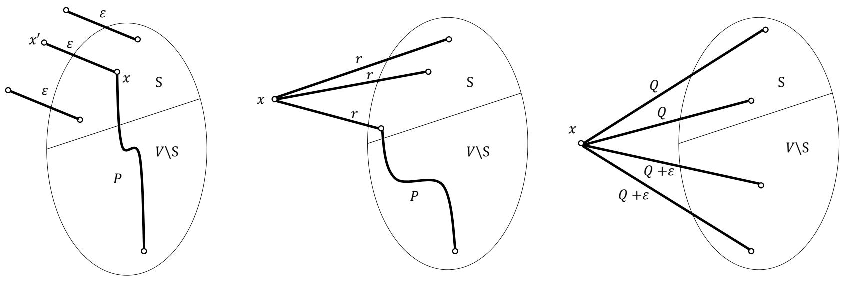

Figure 3 (i) Left figure shows the reduction for Diameter, (ii) the figure in the middle illustrates the reduction for Radius, and (iii) the figure on the right shows the reduction for Median.

■